

RM05/3/2

RM05/3/2 DSKLS TST 1 AH-F931A-MC
CZRMPA0 FICHE 1 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



The main body of the document is a microfiche card containing a grid of 16 columns and 24 rows of data. Each cell in the grid contains a small, high-contrast image of a document page, which is a common format for microfiche storage. The images are arranged in a regular grid pattern across the entire card.

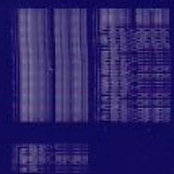
RM05/3/2

RM05/3/2 DSKLS TST 1 AH-F931A-MC
CZRMPA0 FICHE 2 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



Microfiche grid containing multiple frames of data, including text and barcodes.



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

.REM \

IDENTIFICATION

PRODUCT CODE: AC-F930A-MC
PRODUCT NAME: CZRMPA0 RM05/3/2 DSKLS TST 1
DATE CREATED: APRIL 1980
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 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. OPERATOR INTERFACE
 - 1. PROGRAM ID
 - 2. PROGRESS REPORTS
 - 3. PERFORMANCE REPORTS
 - 4. PROGRAM HALTS
 - 5. ERROR REPORTS
- 5. ENVIRONMENTAL SUPPORT
 - 1. PROCESSOR COMPATIBILITY
 - 2. DUAL PORT CONFIGURATIONS
 - 3. MEMORY PARITY HARDWARE
 - 4. MEMORY MANAGEMENT HARDWARE
 - 5. ACT, APT COMPATIBILITY
 - 6. XXDP COMPATIBILITY
 - 7. OPERATING SYSTEM COMPATIBILITY
- 6. TEST DESCRIPTION

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

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM05/3/2 DISKLESS DIAGNOSTIC IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL AND DIAGNOSTIC MEANS TO VERIFY THE OPERABILITY OF THE RM05/3/2 DISK SUBSYSTEM EXCLUDING AND INDEPENDENTLY OF THE STORAGE MODULE DRIVE. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

TO DETECT ERRORS AND FAULTS IN THE RH MASSBUS CONTROLLER;

TO DETECT ERRORS AND FAULTS IN THE RM MASSBUS ADAPTER;

TO RESOLVE HARDWARE FAILURES IN THE RH/RM TO A FIELD REPLACEABLE MODULE OR MODULES.

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM05/3/2 DISK SUBSYSTEM, EXCLUDING THE STORAGE MODULE DISK DRIVE AND THE RH11 OR RH70 MASSBUS CONTROLLER.

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 DISKLESS DIAGNOSTIC:

PDP-11 PROCESSOR
24K 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)

2.2 MEDIA REQUIREMENTS

NONE

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

NONE

3.0 OPERATING PROCEDURE

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

3.1 LOADING

THE PROGRAM MAY BE LOADED BY EITHER PAPER TAPE, USING THE STANDARD PAPER TAPE LOADING PROCEDURE, OR XXDP MEDIA, USING THE APPROPRIATE LOADING DEVICE.

3.2 SWITCH OPTIONS

THE FOLLOWING SWITCH OPTIONS ARE INVOKED WHEN THE APPROPRIATE SWITCH IS ON.

- 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 ARE USED IN CONJUNCTION WITH SW08 TO SPECIFY A PARTICULAR TEST WHICH THE PROGRAM WILL LOOP ON.

3.3 STARTING PROGRAM

THE PROGRAM STARTS AT LOCATION 200 WHICH USES DEFAULT VALUES OF PARAMETERS AND PROVIDES MAXIMUM TESTING WITH THE SWITCH REGISTER EQUAL TO ZERO.

4.0 OPERATOR INTERFACE

4.1 PROGRAM ID

THE PROGRAM TYPES ITS NAME AND MAINDEC NUMBER THE FIRST TIME IT IS STARTED AFTER BEING LOADED.

4.2 PROGRESS REPORTS

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

4.3 PERFORMANCE REPORT

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

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

4.4 PROGRAM HALTS

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

4.5 ERROR REPORTS

THE RM05/3/2 DISKLESS DIAGNOSTIC PROVIDES COMPREHENSIVE ERROR REPORTS INTENDED TO (1) AID IN FAULT RESOLUTION AND (2) MINIMIZE REFERENCES TO PROGRAM LISTINGS.

THE FIRST LINE OF THE ERROR REPORT CONTAINS THE UNIT NUMBER BEING TESTED, 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: SEVERAL LINES OF TEXT WHICH GIVE A COMPREHENSIVE DESCRIPTION OF THE ERROR, AND A LIST OF FAILING MODULES IN ORDER OF DECREASING PROBABILITY. THE ERROR MESSAGE IS NORMALLY FOLLOWED BY ONE OR MORE PAIRS OF LINES CONTAINING DATA HEADERS AND DATA PERTINENT TO THE ERROR, INCLUDING EXPECTED AND ACTUAL TEST RESULTS.

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM05/3/2 DISKLESS DIAGNOSTIC IS EXECUTABLE ON ANY PDP-11 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET.

5.2 DUAL PORT CONFIGURATIONS

THE RM05/3/2 DISKLESS DIAGNOSTIC IS NOT EXECUTABLE ON RM05/3/2 SUBSYSTEMS HAVING THE DUAL PORT OPTION UNLESS THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B) AND NOT TO THE PROGRAMMABLE POSITION (A/B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE WILL NOT BE USED DURING THE EXECUTION OF THE RM05/3/2 DISKLESS DIAGNOSTIC.

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE WILL NOT BE USED DURING THE RM05/3/2

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

DISKLESS DIAGNOSTIC.

5.5 ACT11, APT11 COMPATIBILITY

THE RM05/3/2 DISKLESS DIAGNOSTIC PROGRAM 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 DISKLESS DIAGNOSTIC PROGRAM IS COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES.

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT REQUIRED TO BE COMPATIBLE WITH ANY OPERATING SYSTEM.

6.0 TEST DESCRIPTION

THE PROGRAM IS DESIGNED IN A BOTTOM UP MANNER SUCH THAT EACH TEST GENERALLY USES A MORE COMPLEX SUBSET OF HARDWARE THAN THE PREVIOUS TEST.

MODULE CALLOUT IS PREDICATED ON THE ASSUMPTION THAT EARLIER TESTS HAVE BEEN COMPLETED WITHOUT ERROR AND THAT ERRORS ARE DUE TO SINGLE, NONTRANSIENT HARDWARE FAILURES.

THE 'RM05/3/2 DISKLES DIAGNOSTIC' CAN BE EXECUTED USING AN RH70 OR AN RH11 MASSBUS CONTROLLER.

UNLESS SPECIFIED BY THE OPERATOR OR BY THE ENVIRONMENT TABLE THE TEST IS REPEATED FOR EACH POSSIBLE DEVICE STARTING WITH DEVICE 0.

THE MODULES WHICH MAY BE CALLED OUT DURING THE EXECUTION OF THE TEST ARE AS FOLLOWS:

IF
CS
DS
MASSBUS MODULE

THE RADIAL MODULE (RD) IS NOT TESTED BY THIS PROGRAM.

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

TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE RM05/3/2 CAN COMPLETE A REGISTER TRANSFER ON THE MASSBUS, AND, IN PARTICULAR, TO VERIFY THAT 'TRANSFER' IS NOT STUCK IN AN INACTIVE STATE.

PROCEDURE:

THE PROGRAM WRITES AND READS REMOTE REGISTERS FOR THE SELECTED DEVICE. REGISTER CONTENTS AND PARITY ERRORS ARE IGNORED, AND THE TEST FAILS IF A 'NONEXISTENT DEVICE ERROR' OR BUS TIMEOUT OCCURS FOR EVERY REGISTER ACCESS. IF THE TEST FAILS THE PROGRAM JUMPS TO THE END OF PASS HANDLER WHICH SELECTS THE NEXT DEVICE TO BE TESTED.

PROBABLE FAULT:

THE TEST FAILS IF THE SELECTED DEVICE IS NONEXISTENT OR IS SWITCHED TO THE PROGRAMMABLE POSITION OR TO THE ALTERNATE PORT. THE FOLLOWING FAULTS ARE APPLICABLE ONLY WHEN THE DEVICE IS PRESENT AND IS SWITCHED TO THE APPROPRIATE PORT.

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE
3. CS MODULE

CTOD TEST

PURPOSE:

TO VERIFY THAT DATA CAN BE TRANSFERRED TO AND FROM THE RM05/3/2 USING THE CONTROL BUS AND, IN PARTICULAR, TO VERIFY THAT 'CONTROLLER TO DEVICE' HAS NOT FAILED.

PROCEDURE:

THE TEST WRITES ONES IN REMOTE REGISTERS THEN READS EACH REGISTER WHICH WILL WRITE ZEROS IN THE REGISTER IF 'IF3 CTOD HOLD H' IS STUCK AT ONE. THE TEST THEN READS AS MANY REMOTE REGISTERS AS ARE NECESSARY TO OBTAIN ONE OR MORE ONE BITS.

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

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

MASSBUS INITIALIZE TEST

PURPOSE:

TO VERIFY THAT THE MASSBUS ADAPTER IS BEING INITIALIZED BY THE MASS BUS.

PROCEDURE:

USING CONTROLLER CLEAR TO INITIALIZE THE SELECTED UNIT, THIS TEST THEN READS MASSBUS ADAPTER REGISTERS TO VERIFY THAT AT LEAST ONE BIT IS CLEARED. MASSBUS ADAPTER REGISTERS ARE PRESET TO A NON ZERO VALUE PRIOR TO CONTROLLER CLEAR.

PROBABLE FAULT:

1. ASYNCHRONOUS MASSBUS MODULE
2. IF MODULE
3. CS MODULE

CLEAR STUCK ACTIVE TEST

PURPOSE:

TO VERIFY THAT 'MBA CLR L' ON THE CS MODULE IS NOT STUCK IN AN ACTIVE STATE.

PROCEDURE:

CONTROLLER CLEAR IS USED TO INITIALIZE THE SELECTED UNIT, AFTER WHICH 1'S ARE WRITTEN IN ERROR REGISTERS 1 AND 2 AND MAINTENANCE REGISTER 1. IF ANY 1 BITS CAN BE READ BACK THE TEST IS OK, ELSE, 'MBA CLR L' IS PROBABLY STUCK ACTIVE.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE
3. ASYNCHRONOUS MASSBUS MODULE

TRISTATE TRANSFER 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 PATH TO AND FROM THE MASSBUS ADAPTER TRI-STATE REGISTER BUS IS NOT STUCK AT ONE OR ZERO AND THAT EACH BIT POSITION IS INDEPENDENT.

PROCEDURE:

THIS TEST PRESETS MASSBUS ADAPTER REGISTERS TO A NONZERO VALUE, THEN, ASSUMING THE REGISTERS ARE PRESET, IT CLEARS THEM USING A MOVE INSTRUCTION. THE TEST THEN READS AS MANY REGISTERS AS IS NECESSARY TO OBTAIN ONE OR MORE ZEROS FROM EACH BIT POSITION.

THE TEST CLEARS MASSBUS ADAPTER REGISTERS, THEN, ASSUMING THE REGISTERS ARE CLEARED, IT LOADS THEM WITH ONES AND READS AS MANY REGISTERS AS IS NECESSARY TO OBTAIN ONE OR MORE ONE BITS IN EACH BIT POSITION.

FINALLY, THE TEST WRITES A SINGLE ONE BIT PATTERN IN BIT 0 OF SELECTED REMOTE REGISTERS AND VERIFIES THAT THE PATTERN CAN BE READ BACK. THE ONE BIT IS SHIFTED AND THE TEST REPEATED FOR ALL BIT POSITIONS.

PROBABLE FAULT:

1. ASYNCHRONOUS MASSBUS MODULE
2. IF MODULE
3. CS MODULE
4. DS MODULE

REGISTER SELECT TEST

PURPOSE:

TO VERIFY THAT THE REGISTER SELECT LINES ARE NOT IN A STUCK POSITION.

PROCEDURE:

EACH REGISTER SELECT LINE IS TESTED BY WRITING ZEROS IN THOSE DEVICE REGISTERS FOR WHICH THE LINE MUST BE ZERO, THEN WRITING ONES IN THOSE DEVICE REGISTERS FOR WHICH THE LINE MUST BE ONE. THE ZERO REGISTER IS READ BACK AND IF THE SELECT LINE IS STUCK AT ZERO, THE ZERO REGISTER WILL CONTAIN ONES. THE PROCESS IS REPEATED TO DETECT A STUCK AT ONE FAULT, EXCEPT IN THIS CASE, THE ONES REGISTER IS WRITTEN FIRST.

REGISTER SELECT LINES 1, 2, 4 AND 8 ARE TESTED IN THIS MANNER; SELECT LINE 16 IS EXPLICITLY TESTED IN THE "ILR TEST".

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

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

DRIVE TYPE TEST

PURPOSE:

TO TEST THE 'DRIVE TYPE' REGISTER, RMDT.

PROCEDURE:

THE PROGRAM READS RMDT AND VERIFIES THAT THE RESULT CORRESPONDS TO A SINGLE OR DUAL PORT RM05, RM03 OR RM02 DRIVE.

PROBABLE FAULT:

1. IF MODULE

DEVICE AVAILABLE TEST

PURPOSE:

TO VERIFY THAT DEVICE AVAILABLE STATUS IS SET.

PROCEDURE:

THE PROGRAM TESTS 'DVA', BIT 11 OF RMCS1.

PROBABLE FAULT:

1. IF MODULE

HOLDING REGISTER TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE HOLDING REGISTER IS NOT STUCK AT ONE, STUCK AT ZERO, AND THAT THERE IS NO BIT INTERFERENCE.

PROCEDURE:

THE PROGRAM TRANSFERS ONES, THEN ZEROS TO THE HOLDING

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

REGISTER AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ONE.

THE PROGRAM TRANSFERS ZEROS, THEN ONES TO THE HOLDING REGISTER AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ZERO.

FINALLY, THE TEST TRANSFERS A SHIFTING ONE BIT PATTERN AND VERIFIES THAT EACH BIT IS INDEPENDENT.

PROBABLE FAULT:

1. IF MODULE

CONTROL STATUS #1 TRANSFER TEST

PURPOSE:

TO VERIFY THAT BITS 01 THROUGH 05 OF CONTROL STATUS REGISTER 1 ARE NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THIS TEST WRITES ONES IN CONTROL STATUS REGISTER 1, RMCS1, THEN WRITES ZEROS AND VERIFIES THAT THE BITS ARE NOT STUCK AT ONE. THE GO BIT IS NOT TESTED IN THIS TEST.

NEXT, THE TEST CLEARS THE CONTROL STATUS REGISTER, RMCS1, WRITES ONES IN BITS 01 THROUGH 05 AND VERIFIES THAT THE BITS ARE NOT STUCK AT ZERO. THE GO BIT IS NOT TESTED.

THE TEST TRANSFERS A SHIFTING ONE BIT DATA PATTERN TO AND FROM RMCS1 AND CHECKS FOR ADJACENT BIT INTERFERENCE.

PROBABLE FAULT:

1. IF MODULE

ERROR REGISTER #1 TRANSFER TEST

PURPOSE:

TO VERIFY THAT ERROR REGISTER 1 IS NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THIS TEST WRITES ONES IN ERROR REGISTER 1, RMER1, THEN WRITES ZEROS AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ONE. 'UNSAFE' IS NOT TESTED DURING THIS TEST. IN ORDER TO LIMIT THE

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

PROBABLE FAULTS TO ONE OR TWO MODULES, THE TEST IS EXECUTED IN 3 PARTS WITH EACH PART TESTING THOSE BITS WHOSE PRESET FUNCTIONS ARE DERIVED FROM THE SAME MODULE.

THE TEST WRITES ZEROS IN ERROR REGISTER 1, RMER1, THEN WRITES ONES AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, THE TEST WRITES A SHIFTING ONE BIT PATTERN IN RMER1 AND CHECKS FOR ADJACENT BIT INTERFERENCE.

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE
3. DS MODULE

CLEAR OFFSET STUCK ACTIVE TEST

PURPOSE:

TO VERIFY THAT THE SIGNAL WHICH CLEARS OFFSET MODE IS NOT STUCK IN ACTIVE STATE.

PROCEDURE:

THE TEST WRITES A ONE IN THE OFFSET DIRECTION BIT WHICH IS CLEARED BY THE SIGNAL AND VERIFIES THAT A ONE CAN BE READ BACK.

PROBABLE FAULT:

1. IF MODULE
2. DS MODULE

OFFSET REGISTER TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE OFFSET REGISTER IS NOT STUCK AT ONE, STUCK AT ZERO, AND THAT THERE IS NO ADJACENT BIT INTERFERENCE.

PROCEDURE:

THE OFFSET REGISTER, RMOF, IS WRITTEN WITH ONES, THEN WRITTEN WITH ZEROS AND READ TO VERIFY THAT NONE OF THE BITS ARE STUCK AT ONE.

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

THEN THE OFFSET REGISTER IS WRITTEN WITH ZEROS AND WRITTEN WITH ONES TO VERIFY THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, THE OFFSET REGISTER IS TESTED WITH A SHIFTING ONE BIT PATTERN.

PROBABLE FAULT:

1. IF MODULE

ERROR REGISTER #2 TRANSFER TEST

PURPOSE:

TO VERIFY THAT ERROR REGISTER 2, RMR2, IS NOT STUCK AT ONE, STUCK AT ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THE TEST WRITES ONES THEN WRITES ZEROS IN RMR2 AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ONE. 'SKI' AND 'DVC' ARE NOT TESTED. IN ORDER TO LIMIT THE NUMBER OF PROBABLE FAULTS TO ONE OR TWO MODULES, THE TEST IS EXECUTED IN 3 PARTS WITH EACH PART TESTING THOSE BITS WHOSE PRESET FUNCTIONS ARE DERIVED FROM THE SAME MODULE.

THEN THE TEST WRITES ZEROS IN ERROR REGISTER 2, AND WRITES ONES VERIFYING THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, THE TEST WRITES A SHIFTING ONE BIT PATTERN IN THE REGISTER AND VERIFIES THAT ALL BIT POSITIONS ARE INDEPENDENT.

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE
3. DS MODULE

SERIAL NUMBER TEST

PURPOSE:

TO VERIFY THAT THE SERIAL NUMBER CAN BE READ.

PROCEDURE:

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

THE TEST READS THE SERIAL NUMBER REGISTER SEVERAL TIMES AND VERIFIES THAT THE NUMBER IS THE SAME EACH TIME.

PROBABLE FAULT:

- 1. CS MODULE

CONTROL BUS PARITY DETECTION TEST

PURPOSE:

TO TEST THE RM05/3/2'S PARITY CHECKING LOGIC FOR THE MASSBUS ASYNCHRONOUS CONTROL BUS.

PROCEDURE:

THIS TEST WRITES A SHIFTING ONE BIT DATA PATTERN IN THE DISK ADDRESS REGISTER USING 'PAT' TO CONTROL THE STATE OF THE PARITY BIT. 'PAR' STATUS, BIT 03 OF RMER1, IS CHECKED AFTER EACH PATTERN IS TRANSFERRED..NOTE THE FOLLOWING TABLE SHOWS A SET OF TEST PATTERNS THAT COULD BE USED INSTEAD OF A SHIFTING ONE BIT PATTERN.

DATA PATTERN	PAT	PAR
000000	1	1
075266	0	0
163753	0	0
116535	1	1

PROBABLE FAULT:

- 1. IF MODULE
- 2. ASSYNCHRONOUS MASSBUS MODULE

CONTROL BUS PARITY GENERATION TEST

PURPOSE:

TO TEST THE RM05/3/2'S PARITY GENERATING LOGIC FOR THE MASSBUS ASYNCHRONOUS CONTROL BUS.

PROCEDURE:

THE TEST TRANSFERS A SHIFTING ONE BIT DATA PATTERN TO THE DISK ADDRESS REGISTER. AFTER EACH PATTERN IS READ BACK, 'MASSBUS CONTROL BUS PARITY ERROR' IS TESTED AND SHOULD BE ZERO..NOTE THE FOLLOWING SET OF TEST PATTERNS COULD BE USED INSTEAD OF THE

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

SHIFTING ONE BIT PATTERN.

DATA PATTERN	MCPE
000000	0
056747	0
135672	0
163135	0

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

RMDA, RMDC FAULT TEST

PURPOSE:

TO VERIFY THAT THERE ARE NOT FAULTS WHICH INHIBIT THE PROGRAM FROM WRITING RMDC AND RMDA. SPECIFICALLY, THESE FAULTS INCLUDE:

.'GO H' STUCK HIGH, WHICH WOULD INHIBIT THE REGISTER LOAD FUNCTION,

.'RIP L' STUCK LOW, WHICH WOULD CONSTANTLY CLEAR THE REGISTER

.'EBL' STUCK, WHICH WOULD INHIBIT THE CLOCK FUNCTION.

PROCEDURE:

THE TEST WRITES AND READS BOTH RMDC, AND RMDA. WITH ZEROS, THEN ONES. THE TEST PASSES IF EITHER REGISTER CAN BE WRITTEN WITH ONES.

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE
3. CS MODULE

DISK ADDRESS TRANSFER 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 THE DISK ADDRESS REGISTER IS NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THIS TEST PRESETS THE DISK ADDRESS TO A NONZERO VALUE, THEN USES A MOVE TO CLEAR THE REGISTER. THE TEST THEN READS RMDA AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ONE.

THEN THE TEST PRECLEARS THE MASSBUS ADAPTER DISK ADDRESS REGISTER (RMDA), LOADS IT TO ALL ONES, AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ZERO.

A SHIFTING ONE BIT PATTERN IS TRANSFERRED TO AND FROM THE DISK ADDRESS REGISTER, RMDA, AND THE TEST VERIFIES THAT EACH BIT IS INDEPENDENT.

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

DESIRED CYLINDER TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE DESIRED CYLINDER ADDRESS REGISTER, RMDC, IS NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THIS TEST WRITES ONES IN THE DESIRED CYLINDER REGISTER RMDC, THEN WRITES ZEROS AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ONE.

THEN THE TEST WRITES ZEROS IN THE DESIRED CYLINDER REGISTER, RMDC, WRITES ONES AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, A SHIFTING 1 BIT PATTERN IS TRANSFERRED TO AND FROM RMDC AND THE PROGRAM CHECKS FOR BIT INTERFERENCE.

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

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

ILLEGAL REGISTER TEST

PURPOSE:

TO TEST ILLEGAL REGISTER ERROR DETECTION IN THE RM05/3/2.

PROCEDURE:

THIS TEST READS ALL LEGAL REGISTERS AND VERIFIES THAT 'ILR', BIT 2 OF RMER1 DOES NOT SET. THEN, TO THE EXTENT ALLOWED BY THE MASSBUS CONTROLLER, IT READS ILLEGAL REGISTERS AND VERIFIES THAT 'ILR' IS SET.

PROBABLE FAULT:

1. IF MODULE
2. ASSYNCHRONOUS MASSBUS MODULE

RESET GO BY INIT TEST

PURPOSE:

TO VERIFY THAT GO CAN BE RESET BY INITIALIZE.

PROCEDURE:

THE TEST SETS GO THEN CLEARS GO USING MASSBUS INITIALIZE, I.E., CONTROLLER CLEAR.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

DIAGNOSTIC MODE TEST

PURPOSE:

TO VERIFY THAT 'DIAGNOSTIC MODE', BIT 0 OF RMMR1, IS NOT STUCK AT ONE OR ZERO.

PROCEDURE:

THE RM05/3/2 IS INITIALIZED AND 'DMD' IS CHECKED FOR ZERO. 'DMD' IS WRITTEN WITH ONE AND READ TO VERIFY THAT IT IS NOT STUCK AT ZERO, THEN WRITTEN WITH ZERO AND READ TO VERIFY THAT IT IS NOT

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

STUCK AT ONE.
PROBABLE FAULT:
1. CS MODULE
2. IF MODULE

MOL TEST

PURPOSE:

TO VERIFY THAT 'MEDIUM ON LINE' STATUS CAN BE SET AND RESET USING MAINTENANCE UNIT READY.

PROCEDURE:

AFTER INITIALIZING THE SUBSYSTEM, THE TEST SETS 'DIAGNOSTIC MODE' AND READS THE DRIVE STATUS REGISTER, RMDS, EXPECTING MOL, BIT 12 TO BE ZERO. 'MAINTENANCE UNIT READY', BIT 9 OF RMMR1, IS SET AND MOL SHOULD BE ONE. THE TEST THEN WRITES A ZERO IN MUR AND READS RMDS, VERIFYING THAT 'MEDIUM ON LINE' IS ZERO.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

WRITE LOCK TEST

PURPOSE:

TO VERIFY THAT 'WRITE LOCK' STATUS, WRL, CAN BE SET AND RESET USING 'MAINTENANCE WRITE PROTECT', MWP.

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE PROGRAM SETS MWP, BIT 03 OF RMMR1, AND READS RMDS TO VERIFY THAT WRL, BIT 11 IS SET. THEN MWP IS RESET AND WRL SHOULD BE ZERO.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

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

DRIVE FAULT TEST

PURPOSE:

TO VERIFY THAT 'DEVICE CHECK', DVC, AND 'UNSAFE', UNS, CAN BE SET AND RESET USING 'MAINTENANCE DRIVE FAULT', MDF.

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE PROGRAM SETS MDF, BIT 06 OF RMMR1, AND READS RMER3 TO VERIFY THAT DVC, BIT 07 IS SET RMER1 IS ALSO READ AND UNS, BIT 14 SHOULD ALSO BE SET. THEN MDF IS RESET AND DVC AND UNS SHOULD BE RESET.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

SEEK ERROR TEST

PURPOSE:

TO VERIFY THAT 'SEEK ERROR', SKI, CAN BE SET AND RESET USING 'MAINTENANCE SEEK ERROR', MSER.

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE TEST SETS MSER, BIT 07 OF RMMR1 AND READS RMER3 TO VERIFY THAT SKI, BIT 14 IS SET. MSER IS RESET AND SKI SHOULD RESET.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

PIP TEST

PURPOSE:

TO VERIFY THAT 'POSITIONING IN PROGRESS', PIP, CAN BE SET AND RESET USING 'MAINTENANCE ON CYLINDER', MOC.

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

PROCEDURE:

DIAGNOSTIC MODE IS SET THEN MOC, BIT 08 OF RMMR1 IS SET AND PIP, BIT 13 OF RMDS, SHOULD BE ZERO. MOC IS THEN RESET AND PIP SHOULD BE ONE.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

EBL TEST

PURPOSE:

TO VERIFY THAT END OF BLOCK STATUS 'EBL' CAN BE SET AND RESET USING DIAGNOSTIC END OF BLOCK 'DEBL'.

PROCEDURE:

THE PROGRAM SETS DIAGNOSTIC MODE AND VERIFIES THAT EBL IS RESET. THEN IT SETS DEBL AND VERIFIES THAT EBL IS SET. FINALLY, THE TEST TRANSFERS A SHIFTING ONE BIT TO RMMR1, AND CHECKS FOR DEBL BEING SET BY AN ADJACENT BIT.

PROBABLE FAULT:

1. CS MODULE

LAST SECTOR, LAST TRACK TEST

PURPOSE:

TO VERIFY THE DESIRED TRACK/SECTOR PLA ON THE DS MODULE USING RMMR1, BITS 01 AND 02.

PROCEDURE:

THE TEST WRITES ALL POSSIBLE PATTERNS IN THE DISK ADDRESS REGISTER, RMDA, AND VERIFIES 'LS' AND 'LST' STATUS FOR EACH PATTERN. THE PROCEDURE IS DONE ONCE FOR 16 BIT FORMAT AND ONCE FOR 18 BIT FORMAT.

PROBABLE FAULT:

1. DS MODULE
2. CS MODULE

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

RMDC COUNT TEST

PURPOSE:

TO VERIFY THAT THE DISK ADDRESS REGISTER (RMDC) INCREMENTS PROPERLY.

PROCEDURE:

THE TEST INCREMENTS RMDC USING DIAGNOSTIC END OF BLOCK 'DEBL' AND VERIFIES THE RESULT IN BOTH 16 AND 18 BIT FORMAT.

PROBABLE FAULT:

1. DS MODULE

RMDC COUNT TEST

PURPOSE:

TO VERIFY THAT THE DESIRED CYLINDER REGISTER, RMDC, INCREMENTS PROPERLY.

PROCEDURE:

THE PROGRAM INCREMENTS RMDC USING DIAGNOSTIC END OF BLOCK, 'DEBL', VERIFYING THAT RMDC INCREMENTS THROUGH A COMPLETE CYCLE.

PROBABLE FAULT:

1. DS MODULE

LBT TEST

PURPOSE:

TO INSURE THAT LAST BLOCK TRANSFERRED, 'LBT', CLEARS WHEN RMDC IS WRITTEN, AND SETS WHEN THE LAST SECTOR IS TRANSFERRED.

PROCEDURE:

THE TEST USES DIAGNOSTIC EBL TO SET LBT, AND TRANSFERS TO RMDC TO RESET LBT.

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

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

COMPOSITE ERROR TEST

PURPOSE:

TO TEST 'COMPOSITE ERROR', BIT 14 OF RMDS.

PROCEDURE:

THE TEST USES INITIALIZE AND DIAGNOSTIC MODE TO FORCE ALL ERRORS TO ZERO THEN VERIFIES THAT 'ERR' IS ZERO. EACH ERROR IS INDIVIDUALLY SET AND 'ERR' SHOULD BE ONE FOR EVERY ERROR TESTED. ADDRESSES #2 AND #17 OF THE COMPOSITE ERROR PLA ARE NOT TESTED. 'ABORT' AND 'EXCEPTION' OUTPUTS OF THE PLA ARE NOT TESTED. THE TEST FAILS IF ERR IS NOT ZERO WITH ALL SET ARGUMENTS ZERO OR IF ERR IS NOT ONE WITH ANY SET ARGUMENT ONE.

PROBABLE FAULT:

1. IF MODULE

WRITE GO TEST

PURPOSE:

TO VERIFY THAT GO CAN BE SET.

PROCEDURE:

THE TEST ENABLES THE DEBUG CLOCK, THEN TRANSFERS A NOP FUNCTION CODE AND GO BIT TO RMCS1, VERIFYING THAT GO SETS. ALL FUNCTION CODES ARE TESTED.

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

BRANCH MULTIPLEXOR TEST

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

PURPOSE:

TO VERIFY THAT THE OUTPUT OF THE COMMAND SEQUENCER BRANCH MULTIPLEXOR DOES NOT HAVE A FAULT.

PROCEDURE:

WITH DEBUG CLOCK ENABLED, THE TEST USES VARIOUS FUNCTION CODES AND REGISTER CONDITIONS TO ADDRESS THE TEST BIT MULTIPLEXOR SUCH THAT THE TEST BIT, BIT12 OF RMMR2, CAN BE CHECKED FOR A STUCK FAULT.

PROBABLE FAULT:

1. CS MODULE

SET/RESET GO TEST

PURPOSE:

TO VERIFY THAT GO CAN BE SET AND RESET.

PROCEDURE:

THE SUBSYSTEM IS INITIALIZED AND PUT IN DIAGNOSTIC MODE WITH 'DEBUG CLOCK ENABLE', BIT 14 OF RMMR1 SET. CERTAIN FUNCTION CODES ARE WRITTEN IN RMCS1 AND THE PROGRAM READS RMCS1 TO VERIFY THAT GO IS SET. RMDS IS ALSO READ TO VERIFY THAT 'DRY' IS RESET. THEN THE PROGRAM STEPS THE DEBUG CLOCK USING BIT 15 OF RMMR1 AND VERIFIES THAT 'GO' RESETS AND 'DRY' SETS. USING A FUNCTION CODE THAT RESETS GO AT A DIFFERENT PROM ADDRESS. THE TEST FAILS IF GO DOES NOT SET OR CANNOT BE RESET BY THE COMMAND SEQUENCER. THE TEST ALSO FAILS IF 'DRIVE READY' IS NOT THE COMPLIMENT OF GO.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

END 1 RESET GO TEST

PURPOSE:

TO VERIFY THAT THE COMMAND SEQUENCER CAN RESET GO AT THE END1 LOCATION.

PROCEDURE:

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

THE TEST EXECUTES RELEASE, SEARCH AND ILLEGAL FUNCTION CODE
32 IN DIAGNOSTIC MODE AND VERIFIES THAT GO RESETS ON THE
SPECIFIED CLOCK CYCLE.

PROBABLE FAULT:

1. CS MODULE

SET PULSE TEST

PURPOSE:

TO VERIFY THAT THE COMMAND SEQUENCER CAN GENERATE SET PULSE.

PROCEDURE:

WHICH DEBUG CLOCK ENABLED, THE TEST STEPS THE COMMAND
SEQUENCER THROUGH PARTS OF VARIOUS FUNCTION CODES AND CHECKS
CONTINUE, BIT 06 OF RMMR1 TO DETERMINE IF SET PULSE IS BEING
GENERATED.

PROBABLE FAULT:

1. CS MODULE

SET/RESET IVC TEST

PURPOSE:

TO TEST "INVALID COMMAND" STATUS FOR EACH FUNCTION CODE.

PROCEDURE:

THE PROGRAM RESETS VOLUME VALID USING 'MAINTENANCE UNIT
READY', BIT09 OF RMMR1, THEN LOADS THE FUNCTION CODE AND GO IN
RMCS1. EACH FUNCTION CODE IS TESTED AND 'IVC', BIT 12 OF RMER2
IS CHECKED.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

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

SET LSC TEST

PURPOSE:

TO VERIFY THAT 'LOSS OF SYSTEM CLOCK' CAN SET AND RESET.

PROCEDURE:

THE TEST ENABLES THE DEBUG CLOCK AND SETS THE GO BIT. AFTER WAITING ENOUGH TIME FOR THE ONE SHOT TO SET, THE TEST DISABLES THE DEBUG CLOCK AND VERIFIES THAT LSC SETS.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

DECODE TEST

PURPOSE:

TO VERIFY THAT THE 'DECODE' FLOP ON THE IF MODULE SETS WITH THE LEADING EDGE OF 'SET PULSE' EXCEPT WHEN 'COMPOSITE ERROR' IS ACTIVE.

PROCEDURE:

THE TEST USES 'VOLUME VALID' AND 'OCCUPIED' TO DETERMINE IF THE DECODE FLOP IS SET OR RESET. INITIALLY, VV AND OCCUPIED ARE RESET AND THE TEST EXECUTES THOSE COMMANDS WHICH SET VV OR OCC AND VERIFIES THAT ONE OR BOTH BITS SET. THE SAME COMMANDS ARE EXECUTED AGAIN WITH COMPOSITE ERROR SET, AND THE TEST VERIFIES THAT NEITHER BIT SETS.

PROBABLE FAULT:

1. IF MODULE

SET/RESET VOLUME VALID TEST

PURPOSE:

TO VERIFY THAT 'VOLUME VALID' RESETS WITH THE LEADING EDGE OF UNIT READY, AND SETS WITH PACK ACKNOWLEDGE AND READ IN PRESET COMMANDS.

PROCEDURE:

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
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368

USING 'MAINTENANCE UNIT READY', BIT 9 OF RMMR1, THIS TEST FORCES A ZERO TO ONE TRANSITION OF UNIT READY AND VERIFIES THAT VOLUME VALID, BIT 6 OF RMDS IS ZERO. THEN THE TEST EXECUTES A PACK ACKNOWLEDGE COMMAND, VERIFYING THAT VV SETS. THE PROCEDURE IS REPEATED WITH A READ IN PRESET COMMAND.

PROBABLE FAULT:

1. IF MODULE

ILLEGAL FUNCTION TEST

PURPOSE:

TO TEST ILLEGAL FUNCTION ERROR IN THE RM05/3/2.

PROCEDURE:

WITH DIAGNOSTIC CLOCK ENABLED TO INHIBIT THE COMMAND SEQUENCER, THIS TEST VERIFIES THAT 'ILF', BIT 0 OF RMER1, IS OFF FOR LEGAL FUNCTION CODES AND ON FOR ILLEGAL FUNCTION CODES. THE STATUS OF THE 'GO' BIT IS IGNORED.

PROBABLE FAULT:

1. IF MODULE

OCCUPIED TEST

PURPOSE:

TO VERIFY THAT 'OCCUPIED' IS SET DURING DATA TRANSFERS AND IS RESET FOR ALL OTHER COMMANDS.

PROCEDURE:

FOR EACH DATA TRANSFER COMMAND, 'OCC', BIT 15 OF RMMR1 SHOULD BE ONE, DEBUG CLOCK IS ENABLED TO PREVENT GO FROM RESETTING BEFORE STATUS IS SAMPLED.

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425

READ IN PRESET TEST

PURPOSE:

TO VERIFY THAT 'READ IN PRESET' COMMAND IS DECODED, AND IN PARTICULAR, TO VERIFY THAT 'IF5 READ IN CMD L' IS NOT STUCK AT ONE.

PROCEDURE:

EACH VISIBLE STATUS OR REGISTER BIT WHICH IS CLEARED BY 'READ IN PRESET' IS SET. THEN THE RIP COMMAND IS EXECUTED AND THE TEST VERIFIES THAT ONE OR MORE BITS ARE CLEARED. THE FOLLOWING ARE USED DURING THE TEST.

. ALL BITS OF RMOF ARE SET BY A MOVE INSTRUCTION AND THE TEST PASSES IF USED BITS ARE ZERO AFTER THE RIP COMMAND.

. THE DESIRED CYLINDER REGISTER, RMDC, IS SET WITH A MOVE INSTRUCTION AND THE TEST PASSES IF BITS 00-09 ARE ZERO AFTER THE RIP COMMAND.

. THE DISK ADDRESS REGISTER, RMDA, IS SET WITH A MOVE INSTRUCTION AND THE TEST PASSES IF BITS 00-07, AND BITS 08-15 ARE ZERO AFTER THE RIP COMMAND.

THE TEST FAILS IF NONE OF THE PRESET TERMS ARE ZERO AFTER THE RIP COMMAND.

PROBABLE FAULT:

1. IF MODULE ==

RIP/RMOF TEST

PURPOSE:

TO VERIFY THAT 'READ IN PRESET' RESETS FMT16, ECI AND HCI, BITS 10,11 AND 12 OF RMOF.

PROCEDURE:

FMT16, ECI AND HCI ARE SET, THEN A RIP COMMAND IS EXECUTED AND EACH BIT SHOULD BE ZERO.

PROBABLE FAULT:

1. IF MODULE

1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482

RMDA/RMDC/RIP TEST

PURPOSE:

TO VERIFY THAT 'READ IN PRESET' RESETS THE DESIRED CYLINDER ADDRESS, RMDC, AND THE DISK ADDRESS, RMDA.

PROCEDURE:

RMDA AND RMDC ARE PRESET THEN TESTED FOR ZERO AFTER THE RIP COMMAND.

PROBABLE FAULT:

1. DS MODULE

OFFSET COMMAND TEST

PURPOSE:

TO VERIFY THAT 'OFFSET MODE' SETS WITH OFFSET COMMAND.

PROCEDURE:

THE TEST EXECUTES OFFSET COMMAND AND VERIFIES THAT 'OM', BIT 00 OF RMD5 IS ONE.

PROBABLE FAULT:

1. IF MODULE

RETURN TO CENTER TEST

PURPOSE:

TO VERIFY THAT 'RETURN TO CENTER' RESETS OFFSET MODE.

PROCEDURE:

OFFSET MODE, BIT 00 OF RMD5, IS SET WITH OFFSET COMMAND, THEN THE TEST EXECUTES A RETURN TO CENTER COMMAND AND VERIFIES THAT OFFSET MODE RESETS. OFFSET DIRECTION IS ALSO SET AND CHECKED FOR ZERO AFTER THE COMMAND.

PROBABLE FAULT:

1. IF MODULE

1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539

RMDC CLEAR OFFSET TEST

PURPOSE:

TO VERIFY THAT CLEAR OFFSET IS ACTIVE WHEN THE DESIRED CYLINDER ADDRESS IS WRITTEN.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND, WRITES RMDC, AND VERIFIES THAT OM, BIT 00 OF RMDS IS ZERO.

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

EBL CLEAR OFFSET TEST

PURPOSE:

TO VERIFY THAT OFFSET MODE CLEARS WHEN HEAD SWITCHING OCCURS.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND TO SET OFFSET MODE. AFTER SETTING THE FORMAT BIT AND LOADING THE LAST SECTOR/TRACK ADDRESS IN RMDA, THE TEST FORCES AN EBL AND VERIFIES THAT OFFSET MODE RESETS.

PROBABLE FAULT:

1. DS MODULE

RUN AND GO TEST

PURPOSE:

TO VERIFY THAT 'RUN AND GO' FLOP SETS DURING READ AND WRITE COMMANDS.

PROCEDURE:

1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596

THE RMO5/3/2 IS INITIALIZED AND A DATA TRANSFER COMMAND WITH GO SET IS WRITTEN IN RMCS1. 'RUN AND GO', BIT 14 OF RMMR1 SHOULD BE ONE FOR EACH DATA COMMAND. THE DEBUG CLOCK IS ENABLED SO THAT GO DOES NOT RESET BEFORE STATUS IS TESTED.

PROBABLE FAULT:

1. CS MODULE
2. SYNCHRONOUS MASSBUS MODULE

SET IAE TEST

PURPOSE:

TO VERIFY THAT INVALID ADDRESS ERROR CAN SET.

PROCEDURE:

THE TEST LOADS INVALID SECTOR, TRACK AND CYLINDER ADDRESSES AND EXECUTES A SEARCH COMMAND, VERIFYING THAT 'IAE' SETS. THE PROCESS IS REPEATED WITH A DIFFERENT COMMAND IF THE IAE DOES NOT SET, AND THE TEST FAILS IF IAE CANNOT BE SET.

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

SEARCH, SEEK, READ, WRITE TEST

PURPOSE:

TO VERIFY THAT THE 'SCH SK R OR W' DECODE ON THE IF MODULE IS CORRECT FOR ALL FUNCTION CODES.

PROCEDURE:

THE TEST LOADS INVALID SECTOR, TRACK AND CYLINDER ADDRESSES AND EXECUTES EACH COMMAND TO WHERE SET PULSE IS ACTIVE AND VERIFIES THE DECODE BY CHECKING 'IAE'.

PROBABLE FAULT:

1. IF MODULE

1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653

INVALID TRACK/SECTOR TEST

PURPOSE:

TO VERIFY THAT INVALID TRACK AND SECTOR ADDRESSES ARE DETECTED.

PROCEDURE:

THE TEST LOADS THE TEST PATTERN IN RMDA AND EXECUTES A SEARCH COMMAND, VERIFYING THAT "IAE" SETS.

PROBABLE FAULT:

1. DS MODULE
2. TRACK ADDRESS OPTION JUMPER

INVALID CYLINDER TEST

PURPOSE:

TO VERIFY THAT INVALID CYLINDER ADDRESSES ARE DETECTED.

PROCEDURE:

THE TEST LOADS THE TEST PATTERN IN RMDC AND EXECUTES A SEARCH COMMAND, VERIFYING THAT "IAE" SETS.

PROBABLE FAULTS:

1. DS MODULE
2. CYLINDER ADDRESS OPTION JUMPER

SET AOE TEST

PURPOSE:

TO VERIFY THAT ADDRESS OVERFLOW ERROR IS DETECTED.

PROCEDURE:

THE TEST LOADS THE ADDRESS OF THE LAST SECTOR IN RMDA AND RMDC, THEN INITIATES A DATA COMMAND WITH DEBUG CLOCK ENABLED. END OF BLOCK IS FORCED TO INCREMENT THE SECTOR ADDRESS, AND THE TEST VERIFIES THAT "AOE" IS SET.

1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710

PROBABLE FAULT:

- 1. DS MODULE

SET RMR TEST

PURPOSE:

TO VERIFY THAT 'REGISTER MODIFICATION REFUSED' SETS WHEN A REGISTER IS WRITTEN WHILE GO IS SET, EXCEPT WHEN THE ATTENTION OR MAINTENANCE REGISTER IS WRITTEN.

PROCEDURE:

'DEBUG CLOCK ENABLE' IS SET TO INHIBIT THE COMMAND SEQUENCER, THEN A NOP COMMAND AND GO BIT IS WRITTEN IN RMCS1. WITHOUT STEPPING THE DEBUG CLOCK, THE TEST WRITES RMMR AND RMAS, WHICH SHOULD NOT SET RMR STATUS. THEN RMDA IS WRITTEN AND RMR STATUS, BIT 02 OF RMR1, SHOULD BE ONE.

PROBABLE FAULT:

- 1. IF MODULE

PGM STATUS CHECK

PURPOSE:

TO VERIFY THAT THE PROGRAMMABLE STATUS BIT AND THE DRIVE REQUEST STATUS BIT ARE COMPATABLE.

PROCEDURE:

THE TEST REPORTS AN ERROR IF PGM IS ON AND DRQ IS OFF. PGM IS NOT PREDICTABLE IN THE CASE WHERE DRQ IS ON BECAUSE OF THE PORT SELECT SWITCH.

PROBABLE FAULT:

- 1. IF MODULE

DVA/DPR STATUS CHECK

PURPOSE:

1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767

TO VERIFY THAT DEVICE AVAILABLE STATUS AND DRIVE PRESENT STATUS ARE SET.

PROCEDURE:

DVA AND DPR ARE TESTED AND BOTH SHOULD BE ON.

PROBABLE FAULT:

- 1. IF MODULE

PORT REQUEST TEST, PART 1

PURPOSE:

TO VERIFY THAT THE PORT REQUEST FLOPS ON THE IF MODULE SET WHEN THE PROGRAM READS RMCS1.

PROCEDURE:

THE TEST EXECUTES A RELEASE COMMAND, THEN, ASSUMING THE PORT IS RELEASED, IT READS RMCS1, THEN READS RMMR2 AND VERIFIES THAT ONE OF THE PORT REQUEST FLOPS IS SET.

PROBABLE FAULT:

- 1. IF MODULE
- 2. CS MODULE

PORT REQUEST TEST, PART 2

PURPOSE:

TO VERIFY THAT THE PORT REQUEST FLOPS ON THE IF MODULE SET WHEN THE PROGRAM WRITES RMAS.

PROCEDURE:

THE TEST EXECUTES A RELEASE COMMAND THEN WRITES RMAS AND READS RMMR2, VERIFYING THAT ONE OF THE REQUEST FLOPS IS SET.

PROBABLE FAULT:

- 1. IF MODULE
- 2. CS MODULE

CZ
RH

1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824

PORT REQUEST TEST, PART 3

PURPOSE:

TO VERIFY THAT PORT REQUEST SETS WHEN ANY REGISTER EXCEPT RMA5 IS WRITTEN.

PROCEDURE:

THE TEST WRITES THE DISK ADDRESS REGISTER AND VERIFIES THAT THE PORT REQUEST FLOP IS ON.

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

RELEASE TEST

PURPOSE:

TO VERIFY THAT A RELEASE COMMAND CAN RESET THE REQUEST FLOPS RQA AND RQB IN MAINTANCE REGISTER #2.

PROCEDURE:

THE PROGRAM SETS REQUEST FLOP BY WRITTING THE RMCS1 REGISTER THEN, EXECUTES A RELEASE COMMAND TO RESET THE REQUEST FLOP.

PROBABLE FAULT:

1. IF MODULE

WRITE ATA TEST

PURPOSE:

TO VERIFY THAT ATTENTION CAN BE CLEARED BY WRITING THE ATTENTION SUMMARY REGISTER.

PROCEDURE:

THE PROGRAM RESETS AND SETS UNIT READY WHICH SHOULD CAUSE AN ATTENTION, THEN WRITES THE ATTENTION SUMMARY REGISTER AND

1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881

VERIFIES THAT ATTENTION IS RESET.

PROBABLE FAULT:

- 1. IF MODULE
- 2. CS MODULE

RESET ATA BY GO TEST

PURPOSE:

TO VERIFY THAT ATA RESETS WHEN GO IS ON AND COMPOSITE ERROR IS OFF.

PROCEDURE:

THE PROGRAM SETS MAINTENANCE UNIT READY WHICH SHOULD CAUSE AN ATTENTION. THEN, WITH DEBUG CLOCK ENABLED, GO IS SET, AND ATA SHOULD BE ZERO.

PROBABLE FAULT:

- 1. IF MODULE

UNIT READY ATA TEST

PURPOSE:

TO VERIFY THAT ONE-ZERO AND ZERO-ONE TRANSITIONS OF UNIT READY SET ATTENTION.

PROCEDURE:

THE TEST USES DIAGNOSTIC MODE TO FORCE BOTH TRANSITIONS OF UNIT READY AND VERIFIES THAT ATA SETS WITH EACH TRANSITION.

PROBABLE FAULT:

- 1. IF MODULE

ERROR ATA TEST

PURPOSE:

1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938

TO VERIFY THAT ATTENTION SETS WHEN COMPOSITE ERROR OCCURS WHILE GO IS OFF.

PROCEDURE:

THE PROGRAM CLEARS THE DEVICE AND SETS AN ERROR, THEN VERIFIES ATA IS ON.

PROBABLE FAULT:

1. IF MODULE

REGISTER TRANSFER ATA TEST

PURPOSE:

TO VERIFY THAT ATTENTION SETS WHEN ANY REGISTER, EXCEPT FOR RMAS AND RMCS, IS WRITTEN WHILE COMP ERROR IS SET.

PROCEDURE:

THE PROGRAM FORCES AN ERROR THEN RESETS ATTENTION FROM THE ERROR. THE PROGRAM THEN WRITES RMAS AND RMCS AND VERIFIES THAT NO ATTENTION OCCURS, AND WRITES RMDC AND VERIFIES THAT ATTENTION DOES OCCUR.

PROBABLE FAULT:

1. IF MODULE

P SET ATA TEST

PURPOSE:

TO VERIFY THAT ATA IS SET AT THE COMPLETETION OF AN OFFSET AND RETURN TO CENTER LINE COMMAND.

PROCEDURE:

THE PROGRAM EXECUTES THE COMMANDS USING THE MAINTANCE DEBUG CLOCK AND EXPECTS ATA TO BE SET ON COMPLETETION.

PROBABLE FAULT:

1. IF MODULE

1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995

SET WLE TEST

PURPOSE:

TO VERIFY THAT 'WLE' IS SET OR RESET WHEN IT SHOULD BE.

PROCEDURE:

THE PROGRAM EXECUTES THE FOLLOWING COMMANDS USING THE MAINTANCE DEBUG CLOCK AND EXPECTS WLE SET OR RESET.

EXECUTE WRITE DATA COMMAND WITH MAINTANCE WRITE PROTECT SET, SHOULD EXPECT WLE TO BE SET.

EXECUTE WRITE DATA COMMAND WITHOUT MAINTANCE WRITE PROTECT SET, SHOULD EXPECT WLE TO BE RESET.

EXECUTE READ DATA COMMAND WITH MAINTANCE WRITE PROTECT SET, SHOULD EXPECT WLE TO BE RESET.

EXECUTE READ IN PRESET COMMAND WITH MAINTANCE WRITE PROTECT SET, SHOULD EXPECT WLE TO BE RESET.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

DATA COMMAND TESTS (1, 2, 3)

PURPOSE:

TO VERIFY THE COMMAND SEQUENCER DURING DATA COMMANDS.

PROCEDURE:

THIS TEST, LIKE RECALIBRATE, SEEK, AND SEARCH TESTS, USES THE MAINTENANCE REGISTER TO SIMULATE DRIVE CONDITIONS AND FORCE THE COMMAND SEQUENCER THROUGH EACH BRANCH PATH. ADDITIONAL ITEMS WHICH ARE TESTED INCLUDE OFFSET PLUS AND MINUS ON THE TAG BUS AND 'ENABLE SEARCH', BIT 11 OF RMMR1.

PROBABLE FAULT:

1. CS MODULE

1
683
684

```

;PROGRAM VERSION #001

.TITLE CZRMPAO RM05/3/2 DSKLS TST 1
;*COPYRIGHT (C) 1980
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY MIKE LEAVITT
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C4), 1980.

```

685

```

.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;*      SWITCH      USE
;*      -----      -----
;*      15          HALT ON ERROR
;*      14          LOOP ON TEST
;*      13          INHIBIT ERROR TYPEOUTS
;*      11          INHIBIT ITERATIONS
;*      10          BELL ON ERROR
;*      9           LOOP ON ERROR
;*      8           LOOP ON TEST IN SWR<7:0>
686  *      7          TN128
;*      6          TN64
;*      5          TN32
;*      4          TN16
;*      3          TN8
;*      2          TN4
;*      1          TN2
687  *      0          TN1
688

```

687
688

```

.SBTTL BASIC DEFINITIONS

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

```

001100
104000
000004

```

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

```

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

000000
000001
000002
000003
000004

```

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0 = %0            ;;GENERAL REGISTER
R1 = %1            ;;GENERAL REGISTER
R2 = %2            ;;GENERAL REGISTER
R3 = %3            ;;GENERAL REGISTER
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
000030 SEARCH = 000030  ;SEARCH COMMAND
    
```

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

```

720      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
721      000050      WCD        = 000050      ;WRITE CHECK DATA COMMAND
722      000052      WCH        = 000052      ;WRITE CHECK HEADER AND DATA
723      000054      ILF54      = 000054      ;ILLEGAL COMMAND
724      000056      ILF56      = 000056      ;ILLEGAL COMMAND
725      000060      WD         = 000060      ;WRITE DATA COMMAND
726      000062      WH         = 000062      ;WRITE HEADER AND DATA COMMAND
727      000064      ILF64      = 000064      ;ILLEGAL COMMAND
728      000066      ILF66      = 000066      ;ILLEGAL COMMAND
729      000070      RD         = 000070      ;READ DATA COMMAND
730      000072      RH         = 000072      ;READ HEADER AND DATA COMMAND
731      000074      ILF74      = 000074      ;ILLEGAL COMMAND
732      000076      ILF76      = 000076      ;ILLEGAL COMMAND
733
734      ;*RMDA DISK ADDRESS REGISTER
735
736      ;TRACK ADDRESS DEFINITIONS
737      010000      TA16       = BIT12       ;TRACK ADDRESS 16.
738      004000      TA8        = BIT11       ;TRACK ADDRESS 8.
739      002000      TA4        = BIT10       ;TRACK ADDRESS 4
740      001000      TA2        = BIT09       ;TRACK ADDRESS 2
741      000400      TA1        = BIT08       ;TRACK ADDRESS 1
742
743      ;SECTOR ADDRESS DEFINITIONS
744      000020      SA16       = BIT04       ;SECTOR ADDRESS 16.
745      000010      SA8        = BIT03       ;SECTOR ADDRESS 8.
746      000004      SA4        = BIT02       ;SECTOR ADDRESS 4
747      000002      SA2        = BIT01       ;SECTOR ADDRESS 2
748      000001      SA1        = BIT00       ;SECTOR ADDRESS 1
749
750      ;TRACK & SECTOR MASKS
751      177400      TADMSK     = 177400      ;TRACK ADDRESS MASK
752      000377      SADMSK     = 000377      ;SECTOR ADDRESS MASK
753
754      ;*RMDS DRIVE STATUS REGISTER
755
756      100000      ATA        = BIT15       ;ATTENTION ACTIVE
757      040000      ERR        = BIT14       ;COMPOSITE ERROR
758      020000      PIP        = BIT13       ;POSITIONING IN PROGRESS
759      010000      MOL        = BIT12       ;MEDIUM ON LINE
760      004000      WRL        = BIT11       ;WRITE LOCK
761      002000      LBT        = BIT10       ;LAST BLOCK TRANSFERRED
762      001000      PGM        = BIT09       ;PROGRAMMABLE
763      000400      DPR        = BIT08       ;DRIVE PRESENT
764      000200      DRY        = BIT07       ;DRIVE READY
765      000100      VV         = BIT06       ;VOLUME VALID
766      000001      OM         = BIT00       ;OFFSET MODE ACTIVE
767
768      ;*RMER1 ERROR REGISTER #1
769

```



```

770      100000      DCK      = BIT15      ;DATA CHECK ERROR
771      040000      UNS      = BIT14      ;DRIVE UNSAFE
772      020000      OPI      = BIT13      ;OPERATION INCOMPLETE
773      010000      DTE      = BIT12      ;DRIVE TIMING ERROR
774      004000      WLE      = BIT11      ;WRITE LOCK ERROR
775      002000      IAE      = BIT10      ;INVALID ADDRESS ERROR
776      001000      AOE      = BIT09      ;ADDRESS OVERFLOW ERROR
777      000400      HCRC     = BIT08      ;HEADER CRC ERROR
778      000200      HCE      = BIT07      ;HEADER COMPARE ERROR
779      000100      ECH      = BIT06      ;ECC 'HARD' ERROR
780      000040      WCF      = BIT05      ;WRITE CLOCK FAILURE
781      000020      FER      = BIT04      ;FORMAT ERROR
782      000010      PAR      = BIT03      ;PARITY ERROR
783      000004      RMR      = BIT02      ;REGISTER MODIFICATION REFUSED
784      000002      ILR      = BIT01      ;ILLEGAL REGISTER
785      000001      ILF      = BIT00      ;ILLEGAL FUNCTION
786
787      115760      NDTMSK   = DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER
788      ;'NDTMSK' IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
789      ;COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
790
791      ;*RMAS ATTENTION SUMMARY REGISTER
792
793      000377      ATNMSK   = 377          ;MASK FOR ATTENTION BITS
794
795      ;*RMLA LOOK AHEAD REGISTER
796
797      002000      SC4       = BIT10      ;SECTOR COUNT = 16
798      001000      SC3       = BIT09      ;SECTOR COUNT = 8
799      000400      SC2       = BIT08      ;SECTOR COUNT = 4
800      000200      SC1       = BIT07      ;SECTOR COUNT = 2
801      000100      SC0       = BIT06      ;SECTOR COUNT = 1
802
803      003700      SCTMSK   = 003700     ;SECTOR COUNT MASK
804
805      ;*RMMR1 MAINTENANCE REGISTER #1
806
807      ;WRITE ONLY BITS
808      100000      DBCK     = BIT15      ;DEBUG CLOCK
809      040000      DBEN     = BIT14      ;DEBUG CLOCK ENABLE
810      020000      DEBL     = BIT13      ;DIAGNOSTIC END OF BLOCK
811      010000      DTO      = BIT12      ;DIAGNOSTIC TIMEOUT
812      004000      MCLK     = BIT11      ;MAINTENANCE CLOCK
813      002000      MRD      = BIT10      ;READ DATA
814      001000      MUR      = BIT09      ;UNIT READY
815      000400      MOC      = BIT08      ;ON CYLINDER
816      000200      MSER     = BIT07      ;SEEK ERROR
817      000100      MDF      = BIT06      ;DRIVE FAULT
818      000040      MS       = BIT05      ;SECTOR PULSE
819      000010      MWP      = BIT03      ;WRITE PROTECT
820      000004      MI       = BIT02      ;INDEX PULSE
821      000002      MSC      = BIT01      ;SECTOR COMPARE
822      000001      DMD      = BIT00      ;DIAGNOSTIC MODE
823
824      ;READ ONLY BITS
825      100000      OCC      = BIT15      ;OCCUPIED
826      040000      RG       = BIT14      ;RUN AND GO

```

827	020000	EBL	= BIT13	:END OF BLOCK
828	010000	REX	= BIT12	:EXCEPTION
829	004000	ESRC	= BIT11	:ENABLE SEARCH
830	002000	PLFS	= BIT10	:LOOKING FOR SYNC
831	001000	ECRC	= BIT09	:ENABLE CRC OUT
832	000400	PDA	= BIT08	:DATA AREA
833	000200	PHA	= BIT07	:HEADER AREA
834	000100	CONT	= BIT06	:CONTINUE
835	000040	WC	= BIT05	:WORD CLOCK
836	000020	EECC	= BIT04	:ENABLE ECC OUT
837	000010	MWD	= BIT03	:WRITE DATA BIT
838	000004	LS	= BIT02	:LAST SECTOR
839	000002	LST	= BIT01	:LAST SECTOR AND TRACK
840	000001	DMD	= BIT00	:DIAGNOSTIC MODE
841	051401	MR1AAA	= DMD!MUR!DBEN!MOC!DTO	
842				
843		;*RMDT DRIVE TYPE REGISTER		
844				
845	100000	NSA	= BIT15	:NOT SECTOR ADDRESSED = 0
846	040000	TAP	= BIT14	:TAPE DRIVE = 0
847	020000	MOH	= BIT13	:MOVING HEAD = 1
848	004000	DRQ	= BIT11	:DRIVE REQUEST REQUIRED
849				
850	020024	SNGPRT	= 020024	:SINGLE PORT DRIVE TYPE
851	024024	DULPRT	= 024024	:DUAL PORT DRIVE TYPE
852				
853		;*RMOF OFFSET REGISTER		
854				
855	010000	FMT16	= BIT12	:16 BIT WORD FORMAT
856	004000	ECI	= BIT11	:ECC INHIBIT
857	002000	HCI	= BIT10	:HEADER COMPARE INHIBIT
858	000200	OFD	= BIT07	:OFFSET FORWARD
859	161577	XNUOF	= 161577	:UNUSED BITS OF RMOF
860				
861		;*RMDC DESIRED CYLINDER ADDRESS REGISTER		
862				
863	001777	CYLMSK	= 001777	:MASK FOR CYLINDER ADDRESS
864	176000	XNUDC	= 176000	:UNUSED BITS OF RMDC
865				
866		;*RMMR2 MAINTENANCE REGISTER #2		
867				
868		:READ ONLY BITS		
869	100000	RQA	= BIT15	:PORT A REQUEST
870	040000	RQB	= BIT14	:PORT B REQUEST
871	020000	TAG	= BIT13	:TAG CONTROL
872	010000	TST	= BIT12	:COMMAND SEQUENCE TEST BIT
873	004000	CC	= BIT11	:CONTROL OR CYLINDER TAG
874	002000	CH	= BIT10	:CONTROL OR HEAD TAG
877	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


```

878      000001      BB00      = BIT00      ;TAG BUS
879
880      ;*RMER2 ERROR REGISTER 2
881      100000      BSE       = BIT15      ;BAD SECTOR ERROR
882      040000      SKI       = BIT14      ;SEEK INCOMPLETE
883      020000      OPE       = BIT13      ;OPERATOR PLUG ERROR
884      010000      IVC       = BIT12      ;INVALID COMMAND ERROR
885      004000      LSC       = BIT11      ;LOSS OF SYSTEM CLOCK
886      002000      LBC       = BIT10      ;LOSS OF BIT CLOCK
887      000200      DVC       = BIT07      ;DEVICE CHECK
888      000010      DPE       = BIT03      ;DATA PARITY ERROR
889      001567      XNUER2    = 001567    ;UNUSED BITS OF RMER2
890
891      .SBTTL      PROGRAM MNEMONICS
892
893      100000      MSE       = BIT15      ;MANUFACTURING DETECTED SECTOR ERROR
894      040000      USE       = BIT14      ;USER DETECTED SECTOR ERROR
895
896      .SBTTL      RM REGISTER INDEX VALUES
897
898      000000      RMCS1     = 00        ;CONTROL STATUS REGISTER #1
899      000006      RMDA     = 06        ;DISK ADDRESS REGISTER
900      000012      RMDS     = 12        ;DRIVE STATUS REGISTER
901      000014      RMER1     = 14        ;ERROR REGISTER #1
902      000016      RMAS     = 16        ;ATTENTION SUMMARY REGISTER
903      000020      RMLA     = 20        ;LOOK AHEAD REGISTER
904      000024      RMMR1     = 24        ;MAINTENANCE REGISTER
905      000026      RMDT     = 26        ;DRIVE TYPE REGISTER
906      000030      RMSN     = 30        ;SERIAL NUMBER REGISTER
907      000032      RMOF     = 32        ;OFFSET REGISTER
908      000034      RMDC     = 34        ;DESIRED CYLINDER REGISTER
909      000036      RMHR     = 36        ;HOLDING REGISTER
910      000040      RMMR2     = 40        ;MAINTENANCE REGISTER #2
911      000042      RMER2     = 42        ;ERROR REGISTER #2
912      000044      RMEC1     = 44        ;ECC POSITION REGISTER
913      000046      RMEC2     = 46        ;ECC PATTERN REGISTER
916      000050      ILRG50    = 50        ;ILLEGAL REGISTER 50
          000052      ILRG52    = 52        ;ILLEGAL REGISTER 52
          000054      ILRG54    = 54        ;ILLEGAL REGISTER 54
          000056      ILRG56    = 56        ;ILLEGAL REGISTER 56
          000060      ILRG60    = 60        ;ILLEGAL REGISTER 60
          000062      ILRG62    = 62        ;ILLEGAL REGISTER 62
          000064      ILRG64    = 64        ;ILLEGAL REGISTER 64
          000066      ILRG66    = 66        ;ILLEGAL REGISTER 66
          000070      ILRG70    = 70        ;ILLEGAL REGISTER 70
          000072      ILRG72    = 72        ;ILLEGAL REGISTER 72
          000074      ILRG74    = 74        ;ILLEGAL REGISTER 74
          000076      ILRG76    = 76        ;ILLEGAL REGISTER 76
917
918
919      000077      IDXMSK    = 77        ;MASK FOR REGISTER INDEX NUMBER
920
921      .SBTTL      RH CONTROLLER REGISTER BIT DEFINITIONS
922
923      ;*RMCS1 CONTROL STATUS REGISTER #1
924

```

```

925      100000      SC      = BIT15      ;SPECIAL CONDITION-READ ONLY
926      040000      TRE      = BIT14      ;TRANSFER ERROR
927      020000      MCPE     = BIT13      ;MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
928      002000      PSEL     = BIT10      ;PORT B SELECT
929      001000      A17      = BIT09      ;ADDRESS EXTENSION
930      000400      A16      = BIT08      ;ADDRESS EXTENSION
931      000200      RDY      = BIT07      ;READY-READ ONLY
932      000100      IE       = BIT06      ;INTERRUPT ENABLE
933
934      ;*RMCS2 RH CONTROL STATUS REGISTER #2
935
936      100000      DLT       = BIT15      ;DATA LATE-READ ONLY
937      040000      WCE       = BIT14      ;WRITE CHECK ERROR-READ ONLY
938      020000      UPE       = BIT13      ;UNIBUS PARITY ERROR
939      010000      NED       = BIT12      ;NONEXISTANT DRIVE-READ ONLY
940      004000      NEM       = BIT11      ;NONEXISTANT MEMORY-READ ONLY
941      002000      PGE       = BIT10      ;PROGRAM ERROR-READ ONLY
942      001000      MXF       = BIT09      ;MISSED TRANSFER
943      000400      MDPE      = BIT08      ;MASSBUS DATA BUS PARITY ERROR-READ ONLY
944      000200      OR        = BIT07      ;OUTPUT READY-READ ONLY
945      000100      IR        = BIT06      ;INPUT READY-READ ONLY
946      000040      CLR       = BIT05      ;CONTROLLER CLEAR
947      000020      PAT       = BIT04      ;PARITY TEST
948      000010      BAI       = BIT03      ;UNIBUS ADDRESS INCREMENT INHIBIT
951      000004      U2        = BIT02      ;UNIT SELECT
          000002      U1        = BIT01      ;UNIT SELECT
          000001      U0        = BIT00      ;UNIT SELECT
952
953      ;UNIT SELECT MASK
954
955      000007      UNTMSK    = 7          ;UNIT SELECT MASK
956
957      ;*RMCS3 RH70 CONTROL STATUS REGISTER #3
958
959      100000      APE       = BIT15      ;ADDRESS PARITY ERROR
960      040000      DPEHI     = BIT14      ;DATA PARITY ERROR HIGH WORD
961      020000      DPELO     = BIT13      ;DATA PARITY ERROR LOW WORD
962      010000      WCEHI     = BIT12      ;WRITE CHECK ERROR HIGH WORD
963      004000      WCELO     = BIT11      ;WRITE CHECK ERROR LOW WORD
964      002000      DBL       = BIT10      ;DOUBLE WORD TRANSFER
965      000100      IE        = BIT06      ;INTERRUPT ENABLE
966      000010      IPCK3     = BIT03      ;INVERT PARITY CHECK
967      000004      IPCK2     = BIT02      ;INVERT PARITY CHECK
968      000002      IPCK1     = BIT01      ;INVERT PARITY CHECK
969      000001      IPCK0     = BIT00      ;INVERT PARITY CHECK
970
971      .SBTTL RH11/RH70 CONTROLLER REGISTER INDEX VALUES
972
973      000000      RMCS1     = 00          ;CONTROL, STATUS REGISTER #1
974      000002      RMWC      = 02          ;WORD COUNT REGISTER
975      000004      RMBA      = 04          ;BUS ADDRESS REGISTER
976      000010      RMCS2     = 10          ;CONTROL, STATUS REGISTER #2
977      000022      RMDB      = 22          ;DATA BUFFER
978      000050      RMBAE     = 50          ;BUS ADDRESS EXTENSION
979      000052      RMCS3     = 52          ;CONTROL, STATUS REGISTER #3
980
981      176700      ABASE     = 176700     ;UNIBUS ADDRESS
  
```



```

982      120254      AVECT1 = 120254      ;UNIBUS VECTOR ADDRESS AND PRIORITY
983
985
986      .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      .=174
000174    000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
000176    000000      SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER

          .SBTTL STARTING ADDRESS(ES)
987      000200    000137    004626      JMP      @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
988
          .SBTTL ACT11 HOOKS
          ;*****
          ;HOOKS REQUIRED BY ACT11
          000204      $SVPC=.      ;SAVE PC
          000046      .=46
          054026      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
          000052      .=52
          000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
          000204      .=$SVPC      ;; RESTORE PC

989
990      001100      .=1100
991      .SBTTL APT PARAMETER BLOCK
          ;*****
          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
          ;*****
          001100      $.X=.      ;;SAVE CURRENT LOCATION
          000024      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
          000200      200      ;;FOR APT START UP
          000044      .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
          001100      $APTHDR ;;POINT TO APT HEADER BLOCK
          001100      .=$.X      ;;RESET LOCATION COUNTER
          ;*****
          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
          ;INTERFACE SPEC.

          $APTHD:
          001100    000000    $SHIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
          001102    001222    $MBADR:  .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
          001104    000001    $TSTM:  .WORD 1      ;;RUN TIM OF LONGEST TEST
          001106    000002    $PASTM: .WORD 2      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
          001110    000002    $UNITM: .WORD 2      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
          001112    000042    .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
992      001114
  
```

0

.SBTTL COMMON TAGS

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

Address	Value	Tag Name	Format	Value	Description
001114	001114	\$CMTAG	.WORD	0	::START OF COMMON TAGS
001114	000000	\$STNM	.BYTE	0	::CONTAINS THE TEST NUMBER
001116	000	\$ERFLG	.BYTE	0	::CONTAINS ERROR FLAG
001117	000	\$ICNT	.WORD	0	::CONTAINS SUBTEST ITERATION COUNT
001120	000000	\$LPADR	.WORD	0	::CONTAINS SCOPE LOOP ADDRESS
001122	000000	\$LPERR	.WORD	0	::CONTAINS SCOPE RETURN FOR ERRORS
001124	000000	\$ERTTL	.WORD	0	::CONTAINS TOTAL ERRORS DETECTED
001126	000000	\$ITEMB	.BYTE	0	::CONTAINS ITEM CONTROL BYTE
001130	000	\$ERMAX	.BYTE	1	::CONTAINS MAX. ERRORS PER TEST
001131	001	\$ERRPC	.WORD	0	::CONTAINS PC OF LAST ERROR INSTRUCTION
001132	000000	\$GDADR	.WORD	0	::CONTAINS ADDRESS OF 'GOOD' DATA
001134	000000	\$BDADR	.WORD	0	::CONTAINS ADDRESS OF 'BAD' DATA
001136	000000	\$GDDAT	.WORD	0	::CONTAINS 'GOOD' DATA
001140	000000	\$BDDAT	.WORD	0	::CONTAINS 'BAD' DATA
001142	000000		.WORD	0	::RESERVED--NOT TO BE USED
001144	000000		.WORD	0	
001146	000000		.WORD	0	
001150	000	\$AUTOB	.BYTE	0	::AUTOMATIC MODE INDICATOR
001151	000	\$INTAG	.BYTE	0	::INTERRUPT MODE INDICATOR
001152	000000		.WORD	0	
001154	177570	\$SWR	.WORD	DSWR	::ADDRESS OF SWITCH REGISTER
001156	177570	\$DISPLAY	.WORD	DDISP	::ADDRESS OF DISPLAY REGISTER
001160	177560	\$TKS	.WORD	177560	::TTY KBD STATUS
001162	177562	\$TKB	.WORD	177562	::TTY KBD BUFFER
001164	177564	\$TPS	.WORD	177564	::TTY PRINTER STATUS REG. ADDRESS
001166	177566	\$TPB	.WORD	177566	::TTY PRINTER BUFFER REG. ADDRESS
001170	000	\$NULL	.BYTE	0	::CONTAINS NULL CHARACTER FOR FILLS
001171	002	\$FILLS	.BYTE	2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012	\$FILLC	.BYTE	12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000	\$TPFLG	.BYTE	0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000	\$TMP0	.WORD	0	::USER DEFINED
001176	000000	\$TMP1	.WORD	0	::USER DEFINED
001200	000000	\$TMP2	.WORD	0	::USER DEFINED
001202	000000	\$TMP3	.WORD	0	::USER DEFINED
001204	000000	\$TMP4	.WORD	0	::USER DEFINED
001206	000000	\$TIMES	.WORD	0	::MAX. NUMBER OF ITERATIONS
001210	000000	\$ESCAPE	.WORD	0	::ESCAPE ON ERROR ADDRESS
001212	207	\$BELL	.ASCIZ	<207><377><377>	::CODE FOR BELL
001216	077	\$QUES	.ASCII	/?/	::QUESTION MARK
001217	015	\$CRLF	.ASCII	<15>	::CARRIAGE RETURN
001220	012	\$LF	.ASCIZ	<12>	::LINE FEED

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

Address	Value	Tag Name	Format	Value	Description
001222		\$MAIL			::APT MAILBOX
001222	000000	\$MSGTY	.WORD	AMSGTY	::MESSAGE TYPE CODE
001224	000000	\$FATAL	.WORD	AFATAL	::FATAL ERROR NUMBER
001226	000000	\$TESTN	.WORD	ATESTN	::TEST NUMBER

001230	000000	\$PASS: .WORD	APASS	::PASS COUNT
001232	000000	\$DEVCT: .WORD	ADEVCT	::DEVICE COUNT
001234	000000	\$UNIT: .WORD	AUNIT	::I/O UNIT NUMBER
001236	000000	\$MSGAD: .WORD	AMSGAD	::MESSAGE ADDRESS
001240	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
001242		\$ETABLE:		::APT ENVIRONMENT TABLE
001242	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
001243	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
001244	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
001246	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
001250	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
		.*		BITS 15-11=CPU TYPE
		.*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
		.*		11/70=06,PDQ=07,Q=10
		.*		BIT 10=REAL TIME CLOCK
		.*		BIT 9=FLOATING POINT PROCESSOR
		.*		BIT 8=MEMORY MANAGEMENT
001252	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001253	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
		.*		MEM.TYPE BYTE -- (HIGH BYTE)
		.*		900 NSEC CORE=001
		.*		300 NSEC BIPOLAR=002
		.*		500 NSEC MOS=003
001254	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
		.*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
001256	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001257	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
001260	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001262	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001263	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
001264	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001266	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001267	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
001270	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001272	120254	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001274	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001276	176700	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001300	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
001302	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001304	000000	\$CDW2: .WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001306	000000	\$DDW0: .WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
001310	000000	\$DDW1: .WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
001312	000000	\$DDW2: .WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
001314	000000	\$DDW3: .WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
001316	000000	\$DDW4: .WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
001320	000000	\$DDW5: .WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
001322	000000	\$DDW6: .WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
001324	000000	\$DDW7: .WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
001326		\$ETEND:		
		.MEXIT		

```

0          .SBTTL  USER DEFINED TAGS

001326 000000  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.

001330      000  LSTRK: .BYTE  0          ;LO BYTE = 0
001331      000          .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
  
```

001332

GETBUF:

```

;REGISTER INPUT BUFFER
001332 000000  RMCS1I: .WORD  0          ;CONTROL, STATUS REGISTER #1
001334 000000  RMWCI:  .WORD  0          ;WORD COUNT REGISTER
001336 000000  RMBAI:  .WORD  0          ;BUS ADDRESS REGISTER
001340 000000  RMDAI:  .WORD  0          ;DISK ADDRESS REGISTER
001342 000000  RMCS2I: .WORD  0          ;CONTROL, STATUS REGISTER #2
001344 000000  RMDSI:  .WORD  0          ;DRIVE STATUS REGISTER
001346 000000  RMER1I: .WORD  0          ;ERROR REGISTER #1
001350 000000  RMASI:  .WORD  0          ;ATTENTION SUMMARY REGISTER
001352 000000  RMLAI:  .WORD  0          ;LOOK AHEAD REGISTER
001354 000000  RMDBI:  .WORD  0          ;DATA BUFFER
001356 000000  RMMR1I: .WORD  0          ;MAINTENANCE REGISTER #1
001360 000000  RMDTI:  .WORD  0          ;DRIVE TYPE REGISTER
001362 000000  RMSNI:  .WORD  0          ;SERIAL NUMBER REGISTER
001364 000000  RMOFI:  .WORD  0          ;OFFSET REGISTER
001366 000000  RMDCI:  .WORD  0          ;DESIRED CYLINDER REGISTER
001370 000000  RMHRI:  .WORD  0          ;HOLDING REGISTER
001372 000000  RMMR2I: .WORD  0          ;MAINTENANCE REGISTER #2
001374 000000  RMER2I: .WORD  0          ;ERROR REGISTER #2
001376 000000  RMEC1I: .WORD  0          ;ECC POSITION REGISTER
001400 000000  RMEC2I: .WORD  0          ;ECC PATTERN REGISTER
001402 000000  RMBAEI: .WORD  0          ;BUS ADDRESS EXTENSION REGISTER
001404 000000  RMCS3I: .WORD  0          ;CONTROL, STATUS REGISTER #3
  
```

```

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

001406

PUTBUF:

```

;REGISTER OUTPUT BUFFER
001406 000000  RMCS1O: .WORD  0          ;CONTROL, STATUS REGISTER #1
001410 000000  RMWCO:  .WORD  0          ;WORD COUNT REGISTER
001412 000000  RMBAO:  .WORD  0          ;BUS ADDRESS REGISTER
001414 000000  RMDAO:  .WORD  0          ;DISK ADDRESS REGISTER
001416 000000  RMCS2O: .WORD  0          ;CONTROL, STATUS REGISTER #2
001420 000000  RMDSO:  .WORD  0          ;DRIVE STATUS REGISTER
001422 000000  RMER1O: .WORD  0          ;ERROR REGISTER #1
001424 000000  RMASO:  .WORD  0          ;ATTENTION SUMMARY REGISTER
001426 000000  RMLAO:  .WORD  0          ;LOOK AHEAD REGISTER
001430 000000  RMDBO:  .WORD  0          ;DATA BUFFER
001432 000000  RMMR1O: .WORD  0          ;MAINTENANCE REGISTER #1
001434 000000  RMDTO:  .WORD  0          ;DRIVE TYPE REGISTER
001436 000000  RMSNO:  .WORD  0          ;SERIAL NUMBER REGISTER
  
```


001440	000000	RM0FO:	.WORD	0	:OFFSET REGISTER
001442	000000	RMDCO:	.WORD	0	:DESIRED CYLINDER REGISTER
001444	000000	RMHRO:	.WORD	0	:HOLDING REGISTER
001446	000000	RMMR20:	.WORD	0	:MAINTENANCE REGISTER #2
001450	000000	RMER20:	.WORD	0	:ERROR REGISTER #2
001452	000000	RMEC10:	.WORD	0	:ECC POSITION REGISTER
001454	000000	RMEC20:	.WORD	0	:ECC PATTERN REGISTER
001456	000000	RMBAE0:	.WORD	0	:BUS ADDRESS EXTENSION REGISTER
001460	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.

001462	000000	TSTQUE:	.WORD	0	:CONTAINS DEVICE POINTER
			.BLKW	8.	:TEST QUE FOR DEVICES UNDER TEST
001504	000000		.WORD	0	:TABLE TERMINATOR GOES HERE WHEN :ALL 8. DEVICES ARE UNDER TEST.

001506	172540	\$LPCSR:	.WORD	172540	:KW11-P CONTROL + STATUS REGISTER
001510	172542	\$LPCSB:	.WORD	172542	:KW11-P COUNT SET BUFFER
001512	000104	\$LPVEC:	.WORD	104	:KW11-P INTERRUPT VECTOR
001514	000106		.WORD	106	
001516	177546	\$LLCSR:	.WORD	177546	:KW11-L CONTROL + STATUS REGISTER
001520	000100	\$LLVEC:	.WORD	100	:KW11-L INTERRUPT VECTOR
001522	000102		.WORD	102	
001524	000000	\$PSW:	.WORD		:STORAGE FOR PRIORITY
001526	000000	TIME:	.WORD		:STORAGE FOR ELAPSED TIME
001530	000000	WATCH:	.WORD		:STORAGE FOR REMAINING TIME
001532	000000	CLOCK:	.WORD		:ADDRESS OF START CLOCK SUB
001534	000000	STOP:	.WORD		:ADDRESS OF STOP CLOCK SUB

:PUT TAGS HERE

0

.SBTTL ERROR POINTER TABLE

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

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

001536

\$ERRTB:

1
2
3

;ERROR 1 CANNOT CLEAR NED STATUS

001536 063510
001540 071366
001542 071466
001544 071514

EMT1
EHT1
EDT1
EFT1

4
5
6

;ERROR 2 CANNOT READ OR WRITE ANY DEVICE REG WITHOUT NED

001546 063516
001550 071372
001552 071470
001554 071516

EMT2
EHT2
EDT2
EFT2

7
8
9

;ERROR 3 CANNOT WRITE/READ ONES TO ANY DEVICE REGISTER

001556 063544
001560 000000
001562 000000
001564 000000

EMT3
0
0
0

10
11
12

;ERROR 4 CANNOT CLEAR ANY DEVICE REGISTER BITS W/MASSBUS INIT

001566 063564
001570 000000
001572 000000
001574 000000

EMT4
0
0
0

13
14
15

;ERROR 5 CANNOT WRITE/READ ZEROS TO ALL BIT POSITIONS

001576 063606
001600 071376
001602 071472
001604 071520

EMT5
EHT5
EDT5
EFT5

16
17

;ERROR 6 CANNOT WRITE/READ ONES TO ALL BIT POSITIONS

18	001606	063632	EMT6	
	001610	071376	EHT5	
	001612	071472	EDT5	
	001614	071520	EFT5	
19				
20			:ERROR	7 CANNOT WRITE/READ SHIFTING ONE BIT TO ALL BIT POSITIONS
21			:	OF DEVICE REGISTERS
22				
	001616	063654	EMT7	
	001620	071402	EHT7	
	001622	071472	EDT5	
	001624	071520	EFT5	
23				
24			:ERROR	10 REGISTER SELECT 1 APPEARS S-A-0
25				
	001626	063676	EMT10	
	001630	000000	0	
	001632	000000	0	
	001634	000000	0	
26				
27			:ERROR	11 REGISTER SELECT 1 APPEARS S-A-1
28				
	001636	063714	EMT11	
	001640	000000	0	
	001642	000000	0	
	001644	000000	0	
29				
30			:ERROR	12 REGISTER SELECT 2 APPEARS S-A-0
31				
	001646	063732	EMT12	
	001650	000000	0	
	001652	000000	0	
	001654	000000	0	
32				
33			:ERROR	13 REGISTER SELECT 2 APPEARS S-A-1
34				
	001656	063750	EMT13	
	001660	000000	0	
	001662	000000	0	
	001664	000000	0	
35				
36			:ERROR	14 REGISTER SELECT 4 APPEARS S-A-0
37				
	001666	063766	EMT14	
	001670	000000	0	
	001672	000000	0	
	001674	000000	0	
38				
39			:ERROR	15 REGISTER SELECT 4 APPEARS S-A-1

40	001676	064004	EMT15	
	001700	000000	0	
	001702	000000	0	
	001704	000000	0	
41				
42				:ERROR 16 REGISTER SELECT 8 APPEARS S-A-0
43	001706	064022	EMT16	
	001710	000000	0	
	001712	000000	0	
	001714	000000	0	
44				
45				:ERROR 17 REGISTER SELECT 8 APPEARS S-A-1
46	001716	064040	EMT17	
	001720	000000	0	
	001722	000000	0	
	001724	000000	0	
47				
48				:ERROR 20 CANT WRITE ZEROS RMDA
49	001726	064056	EMT20	
	001730	071366	EHT1	
	001732	071466	EDT1	
	001734	071514	EFT1	
50				
51				:ERROR 21 CANT WRITE ONES RMDA
52	001736	064076	EMT21	
	001740	071366	EHT1	
	001742	071466	EDT1	
	001744	071514	EFT1	
53				
54				:ERROR 22 BIT INTERFERENCE IN WRITING/READING RMDA
55	001746	064116	EMT22	
	001750	071366	EHT1	
	001752	071466	EDT1	
	001754	071514	EFT1	
56				
57				:ERROR 23 CANT WRITE ZEROS RMCS1
58	001756	064132	EMT23	
	001760	071366	EHT1	
	001762	071466	EDT1	
	001764	071514	EFT1	
59				
60				:ERROR 24 CANT WRITE ONES RMCS1
61				

Line	Address	Code	Message
	001766	064152	EMT24
	001770	071366	EHT1
	001772	071466	EDT1
	001774	071514	EFT1
62			
63			;ERROR 25 BIT INTERFERENCE IN WRITING/READING RMCS1
64			
	001776	064172	EMT25
	002000	071366	EHT1
	002002	071466	EDT1
	002004	071514	EFT1
65			
66			;ERROR 26 MBA CLR L IS STUCK ACTIVE
67			
	002006	064206	EMT26
	002010	000000	0
	002012	000000	0
	002014	000000	0
68			
69			;ERROR 27 CANNOT CLEAR RMER1-PAR,RMR,ILF,ILR
70			
	002016	064240	EMT27
	002020	071366	EHT1
	002022	071466	EDT1
	002024	071514	EFT1
71			
72			;ERROR 30 CANNOT CLEAR RMER1-DCK,IAE,AOE,HCRC,HCE,ECH,WCF,FER
73			
	002026	064252	EMT30
	002030	071366	EHT1
	002032	071466	EDT1
	002034	071514	EFT1
74			
75			;ERROR 31 CANNOT CLEAR RMER1-OPI,DTE
76			
	002036	064266	EMT31
	002040	071366	EHT1
	002042	071466	EDT1
	002044	071514	EFT1
77			
78			;ERROR 32 CANNOT WRITE 0 IN RMER1-PAR,RMR,ILF,ILR
79			
	002046	064302	EMT32
	002050	071366	EHT1
	002052	071466	EDT1
	002054	071514	EFT1
80			
81			;ERROR 33 CANNOT WRITE 0 IN RMER1-DCK,IAE,AOE,HCRC,HCE,ECH,WCF,FER
82			
	002056	064316	EMT33

ERROR POINTER TABLE			
002060	071366	EHT1	
002062	071466	EDT1	
002064	071514	EFT1	
83			
84		:ERROR 34	CANNOT WRITE 0 IN RMER1-OPI,DTE
85			
002066	064334	EMT34	
002070	071366	EHT1	
002072	071466	EDT1	
002074	071514	EFT1	
86			
87		:ERROR 35	CANNOT WRITE 1 IN RMER1
88			
002076	064352	EMT35	
002100	071366	EHT1	
002102	071466	EDT1	
002104	071514	EFT1	
89			
90		:ERROR 36	CANNOT WRITE SHIFTING 1 IN RMER1
91			
002106	064366	EMT36	
002110	071366	EHT1	
002112	071466	EDT1	
002114	071514	EFT1	
92			
93		:ERROR 37	CANNOT WRITE ZEROS IN RMDC
94			
002116	064402	EMT37	
002120	071366	EHT1	
002122	071466	EDT1	
002124	071514	EFT1	
95			
96		:ERROR 40	CANNOT WRITE ONES IN RMDC
97			
002126	064416	EMT40	
002130	071366	EHT1	
002132	071466	EDT1	
002134	071514	EFT1	
98			
99		:ERROR 41	BIT INTERFERENCE IN WRITING/READING RMDC
100			
002136	064436	EMT41	
002140	071366	EHT1	
002142	071466	EDT1	
002144	071514	EFT1	
101			
102		:ERROR 42	CANNOT WRITE 1'S IN RMDC OR RMDA
103			
002146	064452	EMT42	
002150	000000	0	

	002152	000000	0	
	002154	000000	0	
104				
105				
106				:ERROR 43 CANNOT CLEAR RMCS1-FUNCTION CODE
	002156	064476	EMT43	
	002160	071366	EHT1	
	002162	071466	EDT1	
	002164	071514	EFT1	
107				
108				
109				:ERROR 44 UNUSED BITS OF RMER2 NOT ZERO
	002166	064510	EMT44	
	002170	071366	EHT1	
	002172	071466	EDT1	
	002174	071514	EFT1	
110				
111				
112				:ERROR 45 CANNOT CLEAR RMER2-OPE,IVC,LSC
	002176	064524	EMT45	
	002200	071366	EHT1	
	002202	071466	EDT1	
	002204	071514	EFT1	
113				
114				
115				:ERROR 46 CANNOT CLEAR RMER2-LBC,DPE
	002206	064540	EMT46	
	002210	071366	EHT1	
	002212	071466	EDT1	
	002214	071514	EFT1	
116				
117				
118				:ERROR 47 CANNOT WRITE ZEROS RMER2-OPE,IVC,LSC
	002216	064554	EMT47	
	002220	071366	EHT1	
	002222	071466	EDT1	
	002224	071514	EFT1	
119				
120				
121				:ERROR 50 CANNOT WRITE ZEROS RMER2-LBC,DPE
	002226	064576	EMT50	
	002230	071366	EHT1	
	002232	071466	EDT1	
	002234	071514	EFT1	
122				
123				
124				:ERROR 51 CANNOT WRITE ONES RMER2
	002236	064620	EMT51	
	002240	071366	EHT1	
	002242	071466	EDT1	

002244	071514	EFT1	
125			
126		:ERROR 52	CANNOT WRITE SHIFTING ONES RMER2
127			
002246	064640	EMT52	
002250	071366	EHT1	
002252	071466	EDT1	
002254	071514	EFT1	
128			
129		:ERROR 53	UNUSED BITS OF RMOF ARE NOT ZERO
130			
002256	064654	EMT53	
002260	071366	EHT1	
002262	071466	EDT1	
002264	071514	EFT1	
131			
132		:ERROR 54	CANNOT WRITE ZEROS RMOF-FMT,ECI,HCI,OFD
133			
002266	064670	EMT54	
002270	071366	EHT1	
002272	071466	EDT1	
002274	071514	EFT1	
134			
135		:ERROR 55	CANNOT WRITE ONES RMOF-FMT,ECI,HCI,OFD
136			
002276	064710	EMT55	
002300	071366	EHT1	
002302	071466	EDT1	
002304	071514	EFT1	
137			
138		:ERROR 56	CANNOT WRITE SHIFTING ONES RMOF
139			
002306	064730	EMT56	
002310	071366	EHT1	
002312	071466	EDT1	
002314	071514	EFT1	
140			
141		:ERROR 57	DEVICE IS NOT AN RM05/3/2
142			
002316	064744	EMT57	
002320	071406	EHT57	
002322	071474	EDT57	
002324	071522	EFT57	
143			
144		:ERROR 60	DEVICE AVAILABLE IS NOT SET
145			
002326	064760	EMT60	
002330	071366	EHT1	
002332	071466	EDT1	
002334	071514	EFT1	

146				
147				
148			;ERROR 61	CANNOT WRITE ZEROS RMHR
	002336	064774	EMT61	
	002340	071366	EHT1	
	002342	071466	EDT1	
	002344	071514	EFT1	
149				
150			;ERROR 62	CANNOT WRITE ONES RMHR
151				
	002346	065014	EMT62	
	002350	071366	EHT1	
	002352	071466	EDT1	
	002354	071514	EFT1	
152				
153			;ERROR 63	CANNOT WRITE SHIFTING ONES RMHR
154				
	002356	065034	EMT63	
	002360	071366	EHT1	
	002362	071466	EDT1	
	002364	071514	EFT1	
155				
156			;ERROR 64	CANNOT CLEAR ILR STATUS
157				
	002366	065050	EMT64	
	002370	071366	EHT1	
	002372	071466	EDT1	
	002374	071514	EFT1	
158				
159			;ERROR 65	ILR ERROR SHOULD NOT BE SET
160				
	002376	065062	EMT65	
	002400	071412	EHT65	
	002402	071476	EDT65	
	002404	071524	EFT65	
161				
162			;ERROR 66	ILR ERROR SHOULD BE SET
163				
	002406	065076	EMT66	
	002410	071412	EHT65	
	002412	071476	EDT65	
	002414	071524	EFT65	
164				
165			;ERROR 67	CANNOT CLEAR PAR STATUS-DPE IS RESET
166				
	002416	065112	EMT67	
	002420	071366	EHT1	
	002422	071466	EDT1	
	002424	071514	EFT1	

167			
168			
169		:ERROR 70	CANNOT CLEAR PAR AND DPE STATUS
	002426	065130	EMT70
	002430	071366	EHT1
	002432	071466	EDT1
	002434	071514	EFT1
170			
171		:ERROR 71	'PAR' ERROR SHOULD NOT BE SET-'PAT' IS OFF
172			
	002436	065150	EMT71
	002440	071416	EHT71
	002442	071500	EDT71
	002444	071526	EFT71
173			
174		:ERROR 72	'PAR' ERROR SHOULD BE SET-'PAT' IS ON
175			
	002446	065174	EMT72
	002450	071416	EHT71
	002452	071500	EDT71
	002454	071526	EFT71
176			
177		:ERROR 73	'MCPE' ERROR SHOULD NOT BE SET
178			
	002456	065220	EMT73
	002460	071416	EHT71
	002462	071500	EDT71
	002464	071526	EFT71
179			
180		:ERROR 74	UNEXPECTED BUS TIMEOUT
181			
	002466	065240	EMT74
	002470	071422	EHT74
	002472	071502	EDT74
	002474	071530	EFT74
182			
183		:ERROR 75	CANT CLEAR 'DMD'
184			
	002476	065250	EMT75
	002500	071366	EHT1
	002502	071466	EDT1
	002504	071514	EFT1
185			
186		:ERROR 76	CANT WRITE ZERO 'DMD'
187			
	002506	065262	EMT76
	002510	071366	EHT1
	002512	071466	EDT1
	002514	071514	EFT1

188

CZ
ER

189			:ERROR 77	CANT WRITE ONE 'DMD'
190	002516	065276		
	002520	071366	EMT77	
	002522	071466	EHT1	
	002524	071514	EDT1	
			EFT1	
191				
192			:ERROR 100	DMD SET BY WRONG BIT
193	002526	065312		
	002530	071416	EMT100	
	002532	071476	EHT71	
	002534	071524	EDT65	
			EFT65	
194				
195			:ERROR 101	CANT CLEAR 'MOL' IN DIAGNOSTIC MODE
196	002536	065332		
	002540	071366	EMT101	
	002542	071466	EHT1	
	002544	071514	EDT1	
			EFT1	
197				
198			:ERROR 102	CANT SET 'MOL' IN DIAGNOSTIC MODE
199	002546	065352		
	002550	071366	EMT102	
	002552	071466	EHT1	
	002554	071514	EDT1	
			EFT1	
200				
201			:ERROR 103	'MUR' SET BY WRONG BIT
202	002556	065372		
	002560	071416	EMT103	
	002562	071476	EHT71	
	002564	071524	EDT65	
			EFT65	
203				
204			:ERROR 104	CANT RESET 'WRL' IN DIAGNOSTIC MODE
205	002566	065416		
	002570	071366	EMT104	
	002572	071466	EHT1	
	002574	071514	EDT1	
			EFT1	
206				
207			:ERROR 105	CANT SET 'WRL' IN DIAGNOSTIC MODE
208	002576	065436		
	002600	071366	EMT105	
	002602	071466	EHT1	
	002604	071514	EDT1	
			EFT1	
209				
210			:ERROR 106	'MWP' SET BY WRONG BIT

211	002606 065456	EMT106	
	002610 071416	EHT71	
	002612 071476	EDT65	
	002614 071524	EFT65	
212			
213			:ERROR 107 CANT RESET 'DVC' USING 'MDVC'
214	002616 065502	EMT107	
	002620 071366	EHT1	
	002622 071466	EDT1	
	002624 071514	EFT1	
215			
216			:ERROR 110 'DVC' IS RESET BUT 'UNS' IS SET
217	002626 065522	EMT110	
	002630 071366	EHT1	
	002632 071466	EDT1	
	002634 071514	EFT1	
218			
219			:ERROR 111 'DVC' IS SET BUT 'UNS' IS NOT SET
220	002636 065544	EMT111	
	002640 071366	EHT1	
	002642 071466	EDT1	
	002644 071514	EFT1	
221			
222			:ERROR 112 CANT SET 'DVC' USING MDVC''
223	002646 065564	EMT112	
	002650 071366	EHT1	
	002652 071466	EDT1	
	002654 071514	EFT1	
224			
225			:ERROR 113 'DVC' IS RESET BUT 'UNS' IS SET
226	002656 065604	EMT113	
	002660 071366	EHT1	
	002662 071466	EDT1	
	002664 071514	EFT1	
227			
228			:ERROR 114 'DVC' IS SET BUT 'UNS' IS NOT SET
229	002666 065630	EMT114	
	002670 071366	EHT1	
	002672 071466	EDT1	
	002674 071514	EFT1	
230			
231			:ERROR 115 'MDF' IS SET BY WRONG BIT
232			

ERROR POINTER TABLE

	002676	065652		EMT115
	002700	071426		EHT115
	002702	071504		EDT115
	002704	071532		EFT115
233				
234			;ERROR 116	CANT RESET 'SKI' USING 'MSER'
235				
	002706	065676		EMT116
	002710	071366		EHT1
	002712	071466		EDT1
	002714	071514		EFT1
236				
237			;ERROR 117	CANT SET 'SKI' USING 'MSER'
238				
	002716	065716		EMT117
	002720	071366		EHT1
	002722	071466		EDT1
	002724	071514		EFT1
239				
240			;ERROR 120	'SKI' SET BY WRONG BIT
241				
	002726	065736		EMT120
	002730	071426		EHT115
	002732	071504		EDT115
	002734	071532		EFT115
242				
243			;ERROR 121	CANT RESET 'PIP' USING 'MOC'
244				
	002736	065762		EMT121
	002740	071366		EHT1
	002742	071466		EDT1
	002744	071514		EFT1
245				
246			;ERROR 122	CANT SET 'PIP' USING 'MOC'
247				
	002746	066002		EMT122
	002750	071366		EHT1
	002752	071466		EDT1
	002754	071514		EFT1
248				
249			;ERROR 123	'MOC' SET BY WRONG BIT
250				
	002756	066022		EMT123
	002760	071426		EHT115
	002762	071504		EDT115
	002764	071532		EFT115
251				
252			;ERROR 124	CANT CLEAR 'EBL'
253				
	002766	066046		EMT124

ERROR POINTER TABLE

	002770	071366	EHT1	
	002772	071466	EDT1	
	002774	071514	EFT1	
254				
255				:ERROR 125 'EBL' NOT ZERO IN DIAGNOSTIC MODE
256				
	002776	066064	EMT125	
	003000	071366	EHT1	
	003002	071466	EDT1	
	003004	071514	EFT1	
257				
258				:ERROR 126 CANT SET 'EBL' USING 'DEBL'
259				
	003006	066104	EMT126	
	003010	071366	EHT1	
	003012	071466	EDT1	
	003014	071514	EFT1	
260				
261				:ERROR 127 'DEBL' SET BY WRONG BIT
262				
	003016	066122	EMT127	
	003020	071426	EHT115	
	003022	071504	EDT115	
	003024	071532	EFT115	
263				
264				:ERROR 130 'LS' NOT CORRECT ACCORDING TO RMDA
265				
	003026	066146	EMT130	
	003030	071432	EHT130	
	003032	071506	EDT130	
	003034	071534	EFT130	
266				
267				:ERROR 131 'LST' NOT CORRECT ACCORDING TO RMDA
268				
	003036	066164	EMT131	
	003040	071432	EHT130	
	003042	071506	EDT130	
	003044	071534	EFT130	
269				
270				:ERROR 132 CANNOT INCREMENT SECTOR ADDRESS USING 'DEBL'
271				
	003046	066202	EMT132	
	003050	071436	EHT132	
	003052	071510	EDT132	
	003054	071536	EFT132	
272				
273				:ERROR 133 CANNOT INCREMENT TRACK ADDRESS USING 'DEBL'
274				
	003056	066222	EMT133	
	003060	071436	EHT132	

	003062 071510	EDT132	
	003064 071536	EFT132	
275			
276			:ERROR 134 UNUSED BITS OF RMDC NOT ZERO
277			
	003066 066242	EMT134	
	003070 071366	EHT1	
	003072 071466	EDT1	
	003074 071514	EFT1	
278			
279			:ERROR 135 'VV' NOT RESET BY UNIT READY
280			
	003076 066256	EMT135	
	003100 071366	EHT1	
	003102 071466	EDT1	
	003104 071514	EFT1	
281			
282			:ERROR 136 SERIAL NUMBER IS INCONSISTENT
283			
	003106 066274	EMT136	
	003110 071366	EHT1	
	003112 071466	EDT1	
	003114 071514	EFT1	
284			
285			:ERROR 137 CANT CLEAR 'GO' BIT
286			
	003116 066306	EMT137	
	003120 071366	EHT1	
	003122 071466	EDT1	
	003124 071514	EFT1	
287			
288			:ERROR 140 CANT INCREMENT CYLINDER USING 'DEBL'
289			
	003126 066324	EMT140	
	003130 071436	EHT132	
	003132 071510	EDT132	
	003134 071536	EFT132	
290			
291			:ERROR 141 CANT RESET 'LBT' BY WRITING RMDA
292			
	003136 066344	EMT141	
	003140 071442	EHT142	
	003142 071510	EDT132	
	003144 071536	EFT132	
293			
294			:ERROR 142 CANT SET 'LBT' USING 'DEBL'
295			
	003146 066360	EMT142	
	003150 071442	EHT142	
	003152 071510	EDT132	

Line	Address	Module	Label	Description
	003154	071536	EFT132	
296				
297				
298				
	003156	066376	EMT143	
	003160	071366	EHT1	
	003162	071466	EDT1	
	003164	071514	EFT1	
299				
300				
301				
	003166	066412	EMT144	
	003170	071366	EHT1	
	003172	071466	EDT1	
	003174	071514	EFT1	
302				
303				
304				
	003176	066434	EMT145	
	003200	071446	EHT145	
	003202	071506	EDT130	
	003204	071534	EFT130	
305				
306				
307				
	003206	066456	EMT146	
	003210	071366	EHT1	
	003212	071466	EDT1	
	003214	071514	EFT1	
308				
309				
310				
	003216	066474	EMT147	
	003220	071366	EHT1	
	003222	071466	EDT1	
	003224	071514	EFT1	
311				
312				
313				
	003226	066506	EMT150	
	003230	071452	EHT150	
	003232	071504	EDT115	
	003234	071532	EFT115	
314				
315				
316				
	003236	066530	EMT151	
	003240	071366	EHT1	
	003242	071466	EDT1	
	003244	071514	EFT1	

CZ
ER

317				
318			:ERROR	152 'DRY' NOT THE COMPLEMENT OF 'GO'
319				
	003246	066542		EMT152
	003250	071366		EHT1
	003252	071466		EDT1
	003254	071514		EFT1
320				
321			:ERROR	153 'GO' RESET EARLY
322				
	003256	066556		EMT153
	003260	071366		EHT1
	003262	071466		EDT1
	003264	071514		EFT1
323				
324			:ERROR	154 'GO' DIDNT RESET ON TIME
325				
	003266	066576		EMT154
	003270	071366		EHT1
	003272	071466		EDT1
	003274	071514		EFT1
326				
327			:ERROR	155 CANT CLEAR CONTINUE
328				
	003276	066616		EMT155
	003300	071366		EHT1
	003302	071466		EDT1
	003304	071514		EFT1
329				
330			:ERROR	156 CONTINUE IS INCORRECT FOR THE FUNCTION CODE
331				
	003306	066634		EMT156
	003310	071452		EHT150
	003312	071504		EDT115
	003314	071532		EFT115
332				
333			:ERROR	157 CANT CLEAR IVC
334				
	003316	066656		EMT157
	003320	071366		EHT1
	003322	071466		EDT1
	003324	071514		EFT1
335				
336			:ERROR	160 IVC IS INCORRECT FOR THE FUNCTION CODE
337				
	003326	066674		EMT160
	003330	071452		EHT150
	003332	071504		EDT115
	003334	071532		EFT115

338
339
340

003336 066724
003340 071366
003342 071466
003344 071514

;ERROR 161 CANT CLEAR LSC

EMT161
EHT1
EDT1
EFT1

341
342
343

003346 066742
003350 071366
003352 071466
003354 071514

;ERROR 162 CANT SET LSC

EMT162
EHT1
EDT1
EFT1

344
345
346

003356 066760
003360 071366
003362 071466
003364 071514

;ERROR 163 COMMAND DECODE WAS ENABLED WITH COMP ERROR SET

EMT163
EHT1
EDT1
EFT1

347
348
349

003366 067002
003370 071366
003372 071466
003374 071514

;ERROR 164 COMMAND DECODE WAS ENABLED WITH COMP ERROR SET

EMT164
EHT1
EDT1
EFT1

350
351
352

003376 067024
003400 000000
003402 000000
003404 000000

;ERROR 165 DECODE DOES NOT SET

EMT165
0
0
0

353
354
355

003406 067050
003410 071366
003412 071466
003414 071514

;ERROR 166 CANT CLEAR OCCUPIED

EMT166
EHT1
EDT1
EFT1

356
357
358

003416 067070
003420 071366
003422 071466
003424 071514

;ERROR 167 ILF SET WITHOUT GO BIT

EMT167
EHT1
EDT1
EFT1

359

360			:ERROR 170	CANT SET VOLUME VALID
361	003426	067112		EMT170
	003430	071452		EHT150
	003432	071504		EDT115
	003434	071532		EFT115
362				
363			:ERROR 171	ILF IS INCORRECT
364	003436	067124		EMT171
	003440	071452		EHT150
	003442	071504		EDT115
	003444	071532		EFT115
365				
366			:ERROR 172	CANT SET OFFSET DIRECTION BIT
367	003446	067140		EMT172
	003450	071366		EHT1
	003452	071466		EDT1
	003454	071514		EFT1
368				
369			:ERROR 173	OCCUPIED IS INCORRECT FOR FUNCTION CODE
370	003456	067156		EMT173
	003460	071452		EHT150
	003462	071504		EDT115
	003464	071532		EFT115
371				
372			:ERROR 174	READ IN PRESET DIDNT CLEAR RMDA, RMDC OR RMOF
373	003466	067200		EMT174
	003470	000000		0
	003472	000000		0
	003474	000000		0
374				
375			:ERROR 175	READ IN PRESET DIDNT CLEAR RMOF
376	003476	067226		EMT175
	003500	071366		EHT1
	003502	071466		EDT1
	003504	071514		EFT1
377				
378			:ERROR 176	READ IN PRESET DIDNT CLEAR RMDA
379	003506	067244		EMT176
	003510	071366		EHT1
	003512	071466		EDT1
	003514	071514		EFT1
380				
381			:ERROR 177	READ IN PRESET DIDNT CLEAR RMDC

382	003516 067262	EMT177	
	003520 071366	EHT1	
	003522 071466	EDT1	
	003524 071514	EFT1	
383			
384			:ERROR 200 CANT SET OFFSET MODE BY OFFSET COMMAND
385			
	003526 067300	EMT200	
	003530 071366	EHT1	
	003532 071466	EDT1	
	003534 071514	EFT1	
386			
387			:ERROR 201 CANT RESET OFFSET MODE BY RTC COMMAND
388			
	003536 067316	EMT201	
	003540 071366	EHT1	
	003542 071466	EDT1	
	003544 071514	EFT1	
389			
390			:ERROR 202 CANT RESET OFD BY RTC COMMAND
391			
	003546 067334	EMT202	
	003550 071366	EHT1	
	003552 071466	EDT1	
	003554 071514	EFT1	
392			
393			:ERROR 203 CANT RESET OM BY RMDC
394			
	003556 067352	EMT203	
	003560 071366	EHT1	
	003562 071466	EDT1	
	003564 071514	EFT1	
395			
396			:ERROR 204 CANT RESET OM BY EBL
397			
	003566 067374	EMT204	
	003570 071366	EHT1	
	003572 071466	EDT1	
	003574 071514	EFT1	
398			
399			:ERROR 205 RUN AND GO NOT CORRECT FOR FUNCTION CODE
400			
	003576 067414	EMT205	
	003600 071452	EHT150	
	003602 071504	EDT115	
	003604 071532	EFT115	
401			
402			:ERROR 206 CANT SET IAE ERROR
403			

	003606	067434		EMT206
	003610	000000		0
	003612	000000		0
	003614	000000		0
404				
405			:ERROR 207	IAE IS INCORRECT FOR FUNCTION CODE
406				
	003616	067464		EMT207
	003620	071452		EHT150
	003622	071504		EDT115
	003624	071532		EFT115
407				
408			:ERROR 210	IAE IS INCORRECT FOR RMDA
409				
	003626	067500		EMT210
	003630	071426		EHT115
	003632	071504		EDT115
	003634	071532		EFT115
410				
411			:ERROR 211	IAE IS INCORRECT FOR RMDC
412				
	003636	067516		EMT211
	003640	071426		EHT115
	003642	071504		EDT115
	003644	071532		EFT115
413				
414			:ERROR 212	CANT SET AOE
415				
	003646	067534		EMT212
	003650	071442		EHT142
	003652	071510		EDT132
	003654	071536		EFT132
416				
417			:ERROR 213	RMR SET WHEN WRITING RMAS OR RMCS
418				
	003656	067546		EMT213
	003660	071456		EHT213
	003662	071504		EDT115
	003664	071532		EFT115
419				
420			:ERROR 214	CANT SET RMR
421				
	003666	067576		EMT214
	003670	071456		EHT213
	003672	071504		EDT115
	003674	071532		EFT115
422				
423			:ERROR 215	DRQ IS 0 AND PGM IS 1
424				
	003676	067610		EMT215

	003700	071366		EHT1
	003702	071466		EDT1
	003704	071514		EFT1
425				
426			:ERROR 216	DVA IS NOT SET
427				
	003706	067630		EMT216
	003710	071366		EHT1
	003712	071466		EDT1
	003714	071514		EFT1
428				
429			:ERROR 217	DPR IS NOT SET
430				
	003716	067644		EMT217
	003720	071366		EHT1
	003722	071466		EDT1
	003724	071514		EFT1
431				
432			:ERROR 220	CANT SET PORT REQUEST BY READING RMCS1
433				
	003726	067660		EMT220
	003730	071462		EHT220
	003732	071512		EDT220
	003734	071540		EFT220
434				
435			:ERROR 221	CANT SET PORT REQUEST BY WRITING RMAS
436				
	003736	067676		EMT221
	003740	071462		EHT220
	003742	071512		EDT220
	003744	071540		EFT220
437				
438			:ERROR 222	CANT SET PORT REQUEST BY WRITING RMDA
439				
	003746	067714		EMT222
	003750	071462		EHT220
	003752	071512		EDT220
	003754	071540		EFT220
440				
441			:ERROR 223	CANT RESET PORT REQUEST BY RELEASE COMMAND
442				
	003756	067732		EMT223
	003760	071462		EHT220
	003762	071512		EDT220
	003764	071540		EFT220
443				
444			:ERROR 224	CANT CLEAR ATA BY RMAS
445				
	003766	067750		EMT224
	003770	071366		EHT1

ERROR POINTER TABLE			
	003772 071466	EDT1	
	003774 071514	EFT1	
446			
447			
448		:ERROR 225	ATA IS RESET BUT RMAS NOT ZERO
	003776 067770	EMT225	
	004000 071366	EHT1	
	004002 071466	EDT1	
	004004 071514	EFT1	
449			
450		:ERROR 226	CANT RESET ATA BY GO
451			
	004006 070012	EMT226	
	004010 071366	EHT1	
	004012 071466	EDT1	
	004014 071514	EFT1	
452			
453		:ERROR 227	ATA NOT SET BY UNIT READY
454			
	004016 070030	EMT227	
	004020 071366	EHT1	
	004022 071466	EDT1	
	004024 071514	EFT1	
455			
456		:ERROR 230	ATA NOT SET BY UNIT READY
457			
	004026 070046	EMT230	
	004030 071366	EHT1	
	004032 071466	EDT1	
	004034 071514	EFT1	
458			
459		:ERROR 231	ATA NOT SET BY COMP ERROR
460			
	004036 070064	EMT231	
	004040 071366	EHT1	
	004042 071466	EDT1	
	004044 071514	EFT1	
461			
462		:ERROR 232	ATA SET/DID NOT SET WHEN REGISTER WRITTEN
463		:	WHILE COMP ERROR WAS SET
464			
	004046 070100	EMT232	
	004050 071456	EHT213	
	004052 071504	EDT115	
	004054 071532	EFT115	
465			
466		:ERROR 233	ATA NOT SET BY COMMAND SEQUENCER
467			
	004056 070122	EMT233	
	004060 071452	EHT150	

ERROR POINTER TABLE

004062	071504	EDT115	
004064	071532	EFT115	
468			
469			
470			:ERROR 234 WLE INCORRECT ACCORDING TO FUNCTION CODE
004066	070144	EMT234	
004070	071452	EHT150	
004072	071504	EDT115	
004074	071532	EFT115	
471			
472			
473			:ERROR 235 CANT CLEAR EXCEPTION
004076	070172	EMT235	
004100	071366	EHT1	
004102	071466	EDT1	
004104	071514	EFT1	
474			
475			
476			:ERROR 236 CANT SET EXCEPTION
004106	070212	EMT236	
004110	071426	EHT115	
004112	071504	EDT115	
004114	071532	EFT115	
477			
478			
479			:ERROR 237 CANT CLEAR IVC
004116	070244	EMT237	
004120	071366	EHT1	
004122	071466	EDT1	
004124	071514	EFT1	
480			
481			
482			:ERROR 240 CANT SET IVC
004126	070244	EMT237	
004130	071452	EHT150	
004132	071504	EDT115	
004134	071532	EFT115	
483			
484			
485			:ERROR 241 OPI NOT SET DURING RECALIBRATE
004136	070310	EMT241	
004140	071366	EHT1	
004142	071466	EDT1	
004144	071514	EFT1	
486			
487			
488			:ERROR 242 RECALIBRATE DID NOT ABORT WHEN DRIVE FAULT SET
004146	070334	EMT242	
004150	071366	EHT1	
004152	071466	EDT1	

004154	071514	EFT1	
489			
490		:ERROR 243	OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
491		:	DROPPED DURING RECALIBRATE
492			
004156	070360	EMT243	
004160	071366	EHT1	
004162	071466	EDT1	
004164	071514	EFT1	
493			
494		:ERROR 244	ATA NOT SET DURING RECALIBRATE
495			
004166	070404	EMT244	
004170	071366	EHT1	
004172	071466	EDT1	
004174	071514	EFT1	
496			
497		:ERROR 245	GO RESET EARLY DURING RECALIBRATE
498			
004176	070422	EMT245	
004200	071366	EHT1	
004202	071466	EDT1	
004204	071514	EFT1	
499			
500		:ERROR 246	GO NOT RESET DURING RECALIBRATE
501			
004206	070440	EMT246	
004210	071366	EHT1	
004212	071466	EDT1	
004214	071514	EFT1	
502			
503		:ERROR 247	INCORRECT TAG BUS DURING RECALIBRATE
504			
004216	070464	EMT247	
004220	071366	EHT1	
004222	071466	EDT1	
004224	071514	EFT1	
505			
506		:ERROR 250	OPI SHOULD HAVE SET DURING SEEK BECAUSE UNIT
507		:	READY DROPPED
508			
004226	070502	EMT250	
004230	071366	EHT1	
004232	071466	EDT1	
004234	071514	EFT1	
509			
510		:ERROR 251	SEEK DID NOT ABORT WHEN DRIVE FAULT SET
511			
004236	070526	EMT251	
004240	071366	EHT1	

ERROR POINTER TABLE				
	004242 071466		EDT1	
	004244 071514		EFT1	
512				
513		:ERROR 252		OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
514		:		DROPPED DURING SEEK
515				
	004246 070552		EMT252	
	004250 071366		EHT1	
	004252 071466		EDT1	
	004254 071514		EFT1	
516				
517		:ERROR 253		ATA NOT SET DURING SEEK
518				
	004256 070576		EMT253	
	004260 071366		EHT1	
	004262 071466		EDT1	
	004264 071514		EFT1	
519				
520		:ERROR 254		GO RESET EARLY DURING SEEK
521				
	004266 070614		EMT254	
	004270 071366		EHT1	
	004272 071466		EDT1	
	004274 071514		EFT1	
522				
523		:ERROR 255		GO DID NOT RESET DURING SEEK
524				
	004276 070632		EMT255	
	004300 071366		EHT1	
	004302 071466		EDT1	
	004304 071514		EFT1	
525				
526		:ERROR 256		INCORRECT TAG BUS DURING SEEK
527				
	004306 070656		EMT256	
	004310 071366		EHT1	
	004312 071466		EDT1	
	004314 071514		EFT1	
528				
529		:ERROR 257		OPI NOT SET DURING SEARCH
530				
	004316 070674		EMT257	
	004320 071366		EHT1	
	004322 071466		EDT1	
	004324 071514		EFT1	
531				
532		:ERROR 260		SEARCH DID NOT ABORT WHEN DRIVE FAULT SET
533				
	004326 070720		EMT260	
	004330 071366		EHT1	

Line	Code	Address	Module	Message
	004332	071466	EDT1	
	004334	071514	EFT1	
534				
535				
536				
537				
	004336	070744	EMT261	
	004340	071366	EHT1	
	004342	071466	EDT1	
	004344	071514	EFT1	
538				
539				
540				
	004346	070770	EMT262	
	004350	071366	EHT1	
	004352	071466	EDT1	
	004354	071514	EFT1	
541				
542				
543				
	004356	071006	EMT263	
	004360	071366	EHT1	
	004362	071466	EDT1	
	004364	071514	EFT1	
544				
545				
546				
	004366	071024	EMT264	
	004370	071366	EHT1	
	004372	071466	EDT1	
	004374	071514	EFT1	
547				
548				
549				
	004376	071050	EMT265	
	004400	071366	EHT1	
	004402	071466	EDT1	
	004404	071514	EFT1	
550				
551				
552				
	004406	071066	EMT266	
	004410	071366	EHT1	
	004412	071466	EDT1	
	004414	071514	EFT1	
553				
554				
555				
	004416	071104	EMT267	
	004420	071366	EHT1	

	004422 071466	EDT1	
	004424 071514	EFT1	
556			
557			:ERROR 270 OPI NOT SET DURING DATA COMMAND
558			
	004426 071120	EMT270	
	004430 071366	EHT1	
	004432 071466	EDT1	
	004434 071514	EFT1	
559			
560			:ERROR 271 DATA COMMAND DID NOT ABORT WHEN DRIVE FAULT SET
561			
	004436 071144	EMT271	
	004440 071366	EHT1	
	004442 071466	EDT1	
	004444 071514	EFT1	
562			
563			:ERROR 272 EBL RESET EARLY DURING DATA COMMAND
564			
	004446 071170	EMT272	
	004450 071366	EHT1	
	004452 071466	EDT1	
	004454 071514	EFT1	
565			
566			:ERROR 273 EBL DIDNT RESET ON TIME DURING DATA COMMAND
567			
	004456 071206	EMT273	
	004460 071366	EHT1	
	004462 071466	EDT1	
	004464 071514	EFT1	
568			
569			:ERROR 274 GO NOT RESET DURING DATA COMMAND
570			
	004466 071224	EMT274	
	004470 071366	EHT1	
	004472 071466	EDT1	
	004474 071514	EFT1	
571			
572			:ERROR 275 RUN AND GO NOT SET DURING DATA COMMAND
573			
	004476 071242	EMT275	
	004500 071366	EHT1	
	004502 071466	EDT1	
	004504 071514	EFT1	
574			
575			:ERROR 276 INCORRECT TAG BUS DURING DATA COMMAND
576			
	004506 071262	EMT276	
	004510 071366	EHT1	
	004512 071466	EDT1	


```
004514 071514          EFT1
577
578          :ERROR 277      OPI NOT SET DURING DATA COMMAND WHEN ON
579          ;              CYLINDER DIDNT DROP
580
004516 071300          EMT277
004520 071366          EHT1
004522 071466          EDT1
004524 071514          EFT1

581
582          :ERROR 300      DATA COMMAND DID NOT ABORT WHEN SEEK ERROR SET
583
004526 071324          EMT300
004530 071366          EHT1
004532 071466          EDT1
004534 071514          EFT1

584
585          :ERROR 301      SEARCH NOT ENABLED DURING DATA COMMAND
586
004536 071350          EMT301
004540 071366          EHT1
004542 071466          EDT1
004544 071514          EFT1

587
588          :PUT ERROR TABLE HERE
589
```

```

1      ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3      004546 011600
4      004550 005740
5      004552 022626
6      004554 104401 004562
       004560 000417
       004620
7      004620 010046
8      004622 104402
9      004624 000240
10
11
12
13
14     004626 000240
15     004630 005227 000000
16     004634 001375
17     004636 000005
18
19
       004640 012706 001114
       004644 005026
       004646 022706 001154
       004652 001374
       004654 012706 001100
       004660 012737 057066 000020
       004666 012737 000340 000022
       004674 012737 057640 000030
       004702 012737 000340 000032
       004710 012737 061400 000034
       004716 012737 000340 000036
       004724 012737 061506 000024
       004732 012737 000340 000026
       004740 013737 053672 053664
       004746 005037 001206
       004752 005037 001210
       004756 112737 000001 001131
       004764 012737 004764 001122
       004772 012737 004772 001124
       005000 013746 000004
       005004 012737 005000 000004
       005012 012737 177570 001154
       005020 012737 177570 001156
       005026 022777 177777 174120
       005034 001012
       005036 000403
       005040 012716 005046
       005044 000002
       005046 012737 000176 001154
       005054 012737 000174 001156

;BADTMO: MOV (SP),R0 ;SAVE PC WHERE THE TIME OUT OCCURED
;TST -(R0) ;ADJUST PC -2
;CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
;TYPE 65$ ;TYPE ASCIZ STRING
;BR 64$ ;GET OVER THE ASCIZ
;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
64$: MOV 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
START: NOP
;INC #0 ;TTY LOOP, WAIT FOR INCREMENT
;BNE -4 ;OF WORD
;RESET ;RESET THE WORLD

.SBTTL INITIALIZE THE COMMON TAGS
;:CLEAR THE COMMON TAGS ($CMTAG) AREA
;MOV # $CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
;CLR (R6)+ ;:CLEAR MEMORY LOCATION
;CMP #SWR,R6 ;:DONE?
;BNE -6 ;:LOOP BACK IF NO
;MOV #STACK,SP ;:SETUP THE STACK POINTER
;:INITIALIZE A FEW VECTORS
;MOV # $SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
;MOV #340,@IOTVEC+2 ;:LEVEL 7
;MOV # $ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
;MOV #340,@EMTVEC+2 ;:LEVEL 7
;MOV # $TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
;MOV #340,@TRAPVEC+2 ;:LEVEL 7
;MOV # $PWRDN,@PWRVEC ;:POWER FAILURE VECTOR
;MOV #340,@PWRVEC+2 ;:LEVEL 7
;MOV $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
;CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
;CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
;MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
;MOV #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
;MOV #,$SLPERR ;: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 @WERRVEC,-(SP) ;:SAVE ERROR VECTOR
;MOV #64$,@WERRVEC ;:SET UP ERROR VECTOR
;MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
;MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
;CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
;BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
; ;AND THE HARDWARE SWR IS NOT = -1
;BR 65$ ;:BRANCH IF NO TIMEOUT
64$: MOV #65$,(SP) ;:SET UP FOR TRAP RETURN
65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
;MOV #DISPREG,DISPLAY

```



```

005062 012637 000004      66$:  MOV      (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
005066 005037 001230      CLR      $PASS              ;;CLEAR PASS COUNT
005072 132737 000200 001243  BITB     #APTSIZE,$ENVM    ;;TEST USER SIZE UNDER APT
005100 001403 000200 001154  BEQ      67$              ;;YES,USE NON-APT SWITCH
005102 012737 001244 001154  MOV      #$$SWREG,$SWR    ;;NO,USE APT SWITCH REGISTER
005110
20      ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
21 005110 012737 004546 000004  MOV      #BADTMO,ERRVEC  ;;SETUP FOR UNEXPECTED TIMEOUT
22 005116 012737 000300 000006  MOV      #PR6,ERRVEC+2  ;;LEVEL 6
23 005124 012746 000140  MOV      #PR3,-(SP)      ;;PUT NEW PS ON STACK
005130 012746 005136  MOV      #68$,-(SP)     ;;PUT NEW PC ON STACK
005134 000002  RTI                    ;;POP NEW PC AND PS
005136
24      68$:
25      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005136 005227 177777      INC      #-1              ;;FIRST TIME?
005142 001055      BNE      69$              ;;BRANCH IF NO
005144 022737 054026 000042  CMP      #SENDAD,@#42    ;;ACT-11?
005152 001451      BEQ      69$              ;;BRANCH IF YES
005154 104401 005222      TYPE    ,70$            ;;TYPE ASCIZ STRING
      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
005160 005737 000042      TST     @#42             ;;ARE WE RUNNING UNDER XXDP/ACT?
005164 001012      BNE      71$              ;;BRANCH IF YES
005166 123727 001242 000001  CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
005174 001406      BEQ      71$              ;;BRANCH IF YES
005176 023727 001154 000176  CMP     $SWR,#SWREG     ;;SOFTWARE SWITCH REG SELECTED?
005204 001005      BNE      72$              ;;BRANCH IF NO
005206 104407      GTSWR                    ;;GET SOFT-SWR SETTINGS
005210 000403      BR      72$
005212 112737 000001 001150 71$:  MOVB    #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
005220 72$:
005220 000426      BR      69$              ;;GET OVER THE ASCIZ
005276      ;;70$: .ASCIZ <CRLF>@CZRMPAO - RM05/3/2 DISKLESS TEST, PART 1@<CRLF>
      69$:
26
27      ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
28      ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
29
30 005276 005037 001326      CLR      XXDP             ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
31 005302 122737 000016 000041  CMPB    #16,@#41        ;;LOADED FROM AN RM05/3/2 ?
32 005310 001160      BNE      5$              ;;BRANCH IF NOT
33 005312 013737 000040 001326  MOV     @#40,XXDP        ;;GET DEVICE INDICATOR AND NUMBER
34 005320 122737 000007 001326  CMPB    #7,XXDP         ;;IS IT A VALID NUMBER ?
35 005326 103002      BHIS    1$              ;;YES
36 005330 105037 001326      CLRB    XXDP            ;;NO, DEFAULT TO DRIVE 0
37 005334 005737 000042      TST     @#42           ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
38 005340 001425      BEQ     3$              ;;BR IF NEITHER
39 005342 104401 005350      TYPE    ,74$           ;;TYPE ASCIZ STRING
005346 000412      BR      73$            ;;GET OVER THE ASCIZ
      ;;74$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
      73$:
40 005374 005046      CLR     -(SP)           ;;CLEAR WORD ON STACK
41 005376 113716 001326  MOVB    XXDP,(SP)       ;;GET DRIVE ADDRESS
42 005402 104403      TYPOS                    ;;TYPE THE ADDRESS
43 005404 001      .BYTE  1              ;;ONLY 1 CHARACTER

```

```

44 005405      000
45 005406 104401 001217      .BYTE 0          ;SUPRESS LEADING ZEROS
46 005412 000517      TYPE 5$CRLF      ;CR-LF
47                                     BR 5$          ;GET NUMBER OF DRIVES
48 005414 005227 177777      3$: INC #-1      ;FIRST TIME THRU HERE ?
49 005420 001114      BNE 5$          ;NO
50 005422 104401 005430      TYPE 76$        ;TYPE ASCIZ STRING
    005426 000410      BR 75$          ;GET OVER THE ASCIZ
    ;:76$: .ASCIZ <CRLF>/TO TEST DRIVE /
    ;:75$:
51 005450 005046      CLR -(SP)       ;CLEAR WORD ON STACK
52 005452 113716 001326      MOVB XXDP,(SP) ;GET DRIVE ADDRESS
53 005456 104403      TYPOS          ;TYPE DRIVE ADDRESS
54 005460      001      .BYTE 1          ;ONLY 1 CHARACTER
55 005461      000      .BYTE 0          ;SUPRESS LEADING ZEROS
56 005462 104401 005470      TYPE 78$        ;TYPE ASCIZ STRING
    005466 000431      BR 77$          ;GET OVER THE ASCIZ
    ;:78$: .ASCIZ /, HALT PROGRAM, REMOVE RRD P PACK AND REPLACE IT/<CRLF>
    ;:77$:
57 005552 104401 005560      TYPE 79$        ;TYPE ASCIZ STRING
    005556 000435      BR 5$          ;GET OVER THE ASCIZ
    ;:79$: .ASCIZ /WITH A WORK PACK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
    ;:5$:
61 005652
62 005652 105737 001150      ;CHECK FOR AUTO MODE OR STANDALONE
63 005656 001506      TSTB $AUTOB    ;RUNNING IN AUTO MODE ?
64 005660 012737 000377 001300 BEQ STANDALONE ;BR IF NO
65                                     MOV #377,$DEV   ;SET DEVICE MAP FOR ALL DRIVES
66
67 005666
68 005666 132737 000200 001243 ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
69 005674 001066      XSIZ: BITB #BIT7,$ENVM ;SIZING ALLOWED ?
70                                     BNE 7$          ;NO
71 005676 005001      CLR R1          ;START FROM DRIVE 0
72 005700 013700 001276      MOV $BASE,R0   ;LOAD THE BASE ADDRESS
73
74 005704 136137 063052 001300 1$: BITB ATNTBL(R1),$DEV ;IS DEVICE PRESENT IN MAP ?
75 005712 001453      BEQ 6$          ;BR IF NO
76 005714 012737 062675 006040 MOV #LODEV,5$  ;GET ADDRESS OF LOAD DEVICE MESSAGE
77 005722 005737 001326      TST XXDP       ;LOADED FROM RM05/3/2 ?
78 005726 001403      BEQ 2$          ;NO
79 005730 123701 001326      CMPB XXDP,R1   ;IS THIS THE DRIVE ?
80 005734 001426      BEQ 3$          ;YES, TRY NEXT DRIVE
81
82 005736 012760 000040 000010 2$: MOV #CLR, RMCS2(R0) ;CLEAR MASS BUS
83 005744 010160 000010      MOV R1, RMCS2(R0) ;LOAD THE DRIVE ADDRESS
84 005750 005760 000012      TST RMD5(R0)    ;TRY TO ACCESS AN RM DRIVE REGISTER
85 005754 012737 062715 006040 MOV #NOTPRS,5$  ;GET ADDRESS OF NOT PRESENT MESSAGE
86 005762 032760 010000 000010 BIT #NED, RMCS2(R0) ;IS DRIVE PRESENT ?
87 005770 001010      BNE 3$          ;BR IF NO
88 005772 012737 062732 006040 MOV #NOTAVL,5$  ;GET ADDRESS OF AVAILABLE MESSAGE
89 006000 032760 004000 000000 BIT #DVA, RMCS1(R0) ;IS DRIVE AVAILABLE ?
90 006006 001401      BEQ 3$          ;BR IF NO
91 006010 000414      BR 6$
92
93 006012 146137 063052 001300 3$: BICB ATNTBL(R1),$DEV ;CLEAR DEVICE FROM BIT MAP
94 006020 104401 001217      4$: TYPE 4$CRLF ;CR-LF

```



```
95 006024 104401 062667          TYPE      ,MSGDRV      ;TYPE 'DRIVE'  
96 006030 010146          MOV        R1,-(SP)    ;:SAVE R1 FOR TYPEOUT  
   006032 104403          TYPOS      ;:GO TYPE--OCTAL ASCII  
   006034   002          .BYTE     2          ;:TYPE 2 DIGIT(S)  
   006035   000          .BYTE     0          ;:SUPPRESS LEADING ZEROS  
97 006036 104401          TYPE      ;:TYPE ERROR MESSAGE  
98 006040 000000          .WORD     0          ;:ADDRESS OF MESSAGE GOES HERE  
99                               5$:  
100 006042 005201          INC        R1          ;INCREMENT THE DRIVE ADDRESS  
101 006044 020127 000007          CMP        R1,#7      ;ALL DRIVES ARE CHECKED ?  
102 006050 003715          BLE        1$          ;BRANCH IF NOT  
103                               6$:  
104 006052 104401 001217          TYPE      ,SCLRF      ;CR-LF  
105 006056 005004          CLR        R4          ;THESE TWO LOOPS ARE ADDED TO  
106 006060 005304          DEC        R4          ;WAIT FOR TTY TO FINISH TYPING.  
107 006062 001376          BNE        .-2  
108 006064 005304          DEC        R4  
109 006066 001376          BNE        .-2  
110  
111 006070 000137 007000          JMP        CMNSTART    ;JUMP TO COMMON START  
112
```

```

1      .SBTTL  STANDALONE INPUT ROUTINES
2
3      STANDALONE:
4      006074 004737 060050      JSR      PC,$TKINT      ;INITIALIZE CONSOLE
5
6      006100 005227 177777      INC      #-1            ;FIRST TIME THRU HERE ?
7      006104 001426              BEQ      3$            ;YES !!
8
9      ;SEE IF THE USER WANTS TO KEEP SAME DEVICES FOR TESTING
10     006106 104401 062234      1$:      TYPE      ,CNSL00      ;MAINTAIN PREVIOUS PARAMETERS ?
11     006106 104401 062234      RDCHR     ;GET RESPONSE
12     006112 104411              MOV      (SP)+,$TMP1    ;ECHO RESPONSE
13     006114 012637 001176      TYPE      $TMP1
14     006120 104401 001176      CMPB     $TMP1,#'Y      ;YES RESPONSE ?
15     006124 123727 001176 000131 BNE      2$            ;NO!!
16     006132 001004              TYPE      ,SCRLF       ;CR-LF
17     006134 104401 001217      JMP      CMNSTART      ;KEEP PREVIOUS PARAMETERS
18     006140 000137 007000
19
20     006144 123727 001176 000116 2$:      CMPB     $TMP1,#'N      ;NO RESPONSE ?
21     006152 001427              BEQ      5$            ;YES, GET NEW PARAMETERS
22     006154 104401 062631      TYPE      ,CNSL08      ;NO, TYPE ' ILLEGAL INPUT '
23     006160 000752              BR       1$            ;TRY AGAIN
24
25     ;SEE IF OPERATOR WANTS HELP TEXT
26     006162 104401 062150      3$:      TYPE      ,MSHELP      ;WANT HELP ?
27     006162 104411              RDCHR     ;GET RESPONSE
28     006166 104411              MOV      (SP)+,$TMP1    ;SAVE AND ECHO RESPONSE
29     006170 012637 001176      TYPE      $TMP1
30     006174 104401 001176      CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ?
31     006200 123727 001176 000131 BEQ      4$            ;YES
32     006206 001407              CMPB     $TMP1,#'N      ;WAS IT A NO RESPONSE ?
33     006210 123727 001176 000116 BEQ      5$            ;YES
34     006216 001405              TYPE      ,CNSL08      ;NO, TYPE ' ILLEGAL INPUT '
35     006220 104401 062631      BR       3$            ;TRY AGAIN
36     006224 000756              TYPE      ,HELP        ;YES - TYPE HELP TEXT
37     006226 104401 103712      4$:
38
39     ;SEE IF USER WANTS TO CHANGE UNIBUS ADDRESS
40     006232 104401 001217      5$:      TYPE      ,SCRLF       ;CR-LF
41     006232 104401 001217      TYPE      ,UBUSQST     ;WANT TO CHANGE ADDRESS ?
42     006236 104401 062201      RDCHR     ;GET RESPONSE
43     006242 104411              MOV      (SP)+,$TMP1    ;SAVE AND ECHO RESPONSE
44     006244 012637 001176      TYPE      $TMP1
45     006250 104401 001176      CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ?
46     006254 123727 001176 000131 BEQ      6$            ;YES
47     006262 001407              CMPB     $TMP1,#'N      ;WAS IT A NO RESPONSE ?
48     006264 123727 001176 000116 BEQ      12$           ;YES
49     006272 001525              TYPE      ,CNSL08      ;NO, TYPE ' ILLEGAL INPUT '
50     006274 104401 062631      BR       5$+4         ;TRY AGAIN
51     006300 000756
52
53     ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
54     006302 104401 062267      6$:      TYPE      ,CNSL01      ;TYPE CURRENT BUS ADDRESS
55     006302 104401 062267      MOV      $BASE,-(SP)    ;SAVE $BASE FOR TYPEOUT
56     006306 013746 001276      TYPOC     ;GO TYPE--OCTAL ASCII(ALL DIGITS)
57     006312 104402

```



```

57 006314 104401 062141          TYPE      ,QUES      ;TYPE " ? "
58 006320 104413          RDOCT      ;GET NEW BUS ADDRESS
59 006322 012637 001176          MOV      (SP)+,$TMP1 ;CARRIAGE RETURN ?
60 006326 001412          BEQ      8$        ;YES-SKIP TO NEXT ENTRY
61 006330 022737 160000 001176          CMP      #160000,$TMP1 ;BASE ADDRESS IN I/O PAGE ?
62 006336 101403          BLOS      7$        ;YES
63 006340 104401 062305          TYPE      ,CNSL02   ;TYPE WARNING MESSAGE
64 006344 000760          BR       6$+4      ;TRY AGAIN
65 006346 013737 001176 001276 7$:      MOV      $TMP1,$BASE ;STORE NEW BUS ADDRESS
66
67 006354 104401 062347          8$:      TYPE      ,CNSL03
68 006360 005046          CLR      -(SP)
69 006362 113716 001272          MOVB     $VECT1,(SP) ;GET CURRENT VECTOR ADDRESS
70 006366 104403          TYPOS
71 006370          .BYTE    3        ;TYPE 3 DIGITS
72 006371          .BYTE    0        ;SUPPRESS LEADING ZEROS
73 006372 104401 062141          TYPE      ,QUES
74 006376 104413          RDOCT      ;GET NEW VECTOR ADDRESS
75 006400 012637 001176          MOV      (SP)+,$TMP1 ;CARRIAGE RETURN?
76 006404 001412          BEQ      10$       ;YES-SKIP TO NEXT ENTRY
77 006406 022737 001000 001176          CMP      #1000,$TMP1 ;VECTOR ADDRESS < 1000 ?
78 006414 101003          BHI      9$        ;YES!!
79 006416 104401 062367          TYPE      ,CNSL04   ;TYPE WARNING MESSAGE
80 006422 000754          BR       8$        ;RETRY
81 006424 113737 001176 001272 9$:      MOVB     $TMP1,$VECT1 ;STORE NEW VECTOR ADDRESS
82
83 006432 104401 062423          10$:     TYPE      ,CNSL05
84 006436 005046          CLR      -(SP)
85 006440 113716 001273          MOVB     $VECT1+1,(SP) ;GET CURRENT BR LEVEL
86 006444 006216          ASR      (SP)
87 006446 006216          ASR      (SP)
88 006450 006216          ASR      (SP)
89 006452 006216          ASR      (SP)
90 006454 006216          ASR      (SP)
91 006456 104403          TYPOS
92 006460          .BYTE    1        ;ONLY 1 DIGIT
93 006461          .BYTE    0        ;SUPPRESS LEADING ZEROS
94 006462 104401 062141          TYPE      ,QUES
95 006466 104413          RDOCT      ;GET NEW PRIORITY
96 006470 012637 001176          MOV      (SP)+,$TMP1 ;CARRIAGE RETURN ?
97 006474 001424          BEQ      12$       ;YES-SKIP TO NEXT ENTRY
98 006476 023727 001176 000007          CMP      $TMP1,#7   ;LEGAL PRIORITY ?
99 006504 002403          BLT      11$       ;YES!!
100 006506 104401 062435          TYPE      ,CNSL06   ;TYPE WARNING MESSAGE
101 006512 000747          BR       10$       ;TRY AGAIN
102 006514 006337 001176          11$:     ASL      $TMP1
103 006520 006337 001176          ASL      $TMP1
104 006524 006337 001176          ASL      $TMP1
105 006530 006337 001176          ASL      $TMP1
106 006534 006337 001176          ASL      $TMP1
107 006540 113737 001176 001273          MOVB     $TMP1,$VECT1+1
108
109          ;DIALOGUE TO INPUT DEVICE NUMBERS
110 006546          12$:
111 006546 005227 177777          INC      #-1
112 006552 001002          BNE     13$
113 006554 104401 062466          TYPE      ,CNSL07   ;TYPE INPUT INSTRUCTIONS
    
```

```

114 006560 104401 001217      13$:  TYPE      , $CRLF      : CR-LF
115 006564 005037 001300      14$:  CLR       $DEVMM    : CLEAR DEVICE MAP
116 006570 104401 062653      TYPE      ,MSDRVS     : TYPE 'DRIVE(S): '
117 006574 104411      RDCHR
118 006576 012637 001176      MOV      (SP)+,$TMP1  : GET RESPONSE
119 006602 023727 001176 000101      CMP      $TMP1,#'A    : IS INPUT 'A' ?
120 006610 001007      BNE      15$         : NO
121 006612 104401 062134      TYPE      ,ALL        : YES, TYPE 'ALL' AND GO
122 006616 012737 000377 001300      MOV      #377,$DEVMM  : SET DEVICE MAP FOR ALL DRIVES
123 006624 000137 005666      JMP      XSIZ        : AUTO SIZE.
124
125 006630 023727 001176 000015 15$:  CMP      $TMP1,#CR    : CARRIAGE RETURN ?
126 006636 001436      BEQ      17$         : YES
127 006640 104401 001176      TYPE      , $TMP1     : ECHO RESPONSE
128 006644 023727 001176 000060      CMP      $TMP1,#'0    : NUMBER < 0 ?
129 006652 002430      BLT     17$         : YES
130 006654 023727 001176 000067      CMP      $TMP1,#'7    : NUMBER > 7 ?
131 006662 003427      BLE     18$         : NO
132 006664 000423      BR       17$         : ILLEGAL INPUT
133
134 006666 104411      16$:  RDCHR
135 006670 012637 001176      MOV      (SP)+,$TMP1  : GET RESPONSE
136 006674 023727 001176 000015      CMP      $TMP1,#CR    : CARRIAGE RETURN ?
137 006702 001432      BEQ     19$         : YES
138 006704 104401 062145      TYPE      ,COMMA      : TYPE ' , '
139 006710 104401 001176      TYPE      , $TMP1     : ECHO RESPONSE
140 006714 023727 001176 000060      CMP      $TMP1,#'0    : NUMBER < 0 ?
141 006722 002404      BLT     17$         : YES
142 006724 023727 001176 000067      CMP      $TMP1,#'7    : NUMBER > 7 ?
143 006732 003403      BLE     18$         : NO
144 006734 104401 062631      17$:  TYPE      ,CNSL08    : TYPE '' ?ILLEGAL INPUT''
145 006740 000711      BR       14$         : RETRY
146
147 006742 013701 001176      18$:  MOV      $TMP1,R1     : R1 = DRIVE NUMBER
148 006746 042701 177770      BIC     #'C7,R1
149 006752 156137 063052 001300      BISB    ATNTBL(R1),$DEVMM : SET DEVICE IN MAP
150 006760 122737 000377 001300      CMPB    #377,$DEVMM  : DONE ?
151 006766 101337      BHI     16$         : NO
152 006770 104401 001217      19$:  TYPE      , $CRLF     : CR-LF
153 006774 000137 005666      JMP     XSIZ        : GO SIZE DEVICES
154

```



```

007226
41 007226 117737 172230 001234 64$:   MOVB   @TSTQUE,$UNIT   ;LOAD UNIT NUMBER
42
43
44   ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND DETERMINE THE LAST TRACK
   ;OF THE DIFFERENT DRIVE TYPES
45 007234 012737 002000 001330   MOV    #TA4,LSTRK     ;ASSUME LAST TRACK FOR RM02/3 = 4.
46 007242 013700 001276           MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
47 007246 012760 000040 000010   MOV    #CLR,RMCS2(R0) ;CLEAR MASSBUS
48 007254 117760 172202 000010   MOVB   @TSTQUE,RMCS2(R0) ;SELECT DEVICE UNDER TEST
49 007262 016002 000026           MOV    RMDT(R0),R2    ;GET RMDT AND
50 007266 042702 177770           BIC    #177770,R2     ;SAVE DRIVE TYPE BITS
51 007272 022702 000007           CMP    #7,R2         ;IS IT AN RM05 ?
52 007276 001003                   BNE    3$            ;NO, MUST BE AN RM02 OR RM03
53 007300 012737 011000 001330   MOV    #TA16!TA2,LSTRK ;YES--SET LAST TRACK = 18.
54 007306
55   3$:
```


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

.SBTTL REGISTER AND STORAGE USAGE

:REGISTER ASSIGNMENTS

:R0 = UNIBUS ADDRESS OF RH CONTROLLER
:R1 = ADDRESS OF ENTRY IN TEST QUE CORRESPONDING TO THE
: UNIT UNDER TEST
:R2,R3 = WORKING REGISTERS FOR TEST IN PROGRESS, MUST BE
: SAVED BY SUBROUTINES
:R4,R5 = GENERAL WORKING REGISTERS, ARE NOT SAVED BY
: SUBROUTINES
:R6 = STACK POINTER
:R7 = LINKAGE REGISTER TO SUBROUTINES

:STORAGE ASSIGNMENTS

:\$TMP0-\$TMP4 TEMPORARY STORAGE, NOT SAVED BY SUBROUTINES
:\$GDDAT,\$BDDAT EXPECTED AND RECEIVED STATUS FOR ERROR TYPEOUT
:\$GDADR,\$BDADR ADDRESS OF EXPECTED AND RECEIVED STATUS IF APPLICABLE,
: ALSO THE ADDRESS OF A REGISTER ERROR
: :
:\$STN = TEST NUMBER
:\$UNIT = NUMBER OF DEVICE BEING TESTED
:\$RGINBF = THE REGISTER INPUT BUFFER HAS A STORAGE LOCATION FOR
: EACH REGISTER, AND IS USED WHEN READING STATUS AND
: CONTROL DATA
:\$RGOTBF = THE REGISTER OUTPUT BUFFER HAS A STORAGE LOCATION FOR
: EACH REGISTER, AND IS USED FOR ASSEMBLING DATA TO BE
: WRITTEN IN REGISTERS
: :
: :

```

1          ;:*****
          ;*TEST 1      TRANSFER TEST
          ;:*****
          TST1:
          SCOPE                ;SCOPE CALL
          NOP
          MOV #STACK,SP        ;LOAD THE STACK POINTER
          MOV $BASE,R0         ;R0 = UNIBUS ADDRESS
          MOV TSTQUE,R1        ;R1 = POINTER TO DEVICE
          MOV #1,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX
2
3 007334 012702 000000        MOV #0,R2                ;R2 = REGISTER INDEX
4
5          ;CLEAR THE MASSBUS AND VERIFY THAT NONEXISTANT DEVICE ERROR IS RESET
6 10$:
7 007340 004737 055614        JSR PC,CNTCLR           ;GO CLEAR CONTROLLER
8 007344 016037 000010 001142 MOV RMCS2(R0),$BDDAT    ;STORE RMCS2 AT $BDDAT
9 007352 032737 010000 001142 BIT #NED,$BDDAT
10 007360 001417                BEQ 20$
11 007362 111137 001140        MOVB (R1),$GDDAT
12 007366 042737 177770 001140 BIC #^CUNTMSK,$GDDAT
13 007374 052737 000100 001140 BIS #IR,$GDDAT
14 007402 010037 001136        MOV R0,$BDADR
15 007406 062737 000010 001136 ADD #RMCS2,$BDADR
16 007414 104001                EMT 1
17 007416 000475                BR 60$
18
19          ;READ THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE READ
20          ;DOES NOT SET 'NED' ERROR
21 20$:
22 007420 010003                MOV R0,R3                ;R3 = REGISTER ADDRESS
23 007422 060203                ADD R2,R3
24 007424 011304                MOV (R3),R4              ;READ REGISTER
25 007426 032760 010000 000010 BIT #NED,RMCS2(R0)      ;IS 'NED' SET??
26 007434 001470                BEQ 70$                  ;NO!!
27
28 007436 004737 055614        JSR PC,CNTCLR           ;GO CLEAR CONTROLLER
29 007442 016037 000010 001142 MOV RMCS2(R0),$BDDAT    ;STORE RMCS2 AT $BDDAT
30 007450 032737 010000 001142 BIT #NED,$BDDAT
31 007456 001417                BEQ 30$
32 007460 111137 001140        MOVB (R1),$GDDAT
33 007464 042737 177770 001140 BIC #^CUNTMSK,$GDDAT
34 007472 052737 000100 001140 BIS #IR,$GDDAT
35 007500 010037 001136        MOV R0,$BDADR
36 007504 062737 000010 001136 ADD #RMCS2,$BDADR
37 007512 104001                EMT 1
38 007514 000436                BR 60$
39
40          ;WRITE THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE WRITE
41          ;DOES NOT SET 'NED' ERROR
42 30$:
43 007516 012713 000000        MOV #0,(R3)              ;WRITE REGISTER
44 007522 032760 010000 000010 BIT #NED,RMCS2(R0)      ;IS 'NED' SET??
45 007530 001432                BEQ 70$                  ;NO!!
46
47          ;COULD NOT READ OR WRITE THE REGISTER WITHOUT SETING 'NED' ERROR -

```



```

48 ;ADVANCE THE REGISTER INDEX AND REPEAT THE TEST FOR THE NEXT
49 ;AVAILABLE DEVICE REGISTER
50 007532 40$:
51 007532 062702 000002 ADD #2,R2 ;ADVANCE TO NEXT REGISTER
52 007536 022702 000002 CMP #RMWC,R2 ;IS THIS RMWC??
53 007542 001773 BEQ 40$ ;YES - TRY NEXT REGISTER
54 007544 022702 000004 CMP #RMBA,R2 ;IS THIS RMBA??
55 007550 001770 BEQ 40$ ;YES - TRY NEXT REGISTER
56 007552 022702 000010 CMP #RMCS2,R2 ;IS THIS RMCS5??
57 007556 001765 BEQ 40$ ;YES - TRY ANOTHER REGISTER
58 007560 022702 000016 CMP #RMAS,R2 ;IS THIS RMAS ??
59 007564 001762 BEQ 40$ ;YES - TRY ANOTHER REGISTER
60 007566 022702 000022 CMP #RMDB,R2 ;IS THIS RMDB??
61 007572 001757 BEQ 40$ ;YES - TRY ANOTHER REGISTER
62 007574 022702 000046 CMP #RMEC2,R2 ;IS THIS A LEGAL REGISTER
63 007600 103257 BHIS 10$ ;YES - TRY THIS REGISTER
64
65 ;GOT 'NONEXISTENT DEVICE' ERROR FOR EVERY REMOTE REGISTER ADDRESS
66 007602 50$:
67 007602 013737 001276 001136 MOV $BASE,$BDADR ;STORE BASE ADDRESS
68 007610 104002 EMT 2
69 007612 000137 053600 60$: JMP $EOSP ;GO SELECT NEXT DEVICE
70
71 007616 70$:
72
73 ;*****
; *TEST 2 CTOD TEST
;*****
007616 TST2:
007616 000004 SCOPE ;SCOPE CALL
007620 000240 NOP
007622 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
007626 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
007632 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
007636 012737 000002 001226 MOV #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
74
75 007644 004737 055614 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
76
77 ;WRITE ONES IN REMOTE REGISTERS
78 007650 012760 000076 000000 MOV #ILF76,RMCS1(R0) ;LOAD RMCS1
79 007656 012760 177777 000006 MOV #-1,RMDA(R0) ;LOAD RMDA
80 007664 012760 001777 000034 MOV #CYLMSK,RMDC(R0) ;LOAD RMDC
81 007672 012760 016200 000032 MOV #^CXNUOF,RMOF(R0) ;LOAD RMOF
82
83 ;READ REMOTE REGISTERS TWICE
84 007700 012702 000001 MOV #1,R2
85 007704 10$:
007704 016037 000000 001332 MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
86 007712 016037 000006 001340 MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
87 007720 016037 000034 001366 MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
88 007726 016037 000032 001364 MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
89 007734 005302 DEC R2
90 007736 100362 BPL 10$
91
92 ;SEE IF ANY ONE BITS CAME BACK
93 007740 042737 177701 001332 BIC #^CILF76,RMCS1I ;IS RMCS1 0??

```

94 007746 001014
 95 007750 005737 001340
 96 007754 001011
 97 007756 042737 176000 001366
 98 007764 001005
 99 007766 042737 161577 001364
 100 007774 001001
 101
 102
 103 007776 104003
 104 010000
 105
 106

BNE 20\$;NO!!
 TST RMDAI ;IS RMDA 0??
 BNE 20\$;NO!!
 BIC #XNUDC,RMDCI ;IS RMDC 0??
 BNE 20\$;NO!!
 BIC #XNUOF,RMOFI ;IS RMOF 0 ??
 BNE 20\$;NO!!

;CANNOT READ/WRITE ANY ONE FROM REMOTE REGISTER
 EMT 3
 20\$:

::*****
 ;*TEST 3 MASSBUS INITIALIZE TEST

::*****
 ;TST3:

010000
 010000 000004
 010002 000240
 010004 012706 001100
 010010 013700 001276
 010014 013701 001462
 010020 012737 000003 001226

SCOPE ;SCOPE CALL
 NOP
 MOV #STACK,SP ;LOAD THE STACK POINTER
 MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
 MOV #3,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

107
 108 010026 004737 055614
 109
 110
 111 010032 012760 000076 000000
 112 010040 012760 177777 000014
 113 010046 012760 177777 000042
 114
 115

JSR PC,CNTCLR ;GO CLEAR CONTROLLER
 ;WRITE ONES IN SELECTED REGISTERS
 MOV #ILF76,RMCS1(R0) ;LOAD RMCS1
 MOV #-1,RMER1(R0) ;LOAD RMER1
 MOV #-1,RMER2(R0) ;LOAD RMER2

116 010054 004737 055614
 117
 118
 119 010060 016037 000000 001332
 120 010066 016037 000014 001346
 121 010074 016037 000042 001374
 122

;INITIALIZE MASSBUS WITH A CLEAR
 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
 ;READ THE REGISTERS THAT WERE WRITTEN
 MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
 MOV RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
 MOV RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER

123
 124 010102 052737 177701 001332
 125 010110 052737 001567 001374
 126 010116 022737 177777 001332
 127 010124 001011
 128 010126 022737 177777 001346
 129 010134 001005
 130 010136 022737 177777 001374
 131 010144 001001
 132

;SEE IF ANY REGISTER BITS WERE CLEARED
 BIS #^CILF76,RMCS1I ;SET ANY BIT NOT WRITTEN
 BIS #XNUER2,RMER2I
 CMP #-1,RMCS1I ;ANY ZEROS IN RMCS1??
 BNE 10\$;YES!!
 CMP #-1,RMER1I ;ANY ZEROS IN RMER1??
 BNE 10\$;YES!!
 CMP #-1,RMER2I ;ANY ZEROS IN RMER2??
 BNE 10\$

133
 134 010146 104004
 135 010150
 136
 137

;NONE OF THE BITS WERE CLEARED
 EMT 4
 10\$:

::*****
 ;*TEST 4 CLEAR STUCK ACTIVE TEST

::*****


```
010150          TST4:
010150 000004          SCOPE          ;SCOPE CALL
010152 000240          NOP
010154 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
010160 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
010164 013701 001462  MOV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
010170 012737 000004 001226  MOV      #4,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
138
139 010176 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
140
141          ;WRITE ONES IN TEST REGISTERS
142 010202 012760 177777 000014  MOV      #-1,RMER1(R0)  ;LOAD RMER1
143 010210 012760 177777 000042  MOV      #-1,RMER2(R0)  ;LOAD RMER2
144 010216 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
145
146          ;READ TEST REGISTERS AND SEE IF ANY BITS ARE ON
147 010224 016037 000014 001346  MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
148 010232 016037 000042 001374  MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
149 010240 016037 000024 001356  MOV      RMMR1(R0),RMMR1I ;STORE RMMR1 IN INPUT BUFFER
150 010246 042737 040000 001346  BIC      #UNS,RMER1I    ;DONT ACCEPT UNSAFE
151 010254 001011          BNE      10$           ;BRANCH IF ANY OTHER BITS ON
152 010256 042737 040200 001374  BIC      #SKI!DVC,RMER2I ;DONT ACCEPT SKI OR DVC
153 010264 001005          BNE      10$           ;BRANCH IF ANY OTHER BITS ON
154 010266 032737 000001 001356  BIT      #DMD,RMMR1I    ;BRANCH IF DMD IS ON
155 010274 001001          BNE      10$
156 010276 104026          EMT      26
157 010300
158
159
```

*TEST 5 TRISTATE TRANSFER TEST

```
010300          TST5:
010300 000004          SCOPE          ;SCOPE CALL
010302 000240          NOP
010304 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
010310 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
010314 013701 001462  MOV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
010320 012737 000005 001226  MOV      #5,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
160
161 010326 005002          CLR      R2             ;CLEAR ERROR FLAGS
162 010330 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
163
164          ;WRITE ONES IN SELECTED REGISTERS
165 010334 012760 000076 000000  MOV      #ILF76,RMCS1(R0) ;LOAD RMCS1
166 010342 012760 177777 000006  MOV      #-1,RMDA(R0)    ;LOAD RMDA
167 010350 012760 177777 000014  MOV      #-1,RMER1(R0)  ;LOAD RMER1
168 010356 012760 177777 000032  MOV      #-1,RMOF(R0)   ;LOAD RMOF
169 010364 012760 177777 000042  MOV      #-1,RMER2(R0)  ;LOAD RMER2
170
171          ;WRITE ZEROS IN SELECTED REGISTERS
172 010372 012760 000000 000000  MOV      #0,RMCS1(R0)   ;LOAD RMCS1
173 010400 012760 000000 000006  MOV      #0,RMDA(R0)   ;LOAD RMDA
174 010406 012760 000000 000014  MOV      #0,RMER1(R0)  ;LOAD RMER1
175 010414 012760 000000 000032  MOV      #0,RMOF(R0)   ;LOAD RMOF
176 010422 012760 000000 000034  MOV      #0,RMDC(R0)   ;LOAD RMDC
177 010430 012760 000000 000042  MOV      #0,RMER2(R0)  ;LOAD RMER2
```

```
178
179
180 010436 016037 000000 001332 ;READ BACK ALL REGISTERS
181 010444 016037 000006 001340     MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
182 010452 016037 000014 001346     MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
183 010460 016037 000032 001364     MOV RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
184 010466 016037 000034 001366     MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
185 010474 016037 000042 001374     MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
186                                     MOV RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
187
188 010502 012702 177777 ;CHECK EACH REGISTER CONTENT FOR ZERO BITS WRITTEN & READ
189 010506 052737 177701 001332     MOV #-1,R2 ;ACCUMULATE ZEROS IN R2
190 010514 052737 161577 001364     BIS #^CILF76,RMCS1I ;SET ALL BITS NOT WRITTEN
191 010522 052737 176000 001366     BIS #XNUOF,RMOFI
192 010530 052737 001567 001374     BIS #XNUDC,RMDCI
193 010536 005137 001332     COM RMCS1I ;COMPLEMENT REGISTER CONTENTS
194 010542 005137 001340     COM RMDAI
195 010546 005137 001346     COM RMER1I
196 010552 005137 001364     COM RMOFI
197 010556 005137 001366     COM RMDCI
198 010562 005137 001374     COM RMER2I
199 010566 043702 001332     BIC RMCS1I,R2 ;ACCUMULATE ALL ZERO BITS
200 010572 043702 001340     BIC RMDAI,R2
201 010576 043702 001346     BIC RMER1I,R2
202 010602 043702 001364     BIC RMOFI,R2
203 010606 043702 001366     BIC RMDCI,R2
204 010612 043702 001374     BIC RMER2I,R2
205 010616 001407     BEQ 10$ ;BRANCH IF EACH BIT IS ZERO
206
207 ;ONE OR MORE BIT POSITIONS ARE NOT ZERO
208 010620 010237 001142     MOV R2,$BDDAT ;SAVE RESULT FOR TYPE
209 010624 005037 001140     CLR $GDDAT ;LOAD EXPECTED RESULT
210 010630 104005     EMT 5
211 010632 052702 000001     BIS #BIT0,R2 ;SET ERROR FLAG
212 010636 10$:
213 010636 004737 055614     JSR PC,CNTCLR ;GO CLEAR CONTROLLER
214
215 ;PRESET SELECTED REGISTERS TO ZEROS
216 010642 012760 000000 000006 ;(ASSUME RMCS1, RMER1, RMER2 WERE CLEARED BY INIT)
217 010650 012760 000000 000032     MOV #0,RMDA(R0) ;LOAD RMDA
218 010656 012760 000000 000034     MOV #0,RMOF(R0) ;LOAD RMOF
219                                     MOV #0,RMDC(R0) ;LOAD RMDC
220
221 010664 012760 000076 000000 ;WRITE ONES IN SELECTED REGISTERS
222 010672 012760 177777 000006     MOV #ILF76,RMCS1(R0) ;LOAD RMCS1
223 010700 012760 016200 000032     MOV #-1,RMDA(R0) ;LOAD RMDA
224 010706 012760 001777 000034     MOV #^CXNUOF,RMOF(R0) ;LOAD RMOF
225 010714 012760 177777 000014     MOV #^CXNUDC,RMDC(R0) ;LOAD RMDC
226 010722 012760 176210 000042     MOV #-1,RMER1(R0) ;LOAD RMER1
227                                     MOV #^CXNUER2,RMER2(R0) ;LOAD RMER2
228
229 010730 016037 000000 001332 ;READ ALL REGISTERS
230 010736 016037 000006 001340     MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
231 010744 016037 000032 001364     MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
232 010752 016037 000034 001366     MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
233 010760 016037 000014 001346     MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
234                                     MOV RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
```



```

234 010766 016037 000042 001374      MOV      RMER2(R0),RMER2I      ;STORE RMER2 IN INPUT BUFFER
235
236                                     ;CHECK EACH REGISTER CONTENT FOR ONE BITS WRITTEN & READ
237 010774 042737 177701 001332      BIC      #^CILF76,RMCS1I      ;CLEAR ALL BITS NOT WRITTEN
238 011002 042737 161577 001364      BIC      #XNUOF,RMOFI
239 011010 042737 176000 001366      BIC      #XNUDC,RMDCI
240 011016 042737 001567 001374      BIC      #XNUER2,RMER2I
241 011024 005002                                     CLR      R2                    ;ACCUMULATE ONES IN R2
242 011026 053702 001332      BIS      RMCS1I,R2            ;ACCUMULATE ALL ONE BITS
243 011032 053702 001340      BIS      RMDAI,R2
244 011036 053702 001364      BIS      RMOFI,R2
245 011042 053702 001366      BIS      RMDCI,R2
246 011046 053702 001346      BIS      RMER1I,R2
247 011052 053702 001374      BIS      RMER2I,R2
248 011056 022702 177777      CMP      #-1,R2              ;SEE IF EACH BIT POSITION WAS ONE
249 011062 001410      BEQ      20$                 ;BRANCH IF NONE STUCK
250
251                                     ;ONE OR MORE BIT POSITIONS ARE NOT ONE
252 011064 010237 001142      MOV      R2,$BDDAT           ;SAVE RESULT FOR TYPE
253 011070 012737 177777 001140      MOV      #-1,$GDDAT         ;EXPECTED RESULT
254 011076 104006                                     EMT      6
255 011100 052702 000002      BIS      #BIT1,R2           ;SET ERROR FLAG
256 011104                                     20$:
257 011104 005702      TST      R2                  ;ANY ERRORS DETECTED ??
258 011106 001126      BNE      30$                 ;YES - DONT DO BIT TEST
259 011110 012702 000001      MOV      #1,R2              ;R2=BIT POSITION
260 011114                                     25$:
011114 004737 055614      JSR      PC,CNTCLR          ;GO CLEAR CONTROLLER
261
262                                     ;WRITE THE BIT PATTERN IN SELECTED DEVICE REGISTERS
263 011120 010260 000006      MOV      R2,RMDA(R0)        ;LOAD RMDA
264 011124 010260 000032      MOV      R2,RMOF(R0)        ;LOAD RMOF
265 011130 010260 000034      MOV      R2,RMDC(R0)        ;LOAD RMDC
266 011134 010260 000014      MOV      R2,RMER1(R0)       ;LOAD RMER1
267 011140 010260 000042      MOV      R2,RMER2(R0)       ;LOAD RMER2
268
269                                     ;READ BACK THE REGISTERS
270 011144 016037 000006 001340      MOV      RMDA(R0),RMDAI     ;STORE RMDA IN INPUT BUFFER
271 011152 016037 000032 001364      MOV      RMOF(R0),RMOFI     ;STORE RMOF IN INPUT BUFFER
272 011160 016037 000034 001366      MOV      RMDC(R0),RMDCI     ;STORE RMDC IN INPUT BUFFER
273 011166 016037 000014 001346      MOV      RMER1(R0),RMER1I   ;STORE RMER1 IN INPUT BUFFER
274 011174 016037 000042 001374      MOV      RMER2(R0),RMER2I   ;STORE RMER2 IN INPUT BUFFER
275
276                                     ;CHECK REGISTER CONTENTS FOR CORRECT PATTERN
277 011202 005003      CLR      R3                  ;R3=ACCUMULATED ONE BIT
278 011204 012704 177777      MOV      #-1,R4             ;R4=ACCUMULATED ZERO BITS
279 011210 013705 001340      MOV      RMDAI,R5          ;GET ANY GOOD BITS FROM RMDA
280 011214 050503      BIS      R5,R3
281 011216 005105      COM      R5
282 011220 040504      BIC      R5,R4
283 011222 013705 001364      MOV      RMOFI,R5          ;GET GOOD BITS FROM RMOF
284 011226 042705 161577      BIC      #XNUOF,R5
285 011232 050503      BIS      R5,R3
286 011234 005105      COM      R5
287 011236 042705 161577      BIC      #XNUOF,R5
288 011242 040504      BIC      R5,R4
289 011244 013705 001366      MOV      RMDCI,R5          ;GET GOOD BITS FROM RMDC
    
```

```

290 011250 042705 176000      BIC    #XNUDC,R5
291 011254 050503            BIS    R5,R3
292 011256 005105            COM    R5
293 011260 042705 176000      BIC    #XNUDC,R5
294 011264 040504            BIC    R5,R4
295 011266 013705 001346      MOV    RMER1I,R5      ;GET GOOD BITS FROM RMER1
296 011272 050503            BIS    R5,R3
297 011274 005105            COM    R5
298 011276 040504            BIC    R5,R4
299 011300 013705 001374      MOV    RMER2I,R5      ;GET GOOD BITS FROM RMER2
300 011304 042705 001567      BIC    #XNUER2,R5
301 011310 050503            BIS    R5,R3
302 011312 005105            COM    R5
303 011314 042705 001567      BIC    #XNUER2,R5
304 011320 040504            BIC    R5,R4
305 011322 010205            MOV    R2,R5          ;RESET ALL ONES IN R3 EXCEPT
306 011324 005105            COM    R5          ;FOR THE TEST BIT
307 011326 040503            BIC    R5,R3
308 011330 040204            BIC    R2,R4          ;RESET TEST BIT IN R4
309 011332 050403            BIS    R4,R3          ;COMBINE ACCUMULATED 1'S + 0'S
310 011334 020302            CMP    R3,R2          ;IS PATTERN OK??
311 011336 001406            BEQ    26$           ;YES!!
312 011340 010237 001140      MOV    R2,$GDDAT      ;SAVE TEST PATTERN
313 011344 010337 001142      MOV    R3,$BDDAT      ;SAVE RESULT
314 011350 104007
315 011352 000404            EMT    7
316                                BR     30$           ;SKIP TO NEXT
    
```

;ADVANCE R2 TO THE NEXT PATTERN AND REPEAT TEST

```

26$:
    ASL    R2           ;SHIFT THE BIT
    BEQ    30$         ;EXIT IF DONE
    JMP    25$
    
```

30\$:

```

:*****
:*TEST 6      REGISTER SELECT TEST
:*
:*NOTE: REGISTER SELECT 16 IS TESTED BY THE "ILR" TEST
:*
:*****
    
```

TST6:

```

011364
011364 000004      SCOPE          ;SCOPE CALL
011366 000240      NOP
011370 012706 001100  MOV    #STACK,SP  ;LOAD THE STACK POINTER
011374 013700 001276  MOV    $BASE,R0   ;R0 = UNIBUS ADDRESS
011400 013701 001462  MOV    TSTQUE,R1  ;R1 = POINTER TO DEVICE
011404 012737 000006 001226  MOV    #6,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
    
```

;THE FOLLOWING TABLE GIVES MASSBUS REGISTER SELECT VALUES FOR
 ;EACH DEVICE REGISTER

REGISTER NAME	REG SEL (16,8,4,2,1)
RMCS1	00000
RMDS	00001
RMER1	00010

330
331
332
333
334
335
336
337
338
339


```

340      :      RMMR1      00011
341      :      RMAS      0010C
342      :      RMDA      00101
343      :      RMDT      00110
344      :      RMLA      00111
345      :      RMSN      01000
346      :      RMOF      01001
347      :      RMDC      01010
348      :      RMHR      01011
349      :      RMMR2      01100
350      :      RMER2      01101
351      :      RMEC1      01110
352      :      RMEC2      01111
353

```

```

: EACH REGISTER SELECT LINE IS TESTED FOR A STUCK AT ONE,
: STUCK AT ZERO FAULT. AS AN EXAMPLE, TO TEST REG SEL 1.
: FOR S-A-0, RMER1 IS WRITTEN WITH ZEROS. THEN THE REGISTER
: THAT HAS THE SAME SELECT VALUE, EXCEPT FOR THE SELECT LINE
: BEING TESTED, IS WRITTEN WITH ONES. IN THIS EXAMPLE,
: RMMR1 IS WRITTEN WITH ONES. IF SELECT LINE 1 IS S-A-0,
: THE ALL ONES WORD WILL BE WRITTEN IN RMER1, AND RMER1
: WILL NOT BE 0 WHEN READ BACK.

```

```

363 011412 005002
364 011414 012703 177777

```

```

CLR R2 ;R2= ZEROS SOURCE
MOV #-1,R3 ;R3= ONES SOURCE

```

:TEST REG SEL 1 FOR S-A-0

```

366      JSR PC,CNTCLR ;GO CLEAR CONTROLLER
367 011420 004737 055614      MOV R2,RMER1(R0) ;LOAD RMER1
368 011424 010260 000014      MOV R2,RMDC(R0) ;LOAD RMDC
369 011430 010260 000034      MOV R3,RMMR1(R0) ;LOAD RMMR1
370 011434 010360 000024      MOV R3,RMHR(R0) ;LOAD RMHR
371 011440 010360 000036      MOV RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
372 011444 016037 000014 001346      MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
373 011452 016037 000034 001366      CMP R3,RMER1I
374 011460 020337 001346      BNE 10$
375 011464 001007      BIS #XNUDC,RMDCI
376 011466 052737 176000 001366      CMP R3,RMDCI
377 011474 020337 001366      BNE 10$
378 011500 001001      EMT 10
379 011502 104010

```

:TEST REG SEL 1 FOR S-A-1
10\$:

```

382 011504
383 011504 004737 055614      JSR PC,CNTCLR ;GO CLEAR CONTROLLER
384 011510 010260 000006      MOV R2,RMDA(R0) ;LOAD RMDA
385 011514 010260 000032      MOV R2,RMOF(R0) ;LOAD RMOF
386 011520 010260 000042      MOV R2,RMER2(R0) ;LOAD RMER2
387 011524 010360 000016      MOV R3,RMAS(R0) ;LOAD RMAS
388 011530 010360 000030      MOV R3,RMSN(R0) ;LOAD RMSN
389 011534 010360 000040      MOV R3,RMMR2(R0) ;LOAD RMMR2
390 011540 016037 000006 001340      MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
391 011546 016037 000032 001364      MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
392 011554 016037 000042 001374      MOV RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
393 011562 020337 001340      CMP R3,RMDAI
394 011566 001015      BNE 20$
395 011570 052737 161577 001364      BIS #XNUOF,RMOFI
396 011576 020337 001364      CMP R3,RMOFI

```

```

397 011602 001007          BNE      20$
398 011604 052737 001567 001374  BIS      #XNUER2,RMER2I
399 011612 020337 001374      CMP      R3,RMER2I
400 011616 001001          BNE      20$
401 011620 104011          EMT      11
402
403
404 011622          :TEST REG SEL 2 FOR S-A-0
20$:
405 011622 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
406 011626 010260 000006      MOV      R2,RMDA(R0)    ;LOAD RMDA
407 011632 010260 000032      MOV      R2,RMOF(R0)    ;LOAD RMOF
408 011636 010260 000042      MOV      R2,RMER2(R0)   ;LOAD RMER2
409 011642 010360 000020      MOV      R3,RMLA(R0)    ;LOAD RMLA
410 011646 010360 000036      MOV      R3,RMHR(R0)    ;LOAD RMHR
411 011652 010360 000046      MOV      R3,RMEC2(R0)   ;LOAD RMEC2
412 011656 016037 000006 001340  MOV      RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
413 011664 016037 000032 001364  MOV      RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
414 011672 016037 000042 001374  MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
415 011700 020337 001340      CMP      R3,RMDAI
416 011704 001015          BNE      30$
417 011706 052737 161577 001364  BIS      #XNUOF,RMOFI
418 011714 020337 001364      CMP      R3,RMOFI
419 011720 001007          BNE      30$
420 011722 052737 001567 001374  BIS      #XNUER2,RMER2I
421 011730 020337 001374      CMP      R3,RMER2I
422 011734 001001          BNE      30$
423 011736 104012          EMT      12
424
425
426 011740          :TEST REG SEL 2 FOR S-A-1
30$:
427 011740 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
428 011744 010260 000014      MOV      R2,RMER1(R0)   ;LOAD RMER1
429 011750 010260 000034      MOV      R2,RMDC(R0)    ;LOAD RMDC
430 011754 012760 000076 000000  MOV      #ILF76,RMCS1(R0) ;LOAD RMCS1
431 011762 010360 000030      MOV      R3,RMSN(R0)    ;LOAD RMSN
432 011766 016037 000014 001346  MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
433 011774 016037 000034 001366  MOV      RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
434 012002 052737 177701 001346  BIS      #^CILF76,RMER1I
435 012010 020337 001346      CMP      R3,RMER1I
436 012014 001007          BNE      40$
437 012016 052737 176000 001366  BIS      #XNUDC,RMDCI
438 012024 020337 001366      CMP      R3,RMDCI
439 012030 001001          BNE      40$
440 012032 104013          EMT      13
441
442
443 012034          :TEST REG SEL 4 FOR S-A-0
40$:
444 012034 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
445 012040 010260 000014      MOV      R2,RMER1(R0)   ;LOAD RMER1
446 012044 010260 000032      MOV      R2,RMOF(R0)    ;LOAD RMOF
447 012050 010260 000034      MOV      R2,RMDC(R0)    ;LOAD RMDC
448 012054 010360 000026      MOV      R3,RMDT(R0)    ;LOAD RMDT
449 012060 010360 000042      MOV      R3,RMER2(R0)   ;LOAD RMER2
450 012064 010360 000044      MOV      R3,RMEC1(R0)   ;LOAD RMEC1
451 012070 016037 000014 001346  MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
452 012076 016037 000032 001364  MOV      RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
453 012104 016037 000034 001366  MOV      RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
    
```



```

454 012112 020337 001346      CMP      R3,RMER1I
455 012116 001015      BNE      50$
456 012120 052737 161577 001364      BIS      #XNUOF,RMOF I
457 012126 020337 001364      CMP      R3,RMOF I
458 012132 001007      BNE      50$
459 012134 052737 176000 001366      BIS      #XNUDC,RMDC I
460 012142 020337 001366      CMP      R3,RMDC I
461 012146 001001      BNE      50$
462 012150 104014      EMT      14
463
464
465 012152      ;TEST REG SEL 4 FOR S-A-1
466 012152 004737 055614      50$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
467 012156 010260 000006      MOV      R2,RMDA(R0)    ;LOAD RMDA
468 012162 010260 000042      MOV      R2,RMER2(R0)   ;LOAD RMER2
469 012166 010360 000012      MOV      R3,RMDS(R0)    ;LOAD RMDS
470 012172 010360 000032      MOV      R3,RMOF(R0)    ;LOAD RMOF
471 012176 016037 000006 001340      MOV      RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
472 012204 016037 000042 001374      MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
473 012212 020337 001340      CMP      R3,RMDAI
474 012216 001007      BNE      60$
475 012220 052737 001567 001374      BIS      #XNUER2,RMER2I
476 012226 020337 001374      CMP      R3,RMER2I
477 012232 001001      BNE      60$
478 012234 104015      EMT      15
479
480
481 012236      ;TEST REG SEL 8 FOR S-A-0
482 012236 004737 055614      60$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
483 012242 010260 000014      MOV      R2,RMER1(R0)   ;LOAD RMER1
484 012246 010260 000006      MOV      R2,RMDA(R0)    ;LOAD RMDA
485 012252 010360 000034      MOV      R3,RMDC(R0)    ;LOAD RMDC
486 012256 010360 000042      MOV      R3,RMER2(R0)   ;LOAD RMER2
487 012262 016037 000014 001346      MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
488 012270 016037 000006 001340      MOV      RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
489 012276 020337 001346      CMP      R3,RMER1I
490 012302 001004      BNE      70$
491 012304 020337 001340      CMP      R3,RMDAI
492 012310 001001      BNE      70$
493 012312 104016      EMT      16
494
495
496 012314      ;TEST REG SEL 8 FOR S-A-1
497 012314 004737 055614      70$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
498 012320 010260 000032      MOV      R2,RMOF(R0)    ;LOAD RMOF
499 012324 010260 000034      MOV      R2,RMDC(R0)    ;LOAD RMDC
500 012330 010260 000042      MOV      R2,RMER2(R0)   ;LOAD RMER2
501 012334 010360 000012      MOV      R3,RMDS(R0)    ;LOAD RMDS
502 012340 010360 000014      MOV      R3,RMER1(R0)   ;LOAD RMER1
503 012344 010360 000006      MOV      R3,RMDA(R0)    ;LOAD RMDA
504 012350 016037 000032 001364      MOV      RMOF(R0),RMOF I ;STORE RMOF IN INPUT BUFFER
505 012356 016037 000034 001366      MOV      RMDC(R0),RMDC I ;STORE RMDC IN INPUT BUFFER
506 012364 016037 000042 001374      MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
507 012372 052737 161577 001364      BIS      #XNUOF,RMOF I
508 012400 001015      BNE      80$
509 012402 022737 176000 001366      CMP      #XNUDC,RMDC I
510 012410 020337 001366      CMP      R3,RMDC I
    
```

511 012414 001007
 512 012416 052737 001567 001374
 513 012424 020337 001374
 514 012430 001001
 515 012432 104017
 516 012434
 517
 521

BNE 80\$
 BIS #XNUER2,RMER2I
 CMP R3,RMER2I
 BNE 80\$
 EMT 17

80\$:

 ;*TEST 7 DRIVE TYPE TEST

 TST7:

012434
 012434 000004
 012436 000240
 012440 012706 001100
 012444 013700 001276
 012450 013701 001462
 012454 012737 000007 001226

SCOPE ;SCOPE CALL
 NOP
 MOV #STACK,SP ;LOAD THE STACK POINTER
 MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
 MOV #7,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

522
 523 012462 016002 000026
 524 012466 022702 020024
 525 012472 001437
 526 012474 022702 024024
 527 012500 001434
 528
 529 012502 022702 020025
 530 012506 001431
 531 012510 022702 024025
 532 012514 001426
 533
 534 012516 022702 020027
 535 012522 001423
 536 012524 022702 024027
 537 012530 001420

MOV RMDT(R0),R2 ;STORE RMDT AT R2
 CMP #SNGPRT,R2 ;SINGLE PORT RM03 ?
 BEQ 10\$;YES !!
 CMP #DULPRT,R2 ;DUAL PORT RM03 ?
 BEQ 10\$;YES !!
 CMP #SNGPRT!BIT0,R2 ;SINGLE PORT RM02 ?
 BEQ 10\$;YES !!
 CMP #DULPRT!BIT0,R2 ;DUAL PORT RM02 ?
 BEQ 10\$;YES !!
 CMP #SNGPRT!BIT1!BIT0,R2 ;SINGLE PORT RM05 ?
 BEQ 10\$;YES !!
 CMP #DULPRT!BIT1!BIT0,R2 ;DUAL PORT RM05 ?
 BEQ 10\$

538
 539 012532 010237 001142
 540 012536 012737 020024 001174
 541 012544 012737 024024 001176
 542 012552 010037 001136
 543 012556 062737 000026 001136
 544 012564 104057
 545 012566 000137 053600
 546 012572

MOV R2,\$BDDAT ;GET RECIEVED DRIVE TYPE
 MOV #SNGPRT,\$TMP0 ;GET EXPECTED DRIVE TYPE
 MOV #DULPRT,\$TMP1
 MOV R0,\$BDADR ;LOAD BAD ADDRESS
 ADD #RMDT,\$BDADR
 EMT 57
 JMP \$EOSP ;GO TO NEXT DEVICE

10\$:

 ;*TEST 10 DEVICE AVAILABLE TEST

 TST10:

012572
 012572 000004
 012574 000240
 012576 012706 001100
 012602 013700 001276
 012606 013701 001462
 012612 012737 000010 001226
 549
 550 012620 004737 055614

SCOPE ;SCOPE CALL
 NOP
 MOV #STACK,SP ;LOAD THE STACK POINTER
 MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
 MOV #10,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
 JSR PC,CNTCLR ;GO CLEAR CONTROLLER


```

551 012624 016037 000000 001142      MOV      RMCS1(R0), $BDDAT      ;STORE RMCS1 AT $BDDAT
552 012632 042737 173777 001142      BIC      #^CDVA, $BDDAT      ;CLEAR ALL BUT DVA
553 012640 001006                BNE      10$                  ;BRANCH IF DVA SET
554 012642 012737 004000 001140      MOV      #DVA, $GDDAT        ;SETUP EXPECTED
555 012650 010037 001136                MOV      R0, $BDADR          ;SETUP REG ADDRESS
556 012654 104060                EMT      60
557 012656                10$:
558
559
;*****
;*TEST 11      HOLDING REGISTER TRANSFER TEST
;*****
TST11:
012656                SCOPE                          ;SCOPE CALL
012656 000004                NOP
012660 000240                MOV      #STACK, SP          ;LOAD THE STACK POINTER
012662 012706 001100                MOV      $BASE, R0           ;R0 = UNIBUS ADDRESS
012666 013700 001276                MOV      TSTQUE, R1          ;R1 = POINTER TO DEVICE
012672 013701 001462                MOV      #11, $TESTN         ;;SET TEST NUMBER IN APT MAIL BOX
012676 012737 000011 001226
560
561 012704 004737 055614                JSR      PC, CNTCLR          ;GO CLEAR CONTROLLER
562 012710 005003                CLR      R3                  ;CLEAR ERROR FLAGS
563 012712 010037 001136                MOV      R0, $BDADR          ;SETUP REGISTER ADDRESS
564 012716 062737 000036 001136                ADD      #RMHR, $BDADR
565
566                ;WRITE ONES THEN ZEROS IN RMHR AND CHECK FOR S-A-1 BITS.
567                ;NOTE THAT IT IS NECESSARY TO WRITE SOME OTHER REGISTER IN
568                ;ORDER TO WRITE THE HOLDING REGISTER, AND RMDA IS USED FOR THIS
569                ;PURPOSE.
570 012724 012760 177777 000006                MOV      #-1, RMDA(R0)       ;LOAD RMDA
571 012732 012760 000000 000006                MOV      #0, RMDA(R0)        ;LOAD RMDA
572 012740 016037 000036 001142                MOV      RMHR(R0), $BDDAT    ;STORE RMHR AT $BDDAT
573 012746 005137 001142                COM      $BDDAT              ;ANY ERROR??
574 012752 001405                BEQ      10$                  ;NO!!
575 012754 005037 001140                CLR      $GDDAT              ;LOAD EXPECTED
576 012760 104061                EMT      61
577 012762 052703 000001                BIS      #BIT0, R3           ;SET ERROR FLAGS
578
579                ;WRITE ZEROS THEN ONES IN RMHR AND CHECK FOR S-A-0 BITS.
580 012766                10$:
581 012766 012760 000000 000006                MOV      #0, RMDA(R0)        ;LOAD RMDA
582 012774 012760 177777 000006                MOV      #-1, RMDA(R0)       ;LOAD RMDA
583 013002 016037 000036 001142                MOV      RMHR(R0), $BDDAT    ;STORE RMHR AT $BDDAT
584 013010 005137 001142                COM      $BDDAT              ;RMHR IS COMPLEMENTED WHEN READ
585 013014 012737 177777 001140                MOV      #-1, $GDDAT         ;SETUP EXPECTED
586 013022 023737 001140 001142                CMP      $GDDAT, $BDDAT      ;ANY ERROR??
587 013030 001403                BEQ      20$                  ;NO!!
588 013032 104062                EMT      62
589 013034 052703 000002                BIS      #BIT1, R3           ;SET ERROR FLAG
590
591                ;IF NO PREVIOUS ERRORS, WRITE AND READ SHIFTING ONE BIT PATTERN.
592 013040                20$:
593 013040 005703                TST      R3                  ;ANY FLAGS SET??
594 013042 001025                BNE      50$                  ;YES!!
595 013044 012702 000001                MOV      #1, R2              ;R2=DATA PATTERN
596 013050                30$:
597 013050 012760 000000 000006                MOV      #0, RMDA(R0)        ;LOAD RMDA

```

CZ
T2

```

598 013056 010260 000006      MOV      R2,RMDA(R0)      ;LOAD RMDA
599 013062 016037 000036 001142  MOV      RMHR(R0), $BDDAT ;STORE RMHR AT $BDDAT
600 013070 005137 001142      COM      $BDDAT          ;RMHR IS COMPLEMENTED
601 013074 023702 001142      CMP      $BDDAT,R2      ;ANY ERROR??
602 013100 001404              BEQ      40$            ;NO!!
603 013102 010237 001140      MOV      R2,$GDDAT      ;SETUP EXPECTED
604 013106 104063              EMT      63
605 013110 000402              BR       50$            ;DO NOT COLLECT ALL ERRORS
606
607 013112 006302      40$:    ASL      R2          ;SHIFT TO NEXT PATTERN
608 013114 001355      BNE     30$            ;CONTINUE IF NOT DONE
609
610 013116      50$:
611
612
:*****
:*TEST 12      CONTROL STATUS #1 TRANSFER TEST
:*****
TST12:
013116              SCOPE              ;SCOPE CALL
013116 000004              NOP
013120 000240              MOV      #STACK,SP     ;LOAD THE STACK POINTER
013122 012706 001100      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
013126 013700 001276      MOV      TSTQUE,R1     ;R1 = POINTER TO DEVICE
013132 013701 001462      MOV      #12,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
013136 012737 000012 001226      CLR      R3            ;R3 = ERROR INDICATOR
613
614 013144 005003      JSR     PC,CNTCLR      ;GO CLEAR CONTROLLER
615 013146 004737 055614
616
617
618
:WRITE ONES IN RMCS1, BITS 01-05, THEN CLEAR. READ AND
:CHECK FOR S-A-1 BITS.
619 013152 012760 000076 000000      MOV      #ILF76,RMCS1(R0) ;LOAD RMCS1
620 013160 004737 055614      JSR     PC,CNTCLR      ;GO CLEAR CONTROLLER
621 013164 016037 000000 001142      MOV      RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
622 013172 042737 177701 001142      BIC     #^CILF76,$BDDAT
623 013200 001410      BEQ     5$
624 013202 005037 001140      CLR     $GDDAT
625 013206 010037 001136      MOV     R0,$BDADR
626 013212 062737 000000 001136      ADD     #RMCS1,$BDADR
627 013220 104043      EMT     43
628
629
630
:WRITE ONES IN RMCS1, BITS 01-05, THEN WRITE ZEROS. READ AND CHECK FOR
:S-A-1 BITS.
631 013222      5$:
632 013222 012760 000076 000000      MOV     #ILF76,RMCS1(R0) ;LOAD RMCS1
633 013230 012760 000000 000000      MOV     #0,RMCS1(R0)    ;LOAD RMCS1
634 013236 016037 000000 001142      MOV     RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
635 013244 042737 177701 001142      BIC     #^CILF76,$BDDAT
636 013252 001412      BEQ     10$
637 013254 005037 001140      CLR     $GDDAT
638 013260 010037 001136      MOV     R0,$BDADR
639 013264 062737 000000 001136      ADD     #RMCS1,$BDADR
640 013272 104023      EMT     23
641 013274 052703 000001      BIS     #BIT0,R3        ;SET ERROR FLAG
642
643
644 013300      10$:
:WRITE ZEROS IN RMCS1, THEN ONES, READ AND CHECK S-A-0 BITS.

```



```

645 013300 012760 000000 000000      MOV      #0,RMCS1(R0)      ;LOAD RMCS1
646 013306 012760 000076 000000      MOV      #ILF76,RMCS1(R0) ;LOAD RMCS1
647 013314 016037 000000 001142      MOV      RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
648 013322 042737 177701 001142      BIC      #^CILF76,$BDDAT
649 013330 012737 000076 001140      MOV      #ILF76,$GDDAT
650 013336 023737 001140 001142      CMP      $GDDAT,$BDDAT
651 013344 001410                BEQ      20$
652 013346 010037 001136                MOV      R0,$BDADR
653 013352 062737 000000 001136      ADD      #RMCS1,$BDADR
654 013360 104024                EMT      24
655 013362 052703 000002      BIS      #BIT1,R3          ;SET ERROR FLAG
656
657
658 013366                ;WRITE A SHIFTING ONE BIT PATTERN IN RMCS1, READ AND CHECK FOR STUCK BITS.
659 013366 005703 20$:                TST      R3                ;OMIT IF ANY ERRORS
660 013370 001035                BNE      50$
661 013372 012702 000002      MOV      #2,R2            ;R2 = TEST PATTERN
662 013376 30$:                MOV      R2,R3            ;R3 = EXPECTED RESULT, BITS 1-5
663 013376 010203                BIC      #^CILF76,R3
664 013400 042703 177701      MOV      #0,RMCS1(R0)     ;LOAD RMCS1
665 013404 012760 000000 000000      MOV      R2,RMCS1(R0)    ;LOAD RMCS1
666 013412 010260 000000      MOV      RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
667 013416 016037 000000 001142      BIC      #^CILF76,$BDDAT
668 013424 042737 177701 001142      CMP      R3,$BDDAT
669 013432 020337 001142      BEQ      40$
670 013436 001410                MOV      R3,$GDDAT
671 013440 010337 001140      MOV      R0,$BDADR
672 013444 010037 001136      ADD      #RMCS1,$BDADR
673 013450 062737 000000 001136      EMT      25
674 013456 104025 40$:                ASL      R2                ;SHIFT TO NEXT BIT
675 013460 006302                BNE      30$              ;CONTINUE IF R2 NOT ZERO
676 013462 001345 50$:
677 013464
678
679
::*****
:*TEST 13      ERROR REGISTER #1 TRANSFER TEST
::*****
TST13:
013464                SCOPE                    ;SCOPE CALL
013464 000004      NOP
013466 000240      MOV      #STACK,SP      ;LOAD THE STACK POINTER
013470 012706 001100      MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
013474 013700 001276      MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
013500 013701 001462      MOV      #13,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
013504 012737 000013 001226      CLR      R3              ;CLEAR ERROR FLAG
680
681 013512 005003
682
683
684 013514 012760 177777 000014 ;WRITE ONES IN RMER1, CLEAR AND CHECK FOR S-A-1 BITS
685                MOV      #-1,RMER1(R0) ;LOAD RMER1
686 013522 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
687 013526 016037 000014 001346      MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
688 013534 013737 001346 001142      MOV      RMER1I,$BDDAT
689 013542 042737 177760 001142      BIC      #^C<PAR!RMR!ILF!ILR>,$BDDAT
690 013550 001410                BEQ      10$
691 013552 005037 001140      CLR      $GDDAT
  
```

692	013556	010037	001136		MOV	R0,\$BDADR	
693	013562	062737	000014	001136	ADD	#RMER1,\$BDADR	
694	013570	104027			EMT	27	
695	013572						10\$:
696	013572	013737	001346	001142	MOV	RMER1I,\$BDDAT	
697	013600	042737	074017	001142	BIC	#^C<DCK!IAE!AOE!HCRC!HCE!ECH!WCF!FER>,\$BDDAT	
698	013606	001410			BEQ	20\$	
699	013610	005037	001140		CLR	\$GDDAT	
700	013614	010037	001136		MOV	R0,\$BDADR	
701	013620	062737	000014	001136	ADD	#RMER1,\$BDADR	
702	013626	104030			EMT	30	
703	013630						20\$:
704	013630	013737	001346	001142	MOV	RMER1I,\$BDDAT	
705	013636	042737	147777	001142	BIC	#^C<OPI!DTE>,\$BDDAT	
706	013644	001410			BEQ	30\$	
707	013646	005037	001140		CLR	\$GDDAT	
708	013652	010037	001136		MOV	R0,\$BDADR	
709	013656	062737	000014	001136	ADD	#RMER1,\$BDADR	
710	013664	104031			EMT	31	
711							
712							:WRITE ONES THEN ZEROS IN RMER1, READ AND CHECK FOR S-A-1 BITS
713	013666						30\$:
714	013666	012760	177777	000014	MOV	#-1,RMER1(R0) ;LOAD RMER1	
715	013674	012760	000000	000014	MOV	#0,RMER1(R0) ;LOAD RMER1	
716	013702	016037	000014	001346	MOV	RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER	
717	013710	013737	001346	001142	MOV	RMER1I,\$BDDAT	
718	013716	042737	177770	001142	BIC	#^C<RMR!ILF!ILR>,\$BDDAT	
719	013724	001412			BEQ	40\$	
720	013726	005037	001140		CLR	\$GDDAT	
721	013732	010037	001136		MOV	R0,\$BDADR	
722	013736	062737	000014	001136	ADD	#RMER1,\$BDADR	
723	013744	104032			EMT	32	
724	013746	052703	000001		BIS	#BIT0,R3 ;SET ERROR FLAG	
725	013752	013737	001346	001142	MOV	RMER1I,\$BDDAT	40\$:
726	013760	042737	074017	001142	BIC	#^C<DCK!IAE!AOE!HCRC!HCE!ECH!WCF!FER>,\$BDDAT	
727	013766	001412			BEQ	50\$	
728	013770	005037	001140		CLR	\$GDDAT	
729	013774	010037	001136		MOV	R0,\$BDADR	
730	014000	062737	000014	001136	ADD	#RMER1,\$BDADR	
731	014006	104033			EMT	33	
732	014010	052703	000001		BIS	#BIT0,R3 ;SET ERROR FLAG	
733	014014						50\$:
734	014014	013737	001346	001142	MOV	RMER1I,\$BDDAT	
735	014022	042737	147777	001142	BIC	#^C<OPI!DTE>,\$BDDAT	
736	014030	001412			BEQ	60\$	
737	014032	005037	001140		CLR	\$GDDAT	
738	014036	010037	001136		MOV	R0,\$BDADR	
739	014042	062737	000014	001136	ADD	#RMER1,\$BDADR	
740	014050	104034			EMT	34	
741	014052	052703	030000		BIS	#BIT,R3	
742							
743							:WRITE ZEROS THEN ONES IN RMER1, READ AND CHECK FOR S-A-0 BITS
744	014056						60\$:
745	014056	012760	000000	000014	MOV	#0,RMER1(R0) ;LOAD RMER1	
746	014064	012760	177777	000014	MOV	#-1,RMER1(R0) ;LOAD RMER1	
747	014072	016037	000014	001142	MOV	RMER1(R0),\$BDDAT ;STORE RMER1 AT \$BDDAT	
748	014100	012737	177777	001140	MOV	#-1,\$GDDAT	


```

749 014106 023737 001140 001142      CMP      $GDDAT,$BDDAT
750 014114 001410                      BEQ      70$
751 014116 010037 001136                      MOV      R0,$BDADR
752 014122 062737 000014 001136      ADD      #RMER1,$BDADR
753 014130 104035                      EMT      35
754 014132 052703 000002                      BIS      #BIT1,R3
755
756                                     ;WRITE A SHIFTING 1 BIT IN RMER1 AND CHECK FOR STUCK BITS
757                                     ;NOTE: DONT TEST UNSAFE OR PARITY
758 014136                                70$:
759 014136 005703                      TST      R3                      ;SKIP THIS PART IF ANY ERRORS
760 014140 001042                      BNE      120$
761 014142 012702 000001                      MOV      #1,R2                      ;R2 = TEST PATTERN
762 014146                                80$:
763 014146 004737 055614                      JSR      PC,CNTCLR                  ;GO CLEAR CONTROLLER
764 014152 010260 000014                      MOV      R2,RMER1(R0)              ;LOAD RMER1
765 014156 016037 000014 001142      MOV      RMER1(R0),$BDDAT          ;STORE RMER1 AT $BDDAT
766 014164 032702 000010                      BIT      #PAR,R2                      ;DONT TEST PAR = 0
767 014170 001003                      BNE      90$
768 014172 042737 000010 001142      BIC      #PAR,$BDDAT
769 014200 032702 040000 90$:          BIT      #UNS,R2                      ;DONT TEST UNS = 0
770 014204 001003                      BNE      100$
771 014206 042737 040000 001142      BIC      #UNS,$BDDAT
772 014214 020237 001142 100$:        CMP      R2,$BDDAT
773 014220 001410                      BEQ      110$
774 014222 010237 001140                      MOV      R2,$GDDAT
775 014226 010037 001136                      MOV      R0,$BDADR
776 014232 062737 000014 001136      ADD      #RMER1,$BDADR
777 014240 104036                      EMT      36
778
779 014242 006302 110$:          ASL      R2                      ;SHIFT TO NEXT BIT
780 014244 001340 120$:          BNE      80$                      ;CONTINUE IF R2 NOT ZERO
781 014246
782
783

```

::*****
 ;*TEST 14 CLEAR OFFSET STUCK ACTIVE TEST

```

014246                                ;*****
014246 000004                                ;TST14:
014250 000240                                SCOPE                                ;SCOPE CALL
014252 012706 001100                      MOV      #STACK,SP                  ;LOAD THE STACK POINTER
014256 013700 001276                      MOV      $BASE,R0                   ;R0 = UNIBUS ADDRESS
014262 013701 001462                      MOV      TSTQUE,R1                  ;R1 = POINTER TO DEVICE
014266 012737 000014 001226      MOV      #14,$TESTN                ;;SET TEST NUMBER IN APT MAIL BOX
784
785 014274 004737 055614                      JSR      PC,CNTCLR                  ;GO CLEAR CONTROLLER
786 014300 012760 177777 000032      MOV      #-1,RMOF(R0)              ;LOAD RMOF
787 014306 016037 000032 001142      MOV      RMOF(R0),$BDDAT          ;STORE RMOF AT $BDDAT
788 014314 042737 177577 001142      BIC      #^COFD,$BDDAT
789 014322 001011                      BNE      10$                        ;BRANCH IF OFD IS A ONE
790 014324 012737 000200 001140      MOV      #OFD,$GDDAT              ;SETUP ERROR MESSAGE
791 014332 010037 001136                      MOV      R0,$BDADR
792 014336 062737 000032 001136      ADD      #RMOF,$BDADR
793 014344 104172                      EMT      172
794 014346 10$:
795                                     ;END OF TEST

```

CZI
T2

796

 :*TEST 15 OFFSET REGISTER TRANSFER TEST

 TST15:

014346					
014346	000004				
014350	000240				
014352	012706	001100			
014356	013700	001276			
014362	013701	001462			
014366	012737	000015	001226		
797					
798	014374	005003			
799	014376	010037	001136		
800	014402	062737	000032	001136	
801	014410	005037	001140		
802					
803					
804	014414	012760	177777	000032	
805	014422	012760	000000	000032	
806	014430	016037	000032	001364	
807					
808					
809	014436	013737	001364	001142	
810	014444	042737	016200	001142	
811	014452	001403			
812	014454	104053			
813	014456	052703	000001		
814					
815					
816	014462				
817	014462	013737	001364	001142	
818	014470	042737	161577	001142	
819	014476	001403			
820	014500	104054			
821	014502	052703	000001		
822					
823					
824	014506				
825	014506	012760	000000	000032	
826	014514	012760	177777	000032	
827	014522	016037	000032	001364	
828					
829					
830	014530	013737	001364	001142	
831	014536	042737	016200	001142	
832	014544	001403			
833	014546	104053			
834	014550	052703	000001		
835					
836					
837	014554				
838	014554	013737	001364	001142	
839	014562	042737	161577	001142	
840	014570	012737	016200	001140	
841	014576	023737	001140	001142	
842	014604	001403			

	SCOPE				
	NOF				
	MOV	#STACK,SP			:LOAD THE STACK POINTER
	MOV	\$BASE,R0			:R0 = UNIBUS ADDRESS
	MOV	TSTQUE,R1			:R1 = POINTER TO DEVICE
	MOV	#15,\$TESTN			::SET TEST NUMBER IN APT MAIL BOX
	CLR	R3			:RESET ERROR FLAGS
	MOV	R0,\$BDADR			:SETUP BAD ADDRESS
	ADD	#RMOF,\$BDADR			
	CLR	\$GDDAT			:SETUP EXPECTED DATA
					:WRITE ONES THEN ZEROS IN RMOF AND CHECK FOR S-A-1 BITS.
	MOV	#-1,RMOF(R0)			:LOAD RMOF
	MOV	#0,RMOF(R0)			:LOAD RMOF
	MOV	RMOF(R0),RMOFI			:STORE RMOF IN INPUT BUFFER
					:CHECK UNUSED BITS OF RMOF FOR ZERO
	MOV	RMOFI,\$BDDAT			:GET UNUSED BITS
	BIC	#^CXNUOF,\$BDDAT			
	BEQ	10\$:BRANCH IF NO ERROR
	EMT	53			
	BIS	#BIT0,R3			:SET ERROR FLAG
					:CHECK USED BITS OF RMOF FOR ZERO
	MOV	RMOFI,\$BDDAT			:GET USED BITS
	BIC	#XNUOF,\$BDDAT			
	BEQ	20\$:BRANCH IF NO ERROR
	EMT	54			
	BIS	#BIT0,R3			:SET ERROR FLAG
					:WRITE ZEROS THEN ONES ON RMOF AND CHECK FOR S-A-0 BITS.
	MOV	#0,RMOF(R0)			:LOAD RMOF
	MOV	#-1,RMOF(R0)			:LOAD RMOF
	MOV	RMOF(R0),RMOFI			:STORE RMOF IN INPUT BUFFER
					:CHECK UNUSED BITS OF RMOF FOR ZERO.
	MOV	RMOFI,\$BDDAT			:GET UNUSED BITS
	BIC	#^CXNUOF,\$BDDAT			
	BEQ	30\$:BRANCH IF NO ERROR
	EMT	53			
	BIS	#BIT0,R3			
					:CHECK USED BITS OF RMOF FOR ONE.
	MOV	RMOFI,\$BDDAT			:GET USED BITS
	BIC	#XNUOF,\$BDDAT			
	MOV	#^CXNUOF,\$GDDAT			:SETUP EXPECTED STATUS
	CMP	\$GDDAT,\$BDDAT			
	BEQ	40\$:BRANCH IF NO ERROR


```

843 014606 104055          EMT      55
844 014610 052703 000002    BIS      #BIT1,R3
845
846          ;IF NO PREVIOUS ERRORS, TEST RMOF WITH SHIFTING ONE BIT PATTERN.
847 014614          40$:
848 014614 005703          TST      R3          ;ANY ERROR??
849 014616 001025          BNE      70$         ;YES!!
850 014620 012702 000001    MOV      #1,R2       ;STARTING DATA PATTERN
851 014624          50$:
852 014624 012760 000000 000032  MOV      #0,RMOF(R0) ;LOAD RMOF
853 014632 010260 000032          MOV      R2,RMOF(R0) ;LOAD RMOF
854 014636 016037 000032 001142  MOV      RMOF(R0), $BDDAT ;STORE RMOF AT $BDDAT
855 014644 010203          MOV      R2,R3       ;SETUP EXPECTED RESULT
856 014646 042703 161577          BIC      #XNUOF,R3   ;CLEAR UNUSED BITS
857 014652 020337 001142          CMP      R3,$BDDAT   ;COMPARE EXPECTED & RECEIVED
858 014656 001403          BEQ      60$         ;BRANCH IF NO ERROR
859 014660 010337 001140          MOV      R3,$GDDAT  ;LOAD EXPECTED
860 014664 104056          EMT      56
861 014666 006302          60$:  ASL      R2          ;SHIFT TO NEXT BIT
862 014670 001355          BNE      50$         ;CONTINUE IF NOT DONE
863
864 014672          70$:
865
866          ;*****
          ;*TEST 16      ERROR REGISTER #2 TRANSFER TEST
          ;*****
          TST16:
          SCOPE          ;SCOPE CALL
          NOP
          MOV      #STACK,SP ;LOAD THE STACK POINTER
          MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
          MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
          MOV      #16,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
867
868 014720 005003          CLR      R3          ;RESET ERROR FLAGS
869 014722 010037 001136          MOV      R0,$BDADR  ;SETUP BAD ADDRESS
870 014726 062737 000042 001136  ADD      #RMR2,$BDADR
871 014734 005037 001140          CLR      $GDDAT     ;SETUP EXPECTED DATA
872 014740 012737 010000 001440  MOV      #FMT16,RMOFO ;SET 16 BIT FORMAT MODE TO ALLOW
873          ;"SSE" TO BE SET IN RMR2
874
875          ;WRITE ONES IN RMR2, CLEAR AND CHECK FOR S-A-1 BITS
876 014746 012760 177777 000042  MOV      #-1,RMR2(R0) ;LOAD RMR2
877
878 014754 004737 055614          JSR      PC,CNTCLR   ;GO CLEAR CONTROLLER
879 014760 013760 001440 000032  MOV      RMOFO,RMOF(R0) ;LOAD RMOF
880 014766 016037 000042 001374  MOV      RMR2(R0),RMR2I ;STORE RMR2 IN INPUT BUFFER
881
882          ;TEST UNUSED BITS FOR ZERO-FAILURE ON IF
883 014774 013737 001374 001142  MOV      RMR2I,$BDDAT
884 015002 042737 176210 001142  BIC      #^CXNUER2,$BDDAT
885 015010 001403          BEQ      10$         ;BRANCH IF NO ERROR
886 015012 104044          EMT      44
887 015014 052703 000001          BIS      #EIT0,R3   ;SET ERROR FLAG
888
889          ;TEST 'DPE', 'IVC', 'LSC' FOR ZERO-FAILURE ON CS, IF
  
```

```

890 015020
891 015020 013737 001374 001142 10$: MOV RMER2I,$BDDAT
892 015026 042737 143777 001142 BIC #^C<OPE!IVC!LSC>,$BDDAT
893 015034 001403 BEQ 20$ ;BRANCH IF NO ERROR
894 015036 104045 EMT 45
895 015040 052703 000001 BIS #BIT0,R3 ;SET ERROR FLAG
896
897 ;TEST 'LBC', 'DPE' FOR ZERO-FAILURE ON DS, IF
898 015044 20$: MOV RMER2I,$BDDAT
899 015044 013737 001374 001142 BIC #^C<LBC!DPE>,$BDDAT
900 015052 042737 175767 001142 BEQ 30$ ;BRANCH IF NO ERROR
901 015060 001403 EMT 46
902 015062 104046 BIS #BIT0,R3 ;SET ERROR FLAG
903 015064 052703 000001
904
905 ;WRITE ONES IN RMER2 THEN WRITE ZEROS AND CHECK FOR S-A-1 BITS.
906 015070 30$: MOV #-1,RMER2(R0) ;LOAD RMER2
907 015070 012760 177777 000042 MOV #0,RMER2(R0) ;LOAD RMER2
908 015076 012760 000000 000042 MOV RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
909 015104 016037 000042 001374
910
911 ;TEST 'DPE', 'IVC', 'LSC' FOR ZERO-FAILURE ON CS, IF
912 015112 013737 001374 001142 MOV RMER2I,$BDDAT
913 015120 042737 143777 001142 BIC #^C<OPE!IVC!LSC>,$BDDAT
914 015126 001403 BEQ 40$ ;BRANCH IF NO ERROR
915 015130 104047 EMT 47
916 015132 052703 000001 BIS #BIT0,R3 ;SET ERROR FLAG
917
918 ;TEST 'LBC', 'DPE' FOR ZERO-FAILURE ON DS, IF
919 015136 40$: MOV RMER2I,$BDDAT
920 015136 013737 001374 001142 BIC #^C<LBC!DPE>,$BDDAT
921 015144 042737 175767 001142 BEQ 50$ ;BRANCH IF NO ERROR
922 015152 001403 EMT 50
923 015154 104050 BIS #BIT0,R3 ;SET ERROR FLAG
924 015156 052703 000001
925
926 ;WRITE ZEROS IN RMER2 THEN WRITE ONES AND CHECK FOR S-A-0 BITS.
927 015162 50$: MOV #0,RMER2(R0) ;LOAD RMER2
928 015162 012760 000000 000042 MOV #-1,RMER2(R0) ;LOAD RMER2
929 015170 012760 177777 000042 MOV RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
930 015176 016037 000042 001374
931 ;TEST UNUSED BITS FOR ZERO-FAILURE ON IF
932 015204 013737 001374 001142 MOV RMER2I,$BDDAT
933 015212 042737 176210 001142 BIC #^CXNUE2,$BDDAT
934 015220 001403 BEQ 60$ ;BRANCH IF NO ERROR
935 015222 104044 EMT 44
936 015224 052703 000001 BIS #BIT0,R3 ;SET ERROR FLAG
937
938 ;TEST USED BITS FOR ONE-FAILURE ON IF
939 015230 60$: MOV RMER2I,$BDDAT
940 015230 013737 001374 001142 BIC #XNUE2,$BDDAT
941 015236 042737 001567 001142 MOV #^CXNUE2,$GDDAT
942 015244 012737 176210 001140 CMP $GDDAT,$BDDAT
943 015252 023737 001140 001142 BEQ 70$ ;BRANCH IF NO ERROR
944 015260 001403 EMT 51
945 015262 104051 BIS #BIT1,R3 ;SET ERROR FLAG
946 015264 052703 000002
  
```



```

947
948
949 015270
950 015270 005703
951 015272 001044
952 015274 012702 000001
953 015300
954 015300 004737 055614
955 015304 013760 001440 000032
956 015312 010260 000042
957 015316 016037 000042 001142
958 015324 032702 040000
959 015330 001003
960 015332 042737 040000 001142
961 015340 032702 000200
962 015344 001003
963 015346 042737 000200 001142
964 015354 010237 001140
965 015360 042737 001567 001140
966 015366 023737 001140 001142
967 015374 001401
968 015376 104052
969
970 015400 006302
971 015402 001336
972
973 015404
974
975
  
```

:IF NO PREVIOUS ERROR, TEST BIT INTERFERENCE WITH SHIFTING ONE BIT.

```

70$:
TST R3 ;ANY ERRORS?
BNE 120$ ;YES!!
MOV #1,R2 ;R2=DATA PATTERN

80$:
JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV RMOFO,RMOF(R0) ;LOAD RMOF
MOV R2,RMER2(R0) ;LOAD RMER2
MOV RMER2(R0),SBDDAT ;STORE RMER2 AT SBDDAT
BIT #SKI,R2 ;IS SKI BEING SET?
BNE 90$ ;YES!!
BIC #SKI,SBDDAT ;DONT TEST SKI FOR ZERO
90$:
BIT #DVC,R2 ;IS DVC BEING SET??
BNE 100$ ;YES!!
BIC #DVC,SBDDAT ;DONT TEST DVC FOR ZERO
100$:
MOV R2,$GDDAT
BIC #XNUER2,$GDDAT ;UNUSED BITS SHOULD BE ZERO
CMP $GDDAT,SBDDAT ;ANY ERRORS??
BEQ 110$ ;NO!!
EMT 52

110$:
ASL R2 ;SHIFT TO NEXT DATA BIT
BNE 80$ ;CONTINUE IF NOT DONE

120$:
  
```

 ;*TEST 17 SERIAL NUMBER TEST

 TST17:

```

015404
015404 000004
015406 000240
015410 012706 001100
015414 013700 001276
015420 013701 001462
015424 012737 000017 001226
976
977 015432 004737 055614
978 015436 010037 001136
979 015442 062737 000030 001136
980 015450 012702 000031
981
982
983 015454 016037 000030 001140
984
985
986 015462
987 015462 016037 000030 001142
988 015470 023737 001140 001142
989 015476 001401
990 015500 104136
991
992
993 015502
  
```

```

SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #17,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV R0,$BDADR ;SETUP REGISTER ADDRESS FOR TYPEOUT
ADD #RMSN,$BDADR
MOV #25.,R2 ;READ RMSN 25 TIMES

;READ RMSN AND USE THE RESULT AS EXPECTED VALUE
MOV RMSN(R0),$GDDAT ;STORE RMSN AT $GDDAT

;READ RMSN AND COMPARE WITH INITIAL VALUE
10$:
MOV RMSN(R0),SBDDAT ;STORE RMSN AT SBDDAT
CMP $GDDAT,SBDDAT
BEQ 20$ ;BRANCH IF SERIAL NUMBER CONSISTENT
EMT 136

;DECREMENT COUNT AND CONTINUE IF NOT DONE
20$:
  
```

```

994 015502 005302          DEC      R2
995 015504 100366          BPL      10$
996 015506          30$:          ;END OF TEST
997
998          ;*****
          ;*TEST 20          CONTROL BUS PARITY DETECTION TEST
          ;*****
          TST20:
          SCOPE          ;SCOPE CALL
          NOP
          MOV      #STACK,SP          ;LOAD THE STACK POINTER
          MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
          MOV      TSTQUE,R1          ;R1 = POINTER TO DEVICE
          MOV      #20,$TESTN          ;;SET TEST NUMBER IN APT MAIL BOX
999
1000          ;SETUP FOR FIRST TEST LOOP (NO ERROR)
1001 015534 005037 001140          CLR      $GDDAT          ;'PAR' SHOULD BE ZERO
1002 015540 111137 001416          MOVB     (R1),RMCS20          ;SETUP RMCS2 VALUE
1003 015544 042737 177770 001416          BIC      #^CUNTMSK,RMCS20
1004 015552 012737 000001 001444          MOV      #1,RMHRO          ;INITIALIZE DATA PATTERN
1005 015560 000402          BR      6$          ;SKIP INCREMENT FIRST TIME
1006 015562 006337 001444          5$:      ASL      RMHRO          ;SHIFT TO NEXT PATTERN
1007
1008          ;CLEAR AND VERIFY THAT 'PAR' IS RESET
1009 015566          6$:
1010 015566 004737 055614          JSR      PC,CNTCLR          ;GO CLEAR CONTROLLER
1011 015572 016037 000014 001142          MOV      RMER1(R0),$BDDAT          ;STORE RMER1 AT $BDDAT
1012 015600 016037 000042 001374          MOV      RMER2(R0),RMER2I          ;STORE RMER2 IN INPUT BUFFER
1013 015606 042737 177767 001142          BIC      #^CPAR,$BDDAT          ;DID 'PAR' RESET?
1014 015614 001415          BEQ     20$          ;YES!!
1015 015616 010037 001136          MOV      R0,$BDADR          ;SETUP REGISTER ADDRESS
1016 015622 062737 000014 001136          ADD      #RMER1,$BDADR
1017 015630 032737 000010 001374          BIT      #DPE,RMER2I          ;IS 'DPE' SET??
1018 015636 001002          BNE     10$          ;YES!!
1019 015640 104067          EMT     67
1020 015642 000453          BR      50$
1021 015644          10$:
1021 015644 104070          EMT     70
1022 015646 000451          BR      50$
1023
1024          ;WRITE TEST PATTERN AND VERIFY 'PAR' STATUS
1025 015650          20$:
1026 015650 013760 001416 000010          MOV      RMCS20,RMCS2(R0)          ;LOAD RMCS2
1027 015656 013760 001444 000036          MOV      RMHRO,RMHR(R0)          ;LOAD RMHR
1028 015664 016037 000014 001142          MOV      RMER1(R0),$BDDAT          ;STORE RMER1 AT $BDDAT
1029 015672 042737 177767 001142          BIC      #^CPAR,$BDDAT
1030 015700 023737 001140 001142          CMP      $GDDAT,$BDDAT          ;IS 'PAR' CORRECT??
1031 015706 001410          BEQ     40$          ;YES!!
1032 015710 032737 000020 001416          BIT      #PAT,RMCS20          ;SHOULD 'PAR' BE SET?
1033 015716 001002          BNE     30$          ;YES!!
1034 015720 104071          EMT     71
1035 015722 000423          BR      50$
1036 015724          30$:
1036 015724 104072          EMT     72
1037 015726 000421          BR      50$          ;SKIP TO NEXT
1038          ;GO TO NEXT PATTERN

```



```

1039 015730 005737 001444      40$:  TST  RMHRO      ;IS DATA PATTERN COMPLETE??
1040 015734 001312              BNE  5$           ;NO!!
1041 015736 032737 000020 001416  BIT  #PAT, RMCS20 ;IS TEST COMPLETE??
1042 015744 001012              BNE  50$          ;YES!!
1043                               ;SETUP FOR SECOND TEST LOOP (ERROR)
1044 015746 052737 000020 001416  BIS  #PAT, RMCS20 ;TURN ON BAD PARITY
1045 015754 012737 000010 001140  MOV  #PAR, $GDDAT ;EXPECT ERROR
1046 015762 012737 000001 001444  MOV  #1, RMHRO    ;INITIALIZE DATA PATTERN
1047 015770 000676              BR   6$           ;START LOOP
1048
1049 015772      50$:                               ;END OF TEST
1050
1051

```

 ;*TEST 21 CONTROL BUS PARITY GENERATION TEST

 TST21:

```

015772      SCOPE                               ;SCOPE CALL
015772 000004      NOP
015774 000240      MOV  #STACK, SP    ;LOAD THE STACK POINTER
015776 012706 001100  MOV  $BASE, R0     ;R0 = UNIBUS ADDRESS
016002 013700 001276  MOV  TSTQUE, R1    ;R1 = POINTER TO DEVICE
016006 013701 001462  MOV  #21, $TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
016012 012737 000021 001226

```

```

1052
1053
1054 016020 012737 000001 001444  ;SETUP FOR TEST (NO ERROR)
1055 016026 005037 001140      MOV  #1, RMHRO    ;INITIALIZE DATA PATTERN
1056 016032 000402              CLR  $GDDAT       ;MCPE SHOULD BE ZERO
1057                               BR   20$           ;DONT SHIFT FIRST TIME
1058

```

```

1059 016034      ;SHIFT DATA PATTERN
1060 016034 006337 001444  10$:  ASL  RMHRO
1061 016040      20$:  JSR  PC, CNTCLR   ;GO CLEAR CONTROLLER
1062 016040 004737 055614  MOV  RMHRO, RMHR(R0) ;LOAD RMHR
1063 016044 013760 001444 000036

```

```

1064
1065
1066 016052 016037 000036 001370  ;TRANSFER DATA TO RH, VERIFY NO 'MCPE' ERROR
1067 016060 016037 000000 001142  MOV  RMHR(R0), RMHRI ;STORE RMHR IN INPUT BUFFER
1068 016066 042737 157777 001142  MOV  RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
1069 016074 001402              BIC  #^CMCPE, $BDDAT ;WAS BAD PARITY DETECTED??
1070 016076 104073              BEQ  30$           ;NO!!
1071 016100 000403              EMT  73
1072                               BR   40$

```

```

1073
1074 016102      ;GO TO NEXT PATTERN
1075 016102 005737 001444  30$:  TST  RMHRO      ;DONE ALL PATTERNS??
1076 016106 001352              BNE  10$          ;NO!!
1077 016110      40$:                               ;END OF TEST
1078

```

 ;*TEST 22 RMDA, RMDC FAULT TEST

 TST22:

```

016110      SCOPE                               ;SCOPE CALL
016110 000004      NOP
016112 000240

```

```

016114 012706 001100      MOV    #STACK,SP      ;LOAD THE STACK POINTER
016120 013700 001276      MOV    $BASE,R0      ;R0 = UNIBUS ADDRESS
016124 013701 001462      MOV    TSTQUE,R1     ;R1 = POINTER TO DEVICE
016130 012737 000022 001226  MOV    #22,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
1080
1081 016136 004737 055614      JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
1082
1083      ;WRITE ZEROS, THEN ONES IN RMDA,RMDC-READ AND TEST FOR S-A-0
1084 016142 012760 000000 000006      MOV    #0,RMDA(R0)   ;LOAD RMDA
1085 016150 012760 000000 000034      MOV    #0,RMDC(R0)   ;LOAD RMDC
1086 016156 012760 177777 000006      MOV    #-1,RMDA(R0)  ;LOAD RMDA
1087 016164 012760 177777 000034      MOV    #-1,RMDC(R0)  ;LOAD RMDC
1088 016172 016037 000006 001340      MOV    RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
1089 016200 016037 000034 001366      MOV    RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
1090 016206 022737 177777 001340      CMP    #-1,RMDAI     ;IS ANY REGISTER ALL ONES??
1091 016214 001410          BEQ    10$           ;YES!!
1092 016216 052737 176000 001366      BIS    #XNUDC,RMDCI  ;SET UNUSED BITS
1093 016224 022737 177777 001366      CMP    #-1,RMDCI
1094 016232 001401          BEQ    10$           ;YES!!
1095 016234 104042          EMT    42
1096
1097 016236          10$:
1098
1099

```

;*TEST 23 DISK ADDRESS TRANSFER TEST

TST23:

```

016236 000004          SCOPE      ;SCOPE CALL
016240 000240          NOP
016242 012706 001100      MOV    #STACK,SP      ;LOAD THE STACK POINTER
016246 013700 001276      MOV    $BASE,R0      ;R0 = UNIBUS ADDRESS
016252 013701 001462      MOV    TSTQUE,R1     ;R1 = POINTER TO DEVICE
016256 012737 000023 001226  MOV    #23,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
1100
1101 016264 005003          CLR    R3            ;R3 = ERROR INDICATOR
1102 016266 004737 055614      JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
1103
1104      ;WRITE ONES IN RMDA, THEN WRITE ZEROS, READ BACK AND CHECK FOR S-A-1 BITS
1105 016272 012760 177777 000006      MOV    #-1,RMDA(R0)  ;LOAD RMDA
1106 016300 012760 000000 000006      MOV    #0,RMDA(R0)   ;LOAD RMDA
1107 016306 016037 000006 001142      MOV    RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
1108 016314 005737 001142      TST    $BDDAT
1109 016320 001412          BEQ    10$
1110 016322 005037 001140      CLR    $GDDAT
1111 016326 010037 001136      MOV    R0,$BDADR
1112 016332 062737 000006 001136      ADD    #RMDA,$BDADR
1113 016340 104020          EMT    20
1114 016342 052703 000001      BIS    #BIT0,R3      ;SET ERROR FLAG
1115      ;WRITE ZEROS IN RMDA, THEN WRITE ONES, READ BACK AND CHECK FOR S-A-0 BITS
1116 016346          10$:
016346 012760 000000 000006      MOV    #0,RMDA(R0)   ;LOAD RMDA
1117 016354 012760 177777 000006      MOV    #-1,RMDA(R0)  ;LOAD RMDA
1118 016362 016037 000006 001142      MOV    RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
1119 016370 023727 001142 177777      CMP    $BDDAT,#-1
1120 016376 001413          BEQ    20$
1121 016400 012737 177777 001140      MOV    #-1,$GDDAT

```



```

1122 016406 010037 001136      MOV      R0,$BDADR
1123 016412 062737 000006 001136  ADD      #RMDA,$BDADR
1124 016420 104021                EMT      21
1125 016422 052703 000002      BIS      #BIT1,R3      ;SET ERROR FLAG
1126                ;WRITE A SHIFTING 1 BIT PATTERN IN RMDA, READ, AND CHECK FOR STUCK BITS
1127 016426 005703 20$:      TST      R3      ;OMIT BIT TEST IF ANY ERROR
1128 016430 001027                BNE      50$
1129 016432 012702 000001      MOV      #1,R2      ;R2 = TEST PATTERN
1130 016436 30$:                MOV      #0,RMDA(R0)  ;LOAD RMDA
                MOV      R2,RMDA(R0)  ;LOAD RMDA
1131 016444 010260 000006 000006  MOV      RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
1132 016450 016037 000006 001142  CMP      $BDDAT,R2
1133 016456 023702 001142      BEQ      40$
1134 016462 001410                MOV      R2,$GDDAT
1135 016464 010237 001140      MOV      R0,$BDADR
1136 016470 010037 001136      ADD      #RMDA,$BDADR
1137 016474 062737 000006 001136  EMT      22
1138 016502 104022                ASL      R2      ;SHIFT TO NEXT BIT
1139 016504 006302                BNE      30$      ;CONTINUE IF R2 NOT ZERO
1140 016506 001353 40$:
1141 016510 50$:
1142
1143

```

 ;*TEST 24 DESIRED CYLINDER TRANSFER TEST

 TST24:

```

016510 000004      SCOPE                ;SCOPE CALL
016510 000240      NOP
016512 000240      MOV      #STACK,SP   ;LOAD THE STACK POINTER
016514 012706 001100  MOV      $BASE,R0    ;R0 = UNIBUS ADDRESS
016520 013700 001276  MOV      TSTQUE,R1   ;R1 = POINTER TO DEVICE
016524 013701 001462  MOV      #24,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
016530 012737 000024 001226  MOV
1144
1145 016536 005003      CLR      R3      ;RESET ERROR FLAGS
1146 016540 004737 055614  JSR      PC,CNTCLR  ;GO CLEAR CONTROLLER
1147 016544 005037 001140  CLR      $GDDAT    ;LOAD EXPECTED
1148 016550 010037 001136  MOV      R0,$BDADR  ;LOAD REG ADDRESS
1149 016554 062737 000034 001136  ADD      #RMDC,$BDADR
1150
1151                ;WRITE ONES IN RMDC AND VERIFY THAT UNUSED BITS ARE ZERO
1152 016562 012760 177777 000034  MOV      #-1,RMDC(R0) ;LOAD RMDC
1153 016570 016037 000034 001142  MOV      RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
1154 016576 042737 001777 001142  BIC      #^CXNUDC,$BDDAT ;CLEAR ALL USED BITS
1155 016604 001403                BEQ      5$      ;BRANCH IF NO ERROR
1156 016606 104134                EMT      134
1157 016610 052703 000001      BIS      #BIT0,R3   ;SET ERROR FLAG
1158
1159                ;WRITE ONES IN RMDC, THEN WRITE ZEROS, READ AND CHECK FOR S-A-1 BITS
1160 016614 5$:                MOV      #-1,RMDC(R0) ;LOAD RMDC
1161 016614 012760 177777 000034  MOV      #0,RMDC(R0)  ;LOAD RMDC
1162 016622 012760 000000 000034  MOV      RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
1163 016630 016037 000034 001142  BIC      #XNUDC,$BDDAT ;CLEAR UNUSED BITS
1164 016636 042737 176000 001142  BEQ      10$      ;BRANCH IF NO ERROR
1165 016644 001403                EMT      37
1166 016646 104037                BIS      #BIT0,R3   ;SET ERROR FLAG
1167 016650 052703 000001

```

```
1168
1169          ;WRITE ZEROS, THEN ONES IN RMDC, READ AND CHECK FOR S-A-0 BITS
1170 016654 10$:
1171 016654 012760 000000 000034      MOV      #0,RMDC(R0)      ;LOAD RMDC
1172 016662 012760 177777 000034      MOV      #-1,RMDC(R0)    ;LOAD RMDC
1173 016670 016037 000034 001142      MOV      RMDC(R0),SBDDAT ;STORE RMDC AT SBDDAT
1174 016676 052737 176000 001142      BIS      #XNUDC,SBDDAT  ;SET UNUSED BITS
1175 016704 012737 177777 001140      MOV      #-1,$GDDAT     ;LOAD EXPECTED RESULT
1176 016712 023737 001140 001142      CMP      $GDDAT,$BDDAT  ;IS RMDC ALL ONES ??
1177 016720 001403                      BEQ      20$             ;YES !!
1178 016722 104040                      EMT      40
1179 016724 052703 000002                      BIS      #BIT1,R3       ;SET ERROR FLAG
1180
1181          ;OMIT BIT TEST IF ANY ERRORS
1182 016730 20$:
1183 016730 005703                      TST      R3
1184 016732 001026                      BNE      60$
1185 016734 012702 000001                      MOV      #1,R2          ;R2 = TEST PATTERN
1186
1187          ;TEST RMDC WITH SHIFTING ONE BIT
1188 016740 30$:
1189 016740 012760 000000 000034      MOV      #0,RMDC(R0)    ;LOAD RMDC
1190 016746 010260 000034                      MOV      R2,RMDC(R0)    ;LOAD RMDC
1191 016752 010203                      MOV      R2,R3          ;R3 = EXPECTED RESULT
1192 016754 042703 176000                      BIC      #XNUDC,R3      ;CLEAR ANY UNUSED BITS
1193 016760 016037 000034 001142      MOV      RMDC(R0),SBDDAT ;STORE RMDC AT SBDDAT
1194 016766 023703 001142                      CMP      $BDDAT,R3
1195 016772 001404                      BEQ      50$
1196 016774 010337 001140                      MOV      R3,$GDDAT
1197 017000 104041                      EMT      41
1198 017002 000402                      BR       60$            ;SKIP TO NEXT
1199
1200 017004 006302 50$:
1201 017006 001354                      ASL      R2              ;SHIFT TO NEXT BIT
1202                      BNE      30$            ;CONTINUE IF R2 NOT ZERO
1203 017010 60$:
1204
1205          ;*****
          ;*TEST 25      ILLEGAL REGISTER TEST
          ;*****
          ;*****
          ;TST25:
1206          SCOPE                      ;SCOPE CALL
1207 017010 000004                      NOP
1208 017012 000240                      MOV      #STACK,SP     ;LOAD THE STACK POINTER
1209 017014 012706 001100                      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
1210 017020 013700 001276                      MOV      TSTQUE,R1     ;R1 = POINTER TO DEVICE
1211 017024 013701 001462                      MOV      #25,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX
1212 017030 012737 000025 001226                      CLR      $GDDAT        ;"ILR" SHOULD BE ZERO
1213 017036 005037 001140                      MOV      #0,R2         ;R2=REGISTER INDEX
1214 017042 012702 000000
1215          ;CLEAR AND VERIFY THAT "ILR" STATUS IS ZERO
1216 10$:
1217 017046 004737 055614                      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
1218 017052 016037 000014 001142                      MOV      RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
1219 017060 042737 177775 001142                      BIC      #^CILR,$BDDAT
```



```

1215 017066 001411          BEQ      20$          ;BRANCH IF ILR IS RESET
1216 017070 005037 001140    CLR      $GDDAT      ;SETUP GOOD DATA, REG ADR
1217 017074 010037 001136    MOV      R0,$BDADR
1218 017100 062737 000014 001136  ADD      #RMER1,$BDADR
1219 017106 104064          EMT      64
1220 017110 000550          BR       140$
1221
1222          ;WRITE THE REGISTER (INDEX=R2) AND TEST ILR STATUS
1223 017112 20$:
1224 017112 010003          MOV      R0,R3          ;R3=REG ADDRESS
1225 017114 060203          ADD      R2,R3
1226 017116 013746 000004    MOV      ERVEC,-(SP)    ;;PUSH ERVEC ON STACK
1227 017122 013746 000006    MOV      ERVEC+2,-(SP) ;;PUSH ERVEC+2 ON STACK
1228 017126 012737 017170 000004  MOV      #40$,ERRVEC   ;SETUP FOR BUS TIMEOUT
1229 017134 012737 000300 000006  MOV      #PR6,ERRVEC+2
1230 017142 005004          CLR      R4            ;R4=REGISTER VALUE
1231 017144 022702 000010    CMP      #RMCS2,R2
1232 017150 001001          BNE     30$
1233 017152 111104          MOV     (R1),R4        ;SELECT DRIVE IF RMCS2
1234 017154 010413 30$:      MOV      R4,(R3)        ;WRITE TEST REGISTER
1235 017156 012637 000006    MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERVEC+2
1236 017162 012637 000004    MOV      (SP)+,ERRVEC   ;;POP STACK INTO ERVEC
1237 017166 000416          BR       60$
1238 017170 012716 017176 40$:      MOV      #45$,(SP)     ;DUMMY RTI ADDRESS
1239 017174 000002          RTI
1240 017176 45$:
1241 017176 012637 000006    MOV      (SP)+,ERRVEC+2 ;;POP STACK INTO ERVEC+2
1242 017202 012637 000004    MOV      (SP)+,ERRVEC   ;;POP STACK INTO ERVEC
1243 017206 020227 000046    CMP      R2,#RMEC2     ;WERE ALL REGISTERS READ??
1244 017212 101003          BHI     50$           ;YES!!
1245 017214 010337 001136    MOV      R3,$BDADR
1246 017220 104074          EMT      74
1247 017222 000503 50$:      BR       140$
1248 017224 60$:
1249 017224 016037 000014 001142  MOV      RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
1250 017232 042737 177775 001142  BIC      #^CILR,$BDDAT
1251 017240 023737 001140 001142  CMP      $GDDAT,$BDDAT ;IS 'ILR' OK??
1252 017246 001411          BEQ     80$           ;YES!!
1253 017250 010337 001174          MOV      R3,$TMP0      ;SAVE ADDRESS
1254 017254 032737 000002 001140  BIT      #ILR,$GDDAT   ;SHOULD 'ILR' BE SET??
1255 017262 001002          BNE     70$           ;YES!!
1256 017264 104065          EMT      65
1257 017266 000401 70$:      BR       80$
1258 017270 104066          EMT      66
1259          ;ADVANCE TO THE NEXT REGISTER ADDRESS
1260 017272 80$:
1261 017272 062702 000002    ADD      #2,R2          ;INCREMENT INDEX
1262 017276 022702 000050    CMP      #50,R2        ;TIME TO TRY RH70 ?
1263 017302 101261          BHI     10$           ;BRANCH IF NOT
1264 017304 103437          BLO     110$          ;BRANCH IF ALREADY CHECKED
1265          :
1266          :      CMP      #RMBAE,R2 ;IS THIS RMBAE??
1267 017306 013746 000004    BNE     100$          ;NO!!
1268 017312 013746 000006    MOV      ERVEC,-(SP)   ;;PUSH ERVEC ON STACK
1268 017312 013746 000006    MOV      ERVEC+2,-(SP) ;;PUSH ERVEC+2 ON STACK

```

```

1269 017316 012737 017414 000004      MOV      #130$,ERRVEC      ;SETUP FOR TIMEOUT
1270 017324 012737 000300 000006      MOV      #PR6,ERRVEC+2
1271          :      PUTCS1      #A17!A16
1272          :      GETBAE
1273          :      BIC      #^C<BIT1!BIT0>,RMBAEI
1274          :      BNE      90$      ;BRANCH IF RH70 CONTROLLER
1275          :      BIS      #ILR,$GDDAT      ;'ILR' SHOULD BE SET
1276          :      POP      ERRVEC+2
1277          :      POP      ERRVEC
1278          :      CMP      #RMCS3+2,R2      ;DONE ALL LEGAL REGISTERS
1279          :      BNE      110$
1280          :      BIS      #ILR,$GDDAT      ;'ILR' SHOULD BE SET
1281 017332 052737 000002 001140      BIS      #ILR,$GDDAT      ;SET ILR
1282 017340 012702 000054          MOV      #54,R2      ;START AT INDEX 54 IF RH70 WITH 22 REG
1283 017344 012760 001400 000000      MOV      #A17!A16,RMCS1(R0)      ;SET EXTEND BITS
1284 017352 016037 000050 001402      MOV      50(R0),RMBAEI      ;READ THE EXTENDED BITS
1285 017360 042737 177774 001402      BIC      #177774,RMBAEI      ;CHOP OFF
1286 017366 001002          BNE      90$      ;BRANCH IF RH70 WITH 22 REG
1287 017370 012702 000050          MOV      #50,R2      ;OTHERWISE NOT A RH70 OR RH70 WITH 32 REG
1288 017374 012637 000006 90$:      MOV      (SP)+,ERRVEC+2
1289 017400 012637 000004          MOV      (SP)+,ERRVEC
1290 017404 022702 000074 110$:      CMP      #74,R2      ;DONE ALL TESTS
1291 017410 101410          BLOS     140$      ;YES!!
1292 017412 000615 120$:      BR      10$
1293
1294 017414 012716 017422 130$:      MOV      #135$,(SP)      ;DUMMY RTI ADDRESS
1295 017420 000002          RTI      ;RESTORE PRIORITY
1296 017422 135$:
1297 017422 012637 000004          MOV      (SP)+,ERRVEC      ;;POP STACK INTO ERRVEC
1298 017426 012637 000006          MOV      (SP)+,ERRVEC+2      ;;POP STACK INTO ERRVEC+2
1299 017432 140$:      ;END OF TEST
1300

```

 ;*TEST 26 RESET GO BY INIT TEST

 TST26:

```

017432          :      SCOPE      ;SCOPE CALL
017432 000004          NOP
017434 000240          MOV      #STACK,SP      ;LOAD THE STACK POINTER
017436 012706 001100      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
017442 013700 001276      MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
017446 013701 001462      MOV      #26,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
017452 012737 000026 001226
1301
1302 017460 004737 055614          JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
1303 017464 010037 001136          MOV      R0,$BADDR      ;SETUP REGISTER ADDRESS FOR MSG
1304
1305          ;SET GO, INITIALIZE AND VERIFY THAT GO IS RESET
1306 017470 012760 000001 000000      MOV      #GO,RMCS1(R0)      ;LOAD RMCS1
1307
1308 017476 004737 055614          JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
1309 017502 016037 000000 001142      MOV      RMCS1(R0),$BDDAT      ;STORE RMCS1 AT $BDDAT
1310 017510 042737 177776 001142      BIC      #^CGO,$BDDAT
1311 017516 001403          BEQ      10$      ;BRANCH IF GO IS RESET
1312 017520 005037 001140          CLR      $GDDAT
1313 017524 104137          EMT      137
1314

```


1315 017526
1316
1317

10\$: ;END OF TEST

::*****
:*TEST 27 DIAGNOSTIC MODE TEST

::*****
TST27:

017526
017526 000004
017530 000240
017532 012706 001100
017536 013700 001276
017542 013701 001462
017546 012737 000027 001226

SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #27,\$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

1318
1319 017554 010037 001136
1320 017560 062737 000024 001136
1321 017566 005003

MOV R0,\$BDADR ;SETUP REGISTER ADDRESS
ADD #RMMR1,\$BDADR
CLR R3 ;INITIALIZE ERROR FLAGS

1322
1323
1324 017570 004737 055614
1325 017574 016037 000024 001142
1326 017602 042737 177776 001142
1327 017610 001403
1328 017612 005037 001140
1329 017616 104075

;INITIALIZE AND VERIFY THAT 'DMD' IS RESET
JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV RMMR1(R0),\$BDDAT ;STORE RMMR1 AT \$BDDAT
BIC #^CDMD,\$BDDAT
BEQ 10\$;BRANCH IF 'DMD' IS ZERO
CLR \$GDDAT
EMT 75

1330
1331
1332 017620
1333 017620 012760 000001 000024
1334 017626 012760 000000 000024
1335 017634 016037 000024 001142
1336 017642 042737 177776 001142
1337 017650 001405
1338 017652 005037 001140
1339 017656 104076
1340 017660 052703 000001

;SET AND RESET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-1
10\$:

MOV #DMD,RMMR1(R0) ;LOAD RMMR1
MOV #0,RMMR1(R0) ;LOAD RMMR1
MOV RMMR1(R0),\$BDDAT ;STORE RMMR1 AT \$BDDAT
BIC #^CDMD,\$BDDAT
BEQ 20\$;BRANCH IF DMD NOT S-A-1
CLR \$GDDAT
EMT 76
BIS #BIT0,R3 ;SET ERROR FLAG

1341
1342
1343 017664
1344 017664 012760 000000 000024
1345 017672 012760 000001 000024
1346 017700 016037 000024 001142
1347 017706 042737 177776 001142
1348 017714 001006
1349 017716 012737 000001 001140
1350 017724 104077
1351 017726 052703 000002

;RESET AND SET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-0
20\$:

MOV #0,RMMR1(R0) ;LOAD RMMR1
MOV #DMD,RMMR1(R0) ;LOAD RMMR1
MOV RMMR1(R0),\$BDDAT ;STORE RMMR1 AT \$BDDAT
BIC #^CDMD,\$BDDAT
BNE 30\$;BRANCH IF DMD NOT S-A-0
MOV #DMD,\$GDDAT
EMT 77
BIS #BIT1,R3 ;SET ERROR FLAG

1352
1353
1354
1355 017732
1356 017732 005703
1357 017734 001027
1358 017736 012702 000002
1359 017742
017742 012760 000000 000024
1360 017750 010260 000024

;IF NO PREVIOUS ERROR, TEST FOR BIT INTERFERENCE WITH SHIFTING
;ONE BIT PATTERN
30\$:

TST R3 ;ANY ERRORS DETECTED??
BNE 60\$;YES!!
MOV #BIT1,R2 ;INITAILIZE DATA PATTERN

40\$:
MOV #0,RMMR1(R0) ;LOAD RMMR1
MOV R2,RMMR1(R0) ;LOAD RMMR1

```

1361 017754 016037 000024 001142      MOV      RMMR1(R0), $BDDAT      ;STORE RMMR1 AT $BDDAT
1362 017762 042737 177776 001142      BIC      #^CDMD, $BDDAT
1363 017770 001407                BEQ      50$                    ;BRANCH IF DMD NOT SET
1364 017772 010237 001174      MOV      R2, $TMP0             ;SAVE DATA
1365 017776 010237 001174      MOV      R2, $TMP0             ;SAVE DATA
1366 020002 005037 001140      CLR      $GDDAT                ;DMD SHOULD BE ZERO
1367 020006 104100                EMT      100
1368
1369
1370 020010                ;SHIFT TO NEXT DATA BIT AND CONTINUE TEST IF NOT DONE
50$:
1371 020010 006302                ASL      R2
1372 020012 001353                BNE     40$
1373 020014                60$:                            ;END OF TEST
1374
1375
::*****
;*TEST 30      MOL TEST
*****
TST30:
020014                SCOPE                            ;SCOPE CALL
020014 000004                NOP
020016 000240                MOV      #STACK, SP            ;LOAD THE STACK POINTER
020020 012706 001100      MOV      $BASE, R0             ;R0 = UNIBUS ADDRESS
020024 013700 001276      MOV      TSTQUE, R1            ;R1 = POINTER TO DEVICE
020030 013701 001462      MOV      #30, $TESTN           ;;SET TEST NUMBER IN APT MAIL BOX
020034 012737 000030 001226      JSR      PC, CNTCLR            ;GO CLEAR CONTROLLER
1376
1377 020042 004737 055614      MOV      R0, $BDADR            ;R0=REGISTER ADDRESS
1378 020046 010037 001136      ADD      #RMDS, $BDADR
1379 020052 062737 000012 001136      CLR      R3                    ;R3=ERROR FLAG
1380 020060 005003
1381
1382
1383 020062 012760 000001 000024      ;SET DIAGNOSTIC MODE AND VERIFY THAT 'MOL' IS ZERO
1384 020070 016037 000012 001142      MOV      #DMD, RMMR1(R0)       ;LOAD RMMR1
1385 020076 042737 167777 001142      MOV      RMDS(R0), $BDDAT      ;STORE RMDS AT $BDDAT
1386 020104 001405                BIC      #^CMOL, $BDDAT
1387 020106 005037 001140      BEQ     10$
1388 020112 104101                CLR      $GDDAT
1389 020114 052703 000001      EMT     101
1390
1391
1392 020120                ;SET MAINTENANCE UNIT READY AND VERIFY THAT 'MOL' IS ONE
10$:
1393 020120 012760 000001 000024      MOV      #DMD, RMMR1(R0)       ;LOAD RMMR1
1394 020126 012760 001001 000024      MOV      #DMD!MUR, RMMR1(R0)   ;LOAD RMMR1
1395 020134 016037 000012 001142      MOV      RMDS(R0), $BDDAT      ;STORE RMDS AT $BDDAT
1396 020142 042737 167777 001142      BIC      #^CMOL, $BDDAT
1397 020150 001006                BNE     20$
1398 020152 012737 010000 001140      MOV      #MOL, $GDDAT
1399 020160 104102                EMT     102
1400 020162 052703 000002      BIS      #BIT1, R3
1401
1402
1403
1404 020166                ;IF NO PREVIOUS ERROR, VERIFY VOLUME VALID IS RESET AND
20$:
1405 020166 005703                TST     R3                      ;ANY ERROR DETECTED??
1406 020170 001057                BNE     70$                      ;YES!!
1407 020172 016037 000012 001142      MOV      RMDS(R0), $BDDAT      ;STORE RMDS AT $BDDAT
    
```



```

1408 020200 042737 177677 001142      BIC    #^CVV,$BDDAT
1409 020206 001403                    BEQ    25$      ;BRANCH IF VV RESET
1410 020210 005037 001140                    CLR    $GDDAT
1411 020214 104135                    EMT    135
1412 020216                    25$:
1413 020216 012702 000001                    MOV    #1,R2      ;INITIALIZE DATA PATTERN
1414 020222                    30$:
      020222 012760 000001 000024                    MOV    #DMD,RMMR1(RO) ;LOAD RMMR1
1415 020230 010260 000024                    MOV    R2,RMMR1(RO)  ;LOAD RMMR1
1416 020234 016037 000012 001142                    MOV    RMDS(RO),$BDDAT ;STORE RMDS AT $BDDAT
1417 020242 042737 167777 001142                    BIC    #^CMOL,$BDDAT
1418 020250 005003                    CLR    R3          ;SETUP EXPECTED 'MOL'
1419 020252 032702 001000                    BIT    #MUR,R2
1420 020256 001402                    BEQ    40$
1421 020260 052703 010000                    BIS    #MOL,R3     ;'MOL' SHOULD BE ONE
1422 020264 020337 001142                    40$:  CMP    R3,$BDDAT ;IS MOL OK??
1423 020270 001405                    BEQ    50$         ;YES!!
1424 020272 010237 001174                    MOV    R2,$TMP0    ;SAVE TEST PATTERN
1425 020276 010337 001140                    MOV    R3,$GDDAT
1426 020302 104103                    EMT    103
1427
1428                    ;SHIFT TO NEXT PATTERN
1429 020304                    50$:
1430 020304 042702 000001                    BIC    #DMD,R2     ;DONT SHIFT DMD
1431 020310 001002                    BNE    60$
1432 020312 012702 000001                    MOV    #DMD,R2     ;DONT TRUNCATE TEST
1433 020316 006302                    60$:  ASL    R2          ;SHIFT TO NEXT BIT
1434 020320 001403                    BEQ    70$         ;BRANCH IF DONE
1435 020322 052702 000001                    BIS    #DMD,R2     ;KEEP DMD ON
1436 020326 000735                    BR     30$         ;CONTINUE
1437
1438 020330                    70$:
1439
1440                    ;*****
                    ;*TEST 31      WRITE LOCK TEST
                    ;*****
      020330                    TST31:
      020330 000004                    SCOPE                    ;SCOPE CALL
      020332 000240                    NOP
      020334 012706 001100                    MOV    #STACK,SP     ;LOAD THE STACK POINTER
      020340 013700 001276                    MOV    $BASE,R0      ;R0 = UNIBUS ADDRESS
      020344 013701 001462                    MOV    TSTQUE,R1     ;R1 = POINTER TO DEVICE
      020350 012737 000031 001226                    MOV    #31,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
1441
1442 020356 005003                    CLR    R3            ;R3=ERROR FLAG
1443 020360 010037 001136                    MOV    R0,$BDADR     ;SETUP REGISTER ADDRESS
1444 020364 062737 000012 001136                    ADD    #RMDS,$BDADR
1445 020372 005037 001140                    CLR    $GDDAT        ;WRL SHOULD BE ZERO
1446 020376 004737 055614                    JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
1447
1448                    ;SET DIAGNOSTIC MODE AND VERIFY 'WRL' IS ZERO
1449 020402 012760 000001 000024                    MOV    #DMD,RMMR1(RO) ;LOAD RMMR1
1450 020410 016037 000012 001142                    MOV    RMDS(RO),$BDDAT ;STORE RMDS AT $BDDAT
1451 020416 042737 173777 001142                    BIC    #^CWRL,$BDDAT
1452 020424 001403                    BEQ    10$           ;BRANCH IF WRL IS ZERO
1453 020426 104104                    EMT    104

```

```

1454 020430 052703 000001          BIS    #BIT0,R3          ;SET ERROR FLAG
1455
1456                                ;SET MAINTENANCE WRITE PROTECT AND VERIFY 'WRL' IS ONE
1457 020434                                10$:
1458 020434 012760 000001 000024    MOV    #DMD,RMMR1(R0)    ;LOAD RMMR1
1459 020442 012760 000011 000024    MOV    #DMD!MWP,RMMR1(R0) ;LOAD RMMR1
1460 020450 016037 000012 001142    MOV    RMDS(R0),SBDDAT  ;STORE RMDS AT SBDDAT
1461 020456 042737 173777 001142    BIC    #^CWRL,SBDDAT
1462 020464 001006                                BNE    20$              ;BRANCH IF WRL IS NOE
1463 020466 012737 004000 001140    MOV    #WRL,$GDDAT      ;WRL SHOULD BE SET
1464 020474 104105                                EMT    105
1465 020476 052703 000002          BIS    #BIT1,R3          ;SET ERROR FLAG
1466
1467                                ;IF NO PREVIOUS ERROR, TEST FOR BIT INTERFERENCE ON 'MWP'
1468 020502                                20$:
1469 020502 005703                                TST    R3              ;ANY OTHER ERROR??
1470 020504 001045                                BNE    70$              ;YES!!
1471 020506 012702 000001          MOV    #1,R2            ;INITIALIZE DATA PATTERN
1472
1473                                ;TRANSFER DATA TO RMMR1, READ RMDS AND VERIFY WRL
1474 020512                                30$:
1475 020512 012760 000001 000024    MOV    #DMD,RMMR1(R0)    ;LOAD RMMR1
1476 020520 010260 000024          MOV    R2,RMMR1(R0)     ;LOAD RMMR1
1477 020524 016037 000012 001142    MOV    RMDS(R0),SBDDAT  ;STORE RMDS AT SBDDAT
1478 020532 042737 173777 001142    BIC    #^CWRL,SBDDAT    ;CLEARUP RECEIVED 'WRL'
1479 020540 005003                                CLR    R3              ;GENERATE EXPECTED 'WRL'
1480 020542 032702 000010          BIT    #MWP,R2
1481 020546 001402                                BEQ    40$
1482 020550 052703 004000          BIS    #WRL,R3          ;WRL SHOULD BE SET
1483 020554 020337 001142          40$: CMP    R3,SBDDAT      ;IS WRL OK??
1484 020560 001405                                BEQ    50$              ;YES!!
1485 020562 010337 001140          MOV    R3,$GDDAT        ;SAVE EXPECTED
1486 020566 010237 001174          MOV    R2,$TMP0         ;SAVE DATA PATTERN
1487 020572 104106                                EMT    106
1488
1489                                ;ADVANCE TO NEXT DATA BIT
1490 020574                                50$:
1491 020574 042702 000001          BIC    #DMD,R2          ;DONT SHIFT DMD
1492 020600 001002                                BNE    60$
1493 020602 012702 000001          MOV    #DMD,R2          ;DONT TRUNCATE TEST
1494 020606 006302          60$: ASL    R2              ;SHIFT DATA BIT
1495 020610 001403                                BEQ    70$              ;EXIT IF DONE
1496 020612 052702 000001          BIS    #DMD,R2          ;KEEP DIAGNOSTIC MODE ON
1497 020616 000735                                BR     30$              ;CONTINUE TEST
1498
1499 020620          70$:                                ;END OF TEST
1500
1501                                ;*****
                                ;*TEST 32      DRIVE FAULT TEST
                                ;*****
020620          TST32:
020620 000004          SCOPE                                ;SCOPE CALL
020622 000240          NOP
020624 012706 001100          MOV    #STACK,SP        ;LOAD THE STACK POINTER
020630 013700 001276          MOV    $BASE,R0         ;R0 = UNIBUS ADDRESS
020634 013701 001462          MOV    TSTQUE,R1        ;R1 = POINTER TO DEVICE
    
```

CZ
T4


```

1502 020640 012737 000032 001226      MOV      #32,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1503 020646 004737 055614              JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
1504 020652 005003              CLR      R3              ;INITIALIZE ERROR FLAGS
1505 020654 010037 001136      MOV      R0,$BDADR      ;SETUP REGISTER ADDRESS
1506 020660 062737 000042 001136      ADD      #RMER2,$BDADR
1507 020666 005037 001140      CLR      $GDDAT        ;'DVC' AND 'UNS' SHOULD BE ZERO
1508
1509      ;SET AND RESET MAINTENANCE DRIVE FAULT, VERIFY THAT 'DVC' IS NOT
1510      ;STUCK-AT-ONE.
1511 020672 012760 000001 000024      MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
1512 020700 012760 000000 000042      MOV      #0,RMER2(R0)   ;LOAD RMER2
1513 020706 012760 000000 000014      MOV      #0,RMER1(R0)   ;LOAD RMER1
1514 020714 016037 000042 001142      MOV      RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
1515 020722 042737 177577 001142      BIC      #^CDVC,$BDDAT  ;IS 'DVC' RESET??
1516 020730 001406              BEQ      10$            ;YES!!
1517 020732 104107              EMT      107
1518 020734 052703 000001              BIS      #BIT0,R3       ;SET ERROR FLAG
1519 020740 012737 040000 001140      MOV      #UNS,$GDDAT
1520
1521      ;VERIFY THAT 'UNS' IS SAME AS 'DVC'
1522 020746 10$:
1523 020746 016037 000014 001142      MOV      RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
1524 020754 042737 137777 001142      BIC      #^CUNS,$BDDAT
1525 020762 023737 001140 001142      C.AP    $GDDAT,$BDDAT  ;IS 'UNS' OK??
1526 020770 001414              BEQ      30$            ;YES!!
1527 020772 010037 001136      MOV      R0,$BDADR      ;SETUP REGISTER ADDRESS
1528 020776 062737 000014 001136      ADD      #RMER1,$BDADR
1529 021004 032737 040000 001140      BIT      #UNS,$GDDAT    ;SHOULD 'UNS' BE ON??
1530 021012 001002              BNE      20$            ;YES!!
1531 021014 104110              EMT      110
1532 021016 000401              BR       30$
1533 021020 20$:
1534 021020 104111              EMT      111
1535
1536      ;RESET AND SET 'MDF', VERIFY THAT 'DVC' IS NOT S-A-0.
1537 021022 30$:
1538 021022 012737 000200 001140      MOV      #DVC,$GDDAT    ;DVC SHOULD BE ON
1539 021030 012760 000001 000024      MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
1540 021036 012760 000101 000024      MOV      #DMD!MDF,RMMR1(R0) ;LOAD RMMR1
1541 021044 016037 000042 001142      MOV      RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
1542 021052 042737 177577 001142      BIC      #^CDVC,$BDDAT
1543 021060 001012              BNE      40$            ;BRANCH IF DVC IS SET
1544 021062 010037 001136      MOV      R0,$BDADR      ;SETUP REGISTER ADDRESS
1545 021066 062737 000042 001136      ADD      #RMER2,$BDADR
1546 021074 104112              EMT      112
1547 021076 052703 000002              BIS      #BIT1,R3       ;SET ERROR FLAG
1548 021102 005037 001140      CLR      $GDDAT        ;UNS SHOULD BE OFF
1549
1550      ;VERIFY THAT 'UNS' IS SAME AS 'DVC'
1551 021106 40$:
1552 021106 005737 001140      TST      $GDDAT         ;CHANGE DVC TO UNS
1553 021112 001403              BEQ      50$
1554 021114 012737 040000 001140      MOV      #UNS,$GDDAT
1555 021122 50$:
1556 021122 016037 000014 001142      MOV      RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
1557 021130 042737 137777 001142      BIC      #^CUNS,$BDDAT
    
```

```
1558 021136 023737 001140 001142      CMP      $GDDAT,$BDDAT
1559 021144 001414                      BEQ      70$      ;BRANCH IF UNS IS OK
1560 021146 010037 001136                      MOV      R0,$BDADR ;SETUP REGISTER ADDRESS
1561 021152 062737 000014 001136      ADD      #RMR1,$BDADR
1562 021160 032737 040000 001140      BIT      #UNS,$GDDAT ;SHOULD UNS BE ON??
1563 021166 001002                      BNE      60$      ;YES!!
1564 021170 104113                      EMT      113
1565 021172 000401                      BR       70$
1566 021174                      60$:
1567 021174 104114                      EMT      114
1568
1569
1570                      ;IF THERE WERE NO PREVIOUS ERRORS, TEST FOR BIT INTERFERENCE ON
1571 021176                      ;'MDF'
1572 021176 005703                      70$:
1573 021200 001056                      TST      R3
1574 021202 012702 000001                      BNE      120$     ;BRANCH IF ANY OTHER ERRORS
1575 021206                      MOV      #1,R2    ;INITIALIZE TEST PATTERN
1576 021214 012760 000001 000024      80$:      MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
1577 021222 010260 000024 000042      MOV      #0,RMR2(R0) ;LOAD RMR2
1578 021226 016037 000042 001142      MOV      R2,RMMR1(R0) ;LOAD RMMR1
1579 021234 042737 177577 001142      MOV      RMR2(R0),$BDDAT ;STORE RMR2 AT $BDDAT
1580 021242 005037 001140                      BIC      #^CDVC,$BDDAT ;GET RESULTS
1581 021246 032702 000100                      CLR      $GDDAT    ;SETUP EXPECTED
1582 021252 001403                      BIT      #MDF,R2   ;WAS MDF SET??
1583 021254 052737 000200 001140      BEQ      90$      ;NO!!
1584 021262 023737 001140 001142      BIS      #DVC,$GDDAT ;YES-DVC SHOULD BE ON
1585 021270 001410                      90$:      CMP      $GDDAT,$BDDAT
1586 021272 010037 001136                      BEQ      100$     ;BRANCH IF DVC IS OK
1587 021276 062737 000042 001136      MOV      R0,$BDADR ;SETUP REGISTER ADDRESS
1588 021304 010237 001174                      ADD      #RMR2,$BDADR
1589 021310 104115                      MOV      R2,$TMP0 ;SAVE TEST PATTERN
1590                      EMT      115
1591                      ;SHIFT TO NEXT BIT POSITION
1592 021312                      100$:
1593 021312 042702 000001                      BIC      #DMD,R2   ;DONT SHIFT DMD
1594 021316 001002                      BNE      110$
1595 021320 012702 000001                      MOV      #DMD,R2   ;DONT TRUNCATE TEST
1596 021324 006302                      110$:     ASL      R2         ;SHIFT
1597 021326 001403                      BEQ      120$     ;EXIT IF DONE
1598 021330 052702 000001                      BIS      #DMD,R2   ;KEEP DMD ON
1599 021334 000724                      BR       80$      ;CONTINUE
1600
1601 021336                      120$:
1602                      ;END OF TEST
1603
;*****
;*TEST 33      SEEK ERROR TEST
;*****
TST33:
021336                      SCOPE
021336 000004                      NOP          ;SCOPE CALL
021340 000240
021342 012706 001100                      MOV      #STACK,SP ;LOAD THE STACK POINTER
021346 013700 001276                      MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
021352 013701 001462                      MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
021356 012737 000033 001226                      MOV      #33,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
```



```

1604
1605 021364 004737 055614 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
1606 021370 005003 CLR R3 ;CLEAR ERROR FLAGS
1607 021372 010037 001136 MOV R0,$BDADR ;SETUP REGISTER ADDRESS
1608 021376 062737 000042 001136 ADD #RMER2,$BDADR
1609
1610 ;SET DIAGNOSTIC MODE AND VERIFY THAT 'SKI' CAN BE RESET
1611 021404 012760 000001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
1612 021412 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
1613 021420 016037 000042 001142 MOV RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
1614 021426 042737 137777 001142 BIC #^CSKI,$BDDAT
1615 021434 001405 BEQ 10$ ;BRANCH IF SKI IS RESET
1616 021436 005037 001140 CLR $GDDAT ;SKI SHOULD BE ZERO
1617 021442 104116 EMT 116
1618 021444 052703 000001 BIS #BIT0,R3 ;SET ERROR FLAG
1619
1620 ;SET MAINTENANCE SEEK ERROR AND VERIFY THAT 'SKI' CAN BE SET
1621 021450 10$:
1622 021450 012760 000001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
1623 021456 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
1624 021464 012760 000201 000024 MOV #DMD!MSER,RMMR1(R0) ;LOAD RMMR1
1625 021472 016037 000042 001142 MOV RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
1626 021500 042737 137777 001142 BIC #^CSKI,$BDDAT
1627 021506 001005 BNE 20$ ;BRANCH IF SKI IS SET
1628 021510 012737 040000 001140 MOV #SKI,$GDDAT ;CANT SET SKI
1629 021516 052703 000002 BIS #BIT1,R3 ;SET ERROR FLAG
1630
1631 ;IF NO PREVIOUS ERROR, CHECK FOR BIT INTERFERENCE SETTING MAINTENANCE
1632 ;SEEK ERROR.
1633 021522 20$:
1634 021522 005703 TST R3
1635 021524 001051 BNE 70$ ;BRANCH IF ANY OTHER ERRORS
1636 021526 012702 000001 MOV #1,R2 ;INITIALIZE TEST PATTERN
1637 021532
1638 021532 012760 000001 000024 30$: MOV #DMD,RMMR1(R0) ;LOAD RMMR1
1639 021540 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
1640 021546 010260 000024 MOV R2,RMMR1(R0) ;LOAD RMMR1
1641 021552 016037 000042 001142 MOV RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
1642 021560 042737 137777 001142 BIC #^CSKI,$BDDAT ;GET SKI STATUS
1643 021566 005037 001140 CLR $GDDAT ;SETUP EXPECTED RESULT
1644 021572 032702 000200 BIT #MSER,R2
1645 021576 001403 BEQ 40$
1646 021600 052737 040000 001140 BIS #SKI,$GDDAT ;SKI SHOULD BE ON
1647 021606 023737 001140 001142 40$: CMP $GDDAT,$BDDAT
1648 021614 001403 BEQ 50$ ;BRANCH IF SKI IS OK
1649 021616 010237 001174 MOV R2,$TMPO ;SAVE TEST PATTERN
1650 021622 104120 EMT 120
1651
1652 ;ADVANCE TEST PATTERN IN R2
1653 021624 50$:
1654 021624 042702 000001 BIC #DMD,R2 ;DONT SHIFT DMD BIT
1655 021630 001002 BNE 60$
1656 021632 012702 000001 MOV #DMD,R2 ;DONT TRUNCATE TEST
1657 021636 006302 60$: ASL R2 ;SHIFT TO NEXT BIT
1658 021640 001403 BEQ 70$ ;EXIT IF DONE
1659 021642 052702 000001 BIS #DMD,R2 ;KEEP DMD ON
1660 021646 000731 BR 30$ ;CONTINUE TEST
  
```

C2
T4

```

1661
1662 021650          70$:                               ;END OF TEST
1663
1664                ;:*****
                ;*TEST 34      PIP TEST
                ;:*****
                TST34:
021650                SCOPE                               ;SCOPE CALL
021650 000004        NOP
021652 000240        MOV      #STACK,SP                   ;LOAD THE STACK POINTER
021654 012706 001100  MOV      $BASE,R0                    ;R0 = UNIBUS ADDRESS
021660 013700 001276  MOV      TSTQUE,R1                   ;R1 = POINTER TO DEVICE
021664 013701 001462  MOV      #34,$TESTN                      ;;SET TEST NUMBER IN APT MAIL BOX
021670 012737 000034 001226  MOV
1665
1666 021676 004737 055614  JSR      PC,CNTCLR                      ;GO CLEAR CONTROLLER
1667 021702 005003        CLR      R3                          ;RESET ERROR FLAGS
1668 021704 010037 001136  MOV      R0,$BDADR                      ;SETUP REGISTER ADDRESS
1669 021710 062737 000012 001136  ADD      #RMDS,$BDADR
1670
1671                ;SET MAINTENANCE ON CYLINDER 'MOC' AND VERIFY THAT 'PIP' CAN BE RESET.
1672 021716 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
1673 021724 012760 000401 000024  MOV      #DMD!MOC,RMMR1(R0) ;LOAD RMMR1
1674 021732 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
1675 021740 042737 157777 001142  BIC      #^CPIP,$BDDAT
1676 021746 001405        BEQ      10$                          ;BRANCH IF PIP IS RESET
1677 021750 005037 001140        CLR      $GDDAT
1678 021754 104121        EMT      121
1679 021756 052703 000001        BIS      #BIT0,R3                      ;SET ERROR FLAG
1680
1681                ;RESET MAINTENANCE ON CYLINDER AND VERIFY THAT 'PIP' CAN BE SET
1682 021762 10$:
1683 021762 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
1684 021770 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
1685 021776 042737 157777 001142  BIC      #^CPIP,$BDDAT
1686 022004 001006        BNE      20$                          ;BRANCH IF PIP IS SET
1687 022006 012737 020000 001140  MOV      #PIP,$GDDAT
1688 022014 104122        EMT      122
1689 022016 052703 000002        BIS      #BIT1,R3                      ;SET ERROR FLAG
1690
1691                ;IF NO PREVIOUS ERROR, TEST FOR ADJACENT BIT SETTING 'MOC'
1692 022022 20$:
1693 022022 005703        TST      R3
1694 022024 001046        BNE      70$                          ;BRANCH IF ANY PREVIOUS ERROR
1695 022026 012702 000001        MOV      #1,R2                          ;INITIALIZE TEST PATTERN
1696
1697                ;WRITE THE TEST PATTERN, CHECK MOC USING PIP
1698 022032 30$:
1699 022032 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
1700 022040 010260 000024        MOV      R2,RMMR1(R0) ;LOAD RMMR1
1701 022044 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
1702 022052 042737 157777 001142  BIC      #^CPIP,$BDDAT ;GET PIP STATUS
1703 022060 005037 001140        CLR      $GDDAT ;SETUP EXPECTED RESULT
1704 022064 032702 000400        BIT      #MOC,R2
1705 022070 001003        BNE      40$
1706 022072 052737 020000 001140  BIS      #PIP,$GDDAT ;PIP SHOULD BE SET
1707 022100 023737 001140 001142 40$:        CMP      $GDDAT,$BDDAT
    
```



```

1708 022106 001403          BEQ      50$          :BRANCH IF PIP OK
1709 022110 010237 001174   MOV      R2,$TMP0     :SAVE TEST PATTERN
1710 022114 104123          EMT      123
1711
1712          :ADVANCE THE TEST PATTERN
1713 022116 50$:
1714 022116 042702 000001   BIC      #DMD,R2      :DONT SHIFT DMD
1715 022122 001002          BNE      60$
1716 022124 012702 000001   MOV      #DMD,R2      :DONT TRUNCATE TEST
1717 022130 006302 60$:          ASL      R2            :SHIFT BIT
1718 022132 001403          BEQ      70$          :EXIT IF DONE
1719 022134 052702 000001   BIS      #DMD,R2      :KEEP DMD ON
1720 022140 000734          BR       30$          :CONTINUE TEST
1721
1722 022142 70$:          :END OF TEST
1723
1724          :*****
          :*TEST 35          EBL TEST
          :*****
          :TST35:
          SCOPE          :SCOPE CALL
          NOP
          MOV      #STACK,SP :LOAD THE STACK POINTER
          MOV      $BASE,R0  :R0 = UNIBUS ADDRESS
          MOV      TSTQUE,R1 :R1 = POINTER TO DEVICE
          MOV      #35,$TSTN  ;;SET TEST NUMBER IN APT MAIL BOX
1725
1726 022170 005003          CLR      R3            :RESET ERROR FLAGS
1727 022172 010037 001136   MOV      R0,$BDADR     :SETUP REGISTER ADDRESS
1728 022176 062737 000024 001136   ADD      #RMMR1,$BDADR
1729 022204 005037 001140          CLR      $GDDAT        :SETUP EXPECTED RESULT
1730
1731          :CLEAR AND VERIFY THAT END OF BLOCK IS RESET
1732 022210 004737 055614          JSR      PC,CNTCLR     :GO CLEAR CONTROLLER
1733 022214 016037 000024 001142   MOV      RMMR1(R0),$BDDAT :STORE RMMR1 AT $BDDAT
1734 022222 042737 157777 001142   BIC      #^CEBL,$BDDAT
1735 022230 001403          BEQ      10$          :BRANCH IF EBL IS RESET
1736 022232 104124          EMT      124
1737 022234 052703 000001   BIS      #BIT0,R3      :SET ERROR FLAG
1738
1739          :SET AND RESET DIAGNOSTIC END OF BLOCK, CHECK FOR EBL S-A-1.
1740 022240 10$:
1741 022240 012760 000001 000024   MOV      #DMD,RMMR1(R0) :LOAD RMMR1
1742 022246 012760 020001 000024   MOV      #DMD!DEBL,RMMR1(R0) :LOAD RMMR1
1743 022254 012760 000001 000024   MOV      #DMD,RMMR1(R0) :LOAD RMMR1
1744 022262 016037 000024 001142   MOV      RMMR1(R0),$BDDAT :STORE RMMR1 AT $BDDAT
1745 022270 042737 157777 001142   BIC      #^CEBL,$BDDAT
1746 022276 001403          BEQ      20$          :BRANCH IF EBL IS RESET
1747 022300 104125          EMT      125
1748 022302 052703 000001   BIS      #BIT0,R3      :SET ERROR FLAG
1749
1750          :RESET AND SET DIAGNOSTIC END OF BLOCK, CHECK FOR EBL S-A-0
1751 022306 20$:
1752 022306 012760 000001 000024   MOV      #DMD,RMMR1(R0) :LOAD RMMR1
1753 022314 012760 020001 000024   MOV      #DMD!DEBL,RMMR1(R0) :LOAD RMMR1
1754 022322 016037 000024 001142   MOV      RMMR1(R0),$BDDAT :STORE RMMR1 AT $BDDAT

```

```

1755 022330 042737 157777 001142      BIC    #^CEBL,$BDDAT
1756 022336 001006                    BNE    30$      ;BRANCH IF EBL IS SET
1757 022340 012737 020000 001140      MOV    #EBL,$GDDAT
1758 022346 104126                    EMT    126
1759 022350 052703 000002            BIS    #BIT1,R3      ;SET ERROR FLAG
1760
1761      ;IF NO PREVIOUS ERRORS, TEST FOR ADJACENT BIT INTERFERENCE ON 'DEBL'.
1762 022354 30$:
1763 022354 005703                    TST    R3
1764 022356 001042                    BNE    70$      ;BRANCH IF ANY ERROR
1765 022360 012702 000001            MOV    #1,R2      ;INITIALIZE TEST PATTERN
1766
1767      ;WRITE, READ AND VERIFY THE TEST PATTERN IN R2
1768 022364 40$:
1769 022364 012760 000001 000024      MOV    #DMD,RMMR1(R0) ;LOAD RMMR1
1770 022372 010260 000024            MOV    R2,RMMR1(R0) ;LOAD RMMR1
1771 022376 016037 000024 001142      MOV    RMMR1(R0),$BDDAT ;STORE RMMR1 AT $BDDAT
1772 022404 042737 157777 001142      BIC    #^CEBL,$BDDAT
1773 022412 010203                    MOV    R2,R3      ;GENERATE EXPECTED RESULT
1774 022414 042703 157777            BIC    #^CEBL,R3
1775 022420 020337 001142            CMP    R3,$BDDAT
1776 022424 001405                    BEQ    50$      ;BRANCH IF EBL IS OK
1777 022426 010337 001140            MOV    R3,$GDDAT ;SAVE EXPECTED RESULT
1778 022432 010237 001174            MOV    R2,$TMP0  ;SAVE TEST PATTERN
1779 022436 104127                    EMT    127
1780
1781      ;SHIFT TO NEXT BIT POSITION
1782 022440 50$:
1783 022440 042702 000001            BIC    #DMD,R2      ;DONT SHIFT DMD
1784 022444 001002                    BNE    60$
1785 022446 012702 000001            MOV    #DMD,R2      ;DONT TRUNCATE DMD
1786 022452 006302 60$:
1787 022454 001403                    ASL    R2          ;SHIFT TO NEXT BIT
1788 022456 052702 000001            BEQ    70$      ;EXIT IF DONE
1789 022462 000740                    BIS    #DMD,R2      ;KEEP DMD ON
1790
1791 022464 70$:
1792
1805
1806

```

```

*****
*TEST 36      LAST SECTOR, LAST TRACK TEST
*
*TRANSFER TEST PATTERN TO RMDA THEN VERIFY LAST SECTOR 'LS' AND LAST
*SECTOR/TRACK 'LST' FOR EACH TRANSFER. THE TABLE BELOW LISTS THE VALUE
*OF RMDA FOR WHICH LS AND LST ARE SET.
*
*
*          LS =      18 BIT MODE      16 BIT MODE
*          LS =      XXX035          XXX037
*
*RM02/03 -   LST=      002035          002037
*RM05     -   LST=      011035          011037
*****

```

```

022464 000004
022466 000240
022470 012706 001100
022474 013700 001276

TST36:
SCOPE
NOP
MOV    #STACK,SP ;LOAD THE STACK POINTER
MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS

```



```

022500 013701 001462      MOV    TSTQUE,R1      ;R1 = POINTER TO DEVICE
022504 012737 000036 001226  MOV    #36,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

1807
1808 022512 005002      CLR    R2            ;INITIALIZE TEST PATTERN
1809 022514 010037 001136  MOV    R0,$BDADR     ;SETUP REGISTER ADDRESS
1810 022520 062737 000024 001136  ADD    #RMMR1,$BDADR
1811 022526 005037 001440      CLR    RMOFO         ;START IN 18 BIT MODE
1812 022532 013737 001330 023022  MOV    LSTRK,80$     ;SETUP LAST TRACK AND
1813 022540 112737 000035 023022  MOVB   #035,80$     ;LAST SECTOR (29.)
1814 022546
1815 022546 004737 055614      JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
1816 022552 013760 001440 000032  MOV    RMOFO,RMOF(R0) ;LOAD RMOF
1817 022560 010260 000006      MOV    R2,RMDA(R0)  ;LOAD RMDA
1818 022564 016037 000024 001356  MOV    RMMR1(R0),RMMR1I ;STORE RMMR1 IN INPUT BUFFER
1819 022572 013737 001356 001142  MOV    RMMR1I,$BDDAT ;VERIFY 'LS'
1820 022600 042737 177773 001142  BIC    #^CLS,$BDDAT
1821 022606 005037 001140      CLR    $GDDAT       ;GENERATE EXPECTED 'LS'
1822 022612 032737 010000 001440  BIT    #FMT16,RMOFO  ;16 BIT MODE ?
1823 022620 001004      BNE    20$          ;YES!!
1824 022622 123702 023022      CMPB   80$,R2       ;18 BIT MODE LAST SECTOR ?
1825 022626 001007      BNE    35$          ;NO !!
1826 022630 000403      BR     30$
1827 022632 123702 023022 20$:  CMPB   80$,R2       ;16 BIT MODE LAST SECTOR ?
1828 022636 001003      BNE    35$          ;NO !!
1829
1830 022640 052737 000004 001140 30$:  BIS    #LS,$GDDAT   ;LS SHOULD BE ON-16 BIT MODE
1831 022646 023737 001140 001142 35$:  CMP    $GDDAT,$BDDAT
1832 022654 001407      BEQ    40$          ;BRANCH IF LS IS CORRECT
1833 022656 010237 001174      MOV    R2,$TMP0     ;SAVE TEST PATTERN
1834 022662 013737 001440 001176  MOV    RMOFO,$TMP1  ;SAVE OFFSET REGISTER FOR ERROR
1835 022670 104130      EMT    130
1836 022672 000454      BR     90$          ;SKIP TO NEXT
1837
1838 022674 013737 001356 001142 40$:  MOV    RMMR1I,$BDDAT ;VERIFY 'LST'
1839 022702 042737 177775 001142  BIC    #^CLST,$BDDAT
1840 022710 005037 001140      CLR    $GDDAT       ;GENERATE EXPECTED 'LST'
1841 022714 032737 010000 001440  BIT    #FMT16,RMOFO  ;16 BIT MODE??
1842 022722 001004      BNE    50$          ;YES!!
1843 022724 023702 023022      CMP    80$,R2       ;18 BIT MODE LAST TRACK/SECTOR ?
1844 022730 001007      BNE    65$          ;NO !!
1845 022732 000403      BR     60$
1846 022734 023702 023022 50$:  CMP    80$,R2       ;16 BIT MODE LAST TRACK/SECTOR ?
1847 022740 001003      BNE    65$          ;NO !!
1848
1849 022742 052737 000002 001140 60$:  BIS    #LST,$GDDAT  ;LST SHOULD BE SET
1850 022750 023737 001140 001142 65$:  CMP    $GDDAT,$BDDAT
1851 022756 001404      BEQ    70$
1852 022760 010237 001174      MOV    R2,$TMP0     ;SAVE TEST PATTERN
1853 022764 104131      EMT    131
1854 022766 000416      BR     90$          ;SKIP TO NEXT
1855
1856      ;ADVANCE TO NEXT TEST PATTERN, CHANGE TO 16 BIT MODE IF ALL
1857      ;18 BIT TESTS DONE.
1858 022770 70$:
1859 022770 005202      INC    R2            ;INCREMENT PATTERN
1860 022772 001265      BNE    10$          ;CONTINUE IF NOT DONE
1861 022774 032737 010000 001440  BIT    #FMT16,RMOFO  ;DONE 16 BIT TEST ?

```

```

1862 023002 001010          BNE      90$          ;YES!!
1863 023004 012737 010000 001440  MOV     #FMT16,RMOFO ;DO 16 BIT FORMAT TEST
1864 023012 112737 000037 023022  MOVB   #037,80$     ;SET LAST SECTOR FOR 16 BIT MODE (31.)
1865 023020 000652          BR      10$
1866
1867 023022 000000          80$:   .WORD 0          ;HOLDS LAST TRACK/SECTOR ADDRESS
1868
1869 023024          90$:
1870
1874
    
```

 ;*TEST 37 RMDA COUNT TEST

 TST37:

```

023024          SCOPE          ;SCOPE CALL
023024 000004          NOP
023026 000240          MOV     #STACK,SP    ;LOAD THE STACK POINTER
023030 012706 001100    MOV     $BASE,R0     ;R0 = UNIBUS ADDRESS
023034 013700 001276    MOV     TSTQUE,R1   ;R1 = POINTER TO DEVICE
023040 013701 001462    MOV     #37,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
023044 012737 000037 001226  MOV     R0,$BDADR   ;SETUP REGISTER ADDRESS
1875
1876 023052 010037 001136    ADD     #RMDA,$BDADR
1877 023056 062737 000006 001136    CLR     RMOFO       ;START WITH 18 BIT FORMAT
1878 023064 005037 001440    MOV     #1,$GDDAT   ;SETUP FIRST COUNT
1879 023070 012737 000001 001140    MOV     #29.,110$  ;LAST SECTOR
1880 023076 012737 000035 023476
1881
1882
1883
1884
1885
1886
1887
1888
1889 023104          ;INCREMENT SECTOR COUNT USING DIAGNOSTIC END OF BLOCK STARTING AT
1890 023104 004737 055614    ;SECTOR 0 AND CONTINUING UNTIL TRACK ADDRESS INCREMENTS
1891 023110 012760 000001 000024    .CLEAR THE MASSBUS
1892 023116 012760 000000 000014    .SET FORMAT
1893 023124 012760 000000 000042    .LOAD SECTOR AND TRACK ADDRESS
1894 023132 012760 000000 000006    .ENABLE DEBUG CLOCK
1895 023140 013760 001440 000032    .SET GO BIT
1896 023146 013737 001440 001174    10$:
1897 023154 012760 040001 000024    JSR     PC,CNTCLR   ;GO CLEAR CONTROLLER
1898 023162 012760 000001 000000    MOV     #DMD,RMMR1(R0) ;LOAD RMMR1
1899
1900
1901 023170          MOV     #0,RMER1(R0) ;LOAD RMER1
1902 023170 012760 060001 000024    MOV     #0,RMER2(R0) ;LOAD RMER2
1903 023176 012760 040001 000024    MOV     #0,RMDA(R0) ;LOAD RMDA
1904 023204 016037 000006 001142    MOV     RMOFO,RMOF(R0) ;LOAD RMOF
1905 023212 023737 001142 001140    MOV     RMOFO,$TMPO ;SAVE OFFSET FOR ERROR TYPE
1906 023220 001402          MOV     #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
1907 023222 104132          MOV     #GO,RMCS1(R0) ;LOAD RMCS1
1908 023224 000416
1909
1910
1911
    
```

 ;SET AND RESET EBL TO INCREMENT RMDA THEN VERIFY RMDA.
 25\$:
 MOV #DMD!DBEN!DEBL,RMMR1(R0) ;LOAD RMMR1
 MOV #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
 MOV RMDA(R0),\$BDDAT ;STORE RMDA AT \$BDDAT
 CMP \$BDDAT,\$GDDAT
 BEQ 30\$;BRANCH IF RMDA OK
 EMT 132
 BR 50\$;OUT OF SYNC-SKIP TO NEXT

 ;ADVANCE EXPECTED SECTOR COUNT AND CONTINUE IF ONE CYCLE NOT
 ;COMPLETE


```

1912 023226          30$:
1913 023226 005237 001140      INC      $GDDAT      ;INCREMENT EXPECTED SECTOR
1914 023232 123737 001142 023476  CMPB    $BDDAT,110$  ;WAS THE LAST SECTOR JUST COUNTED??
1915 023240 001004          BNE     40$          ;NO!!
1916 023242 105037 001140      CLRB   $GDDAT      ;YES-NEXT SECTOR SHOULD BE ZERO
1917 023246 105237 001141      INCB   $GDDAT+1    ;INCREMENT TRACK ADDRESS
1918 023252 105737 001142 40$:  TSTB   $BDDAT      ;HAS A FULL CYCLE BEEN COUNTED??
1919 023256 001401          BEQ    50$          ;YES-DO NEXT
1920 023260 000743          BR     25$          ;CONTINUE SECTOR TEST
1921
1922
1923      ;INCREMENT TRACK COUNT USING DIAGNOSTIC END OF BLOCK.  START AT TRACK 0,
1924      ;LAST SECTOR AND COUNT ONE COMPLETE TRACK CYCLE.
1924 023262          50$:
1925 023262 013737 023476 001414  MOV     110$,RMDAO  ;START SECTOR ADDRESS = 0
1926 023270 012737 000400 001140  MOV     #TA1,$GDDAT ;FIRST VALUE AFTER INCREMENT
1927
1928      :
1929      : .CLEAR THE MASSBUS
1930      : .SET FORMAT
1931      : .LOAD LAST SECTOR ADDRESS AND TEST TRACK ADDRESS
1932      : .ENABLE DEBUG CLOCK
1933      : .SET GO BIT
1933 023276          60$:
1934 023276 004737 055614          JSR     PC,CNTCLR   ;GO CLEAR CONTROLLER
1935 023302 013760 001440 000032  MOV     RMOFO,RMOF(R0) ;LOAD RMOF
1936 023310 013760 001414 000006  MOV     RMDAO,RMDA(R0) ;LOAD RMDA
1937 023316 012760 000001 000024  MOV     #DMD,RMMR1(R0) ;LOAD RMMR1
1938 023324 012760 040001 000024  MOV     #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
1939 023332 012760 000001 000000  MOV     #GO,RMCS1(R0) ;LOAD RMCS1
1940
1941      ;CLOCK RMDA USING DIAGNOSTIC END OF BLOCK
1942 023340 012760 060001 000024  MOV     #DMD!DBEN!DEBL,RMMR1(R0) ;LOAD RMMR1
1943 023346 012760 040001 000024  MOV     #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
1944
1945      ;VERIFY RMDA ACCORDING TO $GDDAT
1946 023354 016037 000006 001142  MOV     RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
1947 023362 023737 001140 001142  CMP     $GDDAT,$BDDAT
1948 023370 001402          BEQ    70$
1949 023372 104133          EMT    133
1950 023374 000441          BR     120$        ;OUT OF SYNC-SKIP TO NEXT
1951
1952      ;SETUP FOR NEXT INCREMENT OF RMDA TRACK ADDRESS
1953 023376          70$:
1954 023376 105237 001141          INCB   $GDDAT+1    ;ADVANCE EXPECTED TRACK
1955 023402 123737 001143 001331  CMPB   $BDDAT+1,LSTRK+1 ;WAS THE LAST TRACK JUST COUNTED??
1956 023410 001002          BNE     80$          ;NO!!
1957 023412 005037 001140          CLR    $GDDAT      ;YES-NEXT TRACK, SECTOR SHOULD BE ZERO
1958 023416 013737 001142 001414 80$:  MOV     $BDDAT,RMDAO ;HAS A FULL CYCLE BEEN COUNTED??
1959 023424 001404          BEQ    90$          ;YES!!
1960 023426 113737 023476 001414  MOVB   110$,RMDAO  ;INCREMENT FROM LAST SECTOR
1961 023434 000720          BR     60$
1962 023436          90$:
1963 023436 032737 010000 001440  BIT    #FMT16,RMOFO ;DONE BOTH FORMATS??
1964 023444 001015          BNE    120$        ;YES!!
1965 023446 012737 010000 001440  MOV     #FMT16,RMOFO ;SET FORMAT BIT FOR 16
1966 023454 012737 000037 023476  MOV     #31,,110$   ;SET LAST SECTOR FOR 16 BIT MODE
1967 023462 000400          BR     100$
1968

```

```

1969 023464 012737 000001 001140 100$: MOV #1,$GDDAT ;SET FIRST COUNT VALUE
1970 023472 000137 023104 JMP 10$ ;REPEAT TEST
1971
1972 023476 000000 110$: .WORD 0 ;STORAGE FOR LAST SECTOR VALUE
1973
1974 023500 120$:
1975
1976
;*****
;*TEST 40 RMDC COUNT TEST
;*****
TST40:
023500
023500 000004 SCOPE ;SCOPE CALL
023502 000240 NOP
023504 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
023510 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
023514 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
023520 012737 000040 001226 MOV #40,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1977
1978 ;CLEAR RMDC, SET FORMAT AND SETUP PROGRAM PARAMETERS
1979 023526 010037 001136 MOV R0,$BDADR ;SETUP REGISTER ADDRESS
1980 023532 062737 000034 001136 ADD #RMDC,$BDADR
1981 023540 005037 001440 CLR RMOFO ;START WITH 18 BIT FORMAT
1982 023544 013737 001330 001414 MOV LSTRK,RMDAO ;SETUP LAST TRACK AND
1983 023552 112737 000035 001414 MOV #29.,RMDAO ;LAST SECTOR
1984 023560 10$:
1985 023560 004737 055614 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
1986 023564 012737 000001 001140 MOV #1,$GDDAT ;LOAD FIRST INCREMENTAL VALUE
1987 023572 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
1988
1989 ; .CLEAR THE MASSBUS
1990 ; .SET OFFSET
1991 ; .LOAD LAST SECTOR AND TRACK ADDRESS
1992 ; .ENABLE DEBUG CLOCK
1993 ; .SET GO BIT
1994 023600 20$:
1995 023600 004737 055614 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
1996 023604 013760 001440 000032 MOV RMOFO,RMOF(R0) ;LOAD RMOF
1997 023612 013737 001440 001174 MOV RMOFO,$TMP0 ;SAVE FOR ERROR MSG
1998 023620 013760 001414 000006 MOV RMDAO,RMDA(R0) ;LOAD RMDA
1999 023626 012760 000001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
2000 023634 012760 040001 000024 MOV #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
2001 023642 012760 000001 000000 MOV #GO,RMCS1(R0) ;LOAD RMCS1
2002
2003 ;CLOCK THE CYLINDER ADDRESS USING DEBL
2004 023650 012760 060001 000024 MOV #DMD!DBEN!DEBL,RMMR1(R0) ;LOAD RMMR1
2005 023656 012760 040001 000024 MOV #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
2006 023664 016037 000034 001142 MOV RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
2007 023672 023737 001140 001142 CMP $GDDAT,$BDDAT
2008 023700 001402 BEQ 30$ ;BRANCH IF RMDC=RMDC+1
2009 023702 104140 EMT 140
2010 023704 000427 BR 60$ ;OUT OF SYNC-SKIP TO END
2011
2012 ;ADVANCE EXPECTED RESULT FOR NEXT INCREMENT
2013 023706 30$:
2014 023706 005237 001140 INC $GDDAT ;ADVANCE NEXT RESULT
2015 023712 022737 002000 001140 CMP #1024.,$GDDAT ;SHOULD NEXT VALUE BE ZERO??
    
```



```

2016 023720 001002          BNE      40$          ;NO!!
2017 023722 005037 001140    CLR      $GDDAT      ;YES-RMDC SHOULD OVERFLOW
2018 023726 005737 001142    40$:    TST      $BDDAT      ;IS ONE CYCLE COMPLETE??
2019 023732 001401          BEQ      50$          ;YES!!
2020 023734 000721          BR       20$          ;CONTINUE
2021 023736          50$:
2022 023736 032737 010000 001440    BIT      #FMT16,RMOFO ;DONE 16 BIT FORMAT MODE ?
2023 023744 001007          BNE      60$          ;YES !!
2024 023746 012737 010000 001440    MOV      #FMT16,RMOFO ;SET 16 BIT FORMAT AND
2025 023754 112737 000037 001414    MOVB    #31.,RMDAO    ;LOAD LAST SECTOR FOR 16 BIT MODE
2026 023762 000676          BR       10$          ;REPEAT TEST
2027 023764          60$:
2028
2029
    
```

 ;*TEST 41 LBT TEST

 TST41:

```

023764          SCOPE          ;SCOPE CALL
023764 000004    NOP
023766 000240    MOV      #STACK,SP    ;LOAD THE STACK POINTER
023770 012706 001100    MOV      $BASE,R0     ;R0 = UNIBUS ADDRESS
023774 013700 001276    MOV      TSTQUE,R1    ;R1 = POINTER TO DEVICE
024000 013701 001462    MOV      #41,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
024004 012737 000041 001226    MOV
2030
2031 024012 010037 001136    MOV      R0,$BDADR    ;SETUP REGISTER ADDRESS
2032 024016 062737 000012 001136    ADD      #RMDS,$BDADR
2033
2034 024024 005037 001440    CLR      RMOFO        ;START WITH 18 BIT MODE
2035 024030 013737 001330 001414    MOV      LSTRK,RMDAO  ;SETUP LAST TRACK AND
2036 024036 112737 000035 001414    MOVB    #29.,RMDAO    ;LAST SECTOR
2037
2038          :
2039          :
2040          :
2041          :
2042          :
2043          :
2044 024044          :
2045 024044 004737 055614    10$:    JSR      PC,CNTCLR    ;GO CLEAR CONTROLLER
2046 024050 012760 001466 000034    MOV      #822.,RMDC(R0) ;LOAD RMDC
2047 024056 013760 001414 000006    MOV      RMDAO,RMDA(R0) ;LOAD RMDA
2048 024064 013760 001440 000032    MOV      RMOFO,RMOF(R0) ;LOAD RMOF
2049 024072 013737 001440 001174    MOV      RMOFO,$TMPO  ;SAVE OFFSET REG FOR ERROR MSG
2050 024100 016037 000012 001142    MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
2051 024106 042737 175777 001142    BIC     #^CLBT,$BDDAT
2052 024114 001403          BEQ      20$          ;BRANCH IF LBT IS RESET
2053 024116 005037 001140    CLR      $GDDAT      ;LBT SHOULD BE ZERO
2054 024122 104141          EMT      141
2055
2056          :
2057          :
2058          :
2059          :
2060          :
2061 024124          20$:
2062 024124 012760 000001 000024    MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
    
```

```

2063 024132 012760 000000 000014      MOV      #0,RMER1(R0)      ;LOAD RMER1
2064 024140 012760 000000 000042      MOV      #0,RMER2(R0)      ;LOAD RMER2
2065 024146 012760 000001 000000      MOV      #GO,RMCS1(R0)     ;LOAD RMCS1
2066 024154 012760 060001 000024      MOV      #DMD!DBEN!DEBL,RMMR1(R0) ;LOAD RMMR1
2067 024162 012760 040001 000024      MOV      #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
2068 024170 016037 000012 001142      MOV      RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
2069 024176 042737 175777 001142      BIC      #^CLBT,$BDDAT
2070 024204 001005                                BNE      30$              ;BRANCH IF LBT IS SET
2071 024206 012737 002000 001140      MOV      #LBT,$GDDAT
2072 024214 104142                                EMT      142
2073 024216 000413                                BR       40$
2074 024220                                30$:
2075 024220 032737 010000 001440      BIT      #FMT16,RMOFO     ;DONE 16 BIT FORMAT ?
2076 024226 001007                                BNE      40$              ;YES !!
2077 024230 012737 010000 001440      MOV      #FMT16,RMOFO     ;SET 16 BIT MODE AND
2078 024236 112737 000037 001414      MOV      #31.,RMDAO       ;LAST SECTOR
2079 024244 000677                                BR       10$              ;TEST AGAIN
2080 024246                                40$:
2081
2082
:*****
:*TEST 42      COMPOSITE ERROR TEST
:*****

TST42:
024246                                SCOPE                      ;SCOPE CALL
024246 000004                                NOP
024250 000240                                MOV      #STACK,SP        ;LOAD THE STACK POINTER
024252 012706 001100                                MOV      $BASE,R0         ;R0 = UNIBUS ADDRESS
024256 013700 001276                                MOV      TSTQUE,R1        ;R1 = POINTER TO DEVICE
024262 013701 001462                                MOV      #42,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
024266 012737 000042 001226                                JSR      PC,CNTCLR        ;GO CLEAR CONTROLLER
2083                                MOV      R0,$BDADR        ;SETUP REGISTER ADDRESS
2084 024274 004737 055614                                ADD      #RMDS,$BDADR
2085 024300 010037 001136
2086 024304 062737 000012 001136
2087
2088 ;USING DIAGNOSTIC MODE, CLEAR ALL ERRORS AND VERIFY THAT COMPOSITE
2089 ;ERROR IS RESET.
2090 024312 012760 000001 000024      MOV      #DMD,RMMR1(R0)   ;LOAD RMMR1
2091 024320 012760 000000 000014      MOV      #0,RMER1(R0)     ;LOAD RMER1
2092 024326 012760 000000 000042      MOV      #0,RMER2(R0)     ;LOAD RMER2
2093 024334 016037 000012 001142      MOV      RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
2094 024342 042737 137777 001142      BIC      #^CERR,$BDDAT
2095 024350 001403                                BEQ      10$              ;BRANCH IF ERR IS RESET
2096 024352 005037 001140                                CLR      $GDDAT
2097 024356 104143                                EMT      143
2098 024360 012737 040000 001140 10$:      MOV      #ERR,$GDDAT
2099
2100 ;SET BOTH ERROR REGISTERS AND VERIFY THAT COMPOSITE ERROR IS SET
2101 024366 012760 177777 000014      MOV      #-1,RMER1(R0)    ;LOAD RMER1
2102 024374 012760 177777 000042      MOV      #-1,RMER2(R0)    ;LOAD RMER2
2103 024402 016037 000012 001142      MOV      RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
2104 024410 042737 137777 001142      BIC      #^CERR,$BDDAT
2105 024416 001001                                BNE      20$              ;BRANCH IF ERR IS SET
2106 024420 104144                                EMT      144
2107
2108 ;VERIFY THAT COMPOSITE ERROR SETS FOR EACH BIT OF RMER1
2109 024422                                20$:

```



```

2110 024422 012702 000001          MOV    #1,R2          ;INITIALIZE TEST PATTERN
2111
2112          ;WRITE THE TEST PATTERN AND VERIFY THAT ERR IS SET
2113 024426          30$:
2114 024426 004737 055614          JSR    PC,CNTCLR      ;GO CLEAR CONTROLLER
2115 024432 012760 000001 000024      MOV    #DMD,RMMR1(RO) ;LOAD RMMR1
2116 024440 012760 000000 000014      MOV    #0,RMER1(RO)   ;LOAD RMER1
2117 024446 012760 000000 000042      MOV    #0,RMER2(RO)   ;LOAD RMER2
2118 024454 010260 000014          MOV    R2,RMER1(RO)   ;LOAD RMER1
2119 024460 016037 000012 001142      MOV    RMD5(RO),SBDDAT ;STORE RMD5 AT SBDDAT
2120 024466 042737 137777 001142      BIC   #^CERR,SBDDAT
2121 024474 001005          BNE   40$            ;BRANCH IF COMPOSITE ERROR SET
2122 024476 010237 001174          MOV    R2,$TMP0       ;SAVE RMER1 TEST PATTERN
2123 024502 005037 001176          CLR   $TMP1          ;SAVE RMER2 TEST PATTERN
2124 024506 104145          EMT   145
2125
2126          ;ADVANCE THE TEST PATTERN FOR RMER1
2127 024510          40$:
2128 024510 006302          ASL   R2
2129 024512 001345          BNE   30$            ;CONTINUE IF TEST NOT DONE
2130
2131          ;VERIFY THAT COMPOSITE ERROR SETS FOR EACH BIT OF RMER2
2132 024514          50$:
2133 024514 012702 000001          MOV    #1,R2          ;INITIALIZE TEST PATTERN
2134 024520 012737 010000 001440      MOV    #FMT16,RMOFO   ;SET 16 BIT FORMAT
2135
2136          ;WRITE THE TEST PATTERN AND VERIFY THAT ERR IS SET
2137 024526          60$:
2138 024526 004737 055614          JSR    PC,CNTCLR      ;GO CLEAR CONTROLLER
2139 024532 012760 000001 000024      MOV    #DMD,RMMR1(RO) ;LOAD RMMR1
2140 024540 012760 000000 000014      MOV    #0,RMER1(RO)   ;LOAD RMER1
2141 024546 012760 000000 000042      MOV    #0,RMER2(RO)   ;LOAD RMER2
2142 024554 010260 000042          MOV    R2,RMER2(RO)   ;LOAD RMER2
2143 024560 013760 001440 000032      MOV    RMOFO,RMOF(RO) ;LOAD RMOF
2144 024566 016037 000012 001142      MOV    RMD5(RO),SBDDAT ;STORE RMD5 AT SBDDAT
2145 024574 042737 137777 001142      BIC   #^CERR,SBDDAT
2146 024602 012737 040000 001140      MOV    #ERR,$GDDAT    ;SETUP EXPECTED VALUE FOR COMP ERROR
2147 024610 032702 001567          BIT   #XNUER2,R2
2148 024614 001402          BEQ   65$            ;BRANCH IF TEST BIT IS A USED BIT
2149 024616 005037 001140          CLR   $GDDAT         ;TEST BIT IS NOT USED - ERR SHOULD BE 0
2150 024622 023737 001140 001142 65$:  CMP   $GDDAT,$BDDAT
2151 024630 001405          BEQ   70$            ;BRANCH IF COMP ERROR IS OK
2152 024632 005037 001174          CLR   $TMP0         ;SAVE RMER1 TEST PATTERN
2153 024636 010237 001176          MOV    R2,$TMP1      ;SAVE RMER2 TEST PATTERN
2154 024642 104145          EMT   145
2155
2156          ;ADVANCE THE TEST PATTERN FOR RMER2
2157 024644          70$:
2158 024644 006302          ASL   R2
2159 024646 001327          BNE   60$            ;CONTINUE IF TEST NOT DONE
2160 024650          80$:
2161
2162          ;*****
          ;*TEST 43      WRITE GO TEST
          ;*****
          ;TST43:

```

024650

```
024650 000004 SCOPE ;SCOPE CALL
024652 000240 NOP
024654 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
024660 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
024664 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
024670 012737 000043 001226 MOV #43,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2163
2164 024676 010037 001136 MOV R0,$BDADR ;COPY RMCS1 ADDRESS
2165 024702 005002 CLR R2 ;INITIALIZE FUNCTION CODE
2166
2167 ;CLEAR THE MASSBUS, SET DIAGNOSTIC MODE AND ENABLE DEBUG CLOCK
2168 024704 10$:
2169 024704 004737 055614 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
2170 024710 012760 000001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
2171 024716 012760 041001 000024 MOV #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
2172 024724 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
2173 024732 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
2174
2175 ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1, VERIFY GO IS SET
2176 024740 010203 MOV R2,R3 ;SETUP FUNCTION CODE
2177 024742 052703 000001 BIS #GO,R3
2178 024746 010360 000000 MOV R3,RMCS1(R0) ;LOAD RMCS1
2179 024752 016037 000000 001142 MOV RMCS1(R0),$BDDAT ;STORE RMCS1 AT $BDDAT
2180 024760 032737 000001 001142 BIT #GO,$BDDAT
2181 024766 001007 BNE 20$ ;BRANCH IF GO IS SET
2182
2183 ;REPORT THE ERROR-CANT SET GO WITH THIS FUNCTION CODE
2184 024770 042737 177700 001142 BIC #^CFNCMSK,$BDDAT
2185 024776 010337 001140 MOV R3,$GDDAT ;SAVE FUNCTION CODE
2186 025002 104146 EMT 146
2187 025004 000405 BR 30$
2188
2189 ;ADVANCE R2 TO THE NEXT FUNCTION CODE
2190 025006 20$:
2191 025006 062702 000002 ADD #2,R2
2192 025012 022702 000076 CMP #ILF76,R2
2193 025016 103332 BHIS 10$
2194
2195 025020 30$: ;END OF TEST
2196
2197
```

```
::*****
;*TEST 44 BRANCH MULTIPLEXOR TEST
*****
```

```
025020 TST44:
025020 000004 SCOPE ;SCOPE CALL
025022 000240 NOP
025024 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
025030 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
025034 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
025040 012737 000044 001226 MOV #44,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2198
2199 025046 010037 001136 MOV R0,$BDADR ;COPY REGISTER ADDRESS
2200 025052 062737 000040 001136 ADD #RMMR2,$BDADR
2201 025060 012702 025304 MOV #100$,R2 ;INITIALIZE TABLE POINTER
2202
2203 ;CLEAR THE MASSBUS AND SET DEBUG CLOCK ENABLE
```



```

2204 025064
2205 025064 004737 055614
2206 025070 012760 000001 000024
2207 025076 012760 041001 000024
2208 025104 012760 000000 000014
2209 025112 012760 000000 000042
2210
2211
2212
2213 025120 016037 000040 001142
2214 025126 032737 010000 001142
2215 025134 001010
2216 025136 042737 167777 001142
2217 025144 012737 010000 001140
2218 025152 104147
2219 025154 000452
2220
2221
2222
2223 025156
2224 025156 111203
2225 025160 052703 000001
2226 025164 042703 177700
2227 025170 010360 000000
2228 025174 010337 001174
2229
2230 025200 116203 000001
2231 025204 042703 177400
2232 025210
2233 025210 012760 141001 000024
2234 025216 012760 041001 000024
2235 025224 005303
2236 025226 001370
2237
2238
2239 025230 016037 000040 001142
2240 025236 042737 167777 001142
2241 025244 016237 000002 001140
2242 025252 023737 001140 001142
2243 025260 001402
2244 025262 104150
2245 025264 000406
2246
2247
2248 025266
2249 025266 062702 000004
2250 025272 105762 000001
2251 025276 100401
2252 025300 000671
2253 025302 000436
2254
2255
2256 025304
2257 025304 000
2258 025305 001
2259 025306 000000
2260

10$:
JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD,RMMR1(R0) ;LOAD RMMR1
MOV #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
MOV #0,RMER1(R0) ;LOAD RMER1
MOV #0,RMER2(R0) ;LOAD RMER2

;THE TEST BIT SHOULD BE ONE BECAUSE THE ADDRESS IS ALL ONES WHEN
;THE COMMAND SEQUENCER IS INITIALIZED.
MOV RMMR2(R0),SBDDAT ;STORE RMMR2 AT SBDDAT
BIT #TST,SBDDAT
BNE 15$ ;BRANCH IF TEST BIT IS ON
BIC #^CTST,SBDDAT ;SETUP FOR ERROR TYPE
MOV #TST,$GDDAT
EMT 147
BR 40$ ;SKIP REST OF TEST

;GET THE FUNCTION CODE FROM THE TABLE AND TRANSFER IT TO THE DEVICE,
;THEN STEP THE COMMAND SEQUENCER ACCORDING TO THE TABLE.
15$:
MOVB (R2),R3
BIS #GO,R3
BIC #^CFNCMSK,R3 ;R3=FUNCTION CODE, GO BIT
MOV R3,RMCS1(R0) ;LOAD RMCS1
MOV R3,$TMP0 ;SAVE R3 FOR ERROR MSG

MOVB 1(R2),R3 ;GET CLOCK COUNT IN R3
BIC #^C377,R3

20$:
MOV #DMD!DBEN!MUR!DBCK,RMMR1(R0) ;LOAD RMMR1
MOV #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
DEC R3 ;DECREMENT CLOCK COUNT
BNE 20$ ;ISSUE CLOCKS TILL ZERO

;GET THE TEST BIT AND COMPARE IT WITH THE TABLE ENTRY
MOV RMMR2(R0),SBDDAT ;STORE RMMR2 AT SBDDAT
BIC #^CTST,SBDDAT
MOV 2(R2),$GDDAT
CMP $GDDAT,SBDDAT
BEQ 30$ ;BRANCH IF TEST BIT OK
EMT 150
BR 40$ ;SKIP REST OF TEST

;MOVE THE TABLE POINTER AND CONTINUE IF NEXT ENTRY POSITIVE
30$:
ADD #4,R2
TSTB 1(R2)
BMI 40$ ;BRANCH IF DONE TEST
BR 10$ ;REPEAT TEST

40$:
BR 200$ ;JUMP OVER TABLE

;TABLE OF FUNCTION CODES, CLOCK COUNTS, AND TEST BITS
100$:
.BYTE NOP ;MUX ADDRESS=DATA COMMAND
.BYTE 1
.WORD 0 ;TEST BIT=0
  
```

2261	025310	000	.BYTE	NOP	;MUX ADDRESS=UNIT READY
2262	025311	002	.BYTE	2	
2263	025312	000000	.WORD	0	;TEST BIT=0
2264					
2265	025314	010	.BYTE	DRVCLR	;MUX ADDRESS=F4
2266	025315	001	.BYTE	1	
2267	025316	010000	.WORD	TST	;TEST BIT=1
2268					
2269	025320	050	.BYTE	WCD	;MUX ADDRESS=F4
2270	025321	001	.BYTE	1	
2271	025322	000000	.WORD	0	;TEST BIT=0
2272					
2273	025324	012	.BYTE	RLEASE	;MUX ADDRESS=F4
2274	025325	001	.BYTE	1	
2275	025326	010000	.WORD	TST	;TEST BIT=1
2276					
2277	025330	052	.BYTE	WCH	;MUX ADDRESS=F4
2278	025331	001	.BYTE	1	
2279	025332	000000	.WORD	0	;TEST BIT=0
2280					
2281	025334	020	.BYTE	RIP	;MUX ADDRESS=F4
2282	025335	001	.BYTE	1	
2283	025336	010000	.WORD	TST	;TEST BIT=1
2284					
2285	025340	060	.BYTE	WD	;MUX ADDRESS=F4
2286	025341	001	.BYTE	1	
2287	025342	000000	.WORD	0	;TEST BIT=0
2288					
2289	025344	022	.BYTE	PAKACK	;MUX ADDRESS=F4
2290	025345	001	.BYTE	1	
2291	025346	010000	.WORD	TST	;TEST BIT=1
2292					
2293	025350	062	.BYTE	WH	;MUX ADDRESS=F4
2294	025351	001	.BYTE	1	
2295	025352	000000	.WORD	0	;TEST BIT=0
2296					
2297	025354	030	.BYTE	SEARCH	;MUX ADDRESS=F4
2298	025355	001	.BYTE	1	
2299	025356	010000	.WORD	TST	;TEST BIT=1
2300					
2301	025360	070	.BYTE	RD	;MUX ADDRESS=F4
2302	025361	001	.BYTE	1	
2303	025362	000000	.WORD	0	;TEST BIT=0
2304					
2305	025364	032	.BYTE	ILF32	;MUX ADDRESS=F4
2306	025365	001	.BYTE	1	
2307	025366	010000	.WORD	TST	;TEST BIT=1
2308					
2309	025370	072	.BYTE	RH	;MUX ADDRESS=F4
2310	025371	001	.BYTE	1	
2311	025372	000000	.WORD	0	;TEST BIT=0
2312					
2313	025374	000	.BYTE		;END OF TABLE
2314	025375	377	.BYTE	-1	
2315	025376	000000	.WORD		
2316	025400				
2317					

200\$:

;END OF TEST

2318

: *TEST 45 SET/RESET GO TEST

TST45:

025400
025400 000004
025402 000240
025404 012706 001100
025410 013700 001276
025414 013701 001462
025420 012737 000045 001226
2319
2320 025426 012702 026042
2321
2322
2323
2324 025432
2325 025432 004737 055614
2326 025436 012760 000001 000024
2327 025444 012760 041001 000024
2328 025452 012760 000000 000014
2329 025460 012760 000000 000042
2330
2331
2332 025466 111203
2333 025470 042703 177701
2334 025474 052703 000001
2335 025500 010360 000000
2336 025504 016037 000000 001142
2337 025512 032737 000001 001142
2338 025520 001011
2339 025522 042737 177700 001142
2340 025530 010337 001140
2341 025534 010037 001136
2342 025540 104151
2343 025542 000536
2344
2345
2346 025544
2347 025544 005037 001140
2348 025550 032737 000001 001142
2349 025556 001003
2350 025560 012737 000200 001140
2351 025566
025566 016037 000012 001142
2352 025574 042737 177577 001142
2353 025602 023737 001140 001142
2354 025610 001406
2355 025612 010037 001136
2356 025616 062737 000012 001136
2357 025624 104152
2358
2359
2360 025626
2361 025626 116204 000001
2362 025632 042704 177400
2363 025636

SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #45,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
MOV #200\$,R2 ;INITIALIZE FUNCTION CODE POINTER
:CLEAR, THEN SET DIAGNOSTIC MODE, CLEAR COMPOSITE ERROR, SET MEDIUM
:ON LINE AND ENABLE DEBUG CLOCK
10\$:
JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD,RMMR1(R0) ;LOAD RMMR1
MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
MOV #0,RMER1(R0) ;LOAD RMER1
MOV #0,RMER2(R0) ;LOAD RMER2
:TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1 AND VERIFY GO IS SET
MOVB (R2),R3 ;GET FUNCTION CODE
BIC #^CILF76,R3 ;CLEAR UNUSED BITS
BIS #GO,R3 ;SET GO
MOV R3,RMCS1(R0) ;LOAD RMCS1
MOV RMCS1(R0),\$BDDAT ;STORE RMCS1 AT \$BDDAT
BIT #GO,\$BDDAT
BNE 20\$;BRANCH IF GO IS SET
BIC #^CFNCMSK,\$BDDAT
MOV R3,\$GDDAT ;SAVE EXPECTED RESULT
MOV R0,\$BDADR ;COPY REGISTER ADDRESS
EMT 151
BR 100\$
:GET READY STATUS AND VERIFY THAT IT IS THE COMPLEMENT OF GO
20\$:
CLR \$GDDAT ;EXPECT DRY TO BE OFF
BIT #GO,\$BDDAT ;WAS GO SET??
BNE 30\$;YES!!
MOV #DRY,\$GDDAT ;GO WAS NOT SET, DRY SHOULD BE
30\$:
MOV RMD5(R0),\$BDDAT ;STORE RMD5 AT \$BDDAT
BIC #^CDRY,\$BDDAT
CMP \$GDDAT,\$BDDAT
BEQ 40\$;BRANCH IF DRY IS OK
MOV R0,\$BDADR ;COPY REGISTER ADDRESS
ADD #RMD5,\$BDADR
EMT 152
:STEP THE DEBUG CLOCK AND VERIFY THAT GO REMAINS SET
40\$:
MOVB 1(R2),R4 ;GET NUMBER OF CLOCK CYCLES
BIC #^C377,R4
50\$:

```

2364 025636 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
2365 025644 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2366 025652 016037 000000 001142      MOV      RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
2367 025660 042737 177700 001142      BIC      #^CFNCMSK, $BDDAT ;CLEAR UNUSED BITS
2368 025666 010037 001136      MOV      R0, $BDADR ;SETUP REGISTER ADDRESS
2369 025672 010337 001140      MOV      R3, $GDDAT ;SAVE EXPECTED RESULT
2370
2371 ;DECREMENT CLOCK COUNT AND EXIT LOOP IF ZERO
2372 025676 005304      DEC      R4
2373 025700 001406      BEQ      60$
2374 025702 032737 000001 001142      BIT      #GO, $BDDAT ;IS GO STILL SET??
2375 025710 001352      BNE      50$ ;YES!!
2376 025712 104153      EMT      153
2377 025714 000451      BR       100$ ;OUT OF SYNC=SKIP TO NEXT
2378
2379 ;GO SHOULD NOW BE RESET AND DRY SHOULD BE SET
2380 025716 032737 000001 001142 60$:      BIT      #GO, $BDDAT ;IS GO RESET??
2381 025716 001405      BEQ      70$ ;YES!!
2382 025724 042737 000001 001140      BIC      #GO, $GDDAT ;SETUP EXPECTED RESULT
2383 025726 104154      EMT      154
2384 025734 000440      BR       100$
2385 025736
2386 025740 012737 000200 001140 70$:      MOV      #DRY, $GDDAT ;EXPECT DRIVE READY TO BE SET
2387 025740 032737 000001 001142      BIT      #GO, $BDDAT ;DID GO RESET??
2388 025746 001402      BEQ      80$ ;YES!!
2389 025754 005037 001140      CLR      $GDDAT ;GO IS SET-
2390 025756
2391 025762 016037 000012 001142 80$:      MOV      RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
2392 025770 042737 177577 001142      BIC      #^CDRY, $BDDAT
2393 025776 010037 001136      MOV      R0, $BDADR ;COPY REGISTER ADDRESS
2394 026002 062737 000012 001136      ADD      #RMDS, $BDADR
2395 026010 023737 001140 001142      CMP      $GDDAT, $BDDAT ;IS DRIVE READY OK??
2396 026016 001401      BEQ      90$ ;YES!!
2397 026020 104152      EMT      152
2398
2399 ;ADVANCE TO THE NEXT FUNCTION CODE TO BE TESTED-EXIT IF DONE
2400 026022 062702 000002 90$:      ADD      #2, R2 ;MOVE TABLE POINTER
2401 026022 105762 000001      TSTB    1(R2) ;END OF TABLE??
2402 026026 100402      BMI     100$ ;YES!!
2403 026032 000137 025432      JMP     10$ ;TEST THIS FUNCTION CODE
2404 026034 000423      BR     300$ ;GO TO NEXT TEST
2405 026040
2406
2407 ;TABLE OF FUNCTION CODES AND CLOCK COUNTS USED DURING TEST
2408 026042 002      200$: .BYTE  ILF02 ;ILLEGAL FUNCTION CODE #2
2409 026042 001      .BYTE  1
2410 026043 004      .BYTE  SEEK ;SEEK COMMAND
2411 026044 001      .BYTE  1
2412 026045 006      .BYTE  RECAL ;RECALIBRATE COMMAND
2413 026046 001      .BYTE  1
2414 026047 014      .BYTE  OFFSET ;OFFSET COMMAND
2415 026050 001      .BYTE  1
2416 026051

```



```

2420
2421 026052 016 .BYTE RTC ;RETURN TO CENTER LINE COMMAND
2422 026053 001 .BYTE 1
2423
2424 026054 024 .BYTE ILF24 ;ILLEGAL FUNCTION CODE #24
2425 026055 001 .BYTE 1
2426
2427 026056 026 .BYTE ILF26 ;ILLEGAL FUNCTION CODE #26
2428 026057 001 .BYTE 1
2429
2430 026060 034 .BYTE ILF34 ;ILLEGAL FUNCTION CODE #34
2431 026061 001 .BYTE 1
2432
2433 026062 036 .BYTE ILF36 ;ILLEGAL FUNCTION CODE #36
2434 026063 001 .BYTE 1
2435
2436 026064 042 .BYTE ILF42 ;ILLEGAL FUNCTION CODE #42
2437 026065 001 .BYTE 1
2438
2439 026066 044 .BYTE ILF44 ;ILLEGAL FUNCTION CODE #44
2440 026067 001 .BYTE 1
2441
2442 026070 046 .BYTE ILF46 ;ILLEGAL FUNCTION CODE #46
2443 026071 001 .BYTE 1
2444
2445 026072 054 .BYTE ILF54 ;ILLEGAL FUNCTION CODE #54
2446 026073 001 .BYTE 1
2447
2448 026074 056 .BYTE ILF56 ;ILLEGAL FUNCTION CODE #56
2449 026075 001 .BYTE 1
2450
2451 026076 064 .BYTE ILF64 ;ILLEGAL FUNCTION CODE #64
2452 026077 001 .BYTE 1
2453
2454 026100 066 .BYTE ILF66 ;ILLEGAL FUNCTION CODE #66
2455 026101 001 .BYTE 1
2456
2457 026102 074 .BYTE ILF74 ;ILLEGAL FUNCTION CODE #74
2458 026103 001 .BYTE 1
2459
2460 026104 076 .BYTE ILF76 ;ILLEGAL FUNCTION CODE #76
2461 026105 001 .BYTE 1
2462
2463 026106 000 .BYTE ;END OF TABLE
2464 026107 377 .BYTE -1
2465
2466 026110 300$: ;END OF TEST
2467
2468

```

```

:*****
:*TEST 46 END 1 RESET GO TEST

```

```

:*****
TST46:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS

```

```

026110
026110 000004
026112 000240
026114 012706 001100
026120 013700 001276

```

```

026124 013701 001462      MOV    TSTQUE,R1      ;R1 = POINTER TO DEVICE
026130 012737 000046 001226  MOV    #46,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
2469
2470 026136 012702 026374      MOV    #100$,R2     ;INITIALIZE TABLE POINTER
2471 026142 010037 001136      MOV    R0,$BDADR    ;COPY RMCS1 ADDRESS
2472
2473      ;CLEAR MASSBUS, THEN SET MEDIUM ON LINE AND ENABLE DEBUG CLOCK
2474 026146 10$:
2475 026146 004737 055614      JSR    PC,CNTCLR    ;GO CLEAR CONTROLLER
2476 026152 012760 000001 000024  MOV    #DMD,RMMR1(R0) ;LOAD RMMR1
2477 026160 012760 041001 000024  MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2478 026166 012760 000000 000014  MOV    #0,RMER1(R0) ;LOAD RMER1
2479 026174 012760 000000 000042  MOV    #0,RMER2(R0) ;LOAD RMER2
2480
2481      ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1, VERIFY GO IS SET
2482 026202 111203      MOVB   (R2),R3      ;GET FUNCTION CODE FROM
2483 026204 042703 177701      BIC    #^CILF76,R3  ;TABLE AND SET GO
2484 026210 052703 000001      BIS    #GO,R3
2485 026214 010337 001140      MOV    R3,$GDDAT    ;SAVE FUNCTION CODE FOR MSG
2486 026220 010360 000000      MOV    R3,RMCS1(R0) ;LOAD RMCS1
2487 026224 016037 000000 001142  MOV    RMCS1(R0),$BDDAT ;STORE RMCS1 AT $BDDAT
2488 026232 032737 000001 001142  BIT    #GO,$BDDAT
2489 026240 001005      BNE    20$         ;BRANCH IF GO IS SET
2490 026242 042737 177700 001142  BIC    #^CFNCMSK,$BDDAT
2491 026250 104151      EMT    151
2492 026252 000447      BR     60$         ;OUT OF SYNC-SKIP
2493
2494      ;GET THE NUMBER OF CLOCK CYCLES FROM THE TABLE, SAVE EXPECTED STATUS
2495 026254 20$:
2496 026254 116204 000001      MOVB   1(R2),R4     ;R4=CLOCK COUNT
2497 026260 042704 177400      BIC    #^C377,R4
2498
2499      ;STEP THE DEBUG CLOCK AND VERIFY GO STATUS ON UNTIL CLOCK COUNT EXPIRES.
2500 026264 30$:
2501 026264 012760 141001 000024  MOV    #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
2502 026272 012760 041001 000024  MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2503 026300 016037 000000 001142  MOV    RMCS1(R0),$BDDAT ;STORE RMCS1 AT $BDDAT
2504 026306 042737 177700 001142  BIC    #^CFNCMSK,$BDDAT
2505 026314 005304      DEC    R4
2506 026316 001406      BEQ    40$         ;BRANCH IF GO SHOULD BE OFF
2507 026320 032737 000001 001142  BIT    #GO,$BDDAT
2508 026326 001356      BNE    30$         ;CONTINUE IF GO IS ON
2509 026330 104153      EMT    153
2510 026332 000417      BR     60$         ;OUT OF SYNC-SKIP
2511
2512      ;VERIFY THAT GO RESET AT END1
2513 026334 40$:
2514 026334 032737 000001 001142  BIT    #GO,$BDDAT    ;DID GO RESET??
2515 026342 001405      BEQ    50$         ;YES!!
2516 026344 042737 000001 001140  BIC    #GO,$GDDAT
2517 026352 104154      EMT    154
2518 026354 000406      BR     60$
2519
2520      ;GET THE NEXT FUNCTION CODE FROM THE TABLE
2521 026356 50$:
2522 026356 062702 000002      ADD    #2,R2
2523 026362 105762 000001      TSTB  1(R2)

```



```

2524 026366 100401          BMI      60$          ;BRANCH IF END OF TABLE
2525 026370 000666          BR       10$          ;TEST THIS FUNCTION CODE
2526 026372 000404          60$:    BR       200$         ;JUMP OVER TABLE
2527
2528          ;TABLE OF FUNCTION CODES AND CLOCK COUNTS USED DURING TEST
2529 026374          100$:
2530 026374          .BYTE   RELEASE      ;RELEASE COMMAND
2531 026375          .BYTE   2
2532
2533 026376          .BYTE   SEARCH      ;SEARCH COMMAND
2534 026377          .BYTE   2
2535
2536 026400          .BYTE   ILF32       ;ILLEGAL FUNCTON #32
2537 026401          .BYTE   2
2538
2539 026402          .BYTE   ;END OF TABLE
2540 026403          .BYTE   -1
2541 026404          200$:    ;END OF TEST
2542
2543          ;:*****
          ;*TEST 47          SET PULSE TEST
          ;:*****
          ;TST47:
          SCOPE          ;SCOPE CALL
          NOP
          MOV      #STACK,SP ;LOAD THE STACK POINTER
          MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
          MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
          MOV      #47,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
2544
2545 026432 010037 001136          MOV      R0,$BDADR      ;COPY REG ADDRESS FOR MSG
2546 026436 062737 000024 001136          ADD      #RMMR1,$BDADR
2547 026444 012702 026702          MOV      #1COS,R2      ;INITIALIZE TABLE POINTER
2548
2549          ;CLEAR THE MASS BUS, ENABLE DEBUG CLOCK, AND RESET ERROR REGISTERS
2550 026450          10$:
2551 026450 004737 055614          JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
2552 026454 012760 000001 000024          MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
2553 026462 012760 041001 000024          MOV      #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
2554 026470 012760 000000 000014          MOV      #0,RMER1(R0)  ;LOAD RMER1
2555 026476 012760 000000 000042          MOV      #0,RMER2(R0)  ;LOAD RMER2
2556
2557          ;VERIFY THAT CONTINUE, "CONT" IS RESET AFTER CLEAR
2558 026504 016037 000024 001142          MOV      RMMR1(R0),$BDDAT ;STORE RMMR1 AT $BDDAT
2559 026512 042737 177677 001142          BIC      #^CCONT,$BDDAT
2560 026520 001404          BEQ      20$          ;BRANCH IF CONT WAS CLEARED
2561 026522 005037 001140          CLR      $GDDAT       ;FOR ERROR MSG
2562 026526 104155          EMT      155
2563 026530 000463          BR       70$
2564
2565          ;GET THE FUNCTION CODE FROM THE TABLE AND TRANSFER IT TO RMCS1
2566 026532          20$:
2567 026532 111203          MOV      (R2),R3
2568 026534 052703 000001          BIS      #GO,R3
2569 026540 042703 177700          BIC      #^CFNCMSK,R3 ;R3=FUNCTION CODE AND GO
2570 026544 010360 000000          MOV      R3,RMCS1(R0) ;LOAD RMCS1

```

```

2571 026550 010337 001174          MOV    R3,$TMP0          ;SAVE FUNCTION CODE FOR MSG
2572
2573          ;GET THE CLOCK COUNT FROM THE TABLE
2574 026554 116203 000001          MOV    1(R2),R3
2575 026560 042703 177400          BIC    #^C377,R3
2576
2577          ;GET THE BIT STREAM FOR CONTINUE FROM THE TABLE
2578 026564 016204 000002          MOV    2(R2),R4
2579
2580          ;STEP THE COMMAND SEQUENCER AND VERIFY CONTINUE STATUS
2581 026570 30$:
2582 026570 012760 141001 000024          MOV    #DMD!DBEN!MUR!DBCK,RMMR1(R0) ;LOAD RMMR1
2583 026576 012760 041001 000024          MOV    #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
2584 026604 016037 000024 001142          MOV    RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
2585 026612 042737 177677 001142          BIC    #^CCONT,SBDDAT
2586 026620 005037 001140          CLR    $GDDAT ;GENERATE EXPECTED CONTINUE
2587 026624 032704 000001          BIT    #BIT0,R4
2588 026630 001403          BEQ    40$
2589 026632 012737 000100 001140          MOV    #CONT,$GDDAT
2590 026640 023737 001140 001142 40$:          CMP    $GDDAT,SBDDAT
2591 026646 001402          BEQ    50$ ;BRANCH IF CONTINUE IS OK
2592 026650 104156          EMT    156
2593 026652 000412          BR     70$ ;SKIP
2594
2595          ;DECREMENT CLOCK COUNT AND SHIFT BIT STREAM
2596 026654 50$:
2597 026654 005303          DEC    R3
2598 026656 001402          BEQ    60$ ;BRANCH IF CLOCK COUNT EXPIRED
2599 026660 006204          ASR    R4 ;SHIFT TO NEXT CONTINUE BIT
2600 026662 000742          BR     30$ ;TEST NEXT CLOCK CYCLE
2601
2602          ;ADVANCE TABLE POINTER-EXIT IF DONE
2603 026664 60$:
2604 026664 062702 000004          ADD    #4,R2
2605 026670 105762 000001          TSTB  1(R2)
2606 026674 100401          BMI    70$ ;EXIT IF CLOCK COUNT NEGATIVE
2607 026676 000664          BR     10$ ;CONTINUE TEST
2608 026700 000442          BR     200$ ;JUMP OVER TABLE
2609
2610          ;TABLE OF FUNCTION CODES, CLOCK COUNTS AND CONTINUE BITS FOR TEST
2611 026702 100$:
2612 026702 000          .BYTE NOP ;NOP COMMAND
2613 026703 004          .BYTE 4 ;4 CLOCKS
2614 026704 000000          .WORD ^B0000 ;CONTINUE=0000
2615
2616 026706 002          .BYTE ILF02 ;ILLEGAL FUNCTION 2
2617 026707 002          .BYTE 2
2618 026710 000000          .WORD ^B00
2619
2620 026712 004          .BYTE SEEK ;SEEK COMMAND
2621 026713 002          .BYTE 2
2622 026714 000000          .WORD ^B00
2623
2624 026716 006          .BYTE RECAL ;RECALIBRATE COMMAND
2625 026717 002          .BYTE 2
2626 026720 000000          .WORD ^B00
2627

```



```

2628 026722 010 .BYTE DRVCLR ;DRIVE CLEAR COMMAND
2629 026723 002 .BYTE 2
2630 026724 000001 .WORD ^B01
2631
2632 026726 012 .BYTE RLEASE ;RELEASE COMMAND
2633 026727 003 .BYTE 3
2634 026730 000000 .WORD ^B000
2635
2636 026732 014 .BYTE OFFSET ;OFFSET COMMAND
2637 026733 002 .BYTE 2
2638 026734 000000 .WORD ^B00
2639
2640 026736 016 .BYTE RTC ;RETURN TO CENTER COMMAND
2641 026737 002 .BYTE 2
2642 026740 000000 .WORD ^B00
2643
2644 026742 020 .BYTE RIP ;READ IN PRESET COMMAND
2645 026743 004 .BYTE 4
2646 026744 000016 .WORD ^B1110
2647
2648 026746 022 .BYTE PAKACK ;PACK ACKNOWLEDGE
2649 026747 004 .BYTE 4
2650 026750 000016 .WORD ^B1110
2651
2652 026752 024 .BYTE ILF24 ;ILLEGAL FUNCTION 24
2653 026753 002 .BYTE 2
2654 026754 000000 .WORD ^B00
2655
2656 026756 026 .BYTE ILF26 ;ILLEGAL FUNCTION 26
2657 026757 002 .BYTE 2
2658 026760 000000 .WORD ^B00
2659
2660 026762 030 .BYTE SEARCH ;SEARCH COMMAND
2661 026763 003 .BYTE 3
2662 026764 000000 .WORD ^B000
2663
2664 026766 032 .BYTE ILF32 ;ILLEGAL FUNCTION 32
2665 026767 003 .BYTE 3
2666 026770 000000 .WORD ^B000
2667
2668 026772 034 .BYTE ILF34 ;ILLEGAL FUNCTION 34
2669 026773 002 .BYTE 2
2670 026774 000000 .WORD ^B00
2671
2672 026776 036 .BYTE ILF36 ;ILLEGAL FUNCTION 36
2673 026777 002 .BYTE 2
2674 027000 000000 .WORD ^B00
2675
2676 027002 000 .BYTE ;END OF TABLE
2677 027003 377 .BYTE -1
2678 027004 000000 .WORD
2679
2680 027006 200$: ;END OF TEST
2681
2682

```

```

*****
;*TEST 50 SET/RESET IVC TEST

```

```

*****
TST50:
027006 000004 SCOPE ;SCOPE CALL
027010 000240 NOP
027012 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
027016 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
027022 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
2683 027026 012737 000050 001226 MOV #50,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2684 027034 010037 001136 MOV R0,$BDADR ;SETUP REG ADDRESS
2685 027040 062737 000042 001136 ADD #RMER2,$BDADR
2686 027046 005002 CLR R2 ;R2=FUNCTION CODE
2687
2688 ;INITIALIZE AND VERIFY THAT IVC STATUS IS ZERO.
2689 027050 10$:
2690 027050 004737 055614 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
2691 027054 012760 000001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
2692 027062 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2693 027070 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
2694 027076 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
2695 027104 016037 000042 001142 MOV RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
2696 027112 042737 167777 001142 BIC #^CIVC,$BDDAT
2697 027120 001404 BEQ 20$ ;BRANCH IF IVC IS ZERO
2698 027122 005037 001140 CLR $GDDAT
2699 027126 104157 EMT 157
2700 027130 000444 BR 40$ ;SKIP REST OF TEST
2701
2702 ;LOAD THE FUNCTION CODE WITH GO BIT, STEP THE COMMAND SEQUENCER OFF
2703 ;ADDRESS 0 AND VERIFY IVC STATUS.
2704 027132 20$:
2705 027132 010203 MOV R2,R3 ;SETUP FUNCTION CODE
2706 027134 052703 000001 BIS #GO,R3
2707 027140 010360 000000 MOV R3,RMCS1(R0) ;LOAD RMCS1
2708 027144 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
2709 027152 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2710 027160 016037 000042 001142 MOV RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
2711 027166 042737 167777 001142 BIC #^CIVC,$BDDAT ;SET ACTUAL STATUS
2712 027174 016237 062752 001140 MOV FNCDTB(R2),$GDDAT ;SETUP EXPECTED STATUS FROM
2713 027202 042737 167777 001140 BIC #^CIVC,$GDDAT ;FUNCTION CODE TABLE
2714 027210 023737 001140 001142 CMP $GDDAT,$BDDAT
2715 027216 001403 BEQ 30$ ;BRANCH IF IVC IS OK
2716 027220 010237 001174 MOV R2,$TMP0 ;SAVE FUNCTION CODE FOR MSG
2717 027224 104160 EMT 160
2718
2719 ;ADVANCE FUNCTION CODE AND REPEAT TEST IF NOT DONE
2720 027226 30$:
2721 027226 062702 000002 ADD #2,R2
2722 027232 022702 000076 CMP #ILF76,R2
2723 027236 103401 BLO 40$ ;BRANCH IF DONE TEST
2724 027240 000703 BR 10$
2725
2726 027242 40$:
2727 ;END OF TEST
2728
*****
;*TEST 51 SET LSC TEST
*****

```



```

027242          TST51:
027242 000004          SCOPE          ;SCOPE CALL
027244 000240          NOP
027246 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
027252 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
027256 013701 001462  MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
027262 012737 000051 001226  MOV      #51,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX

2729
2730 027270 010037 001136  MOV      R0,$BDADR
2731 027274 062737 000042 001136  ADD      #RMR2,$BDADR

2732
2733          ;INITIALIZE AND VERIFY THAT LOSS OF SYSTEM CLOCK, 'LSC', IS RESET
2734 027302 004737 055614  JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
2735 027306 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
2736 027314 012760 040001 000024  MOV      #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
2737 027322 012760 000000 000014  MOV      #0,RMER1(R0)   ;LOAD RMER1
2738 027330 012760 000000 000042  MOV      #0,RMER2(R0)   ;LOAD RMER2
2739 027336 016037 000042 001142  MOV      RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
2740 027344 042737 173777 001142  BIC      #^CLSC,$BDDAT
2741 027352 001403  BEQ      10$           ;BRANCH IF LSC IS ZERO
2742 027354 005037 001140  CLR      $GDDAT
2743 027360 104161  EMT      161

2744
2745          ;WITH DEBUG CLOCK ENABLED, SET GO AND WAIT FOR ONE SHOT TO SET
2746 027362          10$:
027362 012760 000001 000000  MOV      #GO,RMCS1(R0)  ;LOAD RMCS1
2747 027370 012737 000001 001530  MOV      #1,WATCH      ;SET WATCHDOG TIMER VALUE
027376 004777 152130  JSR      PC,@CLOCK     ;START THE CLOCK
2748 027402 005737 001530  20$:  TST      WATCH
2749 027406 001375  BNE      20$           ;WAIT FOR WATCH ZERO
2750 027410 004777 152120  JSR      PC,@STOP     ;STOP THE CLOCK

2751
2752          ;ONE SHOT SHOULD BE SET-DISABLE DIAGNOSTIC CLOCK AND LSC SHOULD SET.
2753 027414 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
2754 027422 016037 000042 001142  MOV      RMER2(R0),$BDDAT ;STORE RMER2 AT $BDDAT
2755 027430 042737 173777 001142  BIC      #^CLSC,$BDDAT
2756 027436 001004  BNE      30$           ;BRANCH IF LSC SET
2757 027440 012737 004000 001140  MOV      #LSC,$GDDAT
2758 027446 104162  EMT      162

2759
2760 027450  30$:          ;END OF TEST
2761
2762          ;*****
          ;*TEST 52      DECODE TEST
          ;*****

027450          TST52:
027450 000004          SCOPE          ;SCOPE CALL
027452 000240          NOP
027454 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
027460 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
027464 013701 001462  MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
027470 012737 000052 001226  MOV      #52,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX

2763
2764 027476 005037 001422  CLR      RMER10        ;NO ERROR FIRST TEST
2765 027502  5$:
2766 027502 004737 030062  JSR      PC,100$      ;INITIALIZE
    
```



```

2767
2768
2769 027506 013760 001422 000014 ;EXECUTE A PACK ACKNOWLEDGE AND CHECK VOLUME VALID
2770 027514 012760 000023 000000     MOV    RMER10,RMER1(R0)      ;LOAD RMER1
2771 027522 012703 000003           MOV    #PACACK!GO, RMCS1(R0) ;LOAD RMCS1
2772 027526           MOV    #3,R3
2773 027526 012760 141001 000024 10$:   MOV    #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
2774 027534 012760 041001 000024     MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2775 027542 005303           DEC    R3
2776 027544 001370           BNE    10$ ;ISSUE NEXT CLOCK IF COUNT NOT 0
2777 027546 016037 000012 001142     MOV    RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
2778 027554 042737 177677 001142     BIC    #^CVV,$BDDAT
2779 027562 001414           BEQ    20$ ;BRANCH IF VV IS ZERO
2780 027564 005737 001422           TST    RMER10
2781 027570 001527           BEQ    70$ ;BRANCH IF VV SHOULD BE SET
2782 027572 005037 001140           CLR    $GDDAT ;SETUP ERROR MESSAGE
2783 027576 010037 001136           MOV    R0,$BDADR
2784 027602 062737 000012 001136     ADD    #RMDS,$BDADR
2785 027610 104163           EMT    163
2786 027612 000522           BR     80$ ;SKIP
2787 027614 004737 030062 20$:   JSR    PC,100$ ;INITIALIZE AND SET DIAGNOSTIC MODE
2788
2789 ;EXECUTE A READ IN PRESET AND CHECK VOLUME VALID
2790 027620 013760 001422 000014     MOV    RMER10,RMER1(R0)      ;LOAD RMER1
2791 027626 012760 000021 000000     MOV    #RIP!GO, RMCS1(R0)    ;LOAD RMCS1
2792 027634 012703 000003           MOV    #3,R3 ;R3=CLOCK COUNT
2793 027640           30$:   MOV    #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
2794 027646 012760 041001 000024     MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2795 027654 005303           DEC    R3
2796 027656 001370           BNE    30$ ;ISSUE NEXT CLOCK IF COUNT NOT ZERO
2797 027660 016037 000012 001142     MOV    RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
2798 027666 042737 177677 001142     BIC    #^CVV,$BDDAT
2799 027674 001414           BEQ    40$ ;BRANCH IF VOLUME VALID NOT SET
2800 027676 005737 001422           TST    RMER10
2801 027702 001462           BEQ    70$ ;BRANCH IF VOLUME VALID SHOULD BE SET
2802 027704 005037 001140           CLR    $GDDAT ;SETUP ERROR MESSAGE
2803 027710 010037 001136           MOV    R0,$BDADR
2804 027714 062737 000012 001136     ADD    #RMDS,$BDADR
2805 027722 104163           EMT    163
2806 027724 000455           BR     80$ ;SKIP
2807 027726           40$:   JSR    PC,100$ ;INITIALIZE AND SET DIAGNOSTIC MODE
2808 027726 004737 030062
2809
2810 ;EXECUTE A WRITE CHECK DATA AND CHECK OCCUPIED
2811 027732 013760 001422 000014     MOV    RMER10,RMER1(R0)      ;LOAD RMER1
2812 027740 012760 000051 000000     MOV    #WCD!GO, RMCS1(R0)    ;LOAD RMCS1
2813 027746 012703 000002           MOV    #2,R3 ;R3=CLOCK COUNT
2814 027752           50$:   MOV    #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
2815 027752 012760 141001 000024     MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2816 027760 012760 041001 000024     MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2817 027766 005303           DEC    R3
2818 027770 001370           BNE    50$ ;ISSUE NEXT CLOCK IF COUNT NOT ZERO
2819 027772 016037 000024 001142     MOV    RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
2820 030000 042737 077777 001142     BIC    #^COCC,$BDDAT
2820 030006 001414           BEQ    60$ ;BRANCH IF OCCUPIED IS RESET
    
```



```

2821 030010 005737 001422          TST      RMER10
2822 030014 001415          BEQ      70$          ;BRANCH IF OCCUPIED SHOULD BE SET
2823 030016 005037 001140          CLR      $GDDAT      ;SETUP ERROR MESSAGE
2824 030022 010037 001136          MOV      RO,$BDADR
2825 030026 062737 000024 001136          ADD      #RMMR1,$BDADR
2826 030034 104164          EMT      164
2827 030036 000410          BR       80$
2828
2829          ;VOLUME VALID AND OCCUPIED DID NOT SET-SEE IF COMP ERROR WAS ACTIVE
2830 030040 60$:
2831 030040 005737 001422          TST      RMER10
2832 030044 001005          BNE      80$          ;BRANCH IF COMP ERROR WAS SET
2833
2834          ;COULD NOT SET VV OR OCCUPIED-SUSPECT DECODE FLOP NOT SETTING
2835 030046 104165          EMT      165
2836
2837          ;REPEAT TEST WITH COMPOSITE ERROR ACTIVE-VERIFY THAT DECODE FLOP
2838          ;DOES NOT SET, AS INDICATED BY VOLUME VALID AND OCCUPIED.
2839 030050 70$:
2840 030050 012737 040000 001422          MOV      #UNS,RMER10 ;USE UNSAFE TO SET COMP ERROR
2841 030056 000611          BR       5$
2842
2843 030060 000510 80$: BR 200$          ;END OF TEST
2844
2845          ;*****
2846          ;SUBROUTINE USED DURING TEST
2847          ;*****
2848
2849          ;USING DIAGNOSTIC MODE, RESET VOLUME VALID AND COMPOSITE ERROR.
2850          ;VERIFY THAT VV, ERR, AND OCC ARE ZERO.
2851 030062 100$:
2852 030062 004737 055614          JSR      PC,CNTCLR    ;GO CLEAR CONTROLLER
2853 030066 012760 000001 000024          MOV      #DMD,RMMR1(RO) ;LOAD RMMR1
2854 030074 012760 041001 000024          MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
2855 030102 012760 000000 000014          MOV      #0,RMER1(RO) ;LOAD RMER1
2856 030110 012760 000000 000042          MOV      #0,RMER2(RO) ;LOAD RMER2
2857 030116 005037 001140          CLR      $GDDAT      ;SETUP FOR ERROR MSG
2858 030122 010037 001136          MOV      RO,$BDADR
2859 030126 062737 000012 001136          ADD      #RMDS,$BDADR
2860 030134 016037 000012 001142          MOV      RMDS(RO),$BDDAT ;STORE RMDS AT $BDDAT
2861 030142 042737 137777 001142          BIC      #^CERR,$BDDAT
2862 030150 001402          BEQ      110$        ;BRANCH IF COMP ERROR ZERO
2863 030152 104143          EMT      143
2864 030154 000447          BR       140$        ;SKIP TEST
2865 030156 110$:
2866 030164 042737 177677 001142          MOV      RMDS(RO),$BDDAT ;STORE RMDS AT $BDDAT
2867 030172 001402          BIC      #^CVV,$BDDAT
2868 030174 104135          BEQ      120$        ;BRANCH IF VOLUME VALID ZERO
2869 030176 000436          EMT      135
2870 030200 120$: BR 140$          ;SKIP TEST
2871 030200 016037 000024 001142          MOV      RMMR1(RO),$BDDAT ;STORE RMMR1 AT $BDDAT
2872 030206 042737 077777 001142          BIC      #^COCC,$BDDAT
2873 030214 001407          BEQ      130$        ;BRANCH IF OCCUPIED ZERO
2874 030216 010037 001136          MOV      RO,$BDADR   ;SETUP ERROR MESSAGE
2875 030222 062737 000024 001136          ADD      #RMMR1,$BDADR
2876 030230 104166          EMT      166
    
```

```

2876 030232 000420          BR      140$          ;SKIP TEST
2877
2878          ;TO VERIFY THAT THE DECODE FLOP IS RESET, LOAD AN ILLEGAL FUNCTION
2879          ;IN RMCS1 AND VERIFY THAT ILF DOES NOT SET.
2880 030234          130$:
2881 030234 012760 000024 000000      MOV      #ILF24,RMCS1(R0)      ;LOAD RMCS1
2882 030242 016037 000014 001142      MOV      RMER1(R0), $BDDAT      ;STORE RMER1 AT $BDDAT
2883 030250 042737 177776 001142      BIC      #^CILF, $BDDAT
2884 030256 001410          BEQ      150$          ;BRANCH IF ILF IS ZERO
2885 030260 010037 001136          MOV      R0, $BDADR          ;SETUP ERROR MESSAGE
2886 030264 062737 000014 001136      ADD      #RMER1, $BDADR
2887 030272 104167          EMT      167
2888 030274 012716 030302 140$:      MOV      #200$, (SP)          ;DONT GO BACK TO TEST
2889
2890 030300 000207 150$:      RTS      PC          ;RETURN TO TEST OR EXIT TEST
2891
2892 030302          200$:
2893
2894          ;*****
          ;*TEST 53          SET/RESET VOLUME VALID TEST
          ;*****
          ;*****
          ;TST53:
          SCOPE          ;SCOPE CALL
          NOP
          MOV      #STACK, SP      ;LOAD THE STACK POINTER
          MOV      $BASE, R0      ;R0 = UNIBUS ADDRESS
          MOV      TSTQUE, R1      ;R1 = POINTER TO DEVICE
          MOV      #53, $TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2895
2896 030330 010037 001136          MOV      R0, $BDADR          ;SETUP REGISTER ADDRESS
2897 030334 062737 000012 001136      ADD      #RMDS, $BDADR
2898 030342 012702 030546          MOV      #100$, R2          ;R2=TABLE POINTER
2899
2900          ;INITIALIZE AND USE DIAGNOSTIC MODE TO RESET VOLUME VALID
2901 030346          10$:
2902 030346 004737 055614          JSR      PC, CNTCLR          ;GO CLEAR CONTROLLER
2903 030352 012760 000001 000024      MOV      #DMD, RMMR1(R0)      ;LOAD RMMR1
2904 030360 012760 041001 000024      MOV      #DMD!MUR!DBEN, RMMR1(R0) ;LOAD RMMR1
2905 030366 012760 000000 000014      MOV      #0, RMER1(R0)        ;LOAD RMER1
2906 030374 012760 000000 000042      MOV      #0, RMER2(R0)        ;LOAD RMER2
2907 030402 016037 000012 001142      MOV      RMDS(R0), $BDDAT      ;STORE RMDS AT $BDDAT
2908 030410 042737 177677 001142      BIC      #^CVV, $BDDAT
2909 030416 001403          BEQ      20$          ;BRANCH IF VOLUME VALID ZERO
2910 030420 005037 001140          CLR      $GDDAT
2911 030424 104135          EMT      135
2912
2913          ;EXECUTE THE FUNCTION CODE IN THE TABLE
2914 030426          20$:
2915 030426 111203          MOVB     (R2), R3          ;GET FUNCTION CODE
2916 030430 042703 177701          BIC      #^CILF76, R3
2917 030434 052703 000001          BIS      #GO, R3
2918 030440 010360 000000          MOV      R3, RMCS1(R0)        ;LOAD RMCS1
2919 030444 116204 000001          MOVB     1(R2), R4          ;GET CLOCK COUNT
2920 030450 042704 177400          BIC      #^C377, R4
2921 030454          30$:
          030454 012760 141001 000024      MOV      #DMD!DBEN!MUR!DBCK, RMMR1(R0) ;LOAD RMMR1
    
```



```

2967 030642 012760 141001 000024 30$: MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
      030642 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
2968 030650 012760 041001 000024 DEC R4
2969 030656 005304 BNE 30$
2970 030660 001370 MOV RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
2971 030662 016037 000014 001142 BIC #^CILF,$BDDAT ;SETUP ACTUAL ILF STATUS
2972 030670 042737 177776 001142 MOV FNCDTB(R2), $GDDAT ;GET EXPECTED ILF STATUS
2973 030676 016237 062752 001140 BIC #^CILF,$GDDAT
2974 030704 042737 177776 001140 CMP $GDDAT,$BDDAT
2975 030712 023737 001140 001142 BEQ 40$ ;BRANCH IF ILF IS OK
2976 030720 001410 MOV R0,$BDADR ;SETUP FOR ERROR MSG
2977 030722 010037 001136 ADD #RMER1,$BDADR
2978 030726 062737 000014 001136 MOV R2,$TMP0
2979 030734 010237 001174 EMT 171
2980 030740 104171
2981
2982 ;ADVANCE TO THE NEXT FUNCTION CODE AND REPEAT TEST
2983 030742 40$:
2984 030742 062702 000002 ADD #2,R2
2985 030746 022702 000076 CMP #ILF76,R2
2986 030752 103401 BLO 50$
2987 030754 000713 BR 10$
2988 030756 50$: ;END OF TEST
2989
2990
;*****
;*TEST 55 OCCUPIED TEST
;*****
030756 TST55:
030756 000004 SCOPE ;SCOPE CALL
030760 000240 NOP
030762 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
030766 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
030772 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
030776 012737 000055 001226 MOV #55,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2991
2992 031004 005002 CLR R2 ;INITIALIZE FUNCTION CODE
2993
2994 ;GET THE DEVICE READY
2995 031006 10$:
2996 031006 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
      031012 000402 BR 20$ ;BRANCH TO 20$ IF NO ERROR
      031014 104000 EMT
2997 031016 000464 BR 50$
2998
2999 ;ENABLE DEBUG CLOCK AND LOAD THE FUNCTION CODE
3000 031020 20$:
3001 031020 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3002 031026 010203 MOV R2,R3 ;ASSEMBLE FUNCTION CODE AND
3003 031030 052703 000001 BIS #GO,R3 ;GO BIT IN R3
3004 031034 010360 000000 MOV R3,RMCS1(R0) ;LOAD RMCS1
3005 031040 012704 000002 MOV #2,R4 ;R4=CLOCK COUNT
3006
3007 ;STEP THE DEBUG CLOCK UNTIL SET PULSE IS ACTIVE
3008 031044 30$:
3009 031044 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
3010 031052 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
  
```



```

3011 031060 005304          DEC      R4
3012 031062 001370          BNE     30$                ;ISSUE NEXT CLOCK TIL R4 ZERO
3013
3014          ;VERIFY OCCUPIED STATUS
3015 031064 016037 000024 001142  MOV     RMMR1(R0), $BDDAT      ;STORE RMMR1 AT $BDDAT
3016 031072 042737 077777 001142  BIC     #^COCC, $BDDAT
3017 031100 005037 001140          CLR     $GDDAT                ;GENERATE OCC FROM AOE
3018 031104 032762 001000 062752  BIT     #AOE, FNCDTB(R2)
3019 031112 001403          BEQ     35$
3020 031114 012737 100000 001140  MOV     #OCC, $GDDAT
3021 031122 023737 001140 001142 35$:  CMP     $GDDAT, $BDDAT
3022 031130 001411          BEQ     40$                ;BRANCH IF OCC IS OK
3023 031132 010237 001174          MOV     R2, $TMP0            ;SAVE FUNCTION CODE
3024 031136 010037 001136          MOV     R0, $BDADR          ;SETUP REGISTER ADDRESS
3025 031142 062737 000024 001136  ADD     #RMMR1, $BDADR
3026 031150 104173          EMT     173
3027 031152 000406          BR      50$
3028
3029          ;ADVANCE TO NEXT FUNCTIONCODE, EXIT IF DONE
3030 031154          40$:
3031 031154 062702 000002          ADD     #2, R2
3032 031160 022702 000076          CMP     #ILF76, R2
3033 031164 103401          BLO     50$                ;EXIT IF DONE
3034 031166 000707          BR      10$
3035
3036 031170          50$:                      ;END OF TEST
3037
3038          ;*****
          ;*TEST 56      READ IN PRESET TEST
          ;*****
          ;*****
          ;*****
          TST56:
          SCOPE                      ;SCOPE CALL
          NOP
          MOV     #STACK, SP          ;LOAD THE STACK POINTER
          MOV     $BASE, R0           ;R0 = UNIBUS ADDRESS
          MOV     TSTQUE, R1          ;R1 = POINTER TO DEVICE
          MOV     #56, $TESTN         ;;SET TEST NUMBER IN APT MAIL BOX
3039
3040          ;CLEAR AND ENABLE DEBUG CLOCK - LEAVE VOLUME VALID RESET
3041 031216 004737 055614          JSR     PC, CNTCLR           ;GO CLEAR CONTROLLER
3042 031222 012760 000001 000024  MOV     #DMD, RMMR1(R0)      ;LOAD RMMR1
3043 031230 012760 041001 000024  MOV     #DMD!MUR!DBEN, RMMR1(R0) ;LOAD RMMR1
3044 031236 012760 000000 000014  MOV     #0, RMER1(R0)        ;LOAD RMER1
3045 031244 012760 000000 000042  MOV     #0, RMER2(R0)        ;LOAD RMER2
3046
3047          ;LOAD ALL ONES IN RMDA, RMDC AND RMOF
3048 031252 012760 177777 000006  MOV     #-1, RMDA(R0)        ;LOAD RMDA
3049 031260 012760 177777 000034  MOV     #-1, RMDC(R0)        ;LOAD RMDC
3050 031266 012760 177777 000032  MOV     #-1, RMOF(R0)        ;LOAD RMOF
3051
3052          ;LOAD READ IN PRESET COMMAND AND STEP THE CLOCK TILL SET PULSE
3053 031274 012760 000021 000000  MOV     #RIP!GO, RMCS1(R0)   ;LOAD RMCS1
3054 031302 012702 000003          MOV     #3, R2              ;R2=CLOCK COUNT
3055 031306
3056 031306 012760 141001 000024 10$:  MOV     #DMD!MUR!DBEN!DBCK, RMMR1(R0) ;LOAD RMMR1
3057 031314 012760 041001 000024  MOV     #DMD!MUR!DBEN, RMMR1(R0) ;LOAD RMMR1
  
```

```

3058 031322 005302          DEC      R2
3059 031324 001370          BNE     10$                ;ISSUE 3 CLOCKS
3060
3061          ;SEE IF RMDA OR RMDC OR RMOF IS ZERO
3062 031326 016002 000006    MOV     RMDA(R0),R2      ;STORE RMDA AT R2
3063 031332 005702          TST     R2
3064 031334 001413          BEQ     20$                ;BRANCH IF RMDA IS ZERO
3065 031336 016002 000034    MOV     RMDC(R0),R2      ;STORE RMDC AT R2
3066 031342 042702 176000    BIC     #XNUDC,R2        ;CLEAR UNUSED BITS
3067 031346 001406          BEQ     20$                ;BRANCH IF RMDC IS ZERO
3068 031350 016002 000032    MOV     RMOF(R0),R2     ;STORE RMOF AT R2
3069 031354 042702 161577    BIC     #XNUOF,R2        ;CLEAR UNUSED BITS
3070 031360 001401          BEQ     20$                ;BRANCH IF RMOF IS ZERO
3071
3072          ;READ IN PRESET COMMAND DIDNT CLEAR ANY OF THE 3 REGISTERS
3073 031362 104174          EMT     174
3074
3075 031364          20$:                ;END OF TEST
3076
3077          ;*****
          ;*TEST 57          RIP/RMOF TEST
          ;*****
          ;TST57:
          SCOPE                ;SCOPE CALL
          NOP
          MOV     #STACK,SP    ;LOAD THE STACK POINTER
          MOV     $BASE,R0     ;R0 = UNIBUS ADDRESS
          MOV     TSTQUE,R1    ;R1 = POINTER TO DEVICE
          MOV     #57,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
3078
3079 031412 010037 001136    MOV     R0,$BDADR        ;SETUP REGISTER ADDRESS AND
3080 031416 062737 000032 001136    ADD     #RMOF,$BDADR
3081 031424 005037 001140    CLR     $GDDAT           ;EXPECTED RMOF
3082
3083          ;INITIALIZE AND SET BITS IN RMOF
3084 031430 004737 055614    JSR     PC,CNTCLR        ;GO CLEAR CONTROLLER
3085 031434 012760 000001 000024    MOV     #DMD,RMMR1(R0)   ;LOAD RMMR1
3086 031442 012760 041001 000024    MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3087 031450 012760 000000 000014    MOV     #0,RMER1(R0)     ;LOAD RMER1
3088 031456 012760 000000 000042    MOV     #0,RMER2(R0)     ;LOAD RMER2
3089 031464 012760 177777 000032    MOV     #-1,RMOF(R0)    ;LOAD RMOF
3090
3091          ;EXECUTE A READ IN PRESET IN DIAGNOSTIC MODE TILL SET PULSE
3092 031472 012760 000021 000000    MOV     #RIP!GO,RMCS1(R0) ;LOAD RMCS1
3093 031500 012702 000003          MOV     #3,R2            ;R2=CLOCK COUNT
3094 031504          10$:
          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
3095 031512 012760 041001 000024    MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3096 031520 005302          DEC     R2
3097 031522 001370          BNE     10$                ;ISSUE 3 CLOCKS
3098
3099          ;VERIFY THAT RMOF IS ZERO
3100 031524 016037 000032 001142    MOV     RMOF(R0),$BDDAT ;STORE RMOF AT $BDDAT
3101 031532 042737 161577 001142    BIC     #XNUOF,$BDDAT
3102 031540 001401          BEQ     20$                ;BRANCH IF RMOF IS ZERO
3103 031542 104175          EMT     175

```


3104
 3105
 3106
 3107

20\$: ;END OF TEST

::*****
 :*TEST 60 RMDA/RMDC/RIP TEST

::*****
 TST60:

031544
 031544 000004
 031546 000240
 031550 012706 001100
 031554 013700 001276
 031560 013701 001462
 031564 012737 000060 001226

SCOPE ;SCOPE CALL
 NOP
 MOV #STACK,SP ;LOAD THE STACK POINTER
 MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
 MOV #60,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

3108
 3109 031572 005037 001140

CLR \$GDDAT

3110
 3111
 3112 031576 004737 055614
 3113 031602 012760 000001 000024
 3114 031610 012760 041001 000024
 3115 031616 012760 000000 000014
 3116 031624 012760 000000 000014
 3117 031632 012760 177777 000006
 3118 031640 012760 177777 000034

;CLEAR, ENABLE DEBUG CLOCK, THEN PRESET RMDA AND RMDC
 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 MOV #0,RMER1(R0) ;LOAD RMER1
 MOV #0,RMER1(R0) ;LOAD RMER1
 MOV #-1,RMDA(R0) ;LOAD RMDA
 MOV #-1,RMDC(R0) ;LOAD RMDC

3119
 3120
 3121 031646 012760 000021 000000
 3122 031654 012702 000003

;EXECUTE READ IN PRESET TILL SET PULSE
 MOV #RIP!GO,RMCS1(R0) ;LOAD RMCS1
 MOV #3,R2

3123 031660
 031660 012760 141001 000024
 3124 031666 012760 041001 000024
 3125 031674 005302
 3126 031676 001370

10\$:
 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 DEC R2
 BNE 10\$;ISSUE 3 CLOCKS

3127
 3128
 3129 031700 016037 000006 001142
 3130 031706 005737 001142
 3131 031712 001406
 3132 031714 010037 001136
 3133 031720 062737 000006 001136
 3134 031726 104176

;VERIFY RMDA IS ZERO
 MOV RMDA(R0),\$BDDAT ;STORE RMDA AT \$BDDAT
 TST \$BDDAT
 BEQ 20\$;BRANCH IF RMDA RESET
 MOV R0,\$BDADR
 ADD #RMDA,\$BDADR
 EMT 176

3135
 3136
 3137 031730
 3138 031730 016037 000034 001142
 3139 031736 042737 176000 001142
 3140 031744 001406
 3141 031746 010037 001136
 3142 031752 062737 000034 001136
 3143 031760 104177

;VERIFY RMDC IS ZERO
 20\$:
 MOV RMDC(R0),\$BDDAT ;STORE RMDC AT \$BDDAT
 BIC #XNUDC,\$BDDAT
 BEQ 30\$;BRANCH IF RMDC RESET
 MOV R0,\$BDADR
 ADD #RMDC,\$BDADR
 EMT 177

3144
 3145 031762
 3146
 3147

30\$: ;END OF TEST

::*****
 :*TEST 61 OFFSET COMMAND TEST

```

*****
TST61:
031762          SCOPE                ;SCOPE CALL
031762 000004  NOP
031764 000240  MOV #STACK,SP          ;LOAD THE STACK POINTER
031766 012706 001100  MOV $BASE,R0          ;R0 = UNIBUS ADDRESS
031772 013700 001276  MOV TSTQUE,R1        ;R1 = POINTER TO DEVICE
031776 013701 001462  MOV #61,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
032002 012737 000061 001226  MOV
3148
3149 032010 010037 001136  MOV R0,$BDADR
3150 032014 062737 000012 001136  ADD #RMDS,$BDADR
3151 032022 012737 000001 001140  MOV #OM,$GDDAT
3152
3153 032030 004737 055370  JSR PC,SETVV        ;GO SET VOLUME VALID
032034 000402  BR 10$             ;BRANCH TO 10$ IF NO ERROR
032036 104000  EMT
3154 032040 000433  BR 40$
3155 032042 10$:
032042 012760 000000 000034  MOV #0,RMDC(R0)    ;LOAD RMDC
3156
3157 ;ENABLE DEBUG CLOCK AND EXECUTE OFFSET COMMAND
3158 032050 012760 041001 000024  MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3159 032056 012760 000015 000000  MOV #OFFSET!GO,RMCS1(R0)   ;LOAD RMCS1
3160 032064 012702 000002  MOV #2,R2                ;R2=CLOCK COUNT
3161 032070 20$:
032070 012760 141001 000024  MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
3162 032076 012760 041001 000024  MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3163 032104 005302  DEC R2
3164 032106 001370  BNE 20$                ;ISSUE 2 CLOCKS
3165
3166 ;VERIFY THAT OFFSET MODE IS SET
3167 032110 016037 000012 001142  MOV RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
3168 032116 042737 177776 001142  BIC #^COM,$BDDAT
3169 032124 001001  BNE 40$                ;BRANCH IF OM IS SET
3170 032126 104200  EMT 200
3171 032130 40$:
3172 ;END OF TEST
3173
*****

```

*TEST 62 RETURN TO CENTER TEST

```

*****
TST62:
032130          SCOPE                ;SCOPE CALL
032130 000004  NOP
032132 000240  MOV #STACK,SP          ;LOAD THE STACK POINTER
032134 012706 001100  MOV $BASE,R0          ;R0 = UNIBUS ADDRESS
032140 013700 001276  MOV TSTQUE,R1        ;R1 = POINTER TO DEVICE
032144 013701 001462  MOV #62,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
032150 012737 000062 001226  MOV
3174
3175 032156 010037 001136  MOV R0,$BDADR
3176 032162 062737 000012 001136  ADD #RMDS,$BDADR
3177
3178 032170 004737 055370  JSR PC,SETVV        ;GO SET VOLUME VALID
032174 000402  BR 10$             ;BRANCH TO 10$ IF NO ERROR
032176 104000  EMT
3179 032200 000465  BR 60$
3180

```



```

3181 ;SET OFFSET DIRECTION AND OFFSET
3182 032202 10$:
3183 032202 012760 000200 000032 MOV #OFD,RMOF(R0) ;LOAD RMOF
3184 032210 004737 055512 JSR PC,SETOM ;GO SET OFFSET MODE
    032214 000401 BR 20$ ;BRANCH TO 20$ IF NO ERROR
    032216 104000 EMT

3185 ;ENABLE DEBUG CLOCK AND EXECUTE RETURN TO CENTER COMMAND
3186 20$:
3187 032220 JSR PC,SETVV ;GO SET VOLUME VALID
3188 032220 004737 055370 BR 30$ ;BRANCH TO 30$ IF NO ERROR
    032224 000402 EMT
    032226 104000 BR 60$

3189 032230 000451
3190 032232 30$:
3191 032232 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3192 032240 012760 000017 000000 MOV #RTC!GO,RMCS1(R0) ;LOAD RMCS1
3193 032246 012702 000002 MOV #2,R2
3194 032252 40$:
3195 032252 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
3196 032260 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3197 032270 005302 DEC R2
3198 032270 001370 BNE 40$ ;ISSUE 2 CLOCKS
3199

3200 032272 016037 000012 001142 ;VERIFY THAT OFFSET MODE IS RESET
3201 032300 042737 177776 001142 MOV RMDS(R0),SBDDAT ;STORE RMDS AT SBDDAT
3202 032306 001403 BIC #^COM,SBDDAT
3203 032310 005037 001140 BEQ 50$ ;BRANCH IF OFFSET MODE RESET
3204 032314 104201 CLR $GDDAT
3205 EMT 201
3206

3207 032316 ;VERIFY THAT OFFSET DIRECTION IS RESET
3208 032316 016037 000032 001142 50$:
3209 032324 042737 177577 001142 MOV RMOF(R0),SBDDAT ;STORE RMOF AT SBDDAT
3210 032332 001410 BIC #^COFD,SBDDAT
3211 032334 005037 001140 BEQ 60$ ;BRANCH IF OFD IS RESET
3212 032340 010037 001136 CLR $GDDAT
3213 032344 062737 000032 001136 MOV R0,$BDADR
3214 032352 104202 ADD #RMOF,$BDADR
3215 032354 EMT 202
3216 60$:
3217 ;END OF TEST

;*****
;*TEST 63 RMDC CLEAR OFFSET TEST
;*****

TST63:
032354 SCOPE ;SCOPE CALL
032354 000004 NOP
032356 000240 MOV #STACK,SP ;LOAD THE STACK POINTER
032360 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
032364 013700 001276 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
032370 013701 001462 MOV #63,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
032374 012737 000063 001226

3218
3219 032402 010037 001136 MOV R0,$BDADR
3220 032406 062737 000012 001136 ADD #RMDS,$BDADR
3221
3222 032414 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
    
```

```

032420 000402 BR 10$ ;BRANCH TO 10$ IF NO ERROR
032422 104000 EMT
3223 032424 000421 BR 40$ ;SKIP REST OF TEST
3224 032426 10$:
3225 032426 004737 055512 JSR PC,SETOM ;GO SET OFFSET MODE
032432 000401 BR 20$ ;BRANCH TO 20$ IF NO ERROR
032434 104000 EMT
3226
3227 ;WRITE THE DESIRED CYLINDER REGISTER AND VERIFY THAT OFFSET IS ZERO
3228 032436 20$:
3229 032436 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
3230 032444 016037 000012 001142 MOV RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
3231 032452 042737 177776 001142 BIC #^COM,$BDDAT
3232 032460 001403 BEQ 40$
3233 032462 005037 001140 CLR $GDDAT
3234 032466 104203 EMT 203
3235
3236 032470 40$: ;END OF TEST
3237
3238

```

 :*TEST 64 EBL CLEAR OFFSET TEST

```

032470 TST64:
032470 000004 SCOPE ;SCOPE CALL
032472 000240 NOP
032474 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
032500 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
032504 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
032510 012737 000064 001226 MOV #64,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
3239
3240 032516 013737 001330 001414 MOV LSTRK,RMDAO ;SETUP LAST TRACK AND
3241 032524 112737 000037 001414 MOV #31,RMDAO ;LAST SECTOR
3242 032532 010037 001136 MOV R0,$BDADR ;SETUP REGISTER FOR ERROR MSG
3243 032536 062737 000012 001136 ADD #RMDS,$BDADR
3244
3245 032544 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
032550 000402 BR 10$ ;BRANCH TO 10$ IF NO ERROR
032552 104000 EMT
3246 032554 000440 BR 30$ ;SKIP REST OF TEST IF ERROR
3247 032556 10$:
3248 032556 004737 055512 JSR PC,SETOM ;GO SET OFFSET MODE
032562 000401 BR 20$ ;BRANCH TO 20$ IF NO ERROR
032564 104000 EMT
3249 032566 20$:
3250 032566 012760 010000 000032 MOV #FMT16,RMOF(R0) ;LOAD RMOF
3251 032574 013760 001414 000006 MOV RMDAO,RMDA(R0) ;LOAD RMDA
3252
3253 ;FORCE END OF BLOCK AND VERIFY THAT OFFSET MODE IS CLEARED
3254 032602 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3255 032610 012760 000001 000000 MOV #GO,RMCS1(R0) ;LOAD RMCS1
3256 032616 012760 061001 000024 MOV #DMD!MUR!DBEN!DEBL,RMMR1(R0) ;LOAD RMMR1
3257 032624 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3258 032632 016037 000012 001142 MOV RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
3259 032640 042737 177776 001142 BIC #^COM,$BDDAT
3260 032646 001403 BEQ 30$ ;BRANCH IF OFFSET IS ZERO
3261 032650 005037 001140 CLR $GDDAT

```



```

3262 032654 104204
3263 032656
3264
3265

032656
032656 000004
032660 000240
032662 012706 001100
032666 013700 001276
032672 013701 001462
032676 012737 000065 001226

3266
3267 032704 005037 001140
3268 032710 005002
3269 032712 010037 001136
3270 032716 062737 000024 001136
3271
3272
3273 032724
3274 032724 004737 055614
3275 032730 012760 000001 000024
3276 032736 012760 040001 000024
3277 032744 012760 000000 000014
3278 032752 012760 000000 000042
3279
3280
3281 032760 010203
3282 032762 052703 000001
3283 032766 012737 000200 001530
032774 004777 146532
3284 033000 010360 000000
3285 033004
033004 016037 000024 001142
3286 033012 042737 137777 001142
3287 033020 023737 001140 001142
3288 033026 001411
3289 033030 005737 001530
3290 033034 001363
3291 033036 004777 146472
3292 033042 010237 001174
3293 033046 104205
3294 033050 000416
3295 033052
3296 033052 004777 146456
3297
3298
3299 033056 062702 000002
3300 033062 022702 000076
3301 033066 103407
3302 033070 020227 000050
3303 033074 103403
3304 033076 012737 040000 001140
3305 033104 000707
3306

```

```

30$: EMT 204 ;END OF TEST

:*****
:*TEST 65 RUN AND GO TEST
:*****

TST65:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #65,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

CLR $GDDAT ;INITIALIZE EXPECTED RESULT
CLR R2 ;INITIALIZE FUNCTION CODE
MOV R0,$BDADR
ADD #RMMR1,$BDADR

;CLEAR THE MASSBUS AND ENABLE DEBUG CLOCK
10$: JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD,RMMR1(R0) ;LOAD RMMR1
MOV #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
MOV #0,RMER1(R0) ;LOAD RMER1
MOV #0,RMER2(R0) ;LOAD RMER2

;LOAD THE FUNCTION CODE AND VERIFY RUN AND GO FLOP
MOV R2,R3 ;ASSEMBLE FUNCTION CODE AND GO
BIS #GO,R3
MOV #200,WATCH ;SET WATCHDOG TIMER VALUE
JSR PC,@CLOCK ;START THE CLOCK
MOV R3,RMCS1(R0) ;LOAD RMCS1

15$: MOV RMMR1(R0),$BDDAT ;STORE RMMR1 AT $BDDAT
BIC #^CRG,$BDDAT
CMP $GDDAT,$BDDAT
BEQ 20$ ;BRANCH IF RUN AND GO FLOP OK
TST WATCH ;TAKE ANOTHER SAMPLE IF CLOCK NOT ZERO
BNE 15$
JSR PC,@STOP ;STOP THE CLOCK
MOV R2,$TMP0 ;SAVE FUNCTION CODE FOR MSG
EMT 20$
BR 40$ ;SKIP REST OF

20$: JSR PC,@STOP ;STOP THE CLOCK

;ADVANCE TO NEXT FUNCTION CODE - EXIT IF DONE
ADD #2,R2
CMP #ILF76,R2
BLO 40$ ;EXIT IF DONE
CMP R2,#WCD ;CHANGE EXPECTED RESULT I IF
BLO 30$ ;DATA COMMAND
MOV #RG,$GDDAT
30$: BR 10$ ;REPEAT TEST

```

CZ
T1

3307 033106
3308
3309

40\$: ;END OF TEST

::*****
:*TEST 66 SET IAE TEST

::*****
TST66:

033106
033106 000004
033110 000240
033112 012706 001100
033116 013700 001276
033122 013701 001462
033126 012737 000066 001226

SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #66,\$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

3310
3311 033134 012702 033302
3312 033140
3313 033140 004737 055370
033144 000402
033146 104000
3314 033150 000453
3315
3316

10\$: MOV #100\$,R2 ;R2=TABLE POINTER

JSR PC,SETVV ;GO SET VOLUME VALID
BR 20\$;BRANCH TO 20\$ IF NO ERROR
EMT
BR 50\$;SKIP REST OF TEST

3317 033152
3318 033152 012760 177777 000006
3319 033160 012760 177777 000034
3320 033166 012760 000000 000032
3321
3322

;LOAD INVALID TRACK, SECTOR AND CYLINDER ADDRESS AND SET FORMAT TO 18
20\$:

MOV #-1,RMDA(R0) ;LOAD RMDA
MOV #-1,RMDC(R0) ;LOAD RMDC
MOV #0,RMOF(R0) ;LOAD RMOF

3323 033174 012760 041001 000024
3324 033202 111203
3325 033204 042703 177701
3326 033210 052703 000001
3327 033214 010360 000000
3328 033220 116204 000001
3329 033224 042704 177400
3330

;ENABLE DEBUG CLOCK AND LOAD FUNCTION CODE IN RMCSI WITH GO ON
MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
MOVB (R2),R3

BIC #^CILF76,R3
BIS #GO,R3
MOV R3,RMCS1(R0) ;LOAD RMCS1
MOVB 1(R2),R4 ;GET CLOCK COUNT
BIC #^C377,R4

3331
3332 033230
3333 033230 012760 141001 000024
3334 033236 012760 041001 000024
3335 033244 005304
3336 033246 001370
3337

;CLOCK THE COMMAND SEQUENCER
30\$:

MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
DEC R4
BNE 30\$

3338
3339 033250 016004 000014
3340 033254 042704 175777
3341 033260 001007
3342

;SEE IF IAE HAS SET
MOV RMER1(R0),R4 ;STORE RMER1 AT R4
BIC #^CIAE,R4
BNE 50\$;BRANCH IF IAE SET

3343
3344 033262 062702 000002
3345 033266 105762 000001
3346 033272 100401
3347 033274 000721
3348

;IAE DID NOT SET - TRY ANOTHER FUNCTION CODE

ADD #2,R2
TSTB 1(R2)
BMI 40\$;BRANCH IF ALL CODES TRIED
BR 10\$

3349
3350 033276
3351 033276 104206

;CANNOT SET IAE WITH ANY COMBINATION OF ADDRESS AND FUNCTION CODE
40\$:

EMT 206


```

3352
3353 033300 000411      50$:   BR      200$           ;JUMP OVER TABLE
3354
3355      ;TABLE OF FUNCTION CODES AND CLOCK COUNTS FOR TEST
3356 033302      100$:
3357 033302      030           .BYTE   SEARCH           ;SEARCH COMMAND
3358 033303      002           .BYTE   2
3359
3360 033304      004           .BYTE   SEEK           ;SEEK COMMAND
3361 033305      002           .BYTE   2
3362
3363 033306      062           .BYTE   WH             ;WRITE HEADER COMMAND
3364 033307      002           .BYTE   2
3365
3366 033310      052           .BYTE   WCH           ;WRITE CHECK HEADER COMMAND
3367 033311      002           .BYTE   2
3368
3369 033312      072           .BYTE   RH             ;READ HEADER COMMAND
3370 033313      002           .BYTE   2
3371
3372 033314      060           .BYTE   WD             ;WRITE DATA COMMAND
3373 033315      002           .BYTE   2
3374
3375 033316      050           .BYTE   WCD           ;WRITE CHECK DATA COMMAND
3376 033317      002           .BYTE   2
3377
3378 033320      070           .BYTE   RD             ;READ DATA COMMAND
3379 033321      002           .BYTE   2
3380
3381 033322      000           .BYTE           ;END OF TABLE
3382 033323      377           .BYTE   -1
3383
3384 033324      200$:           ;END OF TEST
3385
3386

```

```

:*****
:*TEST 67      SEARCH, SEEK, READ, WRITE TEST
:*****

```

```

033324
033324 000004
033326 000240
033330 012706 001100
033334 013700 001276
033340 013701 001462
033344 012737 000067 001226
3387
3388 033352 005002
3389 033354
3390 033354 004737 055370
033360 000402
033362 104000
3391 033364 000472
3392
3393
3394
3395 033366
3396 033366 012760 177777 000006

TST67:
SCOPE           ;SCOPE CALL
NOP
MOV #STACK,SP  ;LOAD THE STACK POINTER
MOV $BASE,R0   ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1  ;R1 = POINTER TO DEVICE
MOV #67,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

10$:   CLR      R2           ;INITIALIZE FUNCTION CODE
      JSR      PC,SETVV     ;GO SET VOLUME VALID
      BR      20$          ;BRANCH TO 20$ IF NO ERROR
      EMT
      BR      50$

;LOAD INVALID TRACK, SECTOR AND CYLINDER ADDRESS AND SET FORMAT
;TO 18 BIT MODE
20$:   MOV      #-1,RMDA(R0) ;LOAD RMDA

```

```

3397 033374 012760 177777 000034      MOV    #-1,RMDC(R0)    ;LOAD RMDC
3398 033402 012760 000000 000032      MOV    #0,RMOF(R0)    ;LOAD RMOF
3399
3400      ;ENABLE DEBUG CLOCK AND LOAD FUNCTION CODE IN RMCS1 WITH GO ON
3401 033410 012760 041001 000024      MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3402 033416 010203                MOV    R2,R3          ;ASSEMBLE CODE AND GO
3403 033420 052703 000001                BIS    #GO,R3
3404 033424 010360 000000                MOV    R3,RMCS1(R0)   ;LOAD RMCS1
3405
3406      ;CLOCK THE COMMAND SEQUENCER TO SET PULSE
3407 033430 012704 000002                MOV    #2,R4
3408 033434                30$:
3409 033442 012760 141001 000024      MOV    #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
3410 033450 005304                MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3411 033452 001370                DEC    R4
3412                        BNE    30$
3413
3414 033454 016037 000014 001142      ;VERIFY IAE ACCORDING TO FUNCTION CODE TABLE
3415 033462 042737 175777 001142      MOV    RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
3416 033470 016237 062752 001140      BIC    #^CIAE,$BDDAT
3417 033476 042737 175777 001140      MOV    FNCDTB(R2), $GDDAT ;ASSEMBLE EXPECTED IAE
3418 033504 023737 001140 001142      BIC    #^CIAE,$GDDAT
3419 033512 001411                CMP    $GDDAT,$BDDAT
3420 033514 010037 001136                BEQ    40$             ;BRANCH IF IAE OK
3421 033520 062737 000014 001136      MOV    R0,$BDADR      ;SET UP ERROR MSG
3422 033526 010237 001174      ADD    #RMER1,$BDADR
3423 033532 104207                MOV    R2,$TMPO
3424 033534 000406                EMT    207
3425                        BR     50$             ;SKIP REST OF TEST
3426
3427 033536                ;ADVANCE TO NEXT FUNCTION CODE - EXIT IF DONE
3428 033536 062702 000002                40$:
3429 033542 023702 000076      ADD    #2,R2
3430 033546 103401                CMP    ILF76,R2
3431 033550 000701                BLO    50$
3432 033552                BR     10$
3433
3434                50$:
3435                        ;END OF TEST
3436
3437      ;*****
3438      ;*TEST 70      INVALID TRACK/SECTOR TEST
3439      ;*****
3440      ;TST70:
3441      SCOPE                ;SCOPE CALL
3442      NOP
3443      MOV    #STACK,SP      ;LOAD THE STACK POINTER
3444      MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
3445      MOV    TSTQUE,R1      ;R1 = POINTER TO DEVICE
3446      MOV    #70,$TESTN     ;:SET TEST NUMBER IN APT MAIL BOX
3447
3448      MOV    LSTRK,RMDAO    ;INITIALIZE TRACK ADDRESS
3449      INCB   RMDAO+1        ;SETUP FIRST INVALID ADDRESS
3450
3451      10$:
3452      JSR    PC,SETVV       ;GO SET VOLUME VALID
3453      BR     20$           ;BRANCH TO 20$ IF NO ERROR
3454      EMT    100$
3455      BR     100$         ;SKIP REST OF TEST
    
```



```

3441
3442
3443
3444 033624
3445 033624 012760 000000 000034
3446 033632 013760 001414 000006
3447 033640 012760 000000 000032
3448
3449
3450 033646 012760 041001 000024
3451 033654 012760 000031 000000
3452 033662 012703 000002
3453 033666
3454 033666 012760 141001 000024
3455 033674 012760 041001 000024
3456 033702 005303
3457 033704 001370
3458
3459
3460 033706 016037 000014 001142
3461 033714 042737 175777 001142
3462 033722 001015
3463 033724 012737 002000 001140
3464 033732 010037 001136
3465 033736 062737 000014 001136
3466 033744 013737 001414 001174
3467 033752 104210
3468 033754 000422
3469
3470
3471 033756
3472 033756 105737 001415
3473 033762 001411
3474 033764 105237 001415
3475 033770 123727 001415 000200
3476 033776 101705
3477 034000 012737 000035 001414
3478
3479 034006 005237 001414
3480 034012 123727 001414 000200
3481 034020 101674
3482 034022
3483
3484

: CLEAR DESIRED CYLINDER, LOAD INVALID TRACK ADDRESS, AND SET
: 18 BIT FORMAT
20$:
MOV #0,RMDC(R0) ;LOAD RMDC
MOV RMDAO,RMDA(R0) ;LOAD RMDA
MOV #0,RMOF(R0) ;LOAD RMOF

: EXECUTE A SEARCH COMMAND TO WHERE "SET PULSE" IS ACTIVE
MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
MOV #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1
MOV #2,R3
30$:
MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
DEC R3
BNE 30$ ;ISSUE 2 CLOCKS

: VERIFY IAE IS SET
MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
BIC #^CIAE,SBDDAT
BNE 40$ ;BRANCH IF IAE IS ON
MOV #IAE,$GDDAT ;SETUP ERROR MESSAGE
MOV R0,$BDADR
ADD #RMER1,$BDADR
MOV RMDAO,$TMPO
EMT 210
BR 100$

: ADVANCE TO NEXT RMDA ADDRESS
40$:
TSTB RMDAO+1 ;TESTING INVALID SECTORS ?
BEQ 50$ ;YES !!
INCB RMDAO+1 ;INCREMENT TRACK ADDRESS
CMPB RMDAO+1,#128. ;DONE ?
BLOS 10$ ;NO, TEST NEXT TRACK ADDRESS
MOV #29.,RMDAO ;LOAD LOAD SECTOR ADDRESS AND
;TRACK 0.
50$:
INC RMDAO ;INCREMENT SECTOR ADDRESS
CMPB RMDAO,#128. ;DONE ?
BLOS 10$ ;NO, TEST NEXT SECTOR ADDRESS

100$:

: *****
: *TEST 71 INVALID CYLINDER TEST
: *****
TST71:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #71,$TESTN ;;SET TEST NUMBER IN API MAIL BOX

3485
3486 034050 012737 001467 001442
3487 034056
10$:
MOV #823.,RMDCO ;SET FIRST INVALID CYLINDER

```

```

3488 034056 004737 055370      JSR    PC,SETVV      ;GO SET VOLUME VALID
      034062 000402      BR     20$          ;BRANCH TO 20$ IF NO ERROR
      034064 104000      EMT
3489 034066 000460      BR     50$          ;SKIP IF ERROR
3490 034070      20$:
3491 034070 013760 001442 000034      MOV    RMDCO,RMDC(RO) ;LOAD RMDC
3492 034076 012760 000000 000006      MOV    #0,RMDA(RO)   ;LOAD RMDA
3493
3494      ;ENABLE DEBUG CLOCK AND EXECUTE SEARCH COMMAND
3495 034104 012760 041001 000024      MOV    #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
3496 034112 012760 000031 000000      MOV    #SEARCH!GO,RMCS1(RO)   ;LOAD RMCS1
3497 034120 012703 000002      MOV    #2,R3
3498 034124      30$:
      034124 012760 141001 000024      MOV    #DMD!MUR!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
3499 034132 012760 041001 000024      MOV    #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
3500 034140 005303
3501 034142 001370      DEC    R3
      BNE  30$          ;ISSUE 2 CLOCKS
3502
3503      ;VERIFY IAE IS SET
3504 034144 016037 000014 001142      MOV    RMER1(RO), $BDDAT      ;STORE RMER1 AT $BDDAT
3505 034152 042737 175777 001142      BIC    #^CIAE,$BDDAT
3506 034160 001015      BNE    40$
3507 034162 012737 002000 001140      MOV    #IAE,$GDDAT          ;BRANCH IF IAE IS SET
3508 034170 010037 001136      MOV    RO,$BDADR           ;SETUP ERROR MESSAGE
3509 034174 062737 000014 001136      ADD    #RMER1,$BDADR
3510 034202 013737 001442 001174      MOV    RMDCO,$TMPO
3511 034210 104211      EMT    211
3512 034212 000406      BR     50$
3513
3514      ;ADVANCE CYLINDER ADDRESS
3515 034214      40$:
3516 034214 005237 001442      INC    RMDCO             ;INCREMENT CYLINDER ADDRESS
3517 034220 023727 001442 002000      CMP    RMDCO,#1024.      ;DONE ?
3518 034226 103713      BLO   10$              ;NO, TEST AGAIN
3519 034230      50$:
3520
3521      ;*****
      ;*TEST 72      SET AOE TEST
      ;*****
      ;*****
      ;TST72:
      SCOPE              ;SCOPE CALL
      NOP
      MOV    #STACK,SP    ;LOAD THE STACK POINTER
      MOV    $BASE,R0     ;R0 = UNIBUS ADDRESS
      MOV    TSTQUE,R1    ;R1 = POINTER TO DEVICE
      MOV    #72,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
3522
3523 034230 000004
3524 034232 000240
3525 034234 012706 001100
3526 034240 013700 001276
3527 034244 013701 001462
3528 034250 012737 000072 001226
3529
3530 034256 005037 001440      CLR    RMOFO             ;18 BIT FORMAT MODE
3531 034262 013737 001330 001414      MOV    LSTRK,RMDAO      ;SETUP LAST TRACK AND
3532 034270 112737 000035 001414      MOVB  #29.,RMDAO       ;LAST SECTOR
3533
3534      ;ENABLE DEBUG CLOCK AND LOAD LAST SECTOR ADDRESS, MEMORY ADDRESS AND
      ;WORD COUNT, THEN LOAD WRITE DATA COMMAND WITH GO SET
3535 034276 004737 055370      10$:
      JSR    PC,SETVV      ;GO SET VOLUME VALID
      BR     15$          ;BRANCH TO 15$ IF NO ERROR
  
```



```

3531 034304 104000          EMT
3532 034306 000504          BR          40$          ;SKIP TEST IF ERROR
3533 034310 012760 041001 000024 15$:          MOV          #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3534 034316 013760 001440 000032          MOV          RMOFO,RMOF(R0) ;LOAD RMOF
3535 034324 013737 001440 001174          MOV          RMOFO,$TMP0 ;SAVE OFFSET REG FOR ERROR MSG
3536 034332 012760 001466 000034          MOV          #822.,RMDC(R0) ;LOAD RMDC
3537 034340 013760 001414 000006          MOV          RMDAO,RMDA(R0) ;LOAD RMDA
3538 034346 012760 177000 000002          MOV          #-512.,RMWC(R0) ;LOAD RMWC
3539 034354 012760 103712 000004          MOV          #BUFFER,RMBA(R0) ;LOAD RMBA
3540 034362 012760 000061 000000          MOV          #WD!GO,RMCS1(R0) ;LOAD RMCS1
3541 034370 012702 000014          MOV          #14,R2 ;R2 = CLOCK COUNT
3542                                     ;CLOCK COUNT 2-05 3-04 14-60...
3543                                     ;CLOCK THE COMMAND SEQUENCER TO GENERATE SET PULSE
3544 034374 012760 141001 000024 20$:          MOV          #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
3545 034374 012760 041001 000024          MOV          #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3546 034402 005302          DEC          R2
3547 034410 001370          BNE          20$          ;ISSUE 2 CLOCKS
3548 034412
3549
3550                                     ;FORCE EBL TO GET TO OVERFLOW ADDRESS
3551 034414 012760 061001 000024          MOV          #DMD!MUR!DBEN!DEBL,RMMR1(R0) ;LOAD RMMR1
3552 034422 012760 041001 000024          MOV          #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3553
3554                                     ;VERIFY THAT ADDRESS OVERFLOW ERROR IS SET
3555 034430 016037 000014 001142          MOV          RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
3556 034436 042737 176777 001142          BIC          #^CAOE,$BDDAT
3557 034444 001012          BNE          30$          ;BRANCH IF AOE IS SET
3558 034446 010037 001136          MOV          R0,$BDADR ;SETUP ERROR MESSAGE
3559 034452 062737 000014 001136          ADD          #RMER1,$BDADR
3560 034460 012737 001000 001140          MOV          #AOE,$GDDAT
3561 034466 104212          EMT          212
3562 034470 000413          BR          40$
3563 034472 30$:          BIT          #FMT16,RMOFO ;END OF TEST
3564 034472 032737 010000 001440          BNE          40$          ;DONE 16 BIT FORMAT TEST ?
3565 034500 001007          MOV          #FMT16,RMOFO ;YES !!
3566 034502 012737 010000 001440          MOV          #31.,RMDAO ;SET 16 BIT FORMAT MODE AND
3567 034510 112737 000037 001414          MOV          #100$,R2 ;LAST SECTOR
3568 034516 000667          BR          10$
3569 034520
3570
3571

```

 ;*TEST 73 SET RMR TEST

```

034520 TST73:          SCOPE          ;SCOPE CALL
034520 000004          NOP
034522 000240          MOV          #STACK,SP ;LOAD THE STACK POINTER
034524 012706 001100          MOV          $BASE,R0 ;R0 = UNIBUS ADDRESS
034530 013700 001276          MOV          TSTQUE,R1 ;R1 = POINTER TO DEVICE
034534 013701 001462          MOV          #73,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
034540 012737 000073 001226          MOV
3572
3573 034546 010037 001136          MOV          R0,$BDADR ;SETUP REGISTER ADDRESS FOR MSG
3574 034552 062737 000014 001136          ADD          #RMER1,$BDADR
3575 034560 012702 034726          MOV          #100$,R2 ;INITIALIZE TABLE POINTER
3576

```

```

3577          ;CLEAR THE DEVICE AND ENABLE DEBUG CLOCK
3578 034564 10$:
3579 034564 004737 055614          JSR    PC,CNTCLR          ;GO CLEAR CONTROLLER
3580 034570 012760 000001 000024  MOV    #DMD,RMMR1(R0)    ;LOAD RMMR1
3581 034576 012760 041001 000024  MOV    #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3582 034604 012760 000000 000014  MOV    #0,RMER1(R0)     ;LOAD RMER1
3583 034612 012760 000000 000042  MOV    #0,RMER2(R0)     ;LOAD RMER2
3584
3585          ;SET GO THEN WRITE THE REGISTER SPECIFIED BY THE TABLE
3586 034620 012760 000001 000000  MOV    #GO,RMCS1(R0)    ;LOAD RMCS1
3587 034626 011203          MOV    (R2),R3          ;GENERATE REGISTER ADDRESS
3588 034630 060003          ADD    R0,R3
3589 034632 012713 041001          MOV    #DMD!MUR!DBEN,(R3) ;WRITE THE REGISTER
3590
3591          ;VERIFY RMR ACCORDING TO TABLE
3592 034636 016037 000014 001142  MOV    RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
3593 034644 042737 177773 001142  BIC    #^CRMR,$BDDAT
3594 034652 016237 000002 001140  MOV    2(R2),$GDDAT    ;GET EXPECTED RESULT FROM TABLE
3595 034660 023737 001140 001142  CMP    $GDDAT,$BDDAT
3596 034666 001411          BEQ    30$             ;BRANCH IF RMR IS OK
3597 034670 011237 001174          MOV    (R2),$TMP0     ;SAVE TEST REGISTER
3598 034674 032762 000004 000002  BIT    #RMR,2(R2)
3599 034702 001002          BNE    20$             ;BRANCH IF ERROR SHOULD BE ONE
3600 034704 104213          EMT    213
3601 034706 000401          BR     30$
3602 034710 20$:
3603 034710 104214          EMT    214
3604
3605          ;ADVANCE TABLE POINTER, EXIT IF DONE
3606 034712 062702 000004 30$:
3607 034716 005712          ADD    #4,R2
3608 034720 100401          TST    (R2)
3609 034722 000720          BMI    40$             ;EXIT IF ENTRY NEGATIVE
3610 034724 000410          BR     10$
3611          40$:
3612          BR     200$          ;JUMP OVER TABLE
3613          ;TABLE OF REGISTER ADDRESSES AND RMR VALUES
3614 034726 000016 100$:
3615 034730 000000          .WORD  RMAS           ;ATTENTION SUMMARY REG
3616          .WORD  0           ;RMR = 0
3617 034732 000024          .WORD  RMMR1          ;MAINTENANCE REG
3618 034734 000000          .WORD  0           ;RMR = 0
3619
3620 034736 000006          .WORD  RMDA           ;DISK ADDRESS REG
3621 034740 000004          .WORD  RMR           ;RMR = 1
3622
3623 034742 177777          .WORD  -1            ;END OF TABLE
3624 034744 000000          .WORD  0
3625
3626 034746 200$:
3627          ;END OF TEST
3628
3629          ;*****
3630          ;*TEST 74          PGM STATUS CHECK
3631          ;*****
3632          TST74:
3633
3634          034746
  
```



```

034746 000004          SCOPE          ;SCOPE CALL
034750 000240          NOP
034752 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
034756 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
034762 013701 001462  MOV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
034766 012737 000074 001226  MOV      #74,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
3629
3630          ;CLEAR AND READ DRIVE TYPE AND DRIVE STATUS
3631 034774 004737 055614  JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
3632 035000 016037 000026 001174  MOV      RMDT(R0),$TMP0  ;STORE RMDT AT $TMP0
3633 035006 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
3634
3635          ;OMIT TEST IF DRQ IS ON - ELSE VERIFY THAT PGM IS OFF
3636 035014 032737 004000 001174  BIT      #DRQ,$TMP0
3637 035022 001014          BNE      10$            ;BRANCH IF DRQ IS ON
3638 035024 042737 176777 001142  BIC      #^CPGM,$BDDAT
3639 035032 001410          BEQ      10$            ;BRANCH IF PGM IS OFF
3640 035034 005037 001140  CLR      $GDDAT
3641 035040 010037 001136  MOV      R0,$BDADR
3642 035044 062737 000012 001134  ADD      #RMDS,$GDADR
3643 035052 104215          EMT      215
3644 035054          10$:          ;END OF TEST
3645
3646          ;:*****
          ;*TEST 75      DVA/DPR STATUS CHECK
          ;:*****
  
```

```

035054          TST75:
035054 000004          SCOPE          ;SCOPE CALL
035056 000240          NOP
035060 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
035064 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
035070 013701 001462  MOV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
035074 012737 000075 001226  MOV      #75,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
3647
3648          ;CLEAR AND VERIFY THAT DVA IS SET
3649 035102 004737 055614  JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
3650 035106 016037 000000 001142  MOV      RMCS1(R0),$BDDAT ;STORE RMCS1 AT $BDDAT
3651 035114 042737 173777 001142  BIC      #^CDVA,$BDDAT
3652 035122 001006          BNE      10$            ;BRANCH IF DVA IS ON
3653 035124 012737 004000 001140  MOV      #DVA,$GDDAT
3654 035132 010037 001136  MOV      R0,$BADR
3655 035136 104216          EMT      216
3656
3657          ;VERIFY THAT DPR IS SET
3658          10$:
3659 035140 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
3660 035146 042737 177377 001142  BIC      #^CDPR,$BDDAT
3661 035154 001011          BNE      20$
3662 035156 012737 000400 001140  MOV      #DPR,$GDDAT
3663 035164 010037 001136  MOV      R0,$BADR
3664 035170 062737 000012 001136  ADD      #RMDS,$BDADR
3665 035176 104217          EMT      217
3666
3667 035200          20$:          ;END OF TEST
3668
3669          ;:*****
  
```

;*TEST 76 PORT REQUEST TEST, PART 1

 TST76:

035200					SCOPE		;SCOPE CALL
035200	000004				NOP		
035202	000240				MOV	#STACK,SP	;LOAD THE STACK POINTER
035204	012706	001100			MOV	\$BASE,R0	;R0 = UNIBUS ADDRESS
035210	013700	001276			MOV	TSTQUE,R1	;R1 = POINTER TO DEVICE
035214	013701	001462			MOV	#76,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
035220	012737	000076	001226				
3670							
3671	035226	004737	055370		JSR	PC,SETVV	;GO SET VOLUME VALID
	035232	000402			BR	10\$;BRANCH TO 10\$ IF NO ERROR
	035234	104000			EMT		
3672	035236	000434			BR	20\$	
3673							
3674							
3675	035240						
3676	035240	012760	000013	000000	MOV	#RELEASE!GO,RMCS1(R0)	;LOAD RMCS1
3677							
3678							
3679	035246	016002	000040				
3680	035252	042702	037777		MOV	RMMR2(R0),R2	;STORE RMMR2 AT R2
3681	035256	001024			BIC	#^C<RQA!RQB>,R2	
3682					BNE	20\$	
3683							
3684	035260	016002	000000				
3685							
3686							
3687	035264	016005	000012				
3688	035270	032705	001000		MOV	RMDS(R0),R5	;STORE RMDS AT R5
3689	035274	001415			BIT	#PGM,R5	;SEE IF PGM IS SET
3690	035276	016037	000040	001142	BEQ	20\$;DONT TEST REQUEST IF PGM IS ZERO
3691	035304	042737	037777	001142	MOV	RMMR2(R0),\$BDDAT	;STORE RMMR2 AT \$BDDAT
3692	035312	001006			BIC	#^C<RQA!RQB>,\$BDDAT	
3693	035314	010037	001136		BNE	20\$;BRANCH IF REQUEST IS SET
3694	035320	062737	000040	001136	MOV	R0,\$BDADR	
3695	035326	104220			ADD	#RMMR2,\$BDADR	
3696	035330				EMT	220	
3697							
3698							

 ;*TEST 77 PORT REQUEST TEST, PART 2

 TST77:

035330					SCOPE		;SCOPE CALL
035330	000004				NOP		
035332	000240				MOV	#STACK,SP	;LOAD THE STACK POINTER
035334	012706	001100			MOV	\$BASE,R0	;R0 = UNIBUS ADDRESS
035340	013700	001276			MOV	TSTQUE,R1	;R1 = POINTER TO DEVICE
035344	013701	001462			MOV	#77,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
035350	012737	000077	001226				
3699							
3700	035356	004737	055370		JSR	PC,SETVV	;GO SET VOLUME VALID
	035362	000402			BR	10\$;BRANCH TO 10\$ IF NO ERROR
	035364	104000			EMT		
3701	035366	000435			BR	20\$	
3702							


```

3703 ;EXECUTE A RELEASE TO RESET REQUEST FLOP
3704 035370 10$:
3705 035370 012760 000013 000000 MOV #RELEASE!GO, RMCS1(R0) ;LOAD RMCS1
3706
3707 ;READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET
3708 035376 016002 000040 MOV RMMR2(R0), R2 ;STORE RMMR2 AT R2
3709 035402 042702 037777 BIC #^C<RQA!RQB>, R2
3710 035406 001025 BNE 20$
3711
3712 ;WRITE THE ATTENTION SUMMARY REGISTER TO SET REQUEST FLOP
3713 035410 012760 177777 000016 MOV #-1, RMA$(R0) ;LOAD RMA$
3714
3715 ;VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET
3716 035416 016005 000012 MOV RMDS(R0), R5 ;STORE RMDS AT R5
3717 035422 032705 001000 BIT #PGM, R5 ;SEE IF PGM IS SET
3718 035426 001415 BEQ 20$ ;DONT TEST REQUEST IF PGM IS ZERO
3719 035430 016037 000040 001142 MOV RMMR2(R0), $BDDAT ;STORE RMMR2 AT $BDDAT
3720 035436 042737 037777 001142 BIC #^C<RQA!RQB>, $BDDAT
3721 035444 001006 BNE 20$
3722 035446 010037 001136 MOV R0, $BDADR
3723 035452 062737 000040 001136 ADD #RMMR2, $BDADR
3724 035460 104221 EMT 221
3725 035462 20$:
3726
3727

```

```

:*****
:*TEST 100 PORT REQUEST TEST, PART 3

```

```

:*****
:TST100:

```

```

035462 SCOPE ;SCOPE CALL
035462 000004 NOP
035464 000240 MOV #STACK, SP ;LOAD THE STACK POINTER
035466 012706 001100 MOV $BASE, R0 ;R0 = UNIBUS ADDRESS
035472 013700 001276 MOV TSTQUE, R1 ;R1 = POINTER TO DEVICE
035476 013701 001462 MOV #100, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
035502 012737 000100 001226
3728 JSR PC, SETVV ;GO SET VOLUME VALID
3729 035510 004737 055370 BR 10$ ;BRANCH TO 10$ IF NO ERROR
035514 000402 EMT
035516 104000 BR 20$
3730 035520 000435
3731
3732

```

```

3733 ;EXECUTE A RELEASE COMMAND TO RESET REQUEST FLOP
3734 035522 012760 000013 000000 10$:
3735 MOV #RELEASE!GO, RMCS1(R0) ;LOAD RMCS1
3736
3737 ;READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET
3738 035530 016002 000040 MOV RMMR2(R0), R2 ;STORE RMMR2 AT R2
3739 035534 042702 037777 BIC #^C<RQA!RQB>, R2
3740 035540 001025 BNE 20$
3741
3742 ;WRITE RMDA TO SET REQUEST FLOP
3743 035542 012760 000000 000006 MOV #0, RMDA(R0) ;LOAD RMDA
3744
3745 ;VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET
3746 035550 016005 000012 MOV RMDS(R0), R5 ;STORE RMDS AT R5
3747 035554 032705 001000 BIT #PGM, R5 ;SEE IF PGM IS SET
035560 001415 BEQ 20$ ;DONT TEST REQUEST IF PGM IS ZERO

```

```

3748 035562 016037 000040 001142      MOV      RMMR2(R0), $BDDAT      ;STORE RMMR2 AT $BDDAT
3749 035570 042737 037777 001142      BIC      #^C<RQA!RQB>, $BDDAT
3750 035576 001006                BNE      20$
3751 035600 010037 001136      MOV      R0, $BDADR
3752 035604 062737 000040 001136      ADD      #RMMR2, $BDADR
3753 035612 104222                EMT      222
3754
3755 035614                20$:
3756
3757
::*****
:*TEST 101      RELEASE TEST
::*****

035614                TST101:
035614 000004                SCOPE                ;SCOPE CALL
035616 000240                NOP
035620 012706 001100      MOV      #STACK, SP      ;LOAD THE STACK POINTER
035624 013700 001276      MOV      $BASE, R0       ;R0 = UNIBUS ADDRESS
035630 013701 001462      MOV      TSTQUE, R1      ;R1 = POINTER TO DEVICE
035634 012737 000101 001226  MOV      #101, $TESTN    ;:SET TEST NUMBER IN APT MAIL BOX

3758
3759                ;REQUEST FLOP SHOULD SET WHEN WRITTING RMCS1
3760
3761 035642 004737 055370      JSR      PC, SETVV       ;GO SET VOLUME VALID
035646 000402                BR      10$             ;BRANCH TO 10$ IF NO ERROR
035650 104000                EMT
3762 035652 000454                BR      40$
3763
3764                ;EXECUTE A RELEASE COMMAND
3765 035654                10$:
3766 035654 012760 041001 000024  MOV      #DMD!DBEN!MUR, RMMR1(R0) ;LOAD RMMR1
3767 035662 012760 000013 000000  MOV      #RLEASE!GO, RMCS1(R0)   ;LOAD RMCS1
3768 035670 012702 000002                MOV      #2, R2
3769 035674                20$:
035674 012760 141001 000024  MOV      #DMD!DBEN!MUR!DBCK, RMMR1(R0) ;LOAD RMMR1
3770 035702 012760 041001 000024  MOV      #DMD!DBEN!MUR, RMMR1(R0) ;LOAD RMMR1
3771 035710 005302                DEC      R2
3772 035712 001370                BNE     20$             ;ISSUE 2 CLOCKS
3773
3774                ;VERIFY REQUEST FLOPS ARE RESET
3775 035714 016037 000026 001174  MOV      RMDT(R0), $TMP0 ;STORE RMDT AT $TMP0
3776 035722 016037 000040 001142  MOV      RMMR2(R0), $BDDAT ;STORE RMMR2 AT $BDDAT
3777 035730 042737 037777 001142  BIC      #^C<RQA!RQB>, $BDDAT
3778 035736 001422                BEQ     40$             ;BRANCH IF REQUESTS ARE RESET
3779 035740 032737 004000 001174  BIT      #DRQ, $TMP0
3780 035746 001410                BEQ     30$             ;BRANCH IF SINGLE PORT DEVICE
3781 035750 032737 100000 001142  BIT      #RQA, $BDDAT
3782 035756 001412                BEQ     40$             ;BRANCH IF RQA IS RESET
3783 035760 032737 040000 001142  BIT      #RQB, $BDDAT
3784 035766 001406                BEQ     40$             ;BRANCH IF RQB IS RESET
3785
3786                ;DRQ IS ZERO AND A REQUEST FLOP IS ON, OR, DRQ IS ONE AND
3787                ;BOTH REQUEST FLOPS ARE ON
3788 035770                30$:
3789 035770 010037 001136      MOV      R0, $BDADR
3790 035774 062737 000040 001136  ADD      #RMMR2, $BDADR
3791 036002 104223                EMT      223
  
```


3792 036004
3793
3794

40\$: ;END OF TEST

::*****
;*TEST 102 WRITE ATA TEST

::*****
TST102:

036004
036004 000004
036006 000240
036010 012706 001100
036014 013700 001276
036020 013701 001462
036024 012737 000102 001226

SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #102,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

3795
3796
3797 036032 004737 055614
3798 036036 012760 000001 000024
3799 036044 012760 000000 000014
3800 036052 012760 000000 000042
3801 036060 012760 001001 000024
3802
3803

;CLEAR THE DEVICE, SET DIAGNOSTIC MODE THEN SET UNIT READY

JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD,RMMR1(R0) ;LOAD RMMR1
MOV #0,RMER1(R0) ;LOAD RMER1
MOV #0,RMER2(R0) ;LOAD RMER2
MOV #DMD!MUR,RMMR1(R0) ;LOAD RMMR1

3804 036066 111102
3805 036070 042702 177770
3806 036074 116203 063052
3807 036100 042703 177400
3808 036104 010360 000016
3809
3810

;WRITE THE ATTENTION SUMMARY REGISTER

MOVB (R1),R2 ;ASSEMBLE THE ATA BIT IN R3
BIC #^CUNTMSK,R2
MOVB ATNTBL(R2),R3
BIC #^CATNMSK,R3
MOV R3,RMAS(R0) ;LOAD RMAS

3811 036110 016037 000012 001142
3812 036116 042737 077777 001142
3813 036124 001411
3814 036126 010037 001136
3815 036132 062737 000012 001136
3816 036140 005037 001140
3817 036144 104224
3818 036146 000417
3819
3820

;READ RMD5 AND VERIFY THAT ATA IS RESET

MOV RMD5(R0),\$BDDAT ;STORE RMD5 AT \$BDDAT
BIC #^CATA,\$BDDAT
BEQ 10\$;BRANCH IF ATA IS RESET
MOV R0,\$BDADR
ADD #RMD5,\$BDADR
CLR \$GDDAT
EMT 224
BR 20\$

3821 036150
3822 036150 016037 000016 001142
3823 036156 005103
3824 036160 040337 001142
3825 036164 001410
3826 036166 005037 001140
3827 036172 010037 001136
3828 036176 062737 000016 001136
3829 036204 104225
3830 036206
3831
3832

;READ RMAS AND VERIFY ATA IS RESET

10\$:
MOV RMAS(R0),\$BDDAT ;STORE RMAS AT \$BDDAT
COM R3
BIC R3,\$BDDAT
BEQ 20\$;BRANCH IF ATA IS RESET
CLR \$GDDAT
MOV R0,\$BDADR
ADD #RMAS,\$BDADR
EMT 225

20\$: ;END OF TEST

::*****
;*TEST 103 RESET ATA BY GO TEST

::*****
TST103:

036206
036206 000004
036210 000240

SCOPE ;SCOPE CALL
NOP

```
036212 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
036216 013700 001276      MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
036222 013701 001462      MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
036226 012737 000103 001226  MOV      #103,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
3833
3834 036234 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
3835 036240 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
3836 036246 012760 000000 000014  MOV      #0,RMER1(R0)   ;LOAD RMER1
3837 036254 012760 000000 000042  MOV      #0,RMER2(R0)   ;LOAD RMER2
3838 036262 012760 041001 000024  MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3839
3840 ;WITH DEBUG CLOCK ENABLED, SET GO AND VERIFY THAT ATA IS RESET
3841 036270 012760 000001 000000  MOV      #GO,RMCS1(R0)  ;LOAD RMCS1
3842 036276 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
3843 036304 042737 077777 001142  BIC      #^CATA,$BDDAT
3844 036312 001410                      BEQ      10$            ;BRANCH IF ATA IS RESET
3845 036314 005037 001140                      CLR      $GDDAT
3846 036320 010037 001136                      MOV      R0,$BDADR
3847 036324 062737 000012 001136  ADD      #RMDS,$BDADR
3848 036332 104226                      EMT      226
3849
3850 036334                      10$:                      ;END OF TEST
3851
3852
```

```
::*****
;*TEST 104      UNIT READY ATA TEST
```

```
::*****
TST104:
```

```
036334 000004      SCOPE                      ;SCOPE CALL
036336 000240      NOP
036340 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
036344 013700 001276      MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
036350 013701 001462      MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
036354 012737 000104 001226  MOV      #104,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
3853
3854 ;SET DIAGNOSTIC MODE AND CLEAR ATA
3855 036362 004737 055614      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
3856 036366 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
3857 036374 012760 000000 000014  MOV      #0,RMER1(R0)   ;LOAD RMER1
3858 036402 012760 000000 000042  MOV      #0,RMER2(R0)   ;LOAD RMER2
3859 036410 012760 177777 000016  MOV      #-1,RMAS(R0)   ;LOAD RMAS
3860
3861 ;SET UNIT READY AND VERIFY ATA IS SET
3862 036416 012760 001001 000024  MOV      #DMD!MUR,RMMR1(R0) ;LOAD RMMR1
3863 036424 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
3864 036432 042737 077777 001142  BIC      #^CATA,$BDDAT
3865 036440 001011                      BNE      10$            ;BRANCH IF ATA IS SET
3866 036442 010037 001136                      MOV      R0,$BDADR
3867 036446 062737 000012 001136  ADD      #RMDS,$BDADR
3868 036454 012737 100000 001140  MOV      #ATA,$GDDAT
3869 036462 104227                      EMT      227
3870
3871 ;CLEAR ATA, RESET UNIT READY, AND VERIFY ATA IS SET
3872 036464                      10$:
3873 036464 012760 177777 000016  MOV      #-1,RMAS(R0)   ;LOAD RMAS
3874 036472 012760 000001 000024  MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
3875 036500 016037 000012 001142  MOV      RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
```


3876 036506 042737 077777 001142
 3877 036514 001011
 3878 036516 010037 001136
 3879 036522 062737 000012 001136
 3880 036530 012737 100000 001140
 3881 036536 104230
 3882
 3883 036540
 3884
 3885

BIC #^CATA,\$BDDAT
 BNE 20\$
 MOV RO,\$BDADR
 ADD #RMDS,\$BDADR
 MOV #ATA,\$GDDAT
 EMT 230

20\$: ;END OF TEST

::*****
 :*TEST 105 ERROR ATA TEST

::*****
 TST105:

036540
 036540 000004
 036542 000240
 036544 012706 001100
 036550 013700 001276
 036554 013701 001462
 036560 012737 000105 001226
 3886
 3887
 3888 036566 004737 055614
 3889 036572 012760 000001 000024
 3890 036600 012760 000000 000014
 3891 036606 012760 000000 000042
 3892 036614 012760 177777 000016
 3893
 3894
 3895 036622 012760 177777 000014
 3896 036630 016037 000012 001142
 3897 036636 042737 077777 001142
 3898 036644 001011
 3899 036646 012737 100000 001140
 3900 036654 010037 001136
 3901 036660 062737 000012 001136
 3902 036666 104231
 3903
 3904 036670
 3905
 3906

SCOPE ;SCOPE CALL
 NOP
 MOV #STACK,SP ;LOAD THE STACK POINTER
 MOV \$BASE,RO ;RO = UNIBUS ADDRESS
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
 MOV #105,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

;CLEAR THE DEVICE AND RESET ATTENTION

JSR PC,CNTCLR ;GO CLEAR CONTROLLER
 MOV #DMD,RMMR1(RO) ;LOAD RMMR1
 MOV #0,RMER1(RO) ;LOAD RMER1
 MOV #0,RMER2(RO) ;LOAD RMER2
 MOV #-1,RMAS(RO) ;LOAD RMAS

;WRITE ONES IN ERROR REGISTER 1 AND VERIFY ATA IS SET

MOV #-1,RMER1(RO) ;LOAD RMER1
 MOV RMDS(RO),\$BDDAT ;STORE RMDS AT \$BDDAT
 BIC #^CATA,\$BDDAT
 BNE 10\$
 MOV #ATA,\$GDDAT
 MOV RO,\$BDADR
 ADD #RMDS,\$BDADR
 EMT 231

10\$: ;END OF TEST

::*****
 :*TEST 106 REGISTER TRANSFER ATA TEST

::*****
 TST106:

036670
 036670 000004
 036672 000240
 036674 012706 001100
 036700 013700 001276
 036704 013701 001462
 036710 012737 000106 001226
 3907
 3908 036716 012702 037044
 3909
 3910
 3911 036722
 3912 036722 004737 055614

SCOPE ;SCOPE CALL
 NOP
 MOV #STACK,SP ;LOAD THE STACK POINTER
 MOV \$BASE,RO ;RO = UNIBUS ADDRESS
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
 MOV #106,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

MOV #100\$,R2 ;INITIALIZE TABLE ADDRESS

;FORCE COMPOSITE ERROR AND RESET ATA

5\$: JSR PC,CNTCLR ;GO CLEAR CONTROLLER


```

3913 036726 012760 000001 000024      MOV    #DMD,RMMR1(R0) ;LOAD RMMR1
3914 036734 012760 177777 000014      MOV    #-1,RMER1(R0) ;LOAD RMER1
3915 036742 012760 177777 000016      MOV    #-1,RMAS(R0)  ;LOAD RMAS
3916
3917      ;WRITE THE TEST REGISTER AND VERIFY ATA
3918 036750 011203      MOV    (R2),R3      ;GENERATE REGISTER ADDRESS
3919 036752 060003      ADD    R0,R3
3920 036754 005013      CLR    (R3)
3921 036756 016037 000012 001142      MOV    RMD5(R0),$BDDAT ;STORE RMD5 AT $BDDAT
3922 036764 042737 077777 001142      BIC    #^CATA,$BDDAT
3923 036772 016237 000002 001140      MOV    2(R2),$GDDAT
3924 037000 023737 001140 001142      CMP    $GDDAT,$BDDAT
3925 037006 001410      BEQ    10$          ;BRANCH IF ATA IS OK
3926 037010 010337 001174      MOV    R3,$TMP0
3927 037014 010037 001136      MOV    R0,$BDADR
3928 037020 062737 000012 001136      ADD    #RMD5,$BDADR
3929 037026 104232      EMT    232
3930
3931      ;MOVE TABLE POINTER - EXIT IF DONE
3932 037030      10$:
3933 037030 062702 000004      ADD    #4,R2
3934 037034 005712      TST    (R2)
3935 037036 100401      BMI    20$          ;EXIT IF ENTRY MINUS
3936 037040 000730      BR     5$
3937 037042 000410      BR     200$        ;JUMP OVER TABLE
3938
3939      ;TABLE OF REGISTER ADDRESSES AND ATA BITS
3940 037044      100$:
3941 037044 000016      .WORD  RMAS
3942 037046 000000      .WORD
3943
3944 037050 000006      .WORD  RMDA
3945 037052 100000      .WORD  ATA
3946
3947 037054 000000      .WORD  RMCS1
3948 037056 000000      .WORD
3949
3950 037060 177777      .WORD  -1          ;END OF TABLE
3951 037062 000000      .WORD
3952
3953 037064      200$:          ;END OF TEST
3954
3955      ;*****
      ;*TEST 107      P SET ATA TEST
      ;*****
      ;TST107:
037064      SCOPE          ;SCOPE CALL
037064 000004      NOP
037066 000240      MOV    #STACK,SP ;LOAD THE STACK POINTER
037070 012706 001100      MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
037074 013700 001276      MOV    TSTQUE,R1 ;R1 = POINTER TO DEVICE
037100 013701 001462      MOV    #107,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
037104 012737 000107 001226      MOV
3956
3957 037112 012702 037254      MOV    #100$,R2  ;INITIALIZE TABLE POINTER
3958 037116
3959 037116 004737 055370      JSR    PC,SETVV  ;GO SET VOLUME VALID

```



```

037122 000402 BR 20$ ;BRANCH TO 20$ IF NO ERROR
037124 104000 EMT
3960 037126 000451 BR 50$
3961
3962 ;EXECUTE THE COMMAND FROM THE TABLE
3963 037130 20$:
3964 037130 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3965 037136 011203 MOV (R2),R3 ;GET FUNCTION CODE
3966 037140 052703 000001 BIS #GO,R3
3967 037144 010360 000000 MOV R3,RMCS1(R0) ;LOAD RMCS1
3968 037150 012703 000003 MOV #3,R3
3969 037154 30$:
3970 037154 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
3971 037162 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
3972 037170 005303 DEC R3
3973 037172 001370 BNE 30$
3974
3975 ;VERIFY THAT ATA IS SET
3976 037174 016037 000012 001142 MOV RMDS(R0),SBDDAT ;STORE RMDS AT SBDDAT
3977 037202 042737 077777 001142 BIC #^CATA,SBDDAT
3978 037210 001013 BNE 40$
3979 037212 010037 001136 MOV R0,SBADR
3980 037216 062737 000012 001136 ADD #RMDS,SBADR
3981 037224 012737 100000 001140 MOV #ATA,$GDDAT
3982 037232 011237 001174 MOV (R2),$TMPO
3983 037236 104233 EMT 233
3984
3985 ;ADVANCE TABLE POINTER-EXIT IF DONE
3986 037240 40$:
3987 037240 062702 000002 ADD #2,R2
3988 037244 005712 TST (R2)
3989 037246 100401 BMI 50$
3990 037250 000722 BR 10$
3991 037252 000403 BR 200$ ;JUMP OVER TABLE
3992
3993 ;TABLE OF FUNCTION CODES
3994 037254 100$:
3995 037254 000014 .WORD OFFSET
3996 037256 000016 .WORD RTC
3997 037260 177777 .WORD -1 ;END OF TABLE
3998 037262
3999
4000 ;*****
; *TEST 110 SET WLE TEST
;*****
;TST110:
037262 000004 SCOPE ;SCOPE CALL
037262 000240 NOP
037266 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
037272 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
037276 013701 001462 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
037302 012737 000110 001226 MOV #110,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
4001
4002 037310 012702 037504 10$: MOV #100$,R2 ;INITIALIZE TABLE POINTER
4003 037314
4004 037314 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
  
```

```

037320 000402 BR 20$ ;BRANCH TO 20$ IF NO ERROR
037322 104000 EMT
4005 037324 000466 BR 50$
4006
4007 ;ENABLE DEBUG CLOCK AND SET WRITE PROTECT ACCORDING TO TABLE
4008 037326 20$:
4009 037326 016203 000002 MOV 2(R2),R3 ;GET WRITE PROTECT FROM TABLE
4010 037332 052703 041001 BIS #DMD!MUR!DBEN,R3 ;SET OTHER MAINT BITS
4011 037336 010360 000024 MOV R3,RMMR1(R0) ;LOAD RMMR1
4012
4013 ;LOAD AND EXECUTE THE COMMAND FROM THE TABLE
4014 037342 011204 MOV (R2),R4 ;GET FUNCTION CODE FROM TABLE
4015 037344 052704 000001 BIS #GO,R4 ;SET GO
4016 037350 010460 000000 MOV R4,RMCS1(R0) ;LOAD RMCS1
4017 037354 012705 000002 MOV #2,R5 ;R5=CLOCK COUNT
4018 037360 052703 100000 30$: BIS #DBCK,R3 ;SET CLOCK
4019 037364 010360 000024 MOV R3,RMMR1(R0) ;LOAD RMMR1
4020 037370 042703 100000 BIC #DBCK,R3 ;RESET CLOCK
4021 037374 010360 000024 MOV R3,RMMR1(R0) ;LOAD RMMR1
4022 037400 005305 DEC R5
4023 037402 001366 BNE 30$ ;ISSUE 2 CLOCKS
4024
4025 ;VERIFY THAT WRITE LOCK ERROR IS ACCORDING TO TABLE
4026 037404 016037 000014 001142 MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
4027 037412 042737 173777 001142 BIC #^CWLE,SBDDAT
4028 037420 026237 000004 001142 CMP 4(R2),SBDDAT
4029 037426 001417 BEQ 40$ ;BRANCH IF WLE IS OK
4030 037430 016237 000004 001140 MOV 4(R2),$GDDAT ;SAVE DATA FOR ERROR MSG
4031 037436 010037 001136 MOV R0,$BDADR
4032 037442 062737 000014 001136 ADD #RMER1,$BDADR
4033 037450 011237 001174 MOV (R2),$TMP0 ;$TMP0=FUNCTION CODE
4034 037454 016237 000002 001176 MOV 2(R2),$TMP1 ;$TMP1=WRITE PROTECT
4035 037462 104234 EMT 234
4036 037464 000406 BR 50$
4037
4038 ;ADVANCE TABLE POINTER TO NEXT FUNCTION CODE-EXIT IF DONE
4039 037466 40$:
4040 037466 062702 000006 ADD #6,R2
4041 037472 005762 000002 TST 2(R2)
4042 037476 100401 BMI 50$
4043 037500 000705 BR 10$ ;REPEAT TEST
4044
4045 037502 000416 50$: BR 200$ ;JUMP OVER TABLE
4046
4047 ;TABLE OF FUNCTION CODES, WRITE PROTECT AND WRITE LOCK ERRORS
4048 037504 100$:
4049 037504 000060 .WORD WD ;WRITE DATA COMMAND
4050 037506 000010 .WORD MWP ;WRITE PROTECT ON
4051 037510 004000 .WORD WLE ;WRITE LOCK ERROR ONE
4052
4053 037512 000060 .WORD WD ;WRITE DATA COMMAND
4054 037514 000000 .WORD ;MWP OFF
4055 037516 000000 .WORD ;WLE OFF
4056
4057 037520 000070 .WORD RD ;READ DATA COMMAND
4058 037522 000010 .WORD MWP ;MWP ON
4059 037524 000000 .WORD ;WLE OFF

```



```

4060
4061 037526 000020          .WORD  RIP          ;READ IN PRESET COMMAND
4062 037530 000010          .WORD  MWP          ;MWP ON
4063 037532 000000          .WORD          ;WLE OFF
4064
4065 037534 000000          .WORD          ;END OF TABLE
4066 037536 177777          .WORD  -1
4067
4068 037540          200$:          ;END OF TEST
4069
4070          ;:*****
          ;*TEST 111      EXCEPTION TEST
          ;:*****
          TST111:
037540          SCOPE          ;SCOPE CALL
037540 000004          NOP
037542 000240          MOV      #STACK,SP    ;LOAD THE STACK POINTER
037544 012706 001100          MOV      $BASE,R0     ;R0 = UNIBUS ADDRESS
037550 013700 001276          MOV      TSTQUE,R1    ;R1 = POINTER TO DEVICE
037554 013701 001462          MOV      #111,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
037560 012737 000111 001226
4071          ;WITH OCCUPIED SET, EACH OF THE FOLLOWING ERRORS SHOULD CAUSE AN
4072          ;EXCEPTION:
4073          ;
4074          ;
4075          ;
4076          ;
4077          ;
4078          ;
4079          ;
4080          ;
4081          ;
4082          ;
4083          ;
4084 037566 012702 040040          MOV      #100$,R2     ;INITIALIZE TABLE POINTER
4085
4086          ;INITIALIZE AND VERIFY THAT EXCEPTION IS RESET
4087 037572          10$:
4088 037572 004737 055614          JSR      PC,CNTCLR    ;GO CLEAR CONTROLLER
4089 037576 016037 000024 001142          MOV      RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
4090 037604 042737 167777 001142          BIC      #^CREX,$BDDAT
4091 037612 001410          BEQ      20$          ;BRANCH IF EXCEPTION IS 0
4092 037614 010037 001136          MOV      R0,$BDADR
4093 037620 062737 000024 001136          ADD      #RMMR1,$BDADR
4094 037626 005037 001140          CLR      $GDDAT
4095 037632 104235          EMT      235
4096 037634          20$:
4097 037634 004737 055370          JSR      PC,SETVV    ;GO SET VOLUME VALID
          037640 000402          BR      30$          ;BRANCH TO 30$ IF NO ERROR
          037642 104000          EMT
4098 037644 000474          BR      60$
4099
4100          ;EXECUTE A WRITE DATA COMMAND TO SET OCCUPIED, RUN AND GO
4101 037646          30$:
4102 037646 012760 000000 000006          MOV      #0,RMDA(R0) ;LOAD RMDA
4103 037654 012760 000000 000034          MOV      #0,RMDC(R0) ;LOAD RMDC
4104 037662 012760 041001 000024          MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1

```



```

4149 040072 004737 055370      JSR    PC,SETVV      ;GO SET VOLUME VALID
      040076 000403          BR      10$         ;BRANCH TO 10$ IF NO ERROR
      040100 104000          EMT
4150 040102 000137 041400      JMP      330$
4151
4152      ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
4153 040106      10$:
4154 040106 012760 041001 000024      MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4155 040114 012760 000000 000014      MOV     #0,RMER1(R0) ;LOAD RMER1
4156 040122 012760 000000 000042      MOV     #0,RMER2(R0) ;LOAD RMER2
4157 040130 012760 000007 000000      MOV     #RECAL!GO,RMCS1(R0) ;LOAD RMCS1
4158
4159      ;STEP COMMAND SEQUENCER TO RECAL COM (2 CLOCKS)
4160 040136 012702 000002      MOV     #2,R2
4161 040142      20$:
      040142 012760 141001 000024      MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4162 040150 012760 041001 000024      MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4163 040156 005302          DEC     R2
4164 040160 001370          BNE     20$
4165
4166      ;DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)
4167 040162 012760 040001 000024      MOV     #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
4168 040170 012702 000002      MOV     #2,R2
4169 040174      30$:
      040174 012760 140001 000024      MOV     #DMD!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4170 040202 012760 040001 000024      MOV     #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
4171 040210 005302          DEC     R2
4172 040212 001370          BNE     30$
4173
4174      ;VERIFY THAT OPI IS SET
4175 040214 016037 000014 001142      MOV     RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
4176 040222 042737 157777 001142      BIC     #^COPI,$BDDAT
4177 040230 001011          BNE     40$
4178 040232 012737 020000 001140      MOV     #OPI,$GDDAT
4179 040240 010037 001136          MOV     R0,$BDADR
4180 040244 062737 000014 001136      ADD     #RMER1,$BDADR
4181 040252 104241          EMT      241
4182
4183 040254 012737 040262 001124 40$:  MOV     #50$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4184
4185      ;VERIFY THAT RECALIBRATE ABORTS DURING EXECUTION
4186 040262      50$:
4187 040262 004737 055370      JSR     PC,SETVV ;GO SET VOLUME VALID
      040266 000403          BR      60$         ;BRANCH TO 60$ IF NO ERROR
      040270 104000          EMT
4188 040272 000137 041400      JMP      330$
4189
4190      ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
4191 040276      60$:
4192 040276 012760 041001 000024      MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4193 040304 012760 000000 000014      MOV     #0,RMER1(R0) ;LOAD RMER1
4194 040312 012760 000000 000042      MOV     #0,RMER2(R0) ;LOAD RMER2
4195 040320 012760 000007 000000      MOV     #RECAL!GO,RMCS1(R0) ;LOAD RMCS1
4196
4197      ;STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
4198 040326 012702 000003      MOV     #3,R2
4199 040332      70$:
  
```

```

4200 040332 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4201 040340 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4202 040346 005302                      DEC      R2
4203 040350 001370                      BNE     70$
4204
4205 ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
4206 040352 012760 041101 000024      MOV      #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
4207
4208 ;STEP 2 CLOCKS AND VERIFY GO IS RESET
4209 040360 012702 000002                      MOV      #2,R2
4210 040364 80$:
4211 040372 012760 141101 000024      MOV      #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
4212 040372 012760 041101 000024      MOV      #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
4213 040400 005302                      DEC      R2
4214 040402 001370                      BNE     80$
4215 040404 016037 000000 001142      MOV      RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
4216 040412 042737 177776 001142      BIC     #^CGO,$BDDAT
4217 040420 001405                      BEQ     90$
4218 040422 010037 001136      MOV      R0,$BDADR
4219 040426 005037 001140      CLR     $GDDAT
4220 040432 104242                      EMT     242
4221 040434 012737 040442 001124 90$:  MOV      #100$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4222
4223 ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
4224 040442 100$:
4225 040442 004737 055370      JSR     PC,SETVV ;GO SET VOLUME VALID
4226 040446 000403                      BR      110$ ;BRANCH TO 110$ IF NO ERROR
4227 040450 104000                      EMT
4228 040452 000137 041400      JMP     330$
4229 ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
4230 040456 110$:
4231 040456 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4232 040464 012760 000000 000014      MOV      #0,RMER1(R0) ;LOAD RMER1
4233 040472 012760 000000 000042      MOV      #0,RMER2(R0) ;LOAD RMER2
4234 040500 012760 000007 000000      MOV      #RECAL!GO,RMCS1(R0) ;LOAD RMCS1
4235 ;STEP THE COMMAND SEQUENCER
4236 040506 012702 000017                      MOV      #15.,R2
4237 040512 120$:
4238 040512 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4239 040520 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4240 040526 005302                      DEC      R2
4241 040530 001370                      BNE     120$
4242
4243 ;VERIFY THAT OPI IS SET
4244 040532 016037 000014 001142      MOV      RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
4245 040540 042737 157777 001142      BIC     #^COPI,$BDDAT
4246 040546 001011                      BNE     130$
4247 040550 010037 001136      MOV      R0,$BDADR
4248 040554 062737 000014 001136      ADD     #RMER1,$BDADR
4249 040562 012737 020000 001140      MOV      #OPI,$GDDAT
4250 040570 104243                      EMT     243
4251
4252 040572 012737 040600 001124 130$:  MOV      #150$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4253

```



```

4254 ;VERIFY ATA SETS IF DRIVE COMPLETES RECALIBRATE (ON CYLINDER SETS)
4255 040600 150$:
4256 040600 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
      040604 000403 BR 160$ ;BRANCH TO 160$ IF NO ERROR
      040606 104000 EMT
4257 040610 000137 041400 JMP 330$
4258
4259 ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
4260 040614 160$:
4261 040614 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4262 040622 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
4263 040630 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
4264 040636 012760 000007 000000 MOV #RECAL!GO,RMCS1(R0) ;LOAD RMCS1
4265
4266 ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4267 040644 012702 000015 MOV #13.,R2
4268 040650 170$:
      040650 012760 141401 000024 MOV #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0) ;LOAD RMMR1
4269 040656 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4270 040664 005302 DEC R2
4271 040666 001370 BNE 170$
4272
4273 ;DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER
4274 040670 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4275 040676 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4276
4277 ;STEP COMMAND SEQUENCER TO SET ATTENTION (3 CLOCKS)
4278 040704 012702 000003 MOV #3,R2
4279 040710 180$:
4280 040710 012760 141401 000024 MOV #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
4281 040716 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4282 040724 005302 DEC R2
4283 040726 001370 BNE 180$
4284
4285 ;VERIFY ATA IS SET
4286 040730 016037 000012 001142 MOV RMDS(R0),SBDDAT ;STORE RMDS AT SBDDAT
4287 040736 042737 077777 001142 BIC #^CATA,SBDDAT
4288 040744 001011 BNE 190$
4289 040746 012737 100000 001140 MOV #ATA,$GDDAT
4290 040754 010037 001136 MOV R0,$BDADR
4291 040760 062737 000012 001136 ADD #RMDS,$BDADR
4292 040766 104244 EMT 244
4293
4294 040770 012737 040776 001124 190$: MOV #200$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4295
4296 ;VERIFY THAT RECALIBRATE ABORTS AFTER EXECUTION DURING WAIT LOOP
4297 040776 200$:
4298 040776 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
      041002 000402 BR 210$ ;BRANCH TO 210$ IF NO ERROR
      041004 104000 EMT
4299 041006 000574 BR 330$
4300
4301 ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
4302 041010 210$:
4303 041010 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4304 041016 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
4305 041024 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2

```

```

4306 041032 012760 000007 000000      MOV      #RECAL!GO, RMCS1(R0)      ;LOAD RMCS1
4307
4308      ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4309 041040 012702 000015      MOV      #13., R2
4310 041044
4311 041044 012760 141401 000024      MOV      #DMD!MUR!DBEN!MOC!DBCK, RMMR1(R0)      ;LOAD RMMR1
4312 041052 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC, RMMR1(R0)      ;LOAD RMMR1
4313 041060 005302
4314 041062 001370      DEC      R2
4315      BNE      220$
4316      ;DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
4317 041064 012760 041001 000024      MOV      #DMD!MUR!DBEN, RMMR1(R0) ;LOAD RMMR1
4318
4319      ;STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
4320 041072 012702 000007      MOV      #7, R2
4321 041076
4322 041076 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK, RMMR1(R0)      ;LOAD RMMR1
4323 041104 012760 041001 000024      MOV      #DMD!MUR!DBEN, RMMR1(R0) ;LOAD RMMR1
4324 041112 005302
4325 041114 001370      DEC      R2
4326      BNE      230$
4327
4328 041116 016037 000000 001142      ;VERIFY THAT GO IS STILL SET
4329 041124 042737 177776 001142      MOV      RMCS1(R0), $BDDAT      ;STORE RMCS1 AT $BDDAT
4330 041132 001006      BIC      #^CGO, $BDDAT
4331 041134 012737 000001 001140      BNE      240$
4332 041142 010037 001136      MOV      #GO, $GDDAT
4333 041146 104245      MOV      R0, $BDADR
4334      EMT      245
4335
4336 041150      ;SET SEEK INCOMPLETE ERROR
4337 041150 012760 041201 000024      240$:   MOV      #DMD!MUR!DBEN!MSER, RMMR1(R0) ;LOAD RMMR1
4338
4339 041156 012702 000003      ;STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
4340 041162      MOV      #3, R2
4341 041162 012760 141201 000024      250$:   MOV      #DMD!MUR!DBEN!MSER!DBCK, RMMR1(R0) ;LOAD RMMR1
4342 041170 012760 041201 000024      MOV      #DMD!MUR!DBEN!MSER, RMMR1(R0) ;LOAD RMMR1
4343 041176 005302      DEC      R2
4344 041200 001370      BNE      250$
4345 041202 016037 000000 001142      MOV      RMCS1(R0), $BDDAT      ;STORE RMCS1 AT $BDDAT
4346 041210 042737 177776 001142      BIC      #^CGO, $BDDAT
4347 041216 001405      BEQ      260$
4348 041220 010037 001136      MOV      R0, $BDADR
4349 041224 005037 001140      CLR      $GDDAT
4350      EMT      246
4351 041232 012737 041240 001124      260$:   MOV      #300$, $LPERR      ;CHANGE LOOP ON ERROR ADDRESS
4352
4353      ;VERIFY THE TAG BUS DURING RECALIBRATE
4354 041240      300$:
4355 041240 004737 055370      JSR      PC, SETVV      ;GO SET VOLUME VALID
4356 041244 000402      BR      310$      ;BRANCH TO 310$ IF NO ERROR
4357 041246 104000      EMT
4358 041250 000453      BR      330$
      ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
  
```



```

4359 041252          310$:
4360 041252 012760 041401 000024      MOV    #DMD!MUR!DBEN!MOC,RMMR1(R0)      ;LOAD RMMR1
4361 041260 012760 000000 000014      MOV    #0,RMER1(R0)                    ;LOAD RMER1
4362 041266 012760 000000 000042      MOV    #0,RMER2(R0)                    ;LOAD RMER2
4363 041274 012760 000007 000000      MOV    #RECAL!GO,RMCS1(R0)             ;LOAD RMCS1
4364 041302 012702 041402          MOV    #400$,R2                        ;INITIALIZE TABLE POINTER
4365
4366
4367 041306          :VERIFY TAG BUS ACCORDING TO TABLE
315$:
4368 041314 042737 150000 001142      MOV    RMMR2(R0),SBDDAT                ;STORE RMMR2 AT SBDDAT
4369 041322 021237 001142          BIC    #RQA!RQB!TST,SBDDAT
4370 041326 001411          CMP    (R2),SBDDAT
4371 041330 011237 001140          BEQ    320$
4372 041334 010037 001136          MOV    (R2),$GDDAT
4373 041340 062737 000040 001136      MOV    R0,SBDADR
4374 041346 104247          ADD    #RMMR2,SBDADR
4375 041350 000413          EMT    247
4376          BR    330$
4377
4378 041352          :ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
320$:
4379 041352- 062702 000002          ADD    #2,R2
4380 041356 005712          TST    (R2)
4381 041360 100407          BMI    330$                          ;EXIT IF ENTRY NEGATIVE
4382
4383          :STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
4384 041362 012760 151001 000024      MOV    #DMD!DBEN!MUR!MOV!DBCK,RMMR1(R0) ;LOAD RMMR1
4385 041370 012760 041401 000024      MOV    #DMD!DBEN!MUR!MOC,RMMR1(R0)    ;LOAD RMMR1
4386 041376 000743          BR    315$
4387 041400 000416          330$: BR    500$                          ;JUMP OVER TABLE
4388
4389          :TABLE OF TAG BUS CONTROL AND BIT VALUES
400$:
4390 041402          .WORD 1777                          ;BUS BITS AT HIGH IMPEDANCE STATE
4391 041402 001777          .WORD 1777
4392 041404 001777          .WORD CC!CH!BB06                    ;CONTROL BITS ENABLED, BIT 6 ON
4393 041406 006100          .WORD CC!CH!BB06
4394 041410 006100          .WORD TAG!CC!CH!BB06                ;TAG COMES ON
4395 041412 026100          .WORD TAG!CC!CH!BB06
4396 041414 026100          .WORD TAG!CC!CH!BB06
4397 041416 026100          .WORD TAG!CC!CH!BB06
4398 041420 026100          .WORD TAG!CC!CH!BB06
4399 041422 026100          .WORD TAG!CC!CH!BB06
4400 041424 026100          .WORD TAG!CC!CH!BB06
4401 041426 006100          .WORD CC!CH!BB06                    ;TAG GOES OFF
4402 041430 006100          .WORD CC!CH!BB06
4403 041432 001777          .WORD 1777                          ;CONTROL BITS DISABLED
4404
4405 041434 177777          .WORD -1                            ;END OF TABLE
4406
4407 041436          500$:
4408          :END OF TEST
4409
:*****
:*TEST 113      SEEK TEST
:*****
041436          :*****
041436 000004      TST113: SCOPE                          ;SCOPE CALL
  
```

```

041440 000240      NOP
041442 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
041446 013700 001276  MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
041452 013701 001462  MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
041456 012737 000113 001226  MOV      #113,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

4410
4411      ;VERIFY THAT OPI SETS IF UNIT READY DROPS DURING COMMAND EXECUTION
4412
4413 041464 004737 055370  JSR      PC,SETVV      ;GO SET VOLUME VALID
041470 000403      BR      10$           ;BRANCH TO 10$ IF NO ERROR
041472 104000      EMT
4414 041474 000137 043114  JMP      330$

4415
4416      ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4417      ;ADDRESS, AND LOAD SEEK COMMAND
4418 041500 10$:
4419 041500 012760 041001 000024  MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4420 041506 012760 000000 000006  MOV      #0,RMDA(R0)      ;LOAD RMDA
4421 041514 012760 000000 000034  MOV      #0,RMDC(R0)      ;LOAD RMDC
4422 041522 012760 000000 000014  MOV      #0,RMER1(R0)     ;LOAD RMER1
4423 041530 012760 000000 000042  MOV      #0,RMER2(R0)     ;LOAD RMER2
4424 041536 012760 000005 000000  MOV      #SEEK!GO,RMCS1(R0) ;LOAD RMCS1
4425
4426      ;STEP COMMAND SEQUENCER TO SEEK COM (2 CLOCKS)
4427 041544 012702 000002  MOV      #2,R2
4428 041550 20$:
041550 012760 141001 000024  MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4429 041556 012760 041001 000024  MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4430 041564 005302      DEC      R2
4431 041566 001370      BNE     20$

4432
4433      ;DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)
4434 041570 012760 040001 000024  MOV      #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
4435 041576 012702 000002  MOV      #2,R2
4436 041602 30$:
041602 012760 140001 000024  MOV      #DMD!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4437 041610 012760 040001 000024  MOV      #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
4438 041616 005302      DEC      R2
4439 041620 001370      BNE     30$

4440
4441      ;VERIFY THAT OPI IS SET
4442 041622 016037 000014 001142  MOV      RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
4443 041630 042737 157777 001142  BIC     #^COPI,$BDDAT
4444 041636 001011      BNE     40$
4445 041640 012737 020000 001140  MOV      #OPI,$GDDAT
4446 041646 010037 001136  MOV      R0,$BDADR
4447 041652 062737 000014 001136  ADD     #RMER1,$BDADR
4448 041660 104250      EMT     250
4449
4450 041662 012737 041670 001124 40$:  MOV      #50$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4451
4452      ;VERIFY THAT SEEK ABORTS DURING EXECUTION
4453 041670 50$:
4454 041670 004737 055370  JSR      PC,SETVV      ;GO SET VOLUME VALID
041674 000403      BR      60$           ;BRANCH TO 60$ IF NO ERROR
041676 104000      EMT
4455 041700 000137 043114  JMP      330$
  
```



```

4456
4457
4458 ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4459 041704 ;ADDRESS, AND LOAD SEEK COMMAND
4460 041704 012760 041001 000024 60$: MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4461 041712 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
4462 041720 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
4463 041726 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
4464 041734 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
4465 041742 012760 000005 000000 MOV #SEEK!GO,RMCS1(R0) ;LOAD RMCS1
4466
4467 ;STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
4468 041750 012702 000003 MOV #3,R2
4469 041754 70$:
4470 041754 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4471 041762 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4472 041770 005302 DEC R2
4473 041772 001370 BNE 70$
4474
4475 ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
4476 041774 012760 041101 000024 MOV #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
4477
4478 ;STEP 2 CLOCKS AND VERIFY GO IS RESET
4479 042002 012702 000002 MOV #2,R2
4480 042006 80$:
4481 042014 012760 141101 000024 MOV #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
4482 042022 005302 MOV #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
4483 042024 001370 DEC R2
4484 042026 016037 000000 001142 BNE 80$
4485 042034 042737 177776 001142 MOV RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
4486 042042 001405 BIC #^CGO,$BDDAT
4487 042044 010037 001136 BEQ 90$
4488 042050 005037 001140 MOV R0,$BDADR
4489 042054 104251 CLR $GDDAT
4490 EMT 251
4491 042056 012737 042064 001124 90$: MOV #100$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4492
4493 ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
4494 042064 100$:
4495 042064 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
4496 042074 000137 043114 BR 110$ ;BRANCH TO 110$ IF NO ERROR
4497 EMT
4498 JMP 330$
4499
4500 ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4501 042100 ;ADDRESS, AND LOAD SEEK COMMAND
4502 042100 012760 041001 000024 110$: MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4503 042106 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
4504 042114 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
4505 042122 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
4506 042130 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
4507 042136 012760 000005 000000 MOV #SEEK!GO,RMCS1(R0) ;LOAD RMCS1
4508
4509 042144 012702 000017 ;STEP THE COMMAND SEQUENCER
MOV #15.,R2

```



```

4510 042150
4511 042150 012760 141001 000024 120$: MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4512 042156 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4513 042164 005302 DEC R2
4514 042166 001370 BNE 120$
4515
4516 ;VERIFY THAT OPI IS SET
4517 042170 016037 000014 001142 MOV RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
4518 042176 042737 157777 001142 BIC #^COPI,$BDDAT
4519 042204 001011 BNE 130$
4520 042206 010037 001136 MOV R0,$BDADR
4521 042212 062737 000014 001136 ADD #RMER1,$BDADR
4522 042220 012737 020000 001140 MOV #OPI,$GDDAT
4523 042226 104252 EMT 252
4524
4525 042230 012737 042236 001124 130$: MOV #150$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4526
4527 ;VERIFY ATA SETS IF DRIVE COMPLETES SEEK (ON CYLINDER SETS)
4528 042236 150$:
4529 042236 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
042242 000403 BR 160$ ;BRANCH TO 160$ IF NO ERROR
042244 104000 EMT
4530 042246 000137 043114 JMP 330$
4531
4532 ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4533 ;ADDRESS, AND LOAD SEEK COMMAND
4534 042252 160$:
4535 042252 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4536 042260 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
4537 042266 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
4538 042274 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
4539 042302 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
4540 042310 012760 000005 000000 MOV #SEEK!GO,RMCS1(R0) ;LOAD RMCS1
4541
4542 ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4543 042316 012702 000015 MOV #13.,R2
4544 042322 170$:
4545 042330 012760 141401 000024 MOV #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0) ;LOAD RMMR1
4546 042336 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4547 042340 005302 DEC R2
4548 042340 001370 BNE 170$
4549
4550 042342 012760 041001 000024 ;DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER
4551 042350 012760 041401 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4552 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4553
4554 042356 012702 000003 ;STEP COMMAND SEQUENCER TO SET ATTENTION (3 CLOCKS)
4555 042362 MOV #3,R2
4556 042362 012760 141401 000024 180$:
4557 042370 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
4558 042376 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4559 042400 005302 DEC R2
4560 042400 001370 BNE 180$
4561
4562 042402 016037 000012 001142 ;VERIFY ATA IS SET
4563 042410 042737 077777 001142 MOV RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
BIC #^CATA,$BDDAT
    
```



```

4564 042416 001011          BNE      190$
4565 042420 012737 100000 001140      MOV      #ATA,$GDDAT
4566 042426 010037 001136          MOV      R0,$BDADR
4567 042432 062737 000012 001136      ADD      #RMDS,$BDADR
4568 042440 104253          EMT      253
4569
4570 042442 012737 042450 001124 190$:  MOV      #200$,$LPERR      ;CHANGE LOOP ON ERROR ADDRESS
4571
4572          ;VERIFY THAT SEEK ABORTS AFTER EXECUTION DURING WAIT LOOP
4573 042450 200$:
4574 042450 004737 055370          JSR      PC,SETVV          ;GO SET VOLUME VALID
         042454 000403          BR       210$             ;BRANCH TO 210$ IF NO ERROR
         042456 104000          EMT
4575 042460 000137 043114          JMP      330$
4576
4577          ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4578          ;ADDRESS AND LOAD SEEK COMMAND
4579 042464 210$:
4580 042464 012760 041401 000024          MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4581 042472 012760 000000 000006          MOV      #0,RMDA(R0) ;LOAD RMDA
4582 042500 012760 000000 000034          MOV      #0,RMDC(R0) ;LOAD RMDC
4583 042506 012760 000000 000014          MOV      #0,RMER1(R0) ;LOAD RMER1
4584 042514 012760 000000 000042          MOV      #0,RMER2(R0) ;LOAD RMER2
4585 042522 012760 000005 000000          MOV      #SEEK:GO,RMCS1(R0) ;LOAD RMCS1
4586
4587          ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4588 042530 012702 000015          MOV      #13.,R2
4589 042534 220$:
4590 042534 012760 141401 000024          MOV      #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
4591 042542 012760 041401 000024          MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4592 042550 005302          DEC      R2
4593 042552 001370          BNE      220$
4594
4595          ;DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
4596 042554 012760 041001 000024          MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4597
4598          ;STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
4599 042562 012702 000007          MOV      #7.,R2
4600 042566 230$:
4601 042566 012760 141001 000024          MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4602 042574 012760 041001 000024          MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4603 042602 005302          DEC      R2
4604 042604 001370          BNE      230$
4605
4606          ;VERIFY THAT GO IS STILL SET
4607 042606 016037 000000 001142          MOV      RMCS1(R0),$BDDAT ;STORE RMCS1 AT $BDDAT
4608 042614 042737 177776 001142          BIC      #^CGO,$BDDAT
4609 042622 001006          BNE      240$
4610 042624 012737 000001 001140          MOV      #GO,$GDDAT
4611 042632 010037 001136          MOV      R0,$BDADR
4612 042636 104254          EMT      254
4613
4614          ;SET SEEK INCOMPLETE ERROR
4615 042640 240$:
         042640 012760 041201 000024          MOV      #DMD!MUR!DBEN!MSER,RMMR1(R0) ;LOAD RMMR1
4616
4617          ;STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)

```

```

4618 042646 012702 000003          MOV      #3,R2
4619 042652          250$:      MOV      #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0) ;LOAD RMMR1
      042652 012760 141201 000024      MOV      #DMD!MUR!DBEN!MSER,RMMR1(R0) ;LOAD RMMR1
4620 042660 012760 041201 000024      DEC      R2
4621 042666 005302          BNE      250$
4622 042670 001370          MOV      RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
4623 042672 016037 000000 001142      BIC      #^CGO,$BDDAT
4624 042700 042737 177776 001142      BEQ      260$
4625 042706 001405          MOV      R0,$BDADR
4626 042710 010037 001136          CLR      $GDDAT
4627 042714 005037 001140          EMT      255
4628 042720 104255
4629
4630 042722 012737 042734 001124 260$:      MOV      #300$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4631 042730 012703 000001          MOV      #1,R3 ;INITIALIZE CYLINDER ADDRESS
4632
4633          ;VERIFY THE TAG BUS DURING SEEK
4634 042734          300$:
4635 042734 004737 055370          JSR      PC,SETVV ;GO SET VOLUME VALID
      042740 000402          BR       310$ ;BRANCH TO 310$ IF NO ERROR
      042742 104000          EMT
4636 042744 000463          BR       330$
4637
4638          ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4639          ;ADDRESS AND LOAD SEEK COMMAND
4640 042746          310$:
4641 042746 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4642 042754 012760 000000 000006      MOV      #0,RMDA(R0) ;LOAD RMDA
4643 042762 010360 000034          MOV      R3,RMDC(R0) ;LOAD RMDC
4644 042766 012760 000000 000014      MOV      #0,RMER1(R0) ;LOAD RMER1
4645 042774 012760 000000 000042      MOV      #0,RMER2(R0) ;LOAD RMER2
4646 043002 012760 000005 000000      MOV      #SEEK!GO,RMCS1(R0) ;LOAD RMCS1
4647 043010 012702 043130          MOV      #400$,R2 ;INITIALIZE TABLE POINTER
4648
4649          ;VERIFY TAG BUS ACCORDING TO TABLE AND CYLINDER IN R3
4650 043014          315$:
4651 043014 016037 000040 001142      MOV      RMMR2(R0), $BDDAT ;STORE RMMR2 AT $BDDAT
4652 043022 042737 150000 001142      BIC      #RQA!RQB!TST,$BDDAT
4653 043030 011237 001140          MOV      (R2),$GDDAT
4654 043034 050337 001140          BIS      R3,$GDDAT ;OR CYLINDER ADDRESS IN
4655 043040 023737 001140 001142      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
4656 043046 001407          BEQ      320$ ;BRANCH IF TAG BUS OK
4657 043050 010037 001136          MOV      R0,$BDADR
4658 043054 062737 000040 001136      ADD      #RMMR2,$BDADR
4659 043062 104256          EMT      256
4660 043064 000413          BR       330$
4661
4662          ;ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
4663 043066          320$:
4664 043066 062702 000002          ADD      #2,R2
4665 043072 005712          TST      (R2)
4666 043074 100407          BMI      330$ ;EXIT IF ENTRY NEGATIVE
4667
4668          ;STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
4668 043076 012760 151001 000024      MOV      #DMD!DBEN!MUR!MOV!DBCK,RMMR1(R0) ;LOAD RMMR1
4669 043104 012760 041401 000024      MOV      #DMD!DBEN!MUR!MOC,RMMR1(R0) ;LOAD RMMR1
4670 043112 000740          BR       315$
  
```


4671
4672
4673 043114
4674 043114 006303
4675 043116 020327 002000
4676 043122 103001
4677 043124 000703
4678 043126 000416
4679
4680
4681 043130
4682 043130 001777
4683 043132 001777
4684 043134 004000
4685 043136 004000
4686 043140 024000
4687 043142 024000
4688 043144 024000
4689 043146 024000
4690 043150 024000
4691 043152 024000
4692 043154 004000
4693 043156 004000
4694 043160 001777
4695
4696 043162 177777
4697
4698 043164
4699
4700

;REPEAT TAG BUS TEST FOR EACH PRIME CYLINDER, I.E., 1,2,4,....
330\$:

```

ASL R3 ;SHIFT TO NEXT CYLINDER
CMP R3,#1024.
BHIS 340$ ;EXIT IF WAS DONE
BR 300$ ;TEST NEXT CYLINDER
340$: BR 500$ ;JUMP OVER TABLE

```

;TABLE OF TAG BUS CONTROL AND BIT VALUES
400\$:

```

.WORD 1777 ;BUS BITS AT HIGH IMPEDANCE STATE
.WORD 1777
.WORD CC ;CONTROL BITS ENABLED, BIT 6 ON
.WORD CC
.WORD TAG!CC ;TAG COMES ON
.WORD TAG!CC
.WORD TAG!CC
.WORD TAG!CC
.WORD TAG!CC
.WORD CC ;TAG GOES OFF
.WORD CC
.WORD 1777 ;CONTROL BITS DISABLED
.WORD -1 ;END OF TABLE

```

500\$: ;END OF TEST

;*TEST 114 SEARCH TEST

TST114:

```

SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #114,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

```

;VERIFY THAT OPI SETS IF UNIT READY DROPS DURING COMMAND EXECUTION

```

JSR PC,SETVV ;GO SET VOLUME VALID
BR 10$ ;BRANCH TO 10$ IF NO ERROR
EMT
JMP 330$

```

;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
;ADDRESS, AND LOAD SEARCH COMMAND
10\$:

```

MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
MOV #0,RMDA(R0) ;LOAD RMDA
MOV #0,RMDC(R0) ;LOAD RMDC
MOV #0,RMER1(R0) ;LOAD RMER1
MOV #0,RMER2(R0) ;LOAD RMER2
MOV #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1

```

043164
043164 000004
043166 000240
043170 012706 001100
043174 013700 001276
043200 013701 001462
043204 012737 000114 001226
4701
4702
4703
4704 043212 004737 055370
043216 000403
043220 104000
4705 043222 000137 045260
4706
4707
4708
4709 043226
4710 043226 012760 041001 000024
4711 043234 012760 000000 000006
4712 043242 012760 000000 000034
4713 043250 012760 000000 000014
4714 043256 012760 000000 000042
4715 043264 012760 000031 000000

```

4716
4717
4718 043272 012702 000002 ;STEP COMMAND SEQUENCER TO SEARCH COM (2 CLOCKS)
4719 043276 20$: MOV #2,R2
043276 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4720 043304 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4721 043312 005302 DEC R2
4722 043314 001370 BNE 20$
4723
4724 ;DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)
4725 043316 012760 040001 000024 MOV #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
4726 043324 012702 000002 MOV #2,R2
4727 043330 30$:
043330 012760 140001 000024 MOV #DMD!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4728 043336 012760 040001 000024 MOV #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1
4729 043344 005302 DEC R2
4730 043346 001370 BNE 30$
4731
4732 ;VERIFY THAT OPI IS SET
4733 043350 016037 000014 001142 MOV RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
4734 043356 042737 157777 001142 BIC #^COPI,$BDDAT
4735 043364 001011 BNE 40$
4736 043366 012737 020000 001140 MOV #OPI,$GDDAT
4737 043374 010037 001136 MOV R0,$BDADR
4738 043400 062737 000014 C01136 ADD #RMER1,$BDADR
4739 043406 104257 EMT 257
4740
4741 043410 012737 043416 001124 40$: MOV #50$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4742
4743 ;VERIFY THAT SEARCH ABORTS DURING EXECUTION
4744 043416 50$:
4745 043416 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
043422 000403 BR 60$ ;BRANCH TO 60$ IF NO ERROR
043424 104000 EMT
4746 043426 000137 045260 JMP 330$
4747
4748 ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4749 ;ADDRESS, AND LOAD SEARCH COMMAND
4750 043432 60$:
4751 043432 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4752 043440 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
4753 043446 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
4754 043454 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
4755 043462 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
4756 043470 012760 000031 000000 MOV #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1
4757
4758 ;STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
4759 043476 012702 000003 MOV #3,R2
4760 043502 70$:
4761 043502 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4762 043510 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4763 043516 005302 DEC R2
4764 043520 001370 BNE 70$
4765
4766 ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
4767 043522 012760 041101 000024 MOV #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
4768

```



```

4769 ;STEP 2 CLOCKS AND VERIFY GO IS RESET
4770 043530 012702 000002 MOV #2,R2
4771 043534 80$: MOV #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
043534 012760 141101 000024 MOV #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
4772 043542 012760 041101 000024 DEC R2
4773 043550 005302 BNE 80$
4774 043552 001370 MOV RMCS1(R0),SBDDAT ;STORE RMCS1 AT SBDDAT
4775 043554 016037 000000 001142 BIC #^CGO,SBDDAT
4776 043562 042737 177776 001142 BEQ 90$
4777 043570 001405 MOV RO,$BDADR
4778 043572 010037 001136 CLR $GDDAT
4779 043576 005037 001140 EMT 260
4780 043602 104260
4781
4782 ;(THE OTHER TWO ABORT TESTS IN THE COMMAND SEQUENCER ARE TESTED
4783 ;DURING DATA COMMAND TESTS)
4784
4785 043604 012737 043612 001124 90$: MOV #100$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4786
4787 ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
4788 043612 100$: JSR PC,SETVV ;GO SET VOLUME VALID
043612 004737 055370 BR 110$ ;BRANCH TO 110$ IF NO ERROR
043616 000403 EMT
043620 104000 JMP 330$
4790 043622 000137 045260
4791
4792 ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4793 ;ADDRESS, AND LOAD SEARCH COMMAND
4794 043626 110$: MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4795 043626 012760 041001 000024 MOV #0,RMDA(R0) ;LOAD RMDA
4796 043634 012760 000000 000006 MOV #0,RMDC(R0) ;LOAD RMDC
4797 043642 012760 000000 000034 MOV #0,RMER1(R0) ;LOAD RMER1
4798 043650 012760 000000 000014 MOV #0,RMER2(R0) ;LOAD RMER2
4799 043656 012760 000000 000042 MOV #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1
4800 043664 012760 000031 000000
4801
4802 ;STEP THE COMMAND SEQUENCER
4803 043672 012702 000023 MOV #19.,R2
4804 043676 120$: MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4805 043676 012760 141001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4806 043704 012760 041001 000024 DEC R2
4807 043712 005302 BNE 120$
4808 043714 001370
4809
4810 ;VERIFY THAT OPI IS SET
4811 043716 016037 000014 001142 MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
4812 043724 042737 157777 001142 BIC #^COPI,SBDDAT
4813 043732 001011 BNE 130$
4814 043734 010037 001136 MOV RO,$BDADR
4815 043740 062737 000014 001136 ADD #RMER1,$BDADR
4816 043746 012737 020000 001140 MOV #OPI,$GDDAT
4817 043754 104261 EMT 261
4818
4819 043756 012737 043764 001124 130$: MOV #150$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4820
4821 ;VERIFY ATA SETS IF DRIVE COMPLETES SEARCH (ON CYLINDER AND
4822 ;SECTOR COMPARE SETS)

```

```

4823 043764
4824 043764 004737 055370      150$:      JSR      PC,SETVV      ;GO SET VOLUME VALID
          043770 000403          BR      160$          ;BRANCH TO 160$ IF NO ERROR
          043772 104000          EMT
4825 043774 000137 045260      JMP      330$
4826
4827
4828      ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4829 044000      ;ADDRESS, AND LOAD SEARCH COMMAND
          160$:
4830 044000 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4831 044006 012760 000000 000006      MOV      #0,RMDA(R0) ;LOAD RMDA
4832 044014 012760 000000 000034      MOV      #0,RMDC(R0) ;LOAD RMDC
4833 044022 012760 000000 000014      MOV      #0,RMER1(R0) ;LOAD RMER1
4834 044030 012760 000000 000042      MOV      #0,RMER2(R0) ;LOAD RMER2
4835 044036 012760 000031 000000      MOV      #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1
4836
4837      ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (17 CLOCKS)
4838 044044 012702 000021      MOV      #17,R2
4839 044050      170$:
          044050 012760 141401 000024      MOV      #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0) ;LOAD RMMR1
4840 044056 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4841 044064 005302
4842 044066 001370      DEC      R2
          BNE      170$
4843
4844      ;DROP ON CYLINDER TO RESET LATCH,AND RAISE INDEX PULSE
4845      ;TO SET FORMAT CHANGE FLOP
4846 044070 012760 041005 000024      MOV      #DMD!MUR!DBEN!MI,RMMR1(R0) ;LOAD RMMR1
4847
4848      ;RAISE ON CYLINDER AND INHIBIT SEARCH TIMEOUT
4849 044076 012760 051401 000024      MOV      #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
4850
4851      ;STEP COMMAND SEQUENCER TO SEARCH ENABLE (2 CLOCKS)
4852 044104 012702 000002      MOV      #2,R2
4853 044110      180$:
4854 044110 012760 151401 000024      MOV      #DMD!MUR!DBEN!MOC!DTO!DBCK,RMMR1(R0) ;LOAD RMMR1
4855 044116 012760 051401 000024      MOV      #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
4856 044124 005302
4857 044126 001370      DEC      R2
          BNE      180$
4858
4859      ;FORCE SECTOR COMPARE BY CLOCKING SECTOR PULSE WITH SECTOR COMPARE
4860      ;ACTIVE
4861 044130 012760 051403 000024      MOV      #DMD!MUR!MOC!DBEN!DTO!MSC,RMMR1(R0) ;LOAD RMMR1
4862 044136 012760 051443 000024      MOV      #DMD!MUR!MOC!DBEN!DTO!MSC!MS,RMMR1(R0) ;LOAD RMMR1
4863 044144 012760 051403 000024      MOV      #DMD!MUR!MOC!DBEN!DTO!MSC,RMMR1(R0) ;LOAD RMMR1
4864
4865      ;CLOCK SEQUENCER TO SET ATA (3 CLOCKS)
4866 044152 012702 000003      MOV      #3,R2 ;R2 = CLOCK COUNT
4867 044156      185$:
          044156 012760 151401 000024      MOV      #DMD!MUR!MOC!DBEN!DTO!DBCK,RMMR1(R0) ;LOAD RMMR1
4868 044164 012760 051401 000024      MOV      #DMD!MUR!MOC!DBEN!DTO,RMMR1(R0) ;LOAD RMMR1
4869 044172 005302
4870 044174 001370      DEC      R2
          BNE      185$
4871
4872      ;VERIFY ATA IS SET
4873 044176 016037 000012 001142      MOV      RMD5(R0), $BDDAT ;STORE RMD5 AT $BDDAT
4874 044204 042737 077777 001142      BIC      #^CATA,$BDDAT
4875 044212 001011      BNE      190$
  
```



```

4876 044214 012737 100000 001140      MOV      #ATA,$GDDAT
4877 044222 010037 001136      MOV      R0,$BDADR
4878 044226 062737 000012 001136      ADD      #RMDS,$BDADR
4879 044234 104262      EMT      262
4880
4881 044236 012737 044244 001124 190$:  MOV      #200$,$LPERR      ;CHANGE LCOJ ON ERROR ADDRESS
4882
4883      ;VERIFY THAT SEARCH ABORTS AFTER EXECUTION DURING SEARCH SEEK LOOP
4884 044244 200$:
4885 044244 004737 055370      JSR      PC,SETVV      ;GO SET VOLUME VALID
      044250 000403      BR       210$          ;BRANCH TO 210$ IF NO ERROR
      044252 104000      EMT
4886 044254 000137 045260      JMP      330$
4887
4888      ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4889      ;ADDRESS AND LOAD SEARCH COMMAND
4890 044260 210$:
4891 044260 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4892 044266 012760 000000 000006      MOV      #0,RMDA(R0) ;LOAD RMDA
4893 044274 012760 000000 000034      MOV      #0,RMDC(R0) ;LOAD RMDC
4894 044302 012760 000000 000014      MOV      #0,RMER1(R0) ;LOAD RMER1
4895 044310 012760 000000 000042      MOV      #0,RMER2(R0) ;LOAD RMER2
4896 044316 012760 000031 000000      MOV      #SEARCH!GO,RCMS1(R0) ;LOAD RCMS1
4897
4898      ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (17 CLOCKS)
4899 044324 012702 000021      MOV      #17,R2
4900 044330 220$:
4901 044330 012760 141401 000024      MOV      #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
4902 044336 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4903 044344 005302      DEC      R2
4904 044346 001370      BNE     220$
4905
4906      ;DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
4907 044350 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4908
4909      ;STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
4910 044356 012702 000007      MOV      #7,R2
4911 044362 230$:
4912 044362 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
4913 044370 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
4914 044376 005302      DEC      R2
4915 044400 001370      BNE     230$
4916
4917      ;VERIFY THAT GO IS STILL SET
4918 044402 016037 000000 001142      MOV      RCMS1(R0),$BDDAT ;STORE RCMS1 AT $BDDAT
4919 044410 042737 177776 001142      BIC     #^CGO,$BDDAT
4920 044416 001006      BNE     240$
4921 044420 012737 000001 001140      MOV      #GO,$GDDAT
4922 044426 010037 001136      MOV      R0,$BDADR
4923 044432 104263      EMT      263
4924
4925      ;SET SEEK INCOMPLETE ERROR
4926 044434 240$:
      044434 012760 041201 000024      MOV      #DMD!MUR!DBEN!MSER,RMMR1(R0) ;LOAD RMMR1
4927
4928      ;STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
4929 044442 012702 000003      MOV      #3,R2

```

```

4930 044446          250$:
      044446 012760 141201 000024  MOV #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0) ;LOAD RMMR1
4931 044454 012760 041201 000024  MOV #DMD!MUR!DBEN!MSER,RMMR1(R0) ;LOAD RMMR1
4932 044462 005302          DEC R2
4933 044464 001370          BNE 250$
4934 044466 016037 000000 001142  MOV RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
4935 044474 042737 177776 001142  BIC #^CGO,$BDDAT
4936 044502 001405          BEQ 260$
4937 044504 010037 001136          MOV R0,$BDADR
4938 044510 005037 001140          CLR $GDDAT
4939 044514 104264          EMT 264
4940
4941 044516 012737 044524 001124 260$: MOV #265$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4942
4943          ;VERIFY THAT SEARCH ABORTS DURING SECTOR COMPARE LOOP
4944 044524          265$:
4945 044524 004737 055370          JSR PC,SETVV ;GO SET VOLUME VALID
      044530 000403          BR 270$ ;BRANCH TO 270$ IF NO ERROR
      044532 104000          EMT
4946 044534 000137 045260          JMP 330$
4947
4948          ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4949          ;ADDRESS, AND LOAD SEARCH COMMAND
4950 044540          270$:
4951 044540 012760 041401 000024  MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4952 044546 012760 000000 000006  MOV #0,RMDA(R0) ;LOAD RMDA
4953 044554 012760 000000 000034  MOV #0,RMDC(R0) ;LOAD RMDC
4954 044562 012760 000000 000014  MOV #0,RMER1(R0) ;LOAD RMER1
4955 044570 012760 000000 000042  MOV #0,RMER2(R0) ;LOAD RMER2
4956 044576 012760 000031 000000  MOV #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1
4957
4958          ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (17 CLOCKS)
4959 044604 012702 000021          MOV #17.,R2
4960 044610          275$:
4961 044616 012760 041401 000024  MOV #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0) ;LOAD RMMR1
4962 044624 005302          MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
4963 044626 001370          DEC R2
4964          BNE 275$
4965          ;DROP ON CYLINDER TO RESET LATCH, AND RAISE INDEX PULSE
4966          ;TO SET FORMAT CHANGE FLOP
4967 044630 012760 041005 000024  MOV #DMD!MUR!DBEN!MI,RMMR1(R0) ;LOAD RMMR1
4968
4969          ;RAISE ON CYLINDER AND INHIBIT SEARCH TIMEOUT
4970 044636 012760 051401 000024  MOV #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
4971
4972          ;STEP COMMAND SEQUENCER TO SEARCH ENABLE (2 CLOCKS)
4973 044644 012702 000002          MOV #2,R2
4974 044650          280$:
4975 044650 012760 151401 000024  MOV #DMD!MUR!DBEN!MOC!DTO!DBCK,RMMR1(R0) ;LOAD RMMR1
4976 044656 012760 051401 000024  MOV #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
4977 044664 005302          DEC R2
4978 044666 001370          BNE 280$
4979
4980          ;VERIFY THAT SEARCH ENABLE IS ON DURING SECTOR COMPARE LOOP
4981 044670 012702 000004          MOV #4,R2 ;R2 = CLOCK COUNT
4982 044674          281$:
  
```



```

5036                                     ;9. COMMAND SEQUENCES TO ALLOW THE PROGRAM
5037                                     ;TO RUN WITH OR WITHOUT THE ECO.
5038 045134                               312$:
5039 045134 012760 141401 000024        MOV    #DMD!DBEN!MUR!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
5040 045142 012760 041401 000024        MOV    #DMD!DBEN!MUR!MOC,RMMR1(R0) ;LOAD RMMR1
5041 045150 005302                        DEC    R2 ;DONE 9. CLOCKS ?
5042 045152 001370                        BNE   312$ ;NO !!
5043 045154 012702 045316                MOV    #450$,R2 ;INITIALIZE NEW TABLE POINTER
5044                                     ;:*****
5045                                     ;:*****
5046                                     ;:VERIFY TAG BUS ACCORDING TO TABLE AND CYLINDER IN R3
5047 045160                               315$:
5048 045160 016037 000040 001142        MOV    RMMR2(R0), $BDDAT ;STORE RMMR2 AT $BDDAT
5049 045166 042737 150000 001142        BIC   #RQA!RQB!TST,$BDDAT
5050 045174 011237 001140                MOV    (R2), $GDDAT
5051 045200 050337 001140                BIS   R3,$GDDAT ;OR CYLINDER ADDRESS IN
5052 045204 023737 001140 001142        CMP   $GDDAT,$BDDAT ;COMPARE EXPECTED AND RECEIVED
5053 045212 001407                        BEQ   320$ ;BRANCH IF TAG BUS OK
5054 045214 010037 001136                MOV    R0,$BDDADR
5055 045220 062737 000040 001136        ADD   #RMMR2,$BDDADR
5056 045226 104266                        EMT   266
5057 045230 000420                        BR    340$
5058                                     ;:ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
5059 045232                               320$:
5060 045232 062702 000002                ADD   #2,R2
5061 045236 005712                        TST   (R2)
5062 045240 100407                        BMI   330$ ;EXIT IF ENTRY NEGATIVE
5063
5064                                     ;:STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
5065 045242 012760 141401 000024        MOV    #DMD!DBEN!MUR!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
5066 045250 012760 041401 000024        MOV    #DMD!DBEN!MUR!MOC,RMMR1(R0) ;LOAD RMMR1
5067 045256 000740                        BR    315$
5068
5069                                     ;:REPEAT TAG BUS TEST FOR EACH PRIME CYLINDER, I.E., 1,2,4,....
5070 045260                               330$:
5071 045260 006303                        ASL   R3 ;SHIFT TO NEXT CYLINDER
5072 045262 020327 002000                CMP   R3,#1024.
5073 045266 103001                        BHIS  340$ ;EXIT IF WAS DONE
5074 045270 000671                        BR    300$ ;TEST NEXT CYLINDER
5075 045272 000424                        BR    500$ ;JUMP OVER TABLE
5076
5077                                     ;:TABLE OF TAG BUS CONTROL AND BIT VALUES
5078 045274                               400$:
5079 045274 001777                        .WORD 1777 ;BUS BITS AT HIGH IMPEDENCE STATE
5080 045276 001777                        .WORD 1777
5081 045300 001777                        .WORD 1777
5082 045302 001777                        .WORD 1777
5083 045304 001777                        .WORD 1777
5084 045306 004000                        .WORD CC ;CONTROL BITS ENABLED, BIT 6 ON
5085 045310 004000                        .WORD CC
5086 045312 024000                        .WORD TAG!CC ;TAG COMES ON
5087 045314 024000                        .WORD TAG!CC
5088 045316                               450$:
5089 045316 024000                        .WORD TAG!CC ;START TABLE HERE FOR HARDWARE ECO CHANGE
5090 045320 024000                        .WORD TAG!CC
5091 045322 024000                        .WORD TAG!CC
  
```



```

5092 045324 024000 .WORD TAG!CC
5093 045326 177777 .WORD -1 ;END TABLE HERE FOR HARDWARE ECO CHANGE
5094
5095 : .WORD CC ;TAG GOES OFF
5096 045330 004000 .WORD CC
5097 045332 001777 .WORD 1777 ;CONTROL BITS DISABLED
5098 045334 001777 .WORD 1777
5099 045336 001777 .WORD 1777
5100 045340 001777 .WORD 1777
5101
5102 045342 177777 .WORD -1 ;END OF TABLE
5103
5104 045344 500$: ;END OF TEST
5105
5106

```

::*****
 :*TEST 115 SEARCH TIMEOUT TEST

::*****
 TST115:

```

045344
045344 000004
045346 000240
045350 012706 001100
045354 013700 001276
045360 013701 001462
045364 012737 000115 001226
5107
5108 045372 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
045376 000402 BR 10$ ;BRANCH TO 10$ IF NO ERROR
045400 104000 EMT
5109 045402 000550 BR 90$
5110
5111 ;ENABLE DEBUG CLOCK AND LOAD SEARCH COMMAND
5112 045404 10$:
5113 045404 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
5114 045412 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
5115 045420 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
5116 045426 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
5117 045434 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
5118 045442 012760 000031 000000 MOV #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1
5119
5120 ;EXECUTE SEARCH TO TEST FOR ON LATCH RESET (17 CLOCKS)
5121 045450 012702 000021 MOV #17.,R2
5122 045454 20$:
5123 045462 012760 141401 000024 MOV #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
5124 045470 005302 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
5125 045472 001370 DEC R2
5126 BNE 20$
5127 ;DROP ON CYLINDER TO RESET LATCH, RAISE ON CYLINDER
5128 045474 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5129 045502 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
5130
5131 ;STEP COMMAND SEQUENCER TO SECTOR COMPARE LOOP (2 CLOCKS)
5132 045510 012702 000002 MOV #2,R2
5133 045514 30$:
5134 045522 012760 151401 000024 MOV #DMD!DBEN!MUR!MOC!DBCK!DTO,RMMR1(R0) ;LOAD RMMR1
045522 012760 051401 000024 MOV #DMD!DBEN!MUR!MOC!DTO,RMMR1(R0) ;LOAD RMMR1

```

```

5135 045530 005302          DEC      R2
5136 045532 001370          BNE     30$
5137
5138          ;THE COMMAND SEQUENCER IS NOW IN THE SECTOR COMPARE LOOP. STEP
5139          ;THROUGH THE LOOP AND VERIFY SEARCH IS ENABLED.
5140 045534 012702 000005          MOV     #5,R2
5141 045540          40$:
5142 045546 012760 151401 000024          MOV     #DMD!DBEN!MUR!MOC!DBCK!DTO,RMMR1(R0) ;LOAD RMMR1
5143 045554 016037 000024 001142          MOV     #DMD!DBEN!MUR!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
5144 045562 042737 173777 001142          MOV     RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
5145 045570 001403          BIC     #^CESRC,$BDDAT
5146 045572 005302          BEQ     50$ ;BRANCH IF SEARCH NOT ENABLED
5147 045574 001361          DEC     R2
5148 045576 000412          BNE     40$
5149 045600 012737 004000 001140 50$:          BR      60$
5150 045606 010037 001136          MOV     #ESRC,$GDDAT ;SETUP ERROR MSG
5151 045612 062737 000024 001136          MOV     R0,$BDADR
5152 045620 104265          ADD     #RMMR1,$BDADR
5153 045622 000440          EMT     265
5154          BR      90$
5155          ;DROP DTO TO ENABLE SEARCH TIMEOUT AND WAIT FOR OPI TO SET.
5156 045624          60$:
5157 045624 012760 041401 000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5158 045632 012737 070000 001530          MOV     #70000,WATCH ;SET WATCHDOG TIMER VALUE
5159 045644          JSR     PC,@CLOCK ;START THE CLOCK
5160 045644 016037 000014 001142 70$:          MOV     RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
5161 045652 042737 157777 001142          BIC     #^COPI,$BDDAT
5162 045660 001017          BNE     80$
5163 045662 005737 001530          TST     WATCH
5164 045666 001366          BNE     70$
5165 045670 004777 133640          JSR     PC,@STOP ;STOP THE CLOCK
5166 045674 012737 020000 001140          MOV     #OPI,$GDDAT ;SETUP ERROR MSG
5167 045702 010037 001136          MOV     RC,$BDADR
5168 045706 062737 000014 001136          ADD     #RMER1,$BDADR
5169 045714 104267          EMT     267
5170 045716 000402          BR      90$
5171          80$:          JSR     PC,@STOP ;STOP THE CLOCK
5172 045724          90$:          ;END OF TEST
5173
5174          ;*****
          ;*TEST 116 DATA COMMAND TESTS (1)
          ;*****
          ;TST116:
          SCOPE ;SCOPE CALL
          NOP
          MOV     #STACK,SP ;LOAD THE STACK POINTER
          MOV     $BASE,R0 ;R0 = UNIBUS ADDRESS
          MOV     TSTQUE,R1 ;R1 = POINTER TO DEVICE
          MOV     #116,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
5175
5176          ;VERIFY DATA COMMAND SETS OPI IF DRIVE NOT READY
5177

```



```

5178 045752 004737 055370      JSR    PC,SETVV      ;GO SET VOLUME VALID
      045756 000402          BR     10$          ;BRANCH TO 10$ IF NO ERROR
      045760 104000          EMT
5179 045762 000471          BR     40$
5180
5181      ;ENABLE DEBUG CLOCK AND LOAD READ COMMAND
5182 045764          10$:
5183 045764 012760 041401 000024      MOV    #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5184 045772 012760 000000 000014      MOV    #0,RMER1(R0) ;LOAD RMER1
5185 046000 012760 000000 000042      MOV    #0,RMER2(R0) ;LOAD RMER2
5186 046006 012760 000000 000006      MOV    #0,RMDA(R0) ;LOAD RMDA
5187 046014 012760 000000 000034      MOV    #0,RMDC(R0) ;LOAD RMDC
5188 046022 012760 000071 000000      MOV    #RD!GO,RMCS1(R0) ;LOAD RMCS1
5189
5190      ;STEP COMMAND SEQUENCER TO UNIT READY TEST (3 CLOCKS)
5191 046030 012702 000003      MOV    #3,R2
5192 046034          20$:
5193 046042 012760 141401 000024      MOV    #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5194 046050 005302 041401 000024      MOV    #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5195 046052 001370          DEC    R2
5196          BNE    20$
5197
5198 046054 012760 040401 000024      ;DROP UNIT READY
      MOV    #DMD!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5199
5200      ;STEP SEQUENCER AND VERIFY OPI SETS (2 CLOCKS)
5201 046062 012702 000002      MOV    #2,R2
5202 046066          30$:
5203 046066 012760 140401 000024      MOV    #DMD!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5204 046074 012760 040401 000024      MOV    #DMD!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5205 046102 005302          DEC    R2
5206 046104 001370          BNE    30$
5207 046106 016037 000014 001142      MOV    RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
5208 046114 042737 157777 001142      BIC    #^COPI,$BDDAT
5209 046122 001011          BNE    40$
5210 046124 012737 020000 001140      MOV    #OPI,$GDDAT
5211 046132 010037 001136          MOV    R0,$BDADR
5212 046136 062737 000014 001136      ADD    #RMER1,$BDADR
5213 046144 104270          EMT    270
5214 046146 012737 046154 001124      40$: MOV    #50$,$LPERR ;CHANGE LOOP ON ERROR TEST
5215
5216      ;VERIFY DATA COMMAND ABORTS AT LOCATION 129
5217 046154          50$:
5218 046154 004737 055370      JSR    PC,SETVV      ;GO SET VOLUME VALID
      046160 000402          BR     60$          ;BRANCH TO 60$ IF NO ERROR
      046162 104000          EMT
5219 046164 000576          BR     150$
5220
5221      ;ENABLE DEBUG CLOCK AND LOAD READ COMMAND
5222 046166          60$:
5223 046166 012760 041401 000024      MOV    #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5224 046174 012760 000000 000014      MOV    #0,RMER1(R0) ;LOAD RMER1
5225 046202 012760 000000 000042      MOV    #0,RMER2(R0) ;LOAD RMER2
5226 046210 012760 000000 000006      MOV    #0,RMDA(R0) ;LOAD RMDA
5227 046216 012760 000000 000034      MOV    #0,RMDC(R0) ;LOAD RMDC
5228 046224 012760 000071 000000      MOV    #RD!GO,RMCS1(R0) ;LOAD RMCS1
  
```



```

5229
5230 ;STEP COMMAND SEQUENCER TO ABORT TEST AT LOCATION 129 (4 CLOCKS)
5231 046232 012702 000004 MOV #4,R2
5232 046236 70$:
046236 012760 141401 000024 MOV #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
046244 012760 041401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5233 046244 012760 041401 000024
5234 046252 005302 DEC R2
5235 046254 001370 BNE 70$
5236
5237 ;SET DEVICE FAULT TO CAUSE ABORT CONDITION
5238 046256 012760 041501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5239
5240 ;STEP THE SEQUENCER THROUGH THE TEST FOR ABORT (1 CLOCK)
5241 046264 012702 000001 MOV #1,R2
5242 046270 80$:
046270 012760 141501 000024 MOV #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
046276 012760 041501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5243 046276 012760 041501 000024
5244 046304 005302 DEC R2
5245 046306 001370 BNE 80$
5246
5247 ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5248 ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5249 046310 012702 000020 MOV #16.,R2 ;MAXIMUM NUMBER OF BIT CLOCKS
5250 046314 85$:
046314 012760 045501 000024 MOV #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0) ;LOAD RMMR1
046322 012760 041501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5251 046322 012760 041501 000024
5252 046330 016037 000024 001142 MOV RMMR1(R0),$BDDAT ;STORE RMMR1 AT $BDDAT
5253 046336 042737 157777 001142 BIC #^CEBL,$BDDAT
5254 046344 001014 BNE 90$ ;BRANCH IF EBL IS SET
5255
5256 046346 005302 DEC R2
5257 046350 001361 BNE 85$ ;CONTINUE BIT CLOCKS IF COUNT NOT 0
5258 046352 012737 020000 001140 MOV #EBL,$GDDAT
5259 046360 010037 001136 MOV RO,$BDADR
5260 046364 062737 000024 001136 ADD #RMMR1,$BDADR
5261 046372 104271 EMT 271
5262 046374 000472 BR 150$
5263
5264 ;STEP THE SEQUENCER THROUGH ITS TEST FOR EBL (2 CLOCKS)
5265 046376 90$:
046376 012702 000002 MOV #2,R2
5266 046376 012702 000002
5267 046402 100$:
046402 012760 141501 000024 MOV #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
046410 012760 041501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5268 046402 012760 141501 000024
5269 046410 012760 041501 000024
5270 046416 005302 DEC R2
5271 046420 001370 BNE 100$
5272
5273 ;ABORT EBL SHOULD NOW BE INACTIVE - FORCE BIT CLOCK USING THE
5274 ;MAINTENANCE REGISTER TO RESET EBL (16 BIT CLOCKS)
5275 046422 012702 000020 MOV #16.,R2 ;MAXIMUM NUMBER OF BIT CLOCKS
5276 046426 110$:
046426 012760 045501 000024 MOV #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0) ;LOAD RMMR1
046434 012760 041501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5277 046434 012760 041501 000024
5278 046442 005302 DEC R2
5279 046444 001370 BNE 110$ ;ISSUE 16 BIT CLOCKS THEN TEST
5280
5281 ;VERIFY EBL IS NOW RESET

```



```

5282 046446
5283 046446 016037 000024 001142 120$: MOV RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
5284 046454 042737 157777 001142 BIC #^CEBL, $BDDAT
5285 046462 001411 BEQ 130$ ;BRANCH IF EBL IS RESET.
5286 046454 005037 001140 CLR $GDDAT
5287 046470 010037 001136 MOV RO, $BDADR
5288 046474 062737 000024 001136 ADD #RMMR1, $BDADR
5289 046502 104273 EMT 273
5290 046504 000426 BR 150$
5291
5292 ;VERIFY GO RESETS WITHIN 4 CLOCK CYCLES
5293 046506 130$:
5294 046506 012702 000004 MOV #4, R2
5295 046512 140$:
5296 046520 012760 141501 000024 MOV #DMD!MUR!MOC!DBEN!MDF!DBCK, RMMR1(R0) ;LOAD RMMR1
5297 046526 005302 041501 000024 MOV #DMD!MUR!MOC!DBEN!MDF, RMMR1(R0) ;LOAD RMMR1
5298 046530 001370 DEC R2
5299 046532 016037 000000 001142 BNE 140$
5300 046540 042737 177776 001142 MOV RMCS1(R0), $BDDAT ;STORE RMCS1 AT $BDDAT
5301 046546 001405 BIC #^CGO, $BDDAT
5302 046550 005037 001140 BEQ 150$
5303 046554 010037 001136 CLR $GDDAT
5304 046560 104274 MOV RO, $BDADR
5305 EMT 274
5306 046562 012737 046570 001124 150$: MOV #200$, $LPERR ;CHANGE LOOP ON ERROR ADDRESS
5307
5308 ;VERIFY SEQUENCER BRANCHES TO SEEK WHEN RUN AND GO FLOP SETS
5309 046570 200$:
5310 046570 004737 055370 JSR PC, SETVV ;GO SET VOLUME VALID
5311 046574 000402 BR 210$ ;BRANCH TO 210$ IF NO ERROR
5312 046576 104000 EMT
5313 046600 000512 BR 250$
5314
5315 ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5316 210$:
5317 MOV #DMD!MUR!MOC!DBEN, RMMR1(R0) ;LOAD RMMR1
5318 MOV #0, RMER1(R0) ;LOAD RMER1
5319 MOV #0, RMER2(R0) ;LOAD RMER2
5320 MOV #0, RMDA(R0) ;LOAD RMDA
5321 MOV #0, RMDC(R0) ;LOAD RMDC
5322 MOV #RD!GO, RMCS1(R0) ;LOAD RMCS1
5323
5324 ;MOVE SEQUENCER TO TEST FOR RUN AND GO AT LOCATION 130 (5 CLOCKS)
5325 MOV #5, R2
5326 220$:
5327 MOV #DMD!MUR!MOC!DBEN!DBCK, RMMR1(R0) ;LOAD RMMR1
5328 MOV #DMD!MUR!MOC!DBEN, RMMR1(R0) ;LOAD RMMR1
5329 DEC R2
5330 BNE 220$
5331
5332 ;VERIFY RUN AND GO IS SET
5333 MOV RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
5334 046700 042737 137777 001142 BIC #^CRG, $BDDAT
5335 046706 001012 BNE 230$
5336 046710 012737 040000 001140 MOV #RG, $GDDAT
5337 046716 010037 001136 MOV RO, $BDADR
  
```



```

5335 046722 062737 000024 001136      ADD    #RMMR1,$BDADR
5336 046730 104275                      EMT    275
5337 046732 000435                      BR     250$
5338
5339      ;VERIFY THAT CYLINDER TAG COMES UP IN ONE CLOCK CYCLE
5340 046734                      230$:
5341 046734 012702 000001      MOV    #1,R2
5342 046740                      240$:
      046740 012760 141401 000024      MOV    #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5343 046746 012760 041401 000024      MOV    #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5344 046754 005302                      DEC    R2
5345 046756 001370                      BNE   240$
5346 046760 016037 000040 001142      MOV    RMMR2(R0),$BDDAT ;STORE RMMR2 AT $BDDAT
5347
5348      ;*****
5349      ;      BIC    #RQA!RQB!TAG,$BDDAT
5350
5351      ;THE FOLLOWING CODE WAS ADDED
5352      ;TO ALLOW THE PROGRAM TO RUN WITH
5353      ;OR WITHOUT THE ECO TO THE CS BOARD.
5354 046766 042737 162000 001142      BIC    #RQA!RQB!TAG!CH,$BDDAT ;HARDWARE ECO CHANGE CAUSES CC AND
5355      ;CH TO SET AT THE SAME TIME
5356      ;*****
5357
5358 046774 012737 004000 001140      MOV    #CC,$GDDAT
5359 047002 023737 001140 001142      CMP    $GDDAT,$BDDAT
5360 047010 001406                      BEQ   250$
5361 047012 010037 001136      MOV    R0,$BDADR
5362 047016 062737 000040 001136      ADD    #RMMR2,$BDADR
5363 047024 104276                      EMT    276
5364
5365 047026 012737 047040 001124 250$: MOV    #260$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
5366
5367      ;VERIFY DATA COMMAND ABORTS AT COMMAND SEQUENCER LOCATIONS 144, 145
5368 047034 012702 000144      MOV    #144,R2 ;INITIALIZE TEST LOCATION
5369 047040                      260$:
5370 047040 004737 055370      JSR    PC,SETVV ;GO SET VOLUME VALID
      047044 000402                      BR     270$ ;BRANCH TO 270$ IF NO ERROR
      047046 104000                      EMT
5371 047050 000553                      BR     320$
5372
5373      ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5374 047052                      270$:
5375 047052 012760 041401 000024      MOV    #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5376 047060 012760 000000 000014      MOV    #0,RMER1(R0) ;LOAD RMER1
5377 047066 012760 000000 000042      MOV    #0,RMER2(R0) ;LOAD RMER2
5378 047074 012760 000000 000006      MOV    #0,RMDA(R0) ;LOAD RMDA
5379 047102 012760 000000 000034      MOV    #0,RMDC(R0) ;LOAD RMDC
5380 047110 012760 000071 000000      MOV    #RD!GO,RMCS1(R0) ;LOAD RMCS1
5381
5382      ;WAIT FOR RUN AND GO TO SET
5383 047116 012737 000310 001530      MOV    #200$,WATCH ;SET WATCHDOG TIMER VALUE
      047124 004777 132402      JSR    PC,@CLOCK ;START THE CLOCK
5384 047130                      280$:
      047130 016037 000024 001142      MOV    RMMR1(R0),$BDDAT ;STORE RMMR1 AT $BDDAT
5385 047136 042737 137777 001142      BIC    #^CRG,$BDDAT
5386 047144 001017                      BNE   290$
    
```



```

5440 047406 005002          CLR      R2          ;INITIALIZE TRACK ADDRESS = 0
5441 047410
5442 047410 004737 055370 330$:   JSR      PC,SETVV    ;GO SET VOLUME VALID
          047414 000402          BR      340$        ;BRANCH TO 340$ IF NO ERROR
          047416 104000          EMT
5443 047420 000570          BR      400$
5444
5445          ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5446 047422 340$:   MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5447 047422 012760 041401 000024          MOV      #0,RMER1(R0) ;LOAD RMER1
5448 047430 012760 000000 000014          MOV      #0,RMER2(R0) ;LOAD RMER2
5449 047436 012760 000000 000042          MOV      R2,RMDA(R0) ;LOAD RMDA
5450 047444 010260 000006          MOV      #0,RMDC(R0) ;LOAD RMDC
5451 047450 012760 000000 000034          MOV      #RD!GO,RMCS1(R0) ;LOAD RMCS1
5452 047456 012760 000071 000000
5453
5454          ;WAIT FOR RUN AND GO TO SET
5455 047464 012737 000310 001530          MOV      #200,WATCH ;SET WATCHDOG TIMER VALUE
          047472 004777 132034          JSR      PC,@CLOCK ;START THE CLOCK
5456 047476 350$:   MOV      RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
          047476 016037 000024 001142          BIC      #^CRG,$BDDAT
5457 047504 042737 137777 001142          BNE      360$
5458 047512 001017          TST      WATCH
5459 047514 005737 001530          BNE      350$
5460 047520 001366          JSR      PC,@STOP ;STOP THE CLOCK
5461 047522 004777 132006          MOV      #RG,$GDDAT
5462 047526 012737 040000 001140          MOV      R0,$BDADR
5463 047534 010037 001136          ADD      #RMMR1,$BDADR
5464 047540 062737 000024 001136          EMT      275
5465 047546 104275          BR      400$
5466 047550 000514
5467 047552 360$:   JSR      PC,@STOP ;STOP THE CLOCK
          047552 004777 131756
5468
5469          ;STEP COMMAND SEQUENCER TO HEAD SEQUENCE, LOCATION 156 (17 CLOCKS)
5470 047556 012703 000021          MOV      #17,R3
5471 047562 370$:   MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
          047562 012760 141401 000024          MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5472 047570 012760 041401 000024          DEC      R3
5473 047576 005303          BNE      370$
5474 047600 001370
5475
5476          ;DROP AND RAISE ON CYLINDER TO RESET ON LATCH
5477 047602 012760 041001 000024          MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5478 047610 012760 041401 000024          MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5479
5480          ;*****
5481          ;      MOV      #450$,R3          ;INITIALIZE TABLE POINTER
5482
5483          ;HARDWARE ECO CHANGE TO THE PLA ON THE
5484          ;CS BOARD.
5485 047616 012703 000004          MOV      #4,R3          ;CLOCK THE SEQUENCER THRU THE FIRST
5486          ;4 COMMAND SEQUENCES TO ALLOW THE PROGRAM
5487          ;TO RUN WITH OR WITHOUT THE ECO.
5488 047622 372$:   MOV      #DMD!DBEN!MUR!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
5489 047622 012760 141401 000024          MOV      #DMD!DBEN!MUR!MOC,RMMR1(R0) ;LOAD RMMR1
5490 047630 012760 041401 000024
  
```



```

5491 047636 005303          DEC      R3          ;DONE 4 CLOCKS ?
5492 047640 001370          BNE     372$         ;NO !!
5493 047642 012703 050014  MOV     #475$,R3    ;INITIALIZE NEW TABLE POINTER
5494                                     ;:*****
5495                                     ;:*****
5496                                     ;:*****
5497 047646          ;VERIFY TAG BUS ACCORDING TO TABLE AND TRACK ADDRESS IN R2
5498 047646 016037 000040 001142 375$: MOV     RMMR2(R0), $BDDAT ;STORE RMMR2 AT $BDDAT
5499 047654 042737 150000 001142  BIC     #RQA!RQB!TST, $BDDAT
5500 047662 011337 001140          MOV     (R3), $GDDAT
5501 047666 010204          MOV     R2, R4      ;GENERATE EXPECTED TAG BUS
5502 047670 000304          SWAB   R4
5503 047672 050437 001140          BIS     R4, $GDDAT
5504 047676 023737 001140 001142  CMP     $GDDAT, $BDDAT
5505 047704 001030          BNE     390$
5506 047706 012760 141401 000024  MOV     #DMD!MUR!MOC!DBEN!DBCK, RMMR1(R0) ;LOAD RMMR1
5507 047714 012760 041401 000024  MOV     #DMD!MUR!MOC!DBEN, RMMR1(R0) ;LOAD RMMR1
5508
5509          ;ADVANCE TO NEXT TABLE ENTRY
5510 047722 062703 000002          ADD     #2, R3
5511 047726 005713          TST    (R3)
5512 047730 100401          BMI    380$
5513 047732 000745          BR     375$
5514
5515          ;SHIFT TO NEXT TRACK ADDRESS-EXIT LOOP IF DONE
5516 047734          380$:
5517 047734 062702 000400          ADD     #TA1, R2    ;ADVANCE TRACK ADDRESS
5518 047740 032737 010000 001330  BIT     #TA16, LSTRK ;IS IT RM05 ?
5519 047746 001403          BEQ    385$        ;NO !!
5520 047750 020237 001330          CMP     R2, LSTRK  ;DONE WITH TRACKS ON RM05 ?
5521 047754 101615          BLOS   330$        ;NO !!
5522 047756          385$:
5523 047756 020237 001330          CMP     R2, LSTRK  ;DONE WITH TRACKS ON RM02/3 ?
5524 047762 101007          BHI    400$        ;YES, EXIT
5525 047764 000611          BR     330$
5526
5527          ;ERROR ON TAG BUS DURING HEAD SEQUENCE
5528 047766          390$:
5529 047766 010037 001136          MOV     R0, $BDADR
5530 047772 062737 000040 001136  ADD     #RMMR2, $BDADR
5531 050000 104276          EMT    276
5532
5533 050002 000440          400$: BR     500$    ;JUMP OVER TABLE
5534
5535          ;TABLE OF TAG BUS DURING HEAD SEQUENCE
5536 050004          450$:
5537 050004 002000          .WORD  CH
5538 050006 002000          .WORD  CH
5539 050010 022000          .WORD  CH!TAG
5540 050012 022000          .WORD  CH!TAG
5541 050014          475$:
5542 050014 022000          .WORD  CH!TAG ;START TABLE HERE FOR HARDWARE ECO CHANGE
5543 050016 022000          .WORD  CH!TAG
5544 050020 022000          .WORD  CH!TAG
5545 050022 022000          .WORD  CH!TAG
5546 050024 177777          .WORD  -1      ;END TABLE HERE FOR HARDWARE ECO CHANGE
5547

```

```

5548          ;          .WORD  CH
5549 050026 002000      .WORD  CH
5550 050030 001777      .WORD 1777
5551 050032 001777      .WORD 1777
5552 050034 001777      .WORD 1777
5553 050036 001777      .WORD 1777
5554 050040 001777      .WORD 1777
5555 050042 001777      .WORD 1777
5556 050044 001777      .WORD 1777
5557 050046 001777      .WORD 1777
5558 050050 001777      .WORD 1777
5559 050052 001777      .WORD 1777
5560 050054 001777      .WORD 1777
5561 050056 001777      .WORD 1777
5562 050060 001777      .WORD 1777
5563 050062 001777      .WORD 1777
5564 050064 001777      .WORD 1777
5565 050066 001777      .WORD 1777
5566 050070 001777      .WORD 1777
5567 050072 001777      .WORD 1777
5568 050074 001777      .WORD 1777
5569 050076 001777      .WORD 1777
5570 050100 001777      .WORD 1777
5571
5572 050102 177777      .WORD -1          ;END OF TABLE
5573
5574 050104          500$:          ;END OF TEST
5575
5576
;*****
;*TEST 117      DATA COMMAND TESTS (2)
;*****
TST117:
050104          SCOPE          ;SCOPE CALL
050104 000004      NOP
050106 000240      MOV      #STACK,SP      ;LOAD THE STACK POINTER
050110 012706 001100  MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
050114 013700 001276  MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
050120 013701 001462  MOV      #117,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
050124 012737 000117 001226
5577
5578          ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT RESET
5579
5580 050132 004737 055370  JSR      PC,SETVV      ;GO SET VOLUME VALID
050136 000402      BR      10$          ;BRANCH TO 10$ IF NO ERROR
050140 104000      EMT
5581 050142 000514      BR      60$
5582
5583          ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND DURING CYLINDER SEQUENCE
5584 050144          10$:
5585 050144 012760 041401 000024  MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0)      ;LOAD RMMR1
5586 050152 012760 000000 000014  MOV      #0,RMER1(R0)      ;LOAD RMER1
5587 050160 012760 000000 000042  MOV      #0,RMER2(R0)      ;LOAD RMER2
5588 050166 012760 000000 000006  MOV      #0,RMDA(R0)      ;LOAD RMDA
5589 050174 012760 000000 000034  MOV      #0,RMDC(R0)      ;LOAD RMDC
5590 050202 012760 000071 000000  MOV      #RD!GO,RMCS1(R0)      ;LOAD RMCS1
5591
5592          ;WAIT FOR RUN AND GO TO SET

```



```

5593 050210 012737 000310 001530      MOV    #200.,WATCH      ;SET WATCHDOG TIMER VALUE
      050216 004777 131310      JSR    PC,@CLOCK        ;START THE CLOCK
5594 050222                                20$:
      050222 016037 000024 001142      MOV    RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
5595 050230 042737 137777 001142      BIC    #^CRG, $BDDAT
5596 050236 001017                                BNE    30$
5597 050240 005737 001530      TST    WATCH
5598 050244 001366                                BNE    20$
5599 050246 004777 131262      JSR    PC,@STOP        ;STOP THE CLOCK
5600 050252 012737 040000 001140      MOV    #RG, $GDDAT
5601 050260 010037 001136      MOV    R0, $BDADR
5602 050264 062737 000024 001136      ADD    #RMMR1, $BDADR
5603 050272 104275      EMT    275
5604 050274 000437      BR     60$
5605 050276                                30$:
      050276 004777 131232      JSR    PC,@STOP        ;STOP THE CLOCK
5606
5607                                ;STEP COMMAND SEQUENCER AND VERIFY OPI SETS (19 CLOCKS)
5608 050302 012702 000023      MOV    #19., R2
5609 050306                                40$:
      050306 012760 141401 000024      MOV    #DMD!MUR!MOC!DBEN!DBCK, RMMR1(R0) ;LOAD RMMR1
5610 050314 012760 041401 000024      MOV    #DMD!MUR!MOC!DBEN, RMMR1(R0) ;LOAD RMMR1
5611 050322 005302      DEC    R2
5612 050324 001370                                BNE    40$
5613 050326 016037 000014 001142      MOV    RMER1(R0), $BDDAT ;STORE RMER1 AT $BDDAT
5614 050334 042737 157777 001142      BIC    #^COPI, $BDDAT
5615 050342 001011                                BNE    50$ ;BRANCH IF OPI SET
5616 050344 012737 020000 001140      MOV    #OPE, $GDDAT
5617 050352 010037 001136      MOV    R0, $BDADR
5618 050356 062737 000014 001136      ADD    #RMER1, $BDADR
5619 050364 104277      EMT    277
5620 050366 012737 050374 001124 50$:      MOV    #60$, $LPERR ;CHANGE LOOP ON ERROR ADDRESS
5621
5622                                ;VERIFY DATA COMMAND ABORTS DURING SEEK WAIT LOOP
5623 050374                                60$:
5624 050374 004737 055370      JSR    PC, SETVV ;GO SET VOLUME VALID
      050400 000402      BR     70$ ;BRANCH TO 70$ IF NO ERROR
      050402 10400^      EMT
5625 050404 000567      BR     140$
5626
5627                                ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5628 050406                                70$:
5629 050406 012760 041401 000024      MOV    #DMD!MUR!MOC!DBEN, RMMR1(R0) ;LOAD RMMR1
5630 050414 012760 000000 000014      MOV    #0, RMER1(R0) ;LOAD RMER1
5631 050422 012760 000000 000042      MOV    #0, RMER2(R0) ;LOAD RMER2
5632 050430 012760 000000 000034      MOV    #0, RMDC(R0) ;LOAD RMDC
5633 050436 012760 000000 000006      MOV    #0, RMDA(R0) ;LOAD RMDA
5634 050444 012760 000071 000000      MOV    #RD!GO, RMCS1(R0) ;LOAD RMCS1
5635
5636                                ;WAIT FOR RUN & GO TO SET
5637 050452 012737 000310 001530      MOV    #200.,WATCH ;SET WATCHDOG TIMER VALUE
      050460 004777 131046      JSR    PC,@CLOCK ;START THE CLOCK
5638 050464                                80$:
      050464 016037 000024 001142      MOV    RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
5639 050472 042737 137777 001142      BIC    #^CRG, $BDDAT
5640 050500 001017                                BNE    90$
5641 050502 005737 001530      TST    WATCH

```

```

5642 050506 001366          BNE      80$
5643 050510 004777 131020      JSR      PC,@STOP          ;STOP THE CLOCK
5644 050514 012737 040000 001140      MOV      #RG,$GDDAT
5645 050522 010037 001136          MOV      R0,$BDADR
5646 050526 062737 000024 001136      ADD      #RMMR1,$BDADR
5647 050534 104275          EMT      275
5648 050536 000512          BR       140$
5649 050540          90$:
050540 004777 130770      JSR      PC,@STOP          ;STOP THE CLOCK
5650
5651          ;STEP COMMAND SEQUENCER TO ON LATCH TEST AT LOCATION 156 (17 CLOCKS)
5652 050544 012702 000021      MOV      #17.,R2
5653 050550          100$:
050550 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5654 050556 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5655 050564 005302          DEC      R2
5656 050566 001370          BNE      100$
5657
5658          ;DROP ON CYLINDER TO RESET LATCH
5659 050570 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5660
5661          ;MOVE COMMAND SEQUENCER TO SEEK WAIT LOOP (31 CLOCKS)
5662 050576 012702 000037      MOV      #31.,R2
5663 050602          110$:
050602 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5664 050610 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5665 050616 005302          DEC      R2
5666 050620 001370          BNE      110$
5667
5668          ;STEP THROUGH SEEK WAIT LOOP (6 CLOCKS) 2 TIMES
5669 050622 012702 000006      MOV      #6.,R2
5670 050626          120$:
050626 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5671 050634 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5672 050642 005302          DEC      R2
5673 050644 001370          BNE      120$
5674
5675          ;SET SEEK INCOMPLETE ERROR TO CAUSE ABORT
5676 050646 012760 041201 000024      MOV      #DMD!MUR!DBEN!MSER,RMMR1(R0) ;LOAD RMMR1
5677
5678          ;CLOCK THE SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
5679 050654 012702 000002      MOV      #2,R2
5680 050660          130$:
050660 012760 141201 000024      MOV      #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0) ;LOAD RMMR1
5681 050666 012760 041201 000024      MOV      #DMD!MUR!DBEN!MSER,RMMR1(R0) ;LOAD RMMR1
5682 050674 005302          DEC      R2
5683 050676 001370          BNE      130$
5684
5685          ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5686          ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5687 050700 012702 000020      MOV      #16.,R2          ;MAXIMUM NUMBER OF BIT CLOCKS
5688 050704          135$:
050704 012760 045501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0) ;LOAD RMMR1
5689 050712 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5690 050720 016037 000024 001142      MOV      RMMR1(R0),$BDDAT ;STORE RMMR1 AT $BDDAT
5691 050726 042737 157777 001142      BIC      #^CEBL,$BDDAT
5692 050734 001013          BNE      140$
    
```



```

5693
5694 050736 005302          DEC      R2
5695 050740 001361          BNE     135$          ;CONTINUE BIT CLOCKS IF COUNT NOT 0
5696 050742 012737 020000 001140  MOV     #EBL,$GDDAT
5697 050750 010037 001136  MOV     RO,$BDADR
5698 050754 062737 000024 001136  ADD     #RMMR1,$BDADR
5699 050762 104300          EMT     300
5700 050764 012737 050772 001124 140$:  MOV     #150$,$LPERR          ;CHANGE LOOP ON ERROR ADDRESS
5701
5702
5703          ;VERIFY DATA COMMAND ABORTS DURING OFFSET IF ON CYLINDER LATCH
5704 050772          ;DOESNT RESET
5705 050772 004737 055370 150$:  JSR     PC,SETVV          ;GO SET VOLUME VALID
5706 051002 000536          BR      160$          ;BRANCH TO 160$ IF NO ERROR
5707
5708          ;LOAD TRACK, SECTOR, AND CYLINDER ADDRESSES
5709 051004 160$:  MOV     #0,RMDC(RO)          ;LOAD RMDC
5710 051004 012760 000000 000034  MOV     #0,RMDA(RO)          ;LOAD RMDA
5711 051012 012760 000000 000006  JSR     PC,SETOM          ;GO SET OFFSET MODE
5712 051020 004737 055512  BR      170$          ;BRANCH TO 170$ IF NO ERROR
5713 051024 000402
5714 051026 104000
5715 051030 000523  BR      220$
5716
5717          ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5718 051032 170$:  MOV     #DMD!MUR!MOC!DBEN,RMMR1(RO)          ;LOAD RMMR1
5719 051040 012760 041401 000024  MOV     #0,RMER1(RO)          ;LOAD RMER1
5720 051046 012760 000000 000014  MOV     #0,RMER2(RO)          ;LOAD RMER2
5721 051054 012760 000071 000042  MOV     #RD!GO,RMCS1(RO)          ;LOAD RMCS1
5722
5723          ;WAIT FOR RUN AND GO TO SET
5724 051062 012737 000310 001530  MOV     #200$,WATCH          ;SET WATCHDOG TIMER VALUE
5725 051070 004777 130436  JSR     PC,@CLOCK          ;START THE CLOCK
5726
5727 180$:  MOV     RMMR1(RO),$BDDAT          ;STORE RMMR1 AT $BDDAT
5728 051074 016037 000024 001142  BIC     #^CRG,$BDDAT
5729 051102 042737 137777 001142  BNE     190$
5730 051110 001017  TST     WATCH
5731 051112 005737 001530  BNE     180$
5732 051116 001366  JSR     PC,@STOP          ;STOP THE CLOCK
5733 051120 004777 130410  MOV     #RG,$GDDAT
5734 051124 012737 040000 001140  MOV     RO,$GDDAT
5735 051132 010037 001140  ADD     #RMMR1,$BDADR
5736 051136 062737 000024 001136  EMT     275
5737 051144 104275  BR      220$
5738 051146 000454 190$:  JSR     PC,@STOP          ;STOP THE CLOCK
5739 051150 004777 130360
5740
5741          ;STEP COMMAND SEQUENCER TO ON LATCH TEST AT LOCATION 156 (17 CLOCKS)
5742 051154 012702 000021 200$:  MOV     #17$,R2
5743 051160 012760 141401 000024  MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO)          ;LOAD RMMR1
5744 051166 012760 041401 000024  MOV     #DMD!MUR!MOC!DBEN,RMMR1(RO)          ;LOAD RMMR1
5745 051174 005302  DEC     R2
  
```



```

5742 051176 001370          BNE      200$
5743
5744          ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
5745          ;AT LOCATION 166
5746 051200 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5747 051206 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
5748
5749          ;MOVE SEQUENCER TO SET OPI AND EBL (39 CLOCKS)
5750 051214 012702 000047          MOV      #39.,R2
5751 051220          210$:
5752 051226 012760 141401 000024      MOV      #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0) ;LOAD RMMR1
5753 051234 005302          MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
5754 051236 001370          DEC      R2
5755          BNE      210$
5756          ;VERIFY OPI IS SET
5757 051240 016037 000014 0C1142      MOV      RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
5758 051246 042737 157777 001142      BIC      #^COPI,SBDDAT
5759 051254 001011          BNE      220$
5760 051256 012737 020000 001140      MOV      #OPI,$GDDAT
5761 051264 010037 001136          MOV      R0,$BDADR
5762 051270 062737 000014 001136      ADD      #RMER1,$BDADR
5763 051276 104277          EMT      277
5764 051300 012737 051306 001124 220$: MOV      #230$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
5765
5766          ;VERIFY DATA COMMAND ABORTS DURING OFFSET WAIT LOOP
5767
5768 051306          230$:
5769 051306 004737 055370          JSR      PC,SETVV ;GO SET VOLUME VALID
5770 051312 000403          BR      240$ ;BRANCH TO 240$ IF NO ERROR
5771 051314 104000          EMT
5772 051316 000137 051726          JMP      310$
5773
5774          ;LOAD SECTOR,TRACK AND CYLINDER ADDRESS
5775 051322          240$:
5776 051322 012760 000000 000006      MOV      #0,RMDA(R0) ;LOAD RMDA
5777 051330 012760 000000 000034      MOV      #0,RMDC(R0) ;LOAD RMDC
5778 051336 004737 055512          JSR      PC,SETOM ;GO SET OFFSET MODE
5779 051342 000402          BR      245$ ;BRANCH TO 245$ IF NO ERROR
5780 051344 104000          EMT
5781 051346 000567          BR      310$
5782
5783          ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5784 051350          245$:
5785 051350 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5786 051356 012760 000000 000014      MOV      #0,RMER1(R0) ;LOAD RMER1
5787 051364 012760 000000 000042      MOV      #0,RMER2(R0) ;LOAD RMER2
5788 051372 012760 000071 000000      MOV      #RD!GO,RMCS1(R0) ;LOAD RMCS1
5789
5790          ;WAIT FOR RUN AND GO TO SET
5791 051400 012737 000310 001530      MOV      #200.,WATCH ;SET WATCHDOG TIMER VALUE
5792 051406 004777 130120          JSR      PC,@CLOCK ;START THE CLOCK
5793
5794          250$:
5795 051412 016037 000024 001142      MOV      RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
5796 051420 042737 137777 001142      BIC      #^CRG,SBDDAT
5797 051426 001017          BNE      260$
5798 051430 005737 001530          TST      WATCH
  
```



```

5790 051434 001366          BNE      250$
5791 051436 004777 130072    JSR      PC,@STOP          ;STOP THE CLOCK
5792 051442 012737 040000 001140  MOV     #RG,$GDDAT
5793 051450 010037 001136    MOV     RO,$BDADR
5794 051454 062737 000024 001136  ADD     #RMMR1,$BDADR
5795 051462 104275          EMT     275
5796 051464 000520          BR      310$
5797 051466          260$:   JSR      PC,@STOP          ;STOP THE CLOCK
5798 051466 004777 130042
5799          ;STEP SEQUENCER TO LOCATION 156 (17 CLOCKS)
5800 051472 012702 000021    MOV     #17.,R2
5801 051476          270$:
5802 051504 012760 141401 000024  MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5803 051512 005302 041401 000024  MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5804 051514 001370          DEC     R2
5805          BNE     270$
5806          ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
5807          ;AT LOCATION 166
5808 051516 012760 041001 000024  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5809 051524 012760 041401 000024  MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5810
5811          ;MOVE SEQUENCER TO LOCATION 174 (37 CLOCKS)
5812 051532 012702 000045    MOV     #37.,R2
5813 051536          280$:
5814 051536 012760 141401 000024  MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5815 051544 012760 041401 000024  MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5816 051552 005302          DEC     R2
5817 051554 001370          BNE     280$
5818
5819 051556 012760 041001 000024  ;DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER RESET
5820          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5821
5822          ;STEP SEQUENCER THROUGH OFFSET WAIT LOOP TWICE (7 CLOCKS)
5823 051564 012702 000007    MOV     #7.,R2
5824 051570 012760 141001 000024  290$:   MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5825 051576 012760 041001 000024  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5826 051604 005302          DEC     R2
5827 051606 001370          BNE     290$
5828
5829 051610 012760 041101 000024  ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
5830          MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5831
5832          ;STEP SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
5833 051616 012702 000002    MOV     #2.,R2
5834 051622 012760 141101 000024  300$:   MOV     #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
5835 051630 012760 041101 000024  MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5836 051636 005302          DEC     R2
5837 051640 001370          BNE     300$
5838
5839          ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5840          ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5841 051642 012702 000020    MOV     #16.,R2          ;MAXIMUM NUMBER OF BIT CLOCKS
5841 051646          305$:

```



```

5842 051646 012760 045501 000024 MOV #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0) ;LOAD RMMR1
5843 051654 012760 041501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5843 051662 016037 000024 001142 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
5844 051670 042737 157777 001142 BIC #^CEBL,SBDDAT
5845 051676 001013 BNE 310$
5846
5847 051700 005302 DEC R2
5848 051702 001361 BNE 305$ ;CONTINUE BIT CLOCKS IF COUNT NOT 0
5849 051704 012737 020000 001140 MOV #EBL,$GDDAT
5850 051712 010037 001136 MOV R0,$BDADR
5851 051716 062737 000024 001136 ADD #RMMR1,$BDADR
5852 051724 104271 EMT 271
5853
5854 051726 012737 051734 001124 310$: MOV #320$,SLPERR ;CHANGE LOOP ON ERROR ADDRESS
5855
5856 ;VERIFY THAT DATA COMMAND ABORTS DURING SECTOR WAIT LOOP AT LOCATION 179
5857
5858 051734 320$:
5858 051734 004737 055370 JSR PC,SETVV ;GO SET VOLUME VALID
5858 051740 000403 BR 330$ ;BRANCH TO 330$ IF NO ERROR
5858 051742 104000 EMT
5859 051744 000137 052372 JMP 420$
5860
5861 ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5862 051750 330$:
5863 051750 012760 041401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5864 051756 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
5865 051764 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
5866 051772 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
5867 052000 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
5868 052006 012760 000071 000000 MOV #RD!GO,RMCS1(R0) ;LOAD RMCS1
5869
5870 ;WAIT FOR RUN AND GO TO SET
5871 052014 012737 000310 001530 MOV #200,.WATCH ;SET WATCHDOG TIMER VALUE
5871 052022 004777 127504 JSR PC,@CLOCK ;START THE CLOCK
5872 052026 340$:
5873 052026 016037 000024 001142 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
5873 052034 042737 137777 001142 BIC #^CRG,SBDDAT
5874 052042 001017 BNE 350$
5875 052044 005737 001530 TST WATCH
5876 052050 001366 BNE 340$
5877 052052 004777 127456 JSR PC,@STOP ;STOP THE CLOCK
5878 052056 012737 040000 001140 MOV #RG,$GDDAT
5879 052064 010037 001136 MOV R0,$BDADR
5880 052070 062737 000024 001136 ADD #RMMR1,$BDADR
5881 052076 104275 EMT 275
5882 052100 000534 BR 420$
5883 052102 350$:
5883 052102 004777 127426 JSR PC,@STOP ;STOP THE CLOCK
5884
5885 ;STEP SEQUENCER TO LOCATION 156 (17 CLOCKS)
5886 052106 012702 000021 MOV #17.,R2
5887 052112 360$:
5888 052112 012760 141401 000024 MOV #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5888 052120 012760 041401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5889 052126 005302 DEC R2
5890 052130 001370 BNE 360$
  
```



```

5891
5892 ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
5893 ;AT LOCATION 166
5894 052132 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5895
5896 ;MOVE SEQUENCER TO SECTOR WAIT (34 CLOCKS)
5897 052140 012702 000042      MOV      #34.,R2
5898 052144      370$:
5899 052152 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5900 052160 005302 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5901 052162 001370      DEC      R2
5902      BNE      370$
5903 ;STEP THROUGH SECTOR WAIT LOOP TWICE AND VERIFY SEARCH IS ENABLED
5904 ;DURING THE LOOP (6 CLOCKS)
5905 052164 012702 000006      MOV      #6.,R2
5906 052170      380$:
5907 052176 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5908 052204 016037 000024 001142      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5909 052212 042737 173777 001142      MOV      RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
5910 052220 001403      BIC      #^CESRC,$BDDAT
5911 052222 005302      BEQ      390$
5912 052224 001361      DEC      R2
5913 052226 000412      BNE      380$
5914 052230 012737 004000 001140 390$:      BR       400$
5915 052236 010037 001136      MOV      #ESRC,$GDDAT
5916 052242 062737 000024 001136      MOV      R0,$BDADR
5917 052250 104301      ADD      #RMMR1,$BDADR
5918 052252 000447      EMT      301
5919      BR       420$
5920 ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
5921 052254      400$:
5922 052254 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5923
5924 052262 012702 000002      ;STEP SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
5925 052266      MOV      #2,R2
5926 052266 012760 141501 000024 410$:
5927 052274 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
5928 052302 005302      MOV      #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5929 052304 001370      DEC      R2
5930      BNE      410$
5931 ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5932 ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5933 052306 012702 000020      MOV      #16.,R2 ;MAXIMUM NUMBER OF BIT CLOCKS
5934 052312 012760 045501 000024 415$:
5935 052320 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0) ;LOAD RMMR1
5936      MOV      #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5937 052326 016037 000024 001142      ;VERIFY EBL IS SET
5938 052334 042737 157777 001142      MOV      RMMR1(R0), $BDDAT ;STORE RMMR1 AT $BDDAT
5939 052342 001013      BIC      #^CEBL,$BDDAT
5940      BNE      420$
5941      DEC      R2
5942 052344 005302      BNE      415$ ;CONTINUE BIT CLOCKS IF COUNT NOT 0

```



```

T117 DATA COMMAND TESTS (2)
5943 052350 012737 020000 001140      MOV    #EBL,$GDDAT
5944 052356 010037 001136      MOV    RO,$BDADR
5945 052362 062737 000024 001136      ADD    #RMMR1,$BDADR
5946 052370 104271      EMT    271
5947
5948 052372      420$:      ;END OF TEST
5949
5950
;*****
;*TEST 120 DATA COMMAND TESTS (3)
;*****
TST120:
052372      000004      SCOPE      ;SCOPE CALL
052374      000240      NOP
052376      012706 001100      MOV    #STACK,SP ;LOAD THE STACK POINTER
052402      013700 001276      MOV    $BASE,RO  ;RO = UNIBUS ADDRESS
052406      013701 001462      MOV    TSTQUE,R1 ;R1 = POINTER TO DEVICE
052412      012737 000120 001226      MOV    #120,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
5951
5952      ;VERIFY THE TAG BUS DURING DATA COMMAND
5953      ;FIRST PART USES OFFSET FORWARD
5954
5955      ;LOAD TEST PARAMETERS IN REGISTER OUTPUT BUFFER
5956 052420 012737 001466 001442      MOV    #822.,RMDCO ;LAST CYLINDER
5957 052426 013737 001330 001414      MOV    LSTRK,RMDAO ;SETUP LAST TRACK AND
5958 052434 112737 000035 001414      MOV    #29.,RMDAO ;LAST SECTOR
5959 052442
5960 052442 012737 000200 001440      5$:      MOV    #OFD,RMOFO ;FORWARD OFFSET
5961 052450 012737 000071 001406      MOV    #RD!GO,RMCS10 ;READ DATA
5962 052456 012737 177400 001410      MOV    #-256.,RMWCO ;WORD COUNT
5963 052464 012737 103712 001412      MOV    #BUFFER,RMBAO ;BUFFER ADDRESS
5964
5965      ;EXECUTE COMMAND AND VERIFY TAG BUS USING SUBROUTINE
5966 052472 004737 052514      JSR    PC,10$
5967
5968      ;SECOND PART USES OFFSET REVERSE
5969      ;LOAD TEST PARAMETERS IN REGISTER OUTPUT BUFFER
5970 052476 112737 000000 001440      MOV    #0,RMOFO ;REVERSE OFFSET
5971
5972      ;EXECUTE COMMAND AND VERIFY TAG BUS USING SUBROUTINE
5973 052504 004737 052514      JSR    PC,10$
5974 052510 000137 053600      JMP    300$
5975
5976      ;*****
5977      ;SUBROUTINE USED DURING TEST
5978      ;*****
5979
5980 052514
5981 052514 004737 055370      10$:      JSR    PC,SETVV ;GO SET VOLUME VALID
5982 052520 000403      BR    20$ ;BRANCH TO 20$ IF NO ERROR
5983 052522 104000      EMT
5984 052524 000137 053416      JMP    160$
5985
5986      ;LOAD TRACK, SECTOR AND CYLINDER ADDRESS, LOAD OFFSET
5987 052530      20$:      MOV    RMDAO,RMDA(RO) ;LOAD RMDA
5988 052536 013760 001414 000006      MOV    RMDCO,RMDC(RO) ;LOAD RMDC
5989 052536 013760 001442 000034

```



```

5988 052544 013760 001440 000032      MOV      RMOFO,RMOF(R0)  ;LOAD RMOF
5989 052552 004737 055512      JSR      PC,SETOM       ;GO SET OFFSET MODE
      052556 000403      BR       30$           ;BRANCH TO 30$ IF NO ERROR
      052560 104000      EMT
5990 052562 000137 053416      JMP      160$
5991
5992      ;LOAD BUFFER ADDRESS AND WORD COUNT
5993 052566 30$:
5994 052566 013760 001410 000002      MOV      RMWCO,RMWC(R0)  ;LOAD RMWC
5995 052574 013760 001412 000004      MOV      RMBAO,RMBA(R0)  ;LOAD RMBA
5996
5997      ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5998 052602 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5999 052610 012760 000000 000014      MOV      #0,RMER1(R0)    ;LOAD RMER1
6000 052616 012760 000000 000042      MOV      #0,RMER2(R0)    ;LOAD RMER2
6001 052624 013760 001406 000000      MOV      RMCS10,RMCS1(R0) ;LOAD RMCS1
6002
6003      ;WAIT FOR RUN AND GO TO SET
6004 052632 012737 000310 001530      MOV      #200.,WATCH     ;SET WATCHDOG TIMER VALUE
      052640 004777 126666      JSR      PC,@CLOCK       ;START THE CLOCK
6005 052644 40$:
      052644 016037 000024 001142      MOV      RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
6006 052652 042737 137777 001142      BIC      #^CRG,SBDDAT
6007 052660 001020      BNE      50$
6008 052662 005737 001530      TST      WATCH
6009 052666 001366      BNE      40$
6010 052670 004777 126640      JSR      PC,@STOP       ;STOP THE CLOCK
6011 052674 012737 040000 001140      MOV      #RG,$GDDAT
6012 052702 010037 001136      MOV      R0,$BDADR
6013 052706 062737 000024 001136      ADD      #RMMR1,$BDADR
6014 052714 104275      EMT
6015 052716 000137 053416      JMP      160$
6016 052722 50$:
      052722 004777 126606      JSR      PC,@STOP       ;STOP THE CLOCK
6017 052726 012704 053420      MOV      #200$,R4       ;R4 = TABLE POINTER
6018
6019      ;STEP SEQUENCER TO HEAD SEQUENCE AT LOCATION 156 (17 CLOCKS)
6020 052732 012705 000021      MOV      #17.,R5        ;R5 = CLOCK COUNT
6021 052736 60$:
      052736 016037 000040 001142      MOV      RMMR2(R0),SBDDAT ;STORE RMMR2 AT SBDDAT
6022
6023      ;*****
6024 052744 013737 001142 001174      MOV      $BDDAT,$TMPO   ;IF CC AND CH ARE SET AT THE SAME TIME
6025      ;THE ECO TO THE CS BOARD IS IMPLEMENTED.
6026
6027 052752 042737 171777 001174      BIC      #^C<CC!CH>,$TMPO ;SAVE CC AND CH BITS
6028 052760 022737 006000 001174      CMP      #CC!CH,$TMPO   ;ARE CC AND CH SET ?
6029 052766 001414      BEQ      65$           ;YES, BRANCH TO NEW LOCATION
6030      ;*****
6031
6032 052770 042737 150000 001142      BIC      #RQA!RQB!TST,$BDDAT
6033 052776 012437 001140      MOV      (R4)+,$GDDAT
6034 053002 053737 001442 001140      BIS      RMDCO,$GDDAT   ;OR CYLINDER ADDRESS
6035 053010 023737 001140 001142      CMP      $GDDAT,$BDDAT
6036 053016 001011      BNE      70$           ;BRANCH IF TAG BUS WRONG
6037 053020 65$:
6038 053020 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1

```



```

6039 053026 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0)      ;LOAD RMMR1
6040 053034 005305                      DEC      R5
6041 053036 001337                      BNE      60$
6042 053040 000407                      BR       80$
6043 053042 010037 001136 70$:    MOV      R0,$BDADR
6044 053046 062737 000040 001136    ADD      #RMMR2,$BDADR
6045 053054 104276                      EMT      276
6046 053056 000557                      BR       160$
6047
6048                                     ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST AT
6049                                     ;LOCATION 166
6050 053060                                     80$:
6051 053060 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
6052 053066 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
6053
6054                                     ;STEP SEQUENCER TO END OF OFFSET AT LOCATION 174 (37 CLOCKS)
6055 053074 012705 000045                      MOV      #37.,R5 ;RELOAD CLOCK COUNT
6056 053100                                     90$:
6057 053100 016037 000040 001142      MOV      RMMR2(R0),$BDDAT ;STORE RMMR2 AT $BDDAT
6058
6059 053106 013737 001142 001174      ;:*****
6060                                     MOV      $BDDAT,$TMPO ;IF CC AND CH ARE SET AT SAME TIME
6061                                     ;THE ECO TO THE CS BOARD IS IMPLEMENTED.
6062 053114 042737 171777 001174      BIC      #^C<<CC!CH>,$TMPO ;SAVE CC AND CH BITS
6063 053122 022737 006000 001174      CMP      #CC!CH,$TMPO ;ARE CC AND CH SET ?
6064 053130 001003                      BNE      92$ ;NO !!
6065 053132 162704 000002                      SUB      #2,R4 ;ADJUST THE TABLE ADDRESS
6066 053136 000441                      BR       115$ ;TO OTHER LOCATION
6067 053140                                     92$:
6068                                     ;:*****
6069
6070 053140 042737 150000 001142      BIC      #RQA!RQB!TST,$BDDAT
6071 053146 011437 001140      MOV      (R4),$GDDAT
6072 053152 032714 002000      BIT      #CH,(R4)
6073 053156 001425                      BEQ      110$ ;BRANCH IF CONTROL/HEADER NOT ON
6074 053160 032714 004000      BIT      #CC,(R4)
6075 053164 001416                      BEQ      100$ ;BRANCH IF HEADER TAG
6076
6077                                     ;CONTROL TAG SHOULD BE ON-SETUP EXPECTED OFFSET
6078 053166 052737 000010 001140      BIS      #BB03,$GDDAT ;ASSUME OFD IS NOT SET
6079 053174 032737 000200 001440      BIT      #OFD,RMOFO
6080 053202 001406                      BEQ      95$
6081 053204 042737 000010 001140      BIC      #BB03,$GDDAT ;RESET BUS BIT 3 - DIRECTION IS REV
6082 053212 052737 000004 001140      BIS      #BB02,$GDDAT
6083 053220 000404                                     95$:
6084                                     BR       110$
6085                                     ;HEADER TAG SHOULD BE ON - SETUP EXPECTED TRACK ADDRESS
6086 053222                                     100$:
6087 053222 113703 001415      MOV      RMDAO+1,R3 ;GET TRACK
6088 053226 050337 001140      BIS      R3,$GDDAT
6089
6090                                     ;COMPARE EXPECTED AND RECEIVED TAG BUS DATA
6091 053232                                     110$:
6092 053232 023737 001140 001142      CMP      $GDDAT,$BDDAT
6093 053240 001013                      BNE      120$
6094 053242                                     115$:

```



```

6095 053242 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)      ;LOAD RMMR1
6096 053250 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0)      ;LOAD RMMR1
6096 053256 062704 000002      ADD      #2,R4      ;MOVE TABLE POINTER
6097 053262 005305      DEC      R5      ;DECREMENT CLOCK COUNT
6098 053264 001305      BNE      90$
6099 053266 000407      BR       130$
6100 053270 010037 001136 120$:  MOV      R0,$BDADR
6101 053274 062737 000040 001136  ADD      #RMMR2,$BDADR
6102 053302 104276      EMT      276
6103 053304 000444      BR       160$
6104
6105      ;DROP ON CYLINDER TO RESET LATCH, RAISE ON CYLINDER TO PASS TEST AT
6106      ;SEQUENCER LOCATION 175
6107 053306 130$:
6108 053306 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
6109 053314 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
6110
6111      ;STEP SEQUENCER TO SECTOR WAIT LOOP (8 CLOCKS)
6112 053322 012705 000010      MOV      #8.,R5
6113 053326 140$:  MOV      RMMR2(R0),$BDDAT ;STORE RMMR2 AT $BDDAT
        053326 016037 000040 001142
6114
6115      ;*****
6116      ;      BIC      #RQA!RQB!TST,$BDDAT
6117
6118      ;THE FOLLOWING CODE WAS ADDED
6119      ;TO ALLOW THE PROGRAM TO RUN WITH
6120      ;OR WITHOUT ECO TO THE CS BOARD.
6121 053334 042737 171777 001142      BIC      #^C<CC!CH>,$BDDAT ;SAVE CC AND CH BITS
6122 053342 012737 006000 001140      MOV      #CC!CH,$GDDAT ;GET EXPECTED DATA
6123      ;*****
6124
6125 053350 023737 001140 001142      CMP      $GDDAT,$BDDAT ;GOOD DATA SAME AS LAST CMP
6126 053356 001011      BNE      150$ ;BRANCH IF ERROR
6127 053360 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
6128 053366 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
6129 053374 005305      DEC      R5
6130 053376 001353      BNE      140$
6131 053400 000406      BR       160$
6132 053402 010037 001136 150$:  MOV      R0,$BDADR
6133 053406 062737 000040 001136  ADD      #RMMR2,$BDADR
6134 053414 104276      EMT      276
6135
6136 053416 000207 160$:  RTS      PC
6137
6138      ;TABLE OF TAG BUS CONTROL AND DATA VALUES
6139 053420 200$:
6140 053420 001777      .WORD    1777 ;LOCATION 0
6141 053422 001777      .WORD    1777 ;LOCATION 25
6142 053424 001777      .WORD    1777 ;LOCATION 26
6143 053426 001777      .WORD    1777 ;LOCATION 128
6144 053430 001777      .WORD    1777 ;LOCATION 129
6145 053432 001777      .WORD    1777 ;LOCATION 130
6146 053434 004000      .WORD    CC ;LOCATION 144
6147 053436 004000      .WORD    CC ;LOCATION 145
6148 053440 024000      .WORD    CC!TAG ;LOCATION 146
6149 053442 024000      .WORD    CC!TAG ;LOCATION 147

```

6150	053444	024000	.WORD	CC!TAG	:LOCATION	148
6151	053446	024000	.WORD	CC!TAG	:LOCATION	149
6152	053450	024000	.WORD	CC!TAG	:LOCATION	150
6153	053452	024000	.WORD	CC!TAG	:LOCATION	151
6154	053454	004000	.WORD	CC	:LOCATION	152
6155	053456	004000	.WORD	CC	:LOCATION	153
6156	053460	001777	.WORD	1777	:LOCATION	154
6157	053462	002000	.WORD	CH	:LOCATION	156
6158	053464	002000	.WORD	CH	:LOCATION	157
6159	053466	022000	.WORD	CH!TAG	:LOCATION	158
6160	053470	022000	.WORD	CH!TAG	:LOCATION	159
6161	053472	022000	.WORD	CH!TAG	:LOCATION	160
6162	053474	022000	.WORD	CH!TAG	:LOCATION	161
6163	053476	022000	.WORD	CH!TAG	:LOCATION	162
6164	053500	022000	.WORD	CH!TAG	:LOCATION	163
6165	053502	002000	.WORD	CH	:LOCATION	164
6166	053504	002000	.WORD	CH	:LOCATION	165
6167	053506	001777	.WORD	1777	:LOCATION	232
6168	053510	001777	.WORD	1777	:LOCATION	233
6169	053512	001777	.WORD	1777	:LOCATION	234
6170	053514	001777	.WORD	1777	:LOCATION	235
6171	053516	001777	.WORD	1777	:LOCATION	236
6172	053520	001777	.WORD	1777	:LOCATION	237
6173	053522	001777	.WORD	1777	:LOCATION	238
6174	053524	001777	.WORD	1777	:LOCATION	239
6175	053526	001777	.WORD	1777	:LOCATION	240
6176	053530	001777	.WORD	1777	:LOCATION	241
6177	053532	001777	.WORD	1777	:LOCATION	242
6178	053534	001777	.WORD	1777	:LOCATION	243
6179	053536	001777	.WORD	1777	:LOCATION	244
6180	053540	001777	.WORD	1777	:LOCATION	245
6181	053542	001777	.WORD	1777	:LOCATION	246
6182	053544	001777	.WORD	1777	:LOCATION	247
6183	053546	001777	.WORD	1777	:LOCATION	248
6184	053550	001777	.WORD	1777	:LOCATION	249
6185	053552	001777	.WORD	1777	:LOCATION	250
6186	053554	001777	.WORD	1777	:LOCATION	251
6187	053556	001777	.WORD	1777	:LOCATION	252
6188	053560	001777	.WORD	1777	:LOCATION	166
6189	053562	006000	.WORD	CC!CH	:LOCATION	169
6190	053564	006000	.WORD	CC!CH	:LOCATION	170
6191	053566	026000	.WORD	CC!CH!TAG	:LOCATION	171
6192	053570	026000	.WORD	CC!CH!TAG	:LOCATION	172
6193	053572	026000	.WORD	CC!CH!TAG	:LOCATION	173
6194	053574	026000	.WORD	CC!CH!TAG	:LOCATION	174
6195	053576	026000	.WORD	CC!CH!TAG	:LOCATION	175,ETC
6196						
6197	053600		300\$:		:END OF TEST	
6198						

054022	001405			BEQ	\$DOAGN	::BRANCH IF NO MONITOR
054024	000005			RESET		::CLEAR THE WORLD
054026	004710			\$ENDAD: JSR	PC,(R0)	::GO TO MONITOR
054030	000240			NOP		::SAVE ROOM
054032	000240			NOP		::FOR
054034	000240			NOP		::ACT11
054036				\$DOAGN:		
054036	000137			JMP	@(PC)+	::RETURN
054040	007152			\$RTNAD: .WORD	READY	
054042	377	377	000	\$ENULL: .BYTE	-1,-1,0	::NULL CHARACTER STRING
				.EVEN		

21


```

1      .SBTTL SUBROUTINES
2      :*****
3      .SBTTL ERROR TYPEOUT ROUTINE
4
5      :*THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
6      :*REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
7      :*
8      :*      .UNIT NUMBER, TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER ARE
9      :*PRINTED ON THE FIRST LINE;
10     :*      .ERROR MESSAGE IS ASSEMBLED, FORMATTED AND PRINTED ON
11     :*ONE OR MORE SUCCEEDING LINES;
12     :*      .PAIRED LINES OF ERROR HEADERS AND ERROR DATA
13     :*ARE PRINTED AFTER THE ERROR MESSAGE.
14
15     054046
16     054046 104414
17     054050 032777 020000 125076
18     054056 001402
19     054060 000137 054662
20
21     054064 104401 001217
22     054070 104401 054676
23     054074 013746 001234
24
25     054100 104403
26     054102 003
27     054103 000
28
29     054104 016000 000026
30     054110 042700 177740
31     054114 012737 054745 054166
32     054122 022700 000024
33     054126 001414
34
35     054130 012737 054740 054166
36     054136 022700 000025
37     054142 001406
38
39     054144 012737 054752 054166
40     054152 022700 000027
41     054156 001004
42     054160 104401 054734
43     054164 104401
44     054166 000000
45
46     054170 005037 054666
47     054174 013737 001226 054666
48     054202 104401 054704
49     054206 013746 054666
50
51     054212 104403
52     054214 003
53     054215 000
54
55     054216 005037 054670
56     054222 113737 001130 054670
57     054230 001406

```

```

ERRTYP:
    SAVREG
    BIT #SW13,@SWR ;INHIBIT TYPEOUTS??
    BEQ 1$ ;NO!!
    JMP 25$ ;YES!!
;TYPE UNIT NUMBER, TEST NUMBER, ERROR NUMBER, AND PROGRAM COUNTER
1$: TYPE ,SCLF
    TYPE ,ERTY00 ;TYPE 'UNTA#'
    MOV $UNIT,-(SP) ;:SAVE $UNIT FOR TYPEOUT
    TYPOS ;:TYPE UNIT NUMBER
    .BYTE 3 ;:GO TYPE--OCTAL ASCII
    .BYTE 0 ;:TYPE 3 DIGIT(S)
;TYPE 'DRIVE TYPE' RM05, RM03 OR RM02 FOR UNIT UNDER TEST
    MOV RMDT(R0),R0 ;:STORE RMDT AT R0
    BIC #177740,R0 ;:SAVE DRIVE TYPE BITS AND
    MOV #$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
    TYPOS ;:TYPE TEST NUMBER
    .BYTE 3 ;:GO TYPE--OCTAL ASCII
    .BYTE 0 ;:TYPE 3 DIGIT(S)
    CLR ERRNMB ;:SUPPRESS LEADING ZEROS
    MOV $ITEMB,ERRNMB ;:LOAD ERROR NUMBER FOR
    BEQ 5$ ;:TYPEOUT
    ;SKIP IF NO ERROR CALLED
5$

```


50	054232	104401	054714		TYPE	,ERTY02		:TYPE 'ERR#'
51	054236	013746	054670		MOV	ERRNMB,-(SP)		::SAVE ERRNMB FOR TYPEOUT
	054242	104403			TYPOS			::TYPE ERROR NUMBER
	054244	003			.BYTE	3		::GO TYPE--OCTAL ASCII
	054245	000			.BYTE	0		::TYPE 3 DIGIT(S)
52	054246	104401	054723	5\$:	TYPE	,ERTY03		::SUPPRESS LEADING ZEROS
53	054252	013746	001132		MOV	\$ERRPC,-(SP)		:TYPE 'PC='
	054256	104403			TYPOS			::SAVE \$ERRPC FOR TYPEOUT
	054260	006			.BYTE	6		::TYPE PROGRAM COUNTER
	054261	001			.BYTE	1		::GO TYPE--OCTAL ASCII
54								::TYPE 6 DIGIT(S)
55	054262	005737	054670	6\$:	:GENERATE POINTER TO ERROR TABLE			:TYPE LEADING ZEROS
56	054266	001575			TST	ERRNMB		UNLESS ERROR NUMBER IS 0
57	054270	104401	001217		BEQ	25\$:WAS AN ERROR CALLED?
58	054274	105037	054674		TYPE	,\$CRLF		:NO!!
59	054300	105037	054675		CLRB	BOTFLG		:YES-TYPE CRLF
60	054304	013700	054670		CLRB	CHRCNT		:CLEAR BOT FLAG
61	054310	006300			MOV	ERRNMB,R0		:CLEAR CHARACTER COUNTER
62	054312	006300			ASL	R0		:R0 POINTS TO FIRST OF
63	054314	006300			ASL	R0		:FOUR ENTRIES IN ERROR
64	054316	062700	001526		ASL	R0		:TABLE
65	054322	011001			ADD	#\$ERRTB-8.,R0		
66					MOV	(R0),R1		:R1 POINTS TO ERROR MESSAGE
67	054324	001507						:TABLE
68					BEQ	16\$:BRANCH IF NO ERROR MESSAGE
69	054326	012102		7\$:	:TYPE THE ERROR MESSAGE			
70	054330	001505			MOV	(R1)+,R2		:R2=ADDRESS OF MESSAGE STRING
71	054332	010237	054500		BEQ	16\$:BRANCH IF END OF MESSAGE
72	054336	005037	054672		MOV	R2,15\$:LOAD ADDRESS OF STRING
73	054342	112203			CLR	BOTADR		:CLEAR BOT ADDRESS
74	054344	001454		8\$:	MOVB	(R2)+,R3		:END OF STRING??
75	054346	122703	000015		BEQ	14\$:YES!!
76	054352	001003			CMPB	#CR,R3		:CARRIAGE RETURN??
77	054354	105037	054675		BNE	9\$:NO!!
78	054360	000770			CLRB	CHRCNT		:YES-CLEAR CHAR COUNT
79	054362	122703	000012	9\$:	BR	8\$:GET NEXT CHARACTER
80	054366	001765			CMPB	#LF,R3		:LINE FEED??
81	054370	122703	000011		BEQ	8\$:YES-GET NEXT CHARACTER
82	054374	001007			CMPB	#HT,R3		:HORIZONTAL TAB??
83	054376	105237	054675	10\$:	BNE	11\$:NO!!
84	054402	132737	000007 054675		INCB	CHRCNT		:ADJUST CHARACTER COUNT
85	054410	001372			BITB	#7,CHRCNT		
86	054412	000407			BNE	10\$		
87	054414	105237	054675	11\$:	BR	12\$:INCREMENT CHARACTER COUNT
88	054420	122703	000040		INCB	CHRCNT		:SPACE??
89	054424	001002			CMPB	#',R3		:NO!!
90	054426	010237	054672		BNE	12\$:SAVE ADDRESS OF SPACE
91	054432	122737	000100 054675	12\$:	MOV	R2,BOTADR		:END OF LINE??
92	054440	103340			CMPB	#64.,CHRCNT		:NO!!
93	054442	013704	054672		BHIS	8\$:GET ADDRESS OF LAST SPACE
94	054446	001007			MOV	BOTADR,R4		:BRANCH IF SPACE DETECTED
95	054450	104401	001217		BNE	13\$		
96	054454	105037	054675		TYPE	,\$CRLF		:TYPE CRLF
97	054460	013702	054500		CLRB	CHRCNT		:CLEAR CHARACTER COUNT
98	054464	000726			MOV	15\$,R2		:SET UP R2 FOR TESTING
					BR	8\$		


```

99 054466 105044 13$: CLRB -(R4) ;REPLACE SPACE
100 054470 112737 177777 054674 MOVB #-1,BOTFLG ;SET BOT FLAG
101 054476 104401 14$: TYPE ;TYPE ERROR MESSAGE STRING
102 054500 000000 15$: .WORD ;STRING ADDRESS GOES HERE
103 054502 105737 054674 TSTB BOTFLG ;WAS STRING TRUNCATED??
104 054506 001707 BEQ 7$ ;NO!!
105 054510 104401 001217 TYPE , $CRLF ;YES-TYPE CRLF
106 054514 105037 054674 CLRB BOTFLG ;CLEAR BOT FLAG
107 054520 105037 054675 CLRB CHRCNT ;CLEAR CHARACTER COUNT
108 054524 013702 054672 MOV BOTADR,R2 ;SETUP R2 FOR TESTING
109 054530 010237 054500 MOV R2,15$ ;SETUP 15$ FOR TYPING
110 054534 112742 000040 MOVB #' -(R2) ;RESTORE SPACE
111 054540 105722 TSTB (R2)+ ;RESTORE R2
112 054542 000677 BR 8$ ;TYPE REST OF STRING
113
114 ;TYPE ERROR HEADER AND ERROR DATA
115 054544
116 054544 016001 000002 16$:
117 054550 001444 17$: MOV 2(R0),R1 ;R1 POINTS TO ERROR HEADER TABLE
118 054552 104401 001217 BEQ 25$ ;BRANCH IF NO HEADER
119 054556 016002 000004 TYPE , $CRLF ;(ASSUME NO DATA)
120 054562 016003 000006 MOV 4(R0),R2 ;R2 POINTS TO DATA ADDRESS TABLE
121 054566 012137 054576 18$: MOV 6(R0),R3 ;R3 POINTS TO FORMAT TABLE
122 054572 001433 MOV (R1)+,19$ ;PUT HEADER ADDRESS FOR TYPE
123 BEQ 25$ ;BRANCH IF END OF HEADERS
124 054574 104401 TYPE ;(ASSUME END OF DATA)
125 054576 000000 19$: .WORD 0 ;HEADER ADDRESS GOES HERE
126 054600 104401 001217 TYPE , $CRLF
127 054604 005702 TST R2 ;DATA WITH HEADER??
128 054606 001767 BEQ 18$ ;NO!!
129 054610 012204 MOV (R2)+,R4 ;R4 POINTS TO DATA ADDRESS
130 054612 012305 MOV (R3)+,R5 ;R5 POINTS TO FORMAT
131 054614 105725 20$: TSTB (R5)+ ;WHAT KIND OF DATA??
132 054616 100407 BMI 22$ ;BINARY
133 054620 001403 BEQ 21$ ;OCTAL
134 054622 013446 MOV @ (R4)+,-(SP) ;DECIMAL
135 054624 104405 TYPDS
136 054626 000405 BR 23$
137 054630 013446 21$: MOV @ (R4)+,-(SP)
138 054632 104402 TYPOC
139 054634 000402 BR 23$
140 054636 013446 22$: MOV @ (R4)+,-(SP)
141 054640 104406 TYPBN
142 054642 005714 23$: TST (R4) ;MORE DATA??
143 054644 001403 BEQ 24$ ;NO!!
144 054646 104401 054731 TYPE ,ERTY04 ;YES-TYPE 2 SPACES
145 054652 000760 BR 20$ ;AND CONTINUE
146 054654 104401 001217 24$: TYPE , $CRLF ;TYPE ONE BLANK LINE
147 054660 000742 BR 18$ ;BEFORE NEXT HEADER
148 054662 104415 25$: RESREG
149 054664 000207 RTS PC
150
151 054666 000000 TSTNMB: .WORD 0 ;TEST NUMBER
152 054670 000000 ERRNMB: .WORD 0 ;ERROR NUMBER
153 054672 000000 BOTADR: .WORD 0 ;BEGINNING OF TEXT ADDRESS
154 054674 000 BOTFLG: .BYTE 0 ;BOT FLAG
155 054675 000 CHRCNT: .BYTE 0 ;CHARACTER COUNT
  
```


156						
157	054676	125	116	111	ERTY00: .ASCIZ	@UNIT#@
158	054704	054	040	124	ERTY01: .ASCIZ	@, TEST#@
159	054714	054	040	105	ERTY02: .ASCIZ	@, ERR#@
160	054723	054	040	120	ERTY03: .ASCIZ	@, PC=@
161	054731	040	040	000	ERTY04: .ASCIZ	@ @
162	054734	040	055	040	ERTY05: .ASCIZ	@ - @
163						
164	054740	122	115	060	\$RM02: .ASCIZ	@RM02@
165	054745	122	115	060	\$RM03: .ASCIZ	@RM03@
166	054752	122	115	060	\$RM05: .ASCIZ	@RM05@
167					.EVEN	
168						


```

1          .SBTTL  CLOCK SUBROUTINES
2
3          ;ROUTINE TO SIZE FOR CLOCKS (KW11-L OR KW11-P)
4
5 054760 000240          SIZCLK: NOP
6 054762 013746 000004  MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
7 054766 013746 000006  MOV      ERRVEC+2,-(SP)    ;;PUSH ERRVEC+2 ON STACK
8 054772 012737 055056 000004  MOV      #10$,ERRVEC      ;LOAD 04 TRAP VECTORS
9 055000 012737 000300 000006  MOV      #PR6,ERRVEC+2
10
11          ;SEE IF A KW11-P CLOCK IS PRESENT - GO TO 10$ IF NOT PRESENT
12 055006 005777 124474  TST      @$LPCSR          ;TEST FOR P CLOCK
13 055012 012737 055220 001532  MOV      #PCLOCK,CLOCK    ;LOAD SUBROUTINE ADDRESS
14 055020 012737 055342 001534  MOV      #PSTOP,STOP      ;LOAD STOP ADDRESS
15 055026 012777 055306 124456  MOV      #PCOUNT,@$LPVEC  ;LOAD P CLOCK INTERRUPT VECTOR
16 055034 012777 000300 124452  MOV      #PR6,@$LPVEC+2
17 055042 013777 001522 124450  MOV      $LLVEC+2,$$LLVEC ;CLEAR L CLOCK INTERRUPT VECTOR
18 055050 005077 124446  CLR      @$LLVEC+2
19 055054 000454  BR       30$
20 055056 012716 055064 10$:  MOV      #15$, (SP)      ;DUMMY RTI ADDRESS
21 055062 000002  RTI                      ;RESTORE PRIORITY
22
23          ;NO P CLOCK-SEE IF L CLOCK IS PRESENT-GO TO 20$ IF NOT PRESET
24 055064 15$:
25 055064 012737 055142 000004  MOV      #20$,ERRVEC      ;CHANGE 04 TRAP VECTOR
26 055072 005777 124420  TST      @$LLCSR          ;TEST FOR L CLOCK
27 055076 012737 055236 001532  MOV      #LCLOCK,CLOCK    ;LOAD SUBROUTINE ADDRESS
28 055104 012737 055350 001534  MOV      #LSTOP,STOP      ;LOAD STOP ADDRESS
29 055112 012777 055306 124400  MOV      #LCOUNT,@$LLVEC  ;LOAD L CLOCK INTERRUPT VECTOR
30 055120 012777 000300 124374  MOV      #PR6,@$LLVEC+2
31 055126 013777 001514 124356  MOV      $LPVEC+2,$$LPVEC;CLEAR P CLOCK INTERRUPT VECTOR
32 055134 005077 124354  CLR      @$LPVEC+2
33 055140 000422  BR       30$
34 055142 012716 055150 20$:  MOV      #25$, (SP)      ;DUMMY RTI ADDRESS
35 055146 000002  RTI                      ;RESTORE PRIORITY
36
37          ;NO CLOCK AVAILABLE - AUGMENT RETURN ADDRESS
38 055150 25$:
39 055150 005037 001532  CLR      CLOCK            ;CLEAR SUBROUTINE ADDRESS
40 055154 012737 001514 001512  MOV      #$LPVEC+2,$$LPVEC;CLEAR P CLOCK INTERRUPT VECTOR
41 055162 005037 001514  CLR      $LPVEC+2
42 055166 012737 001522 001520  MOV      #$LLVEC+2,$$LLVEC;CLEAR L CLOCK INTERRUPT VECTOR
43 055174 005037 001522  CLR      $LLVEC+2
44 055200 062766 000002 000004  ADD      #2,4(SP)         ;CHANGE RETURN ADDRESS
45 055206 30$:
46 055206 012637 000006  MOV      (SP)+,ERRVEC+2    ;;POP STACK INTO ERRVEC+2
47 055212 012637 000004  MOV      (SP)+,ERRVEC     ;;POP STACK INTO ERRVEC
48 055216 000207  RTS      PC
49
50          ;ROUTINES TO START THE CLOCK (KW11-L OR KW11-P)
51 055220 012777 177777 124262 PCLOCK: MOV      #-1,@$LPCSB    ;LOAD COUNT SET BUFFER
52 055226 012777 000135 124252  MOV      #135,@$LPCSR     ;LOAD CONTROL REGISTER
53 055234 000403  BR       PLCLK          ;GO TO COMMON CODE
54
55 055236 012777 000100 124252 LCLOCK: MOV      #100,@$LLCSR  ;LOAD CONTROL REGISTER
56

```

```

57 055244 005037 001526      PLCLK: CLR      TIME      ;CLEAR TIMER COUNT
58 055250 104400              TRAP                    ;:PUSH OLD PSW AND PC ON STACK
   055252 012605              MOV      (SP)+,R5      ;:SAVE THE PSW IN R5
59 055254 010537 001524      MOV      R5,$PSW      ;SAVE PRIORITY
60 055260 042705 177437      BIC      #^CPR7,R5    ;MASK X
61 055264 022705 000300      CMP      #PR6,R5      ;IS PRIORITY TOO HIGH??
62 055270 101005              BHI      40$           ;NO!!
63 055272 012746 000240      MOV      #PR5,-(SP)   ;:PUT NEW PS ON STACK
   055276 012746 055304      MOV      #30$,-(SP)  ;:PUT NEW PC ON STACK
   055302 000002              RTI                    ;:POP NEW PC AND PS
   055304
64 055304 000207      30$:
   40$:      RTS      PC
65
66      ;ROUTINES TO HANDLE CLOCK INTERRUPTS (KW11-L OR KW11-P)
67
68 055306      PCOUNT:
69 055306      LCOUNT:
70 055306 062737 000021 001526      ADD      #17.,TIME    ;ADD 17MS TO ELAPSED TIME
71 055314 103003              BCC      10$          ;BRANCH IF NO OVERFLOW
72 055316 012737 177777 001526      MOV      #-1.,TIME    ;RESTORE MAXIMUM COUNT
73 055324 162737 000021 001530      10$:    SUB      #17.,WATCH ;DECREMENT REMAINING TIME
74 055332 100002              BPL      20$          ;BRANCH IF POSITIVE
75 055334 005037 001530      CLR      WATCH        ;CLEAR REMAINING TIME
76 055340 000002      20$:    RTI                    ;RETURN TO USER
77
78      ;ROUTINES TO STOP THE CLOCK (KW11-L OR KW11-P)
79
80 055342 005077 124140      PSTOP:  CLR      @SLPCSR ;STOP P CLOCK
81 055346 000402              BR      PLSTP         ;GO TO COMMON STOP CODE
82
83 055350 005077 124142      LSTOP:  CLR      @$LLCSR ;STOP L CLOCK
84
85 055354      PLSTP:
86 055354 013746 001524      MOV      $PSW,-(SP)  ;:PUT NEW PS ON STACK
   055360 012746 055366      MOV      #10$,-(SP) ;:PUT NEW PC ON STACK
   055364 000002              RTI                    ;:POP NEW PC AND PS
   055366
87 055366 000207      10$:
88              RTS      PC

```



```
1          .SBTTL SET VOLUME VALID SUBROUTINE
2
3          ;THIS SUBROUTINE INITIALIZES THE SUBSYSTEM AND SETS VOLUME VALID,
4          ;RETURNING WITH THE DRIVE STILL IN DIAGNOSTIC MODE. THE SUBROUTINE
5          ;RETURNS TO THE WORD FOLLOWING THE CALL, EXCEPT WHEN AN ERROR IS
6          ;DETECTED, IN WHICH CASE IT RETURNS TO THE SECOND WORD FOLLOWING THE
7          ;CALL.
8
9          ;CALL: JSR      PC,SETVV      JUMP TO SUBROUTINE
10         BR      ??                  RETURN HERE IF NO ERROR
11         :      ERROR                RETURN HERE IF ERROR
12
13         SETVV:
14         JSR      PC,CNTCLR          ;GO CLEAR CONTROLLER
15         MOV      #DMD,RMMR1(R0)    ;LOAD RMMR1
16         MOV      #DMD!MUR,RMMR1(R0);LOAD RMMR1
17         MOV      #0,RMER1(R0)     ;LOAD RMER1
18         MOV      #0,RMER2(R0)     ;LOAD RMER2
19         MOV      #PACACK!GO, RMCS1(R0);LOAD RMCS1
20         MOV      RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT
21         BIC      #^CVV,$BDDAT
22         BNE      10$              ;BRANCH IF VOLUME VALID SET
23         MOV      R0,$BDADR         ;SETUP FOR ERROR MSG
24         ADD      #RMDS,$BDADR
25         MOV      #VV,$GDDAT
26         ADD      #2,(SP)          ;MOVE RETURN ADDRESS TO ERROR
27         MOVB    #170,@(SP)       ;WRITE ERROR NUMBER
28         MOV      #PACACK,$TMPO
29         RTS      PC              ;RETURN
30
```

```
1      .SBTTL SET OFFSET MODE SUBROUTINE
2
3      :THIS SUBROUTINE EXECUTES AN OFFSET COMMAND AND VERIFIES THAT OFFSET
4      :MODE SETS. THE DRIVE SHOULD BE IN DIAGNOSTIC MODE WHEN CALLING THE
5      :SUBROUTINE, WHICH WILL LEAVE DMD ON. THE SUBROUTINE RETURNS TO THE
6      :WORD FOLLOWING THE CALL UNLESS THERE IS AN ERROR, IN WHICH CASE IT
7      :RETURNS TO THE SECOND WORD FOLLOWING THE CALL
8
9      :CALL: JSR    PC,SETOM      JUMP TO SUBROUTINE
10     :      BR     ??           RETURN HERE IF NO ERROR
11     :      ERROR          RETURN HERE IF ERROR
12
13     055512      SETOM:
14     055512      012760 001001 000024      MOV    #DMD!MUR,RMMR1(R0)      ;LOAD RMMR1
15     055520      012760 000000 000014      MOV    #0,RMER1(R0)           ;LOAD RMER1
16     055526      012760 000000 000042      MOV    #0,RMER2(R0)           ;LOAD RMER2
17     055534      012760 000015 000000      MOV    #OFFSET!GO, RMCS1(R0)   ;LOAD RMCS1
18     055542      016037 000012 001142      MOV    RMDS(R0), $BDDAT        ;STORE RMDS AT $BDDAT
19     055550      042737 177776 001142      BIC    #^COM,$BDDAT
20     055556      001015                      BNE    10$                    ;BRANCH IF OFFSET ON
21     055560      012737 000001 001140      MOV    #OM,$GDDAT
22     055566      010037 001136                      MOV    R0,$BDADR
23     055572      062737 000012 001136      ADD    #RMDS,$BDADR
24     055600      062716 000002                      ADD    #2,(SP)                ;MOVE RETURN ADDRESS TO ERROR
25     055604      112776 000200 000000      MOVB   #200,@(SP)            ;WRITE ERROR NUMBER
26     055612
27     055612      000207
28
```



```

1      .SBTTL CLEAR CONTROLLER SUBROUTINE
2
3      ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTERS,
4      ;AND DRIVES, THEN SELECTS THE DRIVE.
5      ;CALL:
6      ;      JSR      PC,CNTCLR      ;CALL TO ROUTINE
7
8      CNTCLR:
9      055614      010046      001276      000010      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
10     055616      013700      000040      000010      MOV      $BASE,R0      ;R0 = UNIBUS BASE ADDRESS
11     055622      012760      000040      000010      MOV      #CLR, RMCS2(R0) ;CLEAR MASSBUS
12     055630      117760      123626      000010      MOV      @TSTQUE, RMCS2(R0) ;SELECT DEVICE UNDER TEST
13     055636      012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
14     055640      000207      RTS      PC      ;RETURN
15
16     .SBTTL SHUTDOWN SUBROUTINE
17     SHUT:
18     055642      104401      055650      TYPE      ,65$      ;;TYPE ASCIZ STRING
19     055646      000411      BR      64$      ;;GET OVER THE ASCIZ
20     055672      005737      000042      ;;65$: .ASCIZ <CRLF>@TEST WAS HALTED@<CRLF>
21     055676      001402      TST      @#42      ;RUNNING CHAIN MODE OR ACT11 AUTO ACCEPT ?
22     055700      000137      053636      BEQ      10$      ;NO !!
23     055704      000137      004626      JMP      $EOP      ;YES - GO TO END OF PASS
24     10$:      JMP      START      ;GO TO START
  
```

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
    
```

```

055710
055710 010046
055712 010146
055714 010246
055716 010346
055720 010446
055722 010546
055724 016646 000022
055730 016646 000022
055734 016646 000022
055740 016646 000022
055744 000002
    
```

```

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

*RESTORE R0-R5

```

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

```

055746
055746 012666 000022
055752 012666 000022
055756 012666 000022
055762 012666 000022
055766 012605
055770 012604
055772 012603
055774 012602
055776 012601
056000 012600
056002 000002
    
```

2

.SBTTL BINARY TO ASCII AND TYPE ROUTINE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
*BINARY-ASCII NUMBER AND TYPE IT.
*CALL:
*   MOV NUMBER,-(SP)  ;;NUMBER TO BE TYPED
*   TYPBN              ;;TYPE IT
    
```

```

056004 010146
056006 016601 000006
056012 000261
    
```

```

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



```

056014 112737 000060 056056 1$:   MOVB   #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
056022 006101                ROL    R1            ;;GET THIS BIT
056024 001406                BEQ    2$           ;;DONE?
056026 105537 056056        ADCB   $BIN          ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
056032 104401 056056        TYPE   , $BIN       ;;GO TYPE THIS BIT
056036 000241                CLC                    ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
056040 000765                BR     1$           ;;GO DO THE NEXT BIT
056042 012601                MOV    (SP)+,R1      ;;POP THE STACK INTO R1
056044 016666 000002 000004 2$:   MOV    2(SP),4(SP)   ;;ADJUST THE STACK
056052 012616                MOV    (SP)+,(SP)
056054 000002                RTI                    ;;RETURN TO USER
056056 000      000        $BIN:  .BYTE  0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
3      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

```

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

```

```

*   MOV    NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
*   TYPDS                    ;;GO TO THE ROUTINE

```

```

056060 010046                $TYPDS:  MOV    R0,-(SP)      ;;PUSH R0 ON STACK
056062 010146                MOV    R1,-(SP)      ;;PUSH R1 ON STACK
056064 010246                MOV    R2,-(SP)      ;;PUSH R2 ON STACK
056066 010346                MOV    R3,-(SP)      ;;PUSH R3 ON STACK
056070 010546                MOV    R5,-(SP)      ;;PUSH R5 ON STACK
056072 012746 020200        MOV    #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
056076 016605 000020        MOV    20(SP),R5     ;;GET THE INPUT NUMBER
056102 100004                BPL    1$           ;;BR IF INPUT IS POS.
056104 005405                NEG    R5            ;;MAKE THE BINARY NUMBER POS.
056106 112766 000055 000001 1$:   MOVB   #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
056114 005000                CLR    R0            ;;ZERO THE CONSTANTS INDEX
056116 012703 056274        MOV    #$DBLK,R3     ;;SETUP THE OUTPUT POINTER
056122 112723 000040        MOVB   #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
056126 005002                CLR    R2            ;;CLEAR THE BCD NUMBER
056130 016001 056264        MOV    $DTBL(R0),R1  ;;GET THE CONSTANT
056134 160105                3$:   SUB    R1,R5         ;;FORM THIS BCD DIGIT
056136 002402                BLT    4$           ;;BR IF DONE
056140 005202                INC    R2            ;;INCREASE THE BCD DIGIT BY 1
056142 000774                BR     3$
056144 060105                4$:   ADD    R1,R5         ;;ADD BACK THE CONSTANT
056146 005702                TST    R2            ;;CHECK IF BCD DIGIT=0
056150 001002                BNE    5$           ;;FALL THROUGH IF 0
056152 105716                TSTB   (SP)          ;;STILL DOING LEADING 0'S?
056154 100407                BMI    7$           ;;BR IF YES
056156 106316                5$:   ASLB   (SP)          ;;MSD?
056160 103003                BCC    6$           ;;BR IF NO
056162 116663 000001 177777 6$:   MOVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
056170 052702 000060        BIS    #'0,R2        ;;MAKE THE BCD DIGIT ASCII
056174 052702 000040        7$:   BIS    #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
056200 110223                MOVB   R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
056202 005720                TST    (R0)+        ;;JUST INCREMENTING
056204 020027 000010        CMP    R0,#10       ;;CHECK THE TABLE INDEX

```

```

056210 002746          BLT      2$          ;;GO DO THE NEXT DIGIT
056212 003002          BGT      8$          ;;GO TO EXIT
056214 010502          MOV      R5,R2        ;;GET THE LSD
056216 000764          BR       6$          ;;GO CHANGE TO ASCII
056220 105726          8$:    TSTB   (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
056222 100003          BPL     9$          ;;BR IF NO
056224 116663 177777 177776  MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
056232 105013          9$:    CLRB   (R3)          ;;SET THE TERMINATOR
056234 012605          MOV     (SP)+,R5        ;;POP STACK INTO R5
056236 012603          MOV     (SP)+,R3        ;;POP STACK INTO R3
056240 012602          MOV     (SP)+,R2        ;;POP STACK INTO R2
056242 012601          MOV     (SP)+,R1        ;;POP STACK INTO R1
056244 012600          MOV     (SP)+,R0        ;;POP STACK INTO R0
056246 104401 056274  TYPE    $DBLK        ;;NOW TYPE THE NUMBER
056252 016666 000002 000004  MOV     2(SP),4(SP)    ;;ADJUST THE STACK
056260 012616          MOV     (SP)+,(SP)
056262 000002          RTI                    ;;RETURN TO USER
056264 023420          $DTBL: 10000.
056266 001750          1000.
056270 000144          100.
056272 000012          10.
056274          $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
    
```

```

056304 017646 000000          $TYPOS: MOV    @ (SP),-(SP)    ;;PICKUP THE MODE
056310 116637 000001 056527  MOVB   1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
056316 112637 056531          MOVB   (SP)+,$SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
056322 062716 000002          ADD    #2,(SP)      ;;ADJUST RETURN ADDRESS
056326 000406          BR     $TYPON
056330 112737 000001 056527  $TYPOC: MOVB   #1,$OFILL ;;SET THE ZERO FILL SWITCH
056336 112737 000006 056531  MOVB   #6,$SOMODE+1 ;;SET FOR SIX(6) DIGITS
056344 112737 000005 056526  $TYPON: MOVB   #5,$SOCNT ;;SET THE ITERATION COUNT
056352 010346          MOV    R3,-(SP)     ;;SAVE R3
056354 010446          MOV    R4,-(SP)     ;;SAVE R4
    
```

4


```

056356 010546          MOV      R5,-(SP)      ;;SAVE R5
056360 113704 056531  MOVVB   $OMODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
056364 005404          NEG      R4
056366 062704 000006  ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
056372 110437 056530  MOVVB   R4,$OMODE     ;;SAVE IT FOR USE
056376 113704 056527  MOVVB   $OFILL,R4     ;;GET THE ZERO FILL SWITCH
056402 016605 000012  MOV     12(SP),R5     ;;PICKUP THE INPUT NUMBER
056406 005003          CLR     R3           ;;CLEAR THE OUTPUT WORD
056410 006105          1$:    ROL     R5           ;;ROTATE MSB INTO 'C'
056412 000404          BR     3$           ;;GO DO MSB
056414 006105          2$:    ROL     R5           ;;FORM THIS DIGIT
056416 006105          ROL     R5
056420 006105          ROL     R5
056422 010503          MOV     R5,R3
056424 006103          3$:    ROL     R3           ;;GET LSB OF THIS DIGIT
056426 105337 056530  DECB   $OMODE         ;;TYPE THIS DIGIT?
056432 100016          BPL    7$           ;;BR IF NO
056434 042703 177770  BIC    #177770,R3     ;;GET RID OF JUNK
056440 001002          BNE    4$           ;;TEST FOR 0
056442 005704          TST   R4            ;;SUPPRESS THIS 0?
056444 001403          BEQ   5$           ;;BR IF YES
056446 005204          4$:    INC    R4           ;;DON'T SUPPRESS ANYMORE 0'S
056450 052703 000060  BIS    #'0,R3        ;;MAKE THIS DIGIT ASCII
056454 052703 000040  5$:    BIS    #' ,R3     ;;MAKE ASCII IF NOT ALREADY
056460 110337 056524  MOVVB   R3,8$        ;;SAVE FOR TYPING
056464 104401 056524  TYPE   ,8$          ;;GO TYPE THIS DIGIT
056470 105337 056526  7$:    DECB   $OCNT     ;;COUNT BY 1
056474 003347          BGT   2$           ;;BR IF MORE TO DO
056476 002402          BLT   6$           ;;BR IF DONE
056500 005204          INC   R4           ;;INSURE LAST DIGIT ISN'T A BLANK
056502 000744          BR    2$           ;;GO DO THE LAST DIGIT
056504 012605          6$:    MOV    (SP)+,R5     ;;RESTORE R5
056506 012604          MOV    (SP)+,R4     ;;RESTORE R4
056510 012603          MOV    (SP)+,R3     ;;RESTORE R3
056512 016666 000002 000004  MOV    2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
056520 012616          MOV    (SP)+,(SP)
056522 000002          RTI
056524 000          8$:    .BYTE  0           ;;RETURN
056525 000          .BYTE  0           ;;STORAGE FOR ASCII DIGIT
056526 000          $OCNT: .BYTE  0     ;;TERMINATOR FOR TYPE ROUTINE
056527 000          $OFILL: .BYTE  0    ;;OCTAL DIGIT COUNTER
056530 000000          $OMODE: .WORD  0    ;;ZERO FILL SWITCH
                    .SBTTL TYPE ROUTINE  ;;NUMBER OF DIGITS TO TYPE

```

5

```

*****
;*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

```



```

;*
056532 105737 001173 $TYPE: TSTB $TPFLG ::IS THERE A TERMINAL?
056536 100002 BPL 1$ ::BR IF YES
056540 000000 HALT ::HALT HERE IF NO TERMINAL
056542 000430 BR 3$ ::LEAVE
056544 010046 1$: MOV R0,-(SP) ::SAVE R0
056546 017600 000002 MOV @2(SP),R0 ::GET ADDRESS OF ASCIZ STRING
056552 122737 000001 001242 CMPB #APTENV,$ENV ::RUNNING IN APT MODE
056560 001011 BNE 62$ ::NO,GO CHECK FOR APT CONSOLE
056562 132737 000100 001243 BITB #APTSPOOL,$ENVM ::SPOOL MESSAGE TO APT
056570 001405 BEQ 62$ ::NO,GO CHECK FOR CONSOLE
056572 010037 056602 MOV R0,61$ ::SETUP MESSAGE ADDRESS FOR APT
056576 004737 061672 JSR PC,$ATY3 ::SPOOL MESSAGE TO APT
056602 000000 61$: .WORD 0 ::MESSAGE ADDRESS
056604 132737 000040 001243 62$: BITB #APTCSUP,$ENVM ::APT CONSOLE SUPPRESSED
056612 001003 BNE 60$ ::YES,SKIP TYPE OUT
056614 112046 2$: MOV (R0)+,-(SP) ::PUSH CHARACTER TO BE TYPED ONTO STACK
056616 001005 BNE 4$ ::BR IF IT ISN'T THE TERMINATOR
056620 005726 TST (SP)+ ::IF TERMINATOR POP IT OFF THE STACK
056622 012600 60$: MOV (SP)+,R0 ::RESTORE R0
056624 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
056630 000002 RTI ::RETURN
056632 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
056636 001430 BEQ 8$
056640 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
056644 001006 BNE 5$
056646 005726 TST (SP)+ ::POP <CR><LF> EQUIV
056650 104401 TYPE ::TYPE A CR AND LF
056652 001217 $CRLF
056654 105037 057062 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
056660 000755 BR 2$ ::GET NEXT CHARACTER
056662 004737 056744 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
056666 123726 001172 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
056672 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
056674 013746 001170 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
::AND THE NULL CHAR.
056700 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
056704 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
056706 004737 056744 JSR PC,$TYPEC ::GO TYPE A NULL
056712 105337 057062 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
056716 000770 BR 7$ ::LOOP

```

:HORIZONTAL TAB PROCESSOR

```

056720 112716 000040 8$: MOV #'(SP) ::REPLACE TAB WITH SPACE
056724 004737 056744 9$: JSR PC,$TYPEC ::TYPE A SPACE
056730 132737 000007 057062 BITB #7,$CHARCNT ::BRANCH IF NOT AT
056736 001372 BNE 9$ ::TAB STOP
056740 005726 TST (SP)+ ::POP SPACE OFF STACK
056742 000724 BR 2$ ::GET NEXT CHARACTER
056744 $TYPEC:
056744 105777 122210 TSTB @STKS ::CHAR IN KYBD BUFFER?
056750 100022 BPL 10$ ::BR IF NOT
056752 017746 122204 MOV @STKB,-(SP) ::GET CHAR
056756 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS
056762 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF

```



```

056766 001012
056770
056770 105777 122164
056774 100375
056776 117716 122160
057002 042716 177600
057006 122716 000021
057012 001366
057014
057014 005726
057016
057016 105777 122142
057022 100375
057024 116677 000002 122134
057032 122766 000015 000002
057040 001003
057042 105037 057062
057046 000406
057050 122766 000012 000002
057056 001402
057060 105227
057062 000000
057064 000207

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

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

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

6

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($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
    
```

```

057066
057066 104410
057070 032777 040000 122056
057076 001131

$SCOPE:
1$: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
BNE $OVER ;;YES IF SW14=1
:#####START OF CODE FOR THE XOR TESTER#####
$XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
THIS INSTRUCTION TO A 'NOP' (NOP=240)
057100 000416 MOV @ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
057102 013746 000004 MOV #5$,@ERRVEC ;;SET FOR TIMEOUT
057106 012737 057126 000004 TST @#177060 ;;TIME OUT ON XOR?
057114 005737 177060 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
057120 012637 000004 BR $SVLAD ;;GO TO THE NEXT TEST
057124 000500 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
057126 022626 MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR
057130 012637 000004 BR 7$ ;;LOOP ON THE PRESENT TEST
057134 000440
057136 6$:;#####END OF CODE FOR THE XOR TESTER#####
057136 032777 000400 122010 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
057144 001421 BEQ 2$ ;;BR IF NO
057146 005046 CLR -(SP) ;;CLEAR A TEMP. LOCATION
    
```


SCOPE HANDLER ROUTINE

```

057150 117716 122000      MOVB    @SWR,(SP)      ;;PICKUP THE DESIRED TEST NUMBER
057154 001414            BEQ     8$            ;;BRANCH IF BAD TEST NUMBER IN SWR
057156 022716 000120      CMP     #120,(SP)    ;;CHECK THE NUMBER IN THE SWR
057162 002411            BLT     8$            ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
057164 011637 001116      MOV     (SP), $STSTNM ;;UPDATE THE TEST NUMBER
057170 005316            DEC     (SP)         ;;BACKUP BY ONE
057172 006316            ASL     (SP)         ;;SCALE THE TEST NUMBER AS AN INDEX
057174 062716 057400      ADD     #$$SW08TBL,(SP) ;;FORM THE ADDRESS OF TEST POINTER
057200 013637 001122      MOV     @($SP)+,$LPADR ;;SET LOOP ADDRESS TO DESIRED TEST
057204 000466            BR      $OVER        ;;GO LOOP ON THE TEST
057206 005726            8$: TST     ($SP)+      ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
057210 105737 001117      2$: TSTB    $ERFLG    ;;HAS AN ERROR OCCURRED?
057214 001421            BEQ     3$            ;;BR IF NO
057216 123737 001131 001117  CMPB    $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
057224 101015            BHI     3$            ;;BR IF NO
057226 032777 001000 121720  BIT     #BIT09,@SWR  ;;LOOP ON ERROR?
057234 001404            BEQ     4$            ;;BR IF NO
057236 013737 001124 001122 7$: MOV     $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
057244 000446            BR      $OVER
057246 105037 001117      4$: CLRB    $ERFLG    ;;ZERO THE ERROR FLAG
057252 005037 001206      CLR     $TIMES       ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
057256 000415            BR      1$            ;;ESCAPE TO THE NEXT TEST
057260 032777 004000 121666 3$: BIT     #BIT11,@SWR  ;;INHIBIT ITERATIONS?
057266 001011            BNE     1$            ;;BR IF YES
057270 005737 001230      TST     $PASS        ;;IF FIRST PASS OF PROGRAM
057274 001406            BEQ     1$            ;;INHIBIT ITERATIONS
057276 005237 001120      INC     $ICNT        ;;INCREMENT ITERATION COUNT
057302 023737 001206 001120  CMP     $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
057310 002024            BGE     $OVER        ;;BR IF MORE ITERATION REQUIRED
057312 012737 000001 001120 1$: MOV     #1,$ICNT   ;;REINITIALIZE THE ITERATION COUNTER
057320 013737 057376 001206  MOV     $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
057326 105237 001116      $SVLAD: INCB   $STSTNM ;;COUNT TEST NUMBERS
057332 113737 001116 001226  MOVB    $STSTNM,$STSTNM ;;SET TEST NUMBER IN APT MAILBOX
057340 011637 001122      MOV     (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
057344 011637 001124      MOV     (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
057350 005037 001210      CLR     $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
057354 112737 000001 001131  MOVB    #1,$ERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
057362 013777 001116 121566 $OVER: MOV     $STSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
057370 013716 001122      MOV     $LPADR,(SP) ;;FUDGE RETURN ADDRESS
057374 000002            RTI                ;;FIXES PS
057376 000012      $MXCNT: 10.       ;;MAX. NUMBER OF ITERATIONS
057400      $SW08TBL:
057400 007310      .WORD   TST1+2      ;;STARTING ADDRESS OF TEST 1
057402 007620      .WORD   TST2+2      ;;STARTING ADDRESS OF TEST 2
057404 010002      .WORD   TST3+2      ;;STARTING ADDRESS OF TEST 3
057406 010152      .WORD   TST4+2      ;;STARTING ADDRESS OF TEST 4
057410 010302      .WORD   TST5+2      ;;STARTING ADDRESS OF TEST 5
057412 011366      .WORD   TST6+2      ;;STARTING ADDRESS OF TEST 6
057414 012436      .WORD   TST7+2      ;;STARTING ADDRESS OF TEST 7
057416 012574      .WORD   TST10+2     ;;STARTING ADDRESS OF TEST 10
057420 012660      .WORD   TST11+2     ;;STARTING ADDRESS OF TEST 11
057422 013120      .WORD   TST12+2     ;;STARTING ADDRESS OF TEST 12
057424 013466      .WORD   TST13+2     ;;STARTING ADDRESS OF TEST 13
057426 014250      .WORD   TST14+2     ;;STARTING ADDRESS OF TEST 14
057430 014350      .WORD   TST15+2     ;;STARTING ADDRESS OF TEST 15
057432 014674      .WORD   TST16+2     ;;STARTING ADDRESS OF TEST 16
057434 015406      .WORD   TST17+2     ;;STARTING ADDRESS OF TEST 17

```


057436	015510	.WORD	TST20+2	:: STARTING ADDRESS OF TEST	20
057440	015774	.WORD	TST21+2	:: STARTING ADDRESS OF TEST	21
057442	016112	.WORD	TST22+2	:: STARTING ADDRESS OF TEST	22
057444	016240	.WORD	TST23+2	:: STARTING ADDRESS OF TEST	23
057446	016512	.WORD	TST24+2	:: STARTING ADDRESS OF TEST	24
057450	017012	.WORD	TST25+2	:: STARTING ADDRESS OF TEST	25
057452	017434	.WORD	TST26+2	:: STARTING ADDRESS OF TEST	26
057454	017530	.WORD	TST27+2	:: STARTING ADDRESS OF TEST	27
057456	020016	.WORD	TST30+2	:: STARTING ADDRESS OF TEST	30
057460	020332	.WORD	TST31+2	:: STARTING ADDRESS OF TEST	31
057462	020622	.WORD	TST32+2	:: STARTING ADDRESS OF TEST	32
057464	021340	.WORD	TST33+2	:: STARTING ADDRESS OF TEST	33
057466	021652	.WORD	TST34+2	:: STARTING ADDRESS OF TEST	34
057470	022144	.WORD	TST35+2	:: STARTING ADDRESS OF TEST	35
057472	022466	.WORD	TST36+2	:: STARTING ADDRESS OF TEST	36
057474	023026	.WORD	TST37+2	:: STARTING ADDRESS OF TEST	37
057476	023502	.WORD	TST40+2	:: STARTING ADDRESS OF TEST	40
057500	023766	.WORD	TST41+2	:: STARTING ADDRESS OF TEST	41
057502	024250	.WORD	TST42+2	:: STARTING ADDRESS OF TEST	42
057504	024652	.WORD	TST43+2	:: STARTING ADDRESS OF TEST	43
057506	025022	.WORD	TST44+2	:: STARTING ADDRESS OF TEST	44
057510	025402	.WORD	TST45+2	:: STARTING ADDRESS OF TEST	45
057512	026112	.WORD	TST46+2	:: STARTING ADDRESS OF TEST	46
057514	026406	.WORD	TST47+2	:: STARTING ADDRESS OF TEST	47
057516	027010	.WORD	TST50+2	:: STARTING ADDRESS OF TEST	50
057520	027244	.WORD	TST51+2	:: STARTING ADDRESS OF TEST	51
057522	027452	.WORD	TST52+2	:: STARTING ADDRESS OF TEST	52
057524	030304	.WORD	TST53+2	:: STARTING ADDRESS OF TEST	53
057526	030556	.WORD	TST54+2	:: STARTING ADDRESS OF TEST	54
057530	030760	.WORD	TST55+2	:: STARTING ADDRESS OF TEST	55
057532	031172	.WORD	TST56+2	:: STARTING ADDRESS OF TEST	56
057534	031366	.WORD	TST57+2	:: STARTING ADDRESS OF TEST	57
057536	031546	.WORD	TST60+2	:: STARTING ADDRESS OF TEST	60
057540	031764	.WORD	TST61+2	:: STARTING ADDRESS OF TEST	61
057542	032132	.WORD	TST62+2	:: STARTING ADDRESS OF TEST	62
057544	032356	.WORD	TST63+2	:: STARTING ADDRESS OF TEST	63
057546	032472	.WORD	TST64+2	:: STARTING ADDRESS OF TEST	64
057550	032660	.WORD	TST65+2	:: STARTING ADDRESS OF TEST	65
057552	033110	.WORD	TST66+2	:: STARTING ADDRESS OF TEST	66
057554	033326	.WORD	TST67+2	:: STARTING ADDRESS OF TEST	67
057556	033554	.WORD	TST70+2	:: STARTING ADDRESS OF TEST	70
057560	034024	.WORD	TST71+2	:: STARTING ADDRESS OF TEST	71
057562	034232	.WORD	TST72+2	:: STARTING ADDRESS OF TEST	72
057564	034522	.WORD	TST73+2	:: STARTING ADDRESS OF TEST	73
057566	034750	.WORD	TST74+2	:: STARTING ADDRESS OF TEST	74
057570	035056	.WORD	TST75+2	:: STARTING ADDRESS OF TEST	75
057572	035202	.WORD	TST76+2	:: STARTING ADDRESS OF TEST	76
057574	035332	.WORD	TST77+2	:: STARTING ADDRESS OF TEST	77
057576	035464	.WORD	TST100+2	:: STARTING ADDRESS OF TEST	100
057600	035616	.WORD	TST101+2	:: STARTING ADDRESS OF TEST	101
057602	036006	.WORD	TST102+2	:: STARTING ADDRESS OF TEST	102
057604	036210	.WORD	TST103+2	:: STARTING ADDRESS OF TEST	103
057606	036336	.WORD	TST104+2	:: STARTING ADDRESS OF TEST	104
057610	036542	.WORD	TST105+2	:: STARTING ADDRESS OF TEST	105
057612	036672	.WORD	TST106+2	:: STARTING ADDRESS OF TEST	106
057614	037066	.WORD	TST107+2	:: STARTING ADDRESS OF TEST	107
057616	037264	.WORD	TST110+2	:: STARTING ADDRESS OF TEST	110

057620 037542
 057622 040046
 057624 041440
 057626 043166
 057630 045346
 057632 045726
 057634 050106
 057636 052374

```
.WORD TST111+2      ;; STARTING ADDRESS OF TEST 111
.WORD TST112+2      ;; STARTING ADDRESS OF TEST 112
.WORD TST113+2      ;; STARTING ADDRESS OF TEST 113
.WORD TST114+2      ;; STARTING ADDRESS OF TEST 114
.WORD TST115+2      ;; STARTING ADDRESS OF TEST 115
.WORD TST116+2      ;; STARTING ADDRESS OF TEST 116
.WORD TST117+2      ;; STARTING ADDRESS OF TEST 117
.WORD TST120+2      ;; STARTING ADDRESS OF TEST 120
```

.SBTTL ERROR HANDLER ROUTINE

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

057640
 057640 104410
 057642 105237 001117
 057646 001775
 057650 013777 001116 121300
 057656 032777 002000 121270
 057664 001402
 057666 104401 001212
 057672 005237 001126
 057676 011637 001132
 057702 162737 000002 001132
 057710 117737 121216 001130
 057716 032777 020000 121230
 057724 001004
 057726 004737 054046
 057732 104401 0C1217
 057736
 057736 122737 000001 001242
 057744 001007
 057746 113737 001130 057760
 057754 004737 061702
 057760 000
 057761 000
 057762 000777
 057764 005777 121164
 057770 100002
 057772 000000
 057774 104410
 057776 032777 001000 121150
 060004 001402
 060006 013716 001124
 060012 005737 001210
 060016 001402
 060020 013716 001210
 060024

```
$ERROR:
7$:      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
        INCB      $ERFLG      ;;SET THE ERROR FLAG
        BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
        MOV      $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
        BIT      #BIT10,@SWR   ;;BELL ON ERROR?
        BEQ      1$      ;;NO - SKIP
        TYPE     ,SBELL      ;;RING BELL
1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
        MOV      (SP), $ERRPC  ;;GET ADDRESS OF ERROR INSTRUCTION
        SUB      #2, $ERRPC
        MOV      @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
        BIT      #BIT13,@SWR   ;;SKIP TYPEOUT IF SET
        BNE      20$      ;;SKIP TYPEOUTS
        JSR      PC,ERRTP     ;;GO TO USER ERROR ROUTINE
        TYPE     , $CRLF
20$:     CMPB     #APTENV, $ENV  ;;RUNNING IN APT MODE
        BNE      2$      ;;NO, SKIP APT ERROR REPORT
        MOV      $ITEMB, 21$   ;;SET ITEM NUMBER AS ERROR NUMBER
        JSR      PC, $ATY4    ;;REPORT FATAL ERROR TO APT
21$:     .BYTE   0
        .BYTE   0
22$:     BR      22$      ;;APT ERROR LOOP
2$:      TST     @SWR      ;;HALT ON ERROR
        BPL     3$      ;;SKIP IF CONTINUE
        HALT    ;;HALT ON ERROR!
3$:     CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
        BIT     #BIT09,@SWR   ;;LOOP ON ERROR SWITCH SET?
        BEQ     4$      ;;BR IF NO
        MOV     $LPERR, (SP)  ;;FUDGE RETURN FOR LOOPING
4$:     TST     $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
        BEQ     5$      ;;BR IF NONE
5$:     MOV     $ESCAPE, (SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
```


060024 022737 054026 000042
060032 001001
060034 000000
060036
060036 000002
8

CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
BNE 6\$;;BRANCH IF NO
HALT ;;YES
6\$:
RTI ;;RETURN
.SBTTL TTY INPUT ROUTINE

060040 000000
060042 000000
060044 000000
060046 060047

.ENABL LSB
\$TKCNT: .WORD 0 ;;NUMBER OF ITEMS IN QUEUE
\$TKQIN: .WORD 0 ;;INPUT POINTER
\$TKQOUT: .WORD 0 ;;OUTPUT POINTER
\$TKQSRV: .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

060050 005037 060040
060054 012737 060046 060042
060062 013737 060042 060044
060070 012737 060120 000060
060076 012737 000200 000062
060104 005777 121052
060110 012777 000100 121042
060116 000207

\$TKINT: CLR \$TKCNT ;;CLEAR COUNT OF ITEMS IN QUEUE
MOV #TKQSRV,\$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)

060120 117746 121036
060124 042716 177600
060130 021627 000003
060134 001007
060136 104401 061234
060142 004737 060050
060146 005726
060150 000137 055642
060154 021627 000007
060160 001004
060162 022737 000176 001154
060170 001500

\$TKSRV: MCVB @TKB,-(SP) ;;PICKUP THE CHARACTER
BIC #^C177,(SP) ;;STRIP THE JUNK
CMP (SP),#3 ;;IS IT A CONTROL C?
BNE 1\$;;BRANCH IF NO
TYPE ,SCNTLC ;;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

060172
060172 022737 000001 060040
060200 001004
060202 104401 001212

2\$:
CMP #1,\$TKCNT ;;IS THE QUEUE FULL?
BNE 3\$;;BRANCH IF NO
TYPE ,SBELL ;;RING THE TTY BELL

```

060206 005726          TST      (SP)+          ;;CLEAN CHARACTER OFF OF STACK
060210 000451          BR       5$              ;;EXIT
060212 021627 000023  3$:    CMP      (SP),#23          ;;IS IT A CONTROL-S?
060216 001021          BNE      32$              ;;BRANCH IF NO
060220 005077 120734          CLR      @STKS          ;;DISABLE TTY KEYBOARD INTERRUPTS
060224 005726          TST      (SP)+          ;;CLEAN CHAR OFF STACK
060226 105777 120726  31$:  TSTB    @STKS          ;;WAIT FOR A CHAR
060232 100375          BPL      31$              ;;LOOP UNTIL ITS THERE
060234 117746          MOVB    @STKB,-(SP)      ;;GET THE CHARACTER
060240 042716 177600          BIC      #^C177,(SP)    ;;MAKE IT 7-BIT ASCII
060244 022627 000021          CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
060250 001366          BNE      31$              ;;BRANCH IF NO
060252 012777 000100 120700  MOV     #100,@STKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
060260 000002          RTI                    ;;RETURN
060262 005237 060040  32$:  INC     $STKCNT          ;;COUNT THIS CHARACTER
060266 021627 000140          CMP      (SP),#140      ;;IS IT UPPER CASE?
060272 002405          BLT     4$              ;;BRANCH IF YES
060274 021627 000175          CMP      (SP),#175      ;;IS IT A SPECIAL CHAR?
060300 003002          BGT     4$              ;;BRANCH IF YES
060302 042716 000040          BIC      #40,(SP)       ;;MAKE IT UPPER CASE
060306 112677 177530  4$:    MOVB    (SP)+,@STKQIN  ;;AND PUT IT IN QUEUE
060312 005237 060042          INC     $STKQIN         ;;UPDATE THE POINTER
060316 023727 060042 060047  CMP     $STKQIN,$STKQEND ;;GO OFF THE END?
060324 001003          BNE     5$              ;;BRANCH IF NO
060326 012737 060046 060042  MOV     #$STQSRT,$STKQIN ;;RESET THE POINTER
060334 000002  5$:    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.
  
```

```

060336 022737 000176 001154 $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
060344 001124          BNE     15$              ;;EXIT IF NOT
060346 105777 120606          TSTB    @STKS          ;;IS A CHAR WAITING?
060352 100121          BPL     15$              ;;IF NOT, EXIT
060354 117746 120602          MOVB    @STKB,-(SP)    ;;YES
060360 042716 177600          BIC     #^C177,(SP)    ;;MAKE IT 7-BIT ASCII
060364 021627 000007          CMP     (SP),#7        ;;IS IT A CONTROL-G?
060370 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.
  
```

```

060372 123727 001150 000001 5$:    CMPB    $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?
060400 001674          BEQ     2$              ;;BRANCH IF YES
060402 005726          TST     (SP)+          ;;CLEAR CONTROL-G OFF STACK
060404 004737 060050          JSR     PC,$TKINT      ;;FLUSH THE TTY INPUT QUEUE
060410 005077 120544          CLR     @STKS          ;;DISABLE TTY KEYBOARD INTERRUPTS
060414 112737 000001 001151  MOVB    #1,$INTAG      ;;SET INTERRUPT MODE INDICATOR

060422 104401 061246          TYPE    ,SCNTLG        ;;ECHO THE CONTROL-G (^G)
060426 104401 061253  $GTSWR: TYPE    $MSWR      ;;TYPE CURRENT CONTENTS
060432 013746 000176          MOV     SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
060436 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  
```



```

060440 104401 061264          TYPE      , $MNEW      ;; PROMPT FOR NEW SWR
060444 005046          CLR        -(SP)      ;; CLEAR COUNTER
060446 005046          CLR        -(SP)      ;; THE NEW SWR
060450 105777 120504      7$:      TSTB      @ $TKS      ;; CHAR THERE?
060454 100375          BPL        7$      ;; IF NOT TRY AGAIN

060456 117746 120500          MOVB      @ $TKB, -(SP)  ;; PICK UP CHAR
060462 042716 177600          BIC        # ^C177, (SP)  ;; MAKE IT 7-BIT ASCII

060466 021627 000003          CMP        (SP), #3      ;; IS IT A CONTROL-C?
060472 001015          BNE        9$      ;; BRANCH IF NOT
060474 104401 061234          TYPE      , $CNTLC     ;; YES, ECHO CONTROL-C (^C)
060500 062706 000006          ADD        #6, SP      ;; CLEAN UP STACK
060504 123727 001151 000001  CMPB      $INTAG, #1    ;; REENABLE TTY KEYBOARD INTERRUPTS?
060512 001003          BNE        8$      ;; BRANCH IF NO
060514 012777 000100 120436  MOV        #100, @ $TKS  ;; ALLOW TTY KEYBOARD INTERRUPTS
060522 000137 055642      8$:      JMP        SHUT        ;; CONTROL-C RESTART

060526 021627 000025      9$:      CMP        (SP), #25    ;; IS IT A CONTROL-U?
060532 001005          BNE        10$     ;; BRANCH IF NOT
060534 104401 061241          TYPE      , $CNTLU     ;; YES, ECHO CONTROL-U (^U)
060540 062706 000006      20$:     ADD        #6, SP      ;; IGNORE PREVIOUS INPUT
060544 000737          BR        19$     ;; LET'S TRY IT AGAIN

060546 021627 000015      10$:     CMP        (SP), #15    ;; IS IT A <CR>?
060552 001022          BNE        16$     ;; BRANCH IF NO
060554 005766 000004          TST        4(SP)     ;; YES, IS IT THE FIRST CHAR?
060560 001403          BEQ        11$     ;; BRANCH IF YES
060562 016677 000002 120364  MOV        2(SP), @ $SWR  ;; SAVE NEW SWR
060570 062706 000006      11$:     ADD        #6, SP      ;; CLEAN UP STACK
060574 104401 001217      14$:     TYPE      , $CRLF     ;; ECHO <CR> AND <LF>
060600 123727 001151 000001  CMPB      $INTAG, #1    ;; RE-ENABLE TTY KBD INTERRUPTS?
060606 001003          BNE        15$     ;; BRANCH IF NOT
060610 012777 000100 120342  MOV        #100, @ $TKS  ;; RE-ENABLE TTY KBD INTERRUPTS
060616 000002      15$:     RTI        ;; RETURN
060620 004737 056744      16$:     JSR        PC, $TYPEC  ;; ECHO CHAR
060624 021627 000060          CMP        (SP), #60   ;; CHAR < 0?
060630 002420          BLT        18$     ;; BRANCH IF YES
060632 021627 000067          CMP        (SP), #67   ;; CHAR > 7?
060636 003015          BGT        18$     ;; BRANCH IF YES
060640 042726 000060          BIC        #60, (SP)+  ;; STRIP-OFF ASCII
060644 005766 000002          TST        2(SP)     ;; IS THIS THE FIRST CHAR
060650 001403          BEQ        17$     ;; BRANCH IF YES
060652 006316          ASL        (SP)     ;; NO, SHIFT PRESENT
060654 006316          ASL        (SP)     ;; CHAR OVER TO MAKE
060656 006316          ASL        (SP)     ;; ROOM FOR NEW ONE.
060660 005266 000002      17$:     INC        2(SP)     ;; KEEP COUNT OF CHAR
060664 056616 177776          BIS        -2(SP), (SP)  ;; SET IN NEW CHAR
060670 000667          BR        7$      ;; GET THE NEXT ONE
060672 104401 001216      18$:     TYPE      , $QUES     ;; TYPE ?<CR><LF>
060676 000720          BR        20$     ;; 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
;
060700 011646          $RDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC AND
060702 016666 000004 000002 MOV      4(SP),2(SP)      ;; THE PS
060710 005066 000004      CLR      4(SP)          ;; GET READY FOR A CHARACTER
060714 005046          CLR      -(SP)          ;; PUT NEW PS ON STACK
060716 012746 060724      MOV      #64$,-(SP)      ;; PUT NEW PC ON STACK
060722 000002          RTI                      ;; POP NEW PC AND PS
060724
060724 005737 060040 64$:   TST      $TKCNT          ;; WAIT ON A CHARACTER
060730 001775 1$:         BEQ      1$
060732 005337 060040      DEC      $TKCNT          ;; DECREMENT THE COUNTER
060736 117766 177102 000004 MOVB   @$TKQOUT,4(SP)      ;; GET ONE CHARACTER
060744 005237 060044      INC      $TKQOUT          ;; UPDATE THE POINTER
060750 023727 060044 060047 CMP    $TKQOUT,$$TKQEND  ;; DID IT GO OFF OF THE END?
060756 001003          BNE      2$
060760 012737 060046 060044 MOV    #$TKQSRT,$TKQOUT  ;; RESET THE POINTER
060766 000002          RTI                      ;; RETURN
; *****
; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
; *CALL:
; *      RDLIN          ;; INPUT A STRING FROM THE TTY
; *      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
; *                  ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
;
060770 010346          $RDLIN: MOV      R3, -(SP)          ;; SAVE R3
060772 005046          CLR      -(SP)          ;; CLEAR THE RUBOUT KEY
060774 012703 061224 1$:   MOV      $$TTYIN,R3          ;; GET ADDRESS
061000 022703 061234 2$:   CMP      $$TTYIN+8.,R3      ;; BUFFER FULL?
061004 101456          BLOS     4$
061006 104411          RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
061010 112613          MOVB   (SP)+,(R3)          ;; GET CHARACTER
061012 122713 000177 10$:  CMPB   #177,(R3)          ;; IS IT A RUBOUT
061016 001022          BNE      5$
061020 005716          TST      (SP)
061022 001007          BNE      6$
061024 112737 000134 061222 MOVB   #'\",9$          ;; TYPE A BACK SLASH
061032 104401 061222      TYPE    ,9$
061036 012716 177777 6$:   MOV      #-1,(SP)          ;; SET THE RUBOUT KEY
061042 005303          DEC      R3
061044 020327 061224      CMP      R3,$$ITYIN      ;; BACKUP BY ONE
061050 103434          BLO     4$
061052 111337 061222      MOVB   (R3),9$          ;; STACK EMPTY?
061056 104401 061222      TYPE    ,9$          ;; BR IF YES
061062 000746          BR      2$
061064 005716          5$:   TST      (SP)          ;; SETUP TO TYPEOUT THE DELETED CHAR.
061066 001406          BEQ      7$          ;; GO TYPE
061070 112737 000134 061222 MOVB   #'\",9$          ;; GO READ ANOTHER CHAR.
061076 104401 061222      TYPE    ,9$          ;; RUBOUT KEY SET?
061102 005016          CLR      (SP)          ;; BR IF NO
061104 122713 000025 7$:   CMPB   #25,(R3)          ;; TYPE A BACK SLASH
061110 001003          BNE     8$          ;; CLEAR THE RUBOUT KEY
                                ;; IS CHARACTER A CTRL U?
                                ;; BR IF NO
    
```



```

061112 104401 061241          TYPE      ,SCNTLU      ;;TYPE A CONTROL 'U'
061116 000726          BR          1$          ;;GO START OVER
061120 122713 000022      8$:      CMPB      #22,(R3)      ;;IS CHARACTER A '^R'?
061124 001011          BNE          3$          ;;BRANCH IF NO
061126 105013          CLR      (R3)          ;;CLEAR THE CHARACTER
061130 104401 001217      TYPE      ,SCRLF      ;;TYPE A 'CR' & 'LF'
061134 104401 061224      TYPE      ,STTYIN     ;;TYPE THE INPUT STRING
061140 000717          BR          2$          ;;GO PICKUP ANOTHER CHACTER
061142 104401 001216      4$:      TYPE      ,SQUES      ;;TYPE A '?'
061146 000712          BR          1$          ;;CLEAR THE BUFFER AND LOOP
061150 111337 061222      3$:      MOV      (R3),9$      ;;ECHO THE CHARACTER
061154 104401 061222      TYPE      ,9$
061160 122723 000015      CMPB      #15,(R3)+      ;;CHECK FOR RETURN
061164 001305          BNE          2$          ;;LOOP IF NOT RETURN
061166 105063 177777      CLR      -1(R3)          ;;CLEAR RETURN (THE 15)
061172 104401 001220      TYPE      ,SLF          ;;TYPE A LINE FEED
061176 005726          TST      (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
061200 012603          MOV      (SP)+,R3        ;;RESTORE R3
061202 011646          MOV      (SP),-(SP)       ;;ADJUST THE STACK AND PUT ADDRESS OF THE
061204 016666 000004 000002 MOV      4(SP),2(SP)      ;; FIRST ASCII CHARACTER ON IT
061212 012766 061224 000004 MOV      #$TTYIN,4(SP)
061220 000002          RTI          ;;RETURN
061222 000          9$:      .BYTE      0          ;;STORAGE FOR ASCII CHAR. TO TYPE
061223 000          .BYTE      0          ;;TERMINATOR
061224          $TTYIN: .BLKB      8          ;;RESERVE 8 BYTES FOR TTY INPUT
061234 136 103 015 $CNTLC: .ASCIZ /^C/<15><12> ;;CONTROL 'C'
061241 136 125 015 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
061246 136 107 015 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
061253 015 012 123 $MSWR: .ASCIZ <15><12>/SWR = /
061264 040 040 116 $MNEW: .ASCIZ / NEW = /

```

9

.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

```

```

061276 011646 000004 000002 $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
061300 016666          MOV      4(SP),2(SP)      ;;INPUT NUMBER
061306 010046          MOV      R0,-(SP)          ;;PUSH R0 ON STACK
061310 010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
061312 010246          MOV      R2,-(SP)          ;;PUSH R2 ON STACK
061314 104412      1$:      RDLIN          ;;READ AN ASCII LINE
061316 012600          MOV      (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
061320 005001          CLR      R1          ;;CLEAR DATA WORD
061322 005002          CLR      R2
061324 112046      2$:      MOV      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
061326 001412          BEQ      3$          ;;IF ZERO GET OUT
061330 006301          ASL      R1          ;;*2
061332 006102          ROL      R2
061334 006301          ASL      R1          ;;*4
061336 006102          ROL      R2
061340 006301          ASL      R1          ;;*8

```

```

061342 006102          ROL      R2
061344 042716 177770  BIC      #^C7,(SP)      ;;STRIP THE ASCII JUNK
061350 062601          ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
061352 000764          BR       2$              ;;LOOP
061354 005726          3$:   TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
061356 010166 000012  MOV      R1,12(SP)      ;;SAVE THE RESULT
061362 010237 061376  MOV      R2,$HIOCT
061366 012602          MOV      (SP)+,R2      ;;POP STACK INTO R2
061370 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
061372 012600          MOV      (SP)+,R0      ;;POP STACK INTO R0
061374 000002          RTI
061376 000000          $HIOCT: .WORD 0      ;;RETURN
                                .SBTTL TRAP DECODER      ;;HIGH ORDER BITS GO HERE
    
```

10

```

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

```

061400 016646 000002  $TRAP: MOV      2(SP),-(SP)      ;;ASSUME THE STATUS OF
061404 042716 000020  BIC      #20,(SP)      ;; THE CALLER--DO NOT ALLOW
061410 012746 061416  MOV      #1$,-(SP)      ;; T-BIT TRAPS
061414 000002          RTI              ;;SET THE NEW STATUS
061416 010046          1$:   MOV      R0,-(SP)      ;;SAVE R0
061420 016600 000002  MOV      2(SP),R0      ;;GET TRAP ADDRESS
061424 005740          TST      -(R0)        ;;BACKUP BY 2
061426 111000          MOVB     (R0),R0      ;;GET RIGHT BYTE OF TRAP
061430 006300          ASL      R0          ;;POSITION FOR INDEXING
061432 016000 061452  MOV      $TRPAD(R0),R0  ;;INDEX TC TABLE
061436 000200          RTS      R0          ;;GO TO ROUTINE
    
```

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

```

061440 011646          $TRAP2: MOV      (SP),-(SP)      ;;MOVE THE PC DOWN
061442 016666 000004  MOV      4(SP),2(SP)    ;;MOVE THE PSW DOWN
061450 000002          RTI              ;;RESTORE THE PSW
    
```

.SBTTL TRAP TABLE

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

```

:      ROUTINE
:      -----
061452 061440  $TRPAD: .WORD  $TRAP2
061454 056532  $TYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
061456 056330  $TYPOC ;;CALL=TYPOC      TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
061460 056304  $TYPOS ;;CALL=TYPOS      TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
061462 056344  $TYPON ;;CALL=TYPON      TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
061464 056060  $TYPDS ;;CALL=TYPDS      TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
061466 056004  $TYPBN ;;CALL=TYPBN      TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

061470 060426  $GTSWR ;;CALL=GTSWR      TRAP+7(104407)  GET SOFT-SWR SETTING

061472 060336  $CKSWR ;;CALL=CKSWR      TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
    
```



```
061474 060700 SRDCHR ::CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
061476 060770 SRDLIN ::CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
061500 061276 SRDOCT ::CALL=RDOCT TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
061502 055710 $SAVREG ::CALL=SAVREG TRAP+14(104414) SAVE R0-R5 ROUTINE
061504 055746 $RESREG ::CALL=RESREG TRAP+15(104415) RESTORE R0-R5 ROUTINE
11 .SBTTL POWER DOWN AND UP ROUTINES
```

```
*****
:POWER DOWN ROUTINE
061506 012737 061646 000024 $PWRDN: MOV $ILLUP,@#PWRVEC ::SET FOR FAST UP
061514 012737 000340 000026 MOV #340,@#PWRVEC+2 ::PRIO:7
061522 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
061524 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
061526 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
061530 010346 MOV R3,-(SP) ::PUSH R3 ON STACK
061532 010446 MOV R4,-(SP) ::PUSH R4 ON STACK
061534 010546 MOV R5,-(SP) ::PUSH R5 ON STACK
061536 017746 117412 MOV @SWR,-(SP) ::PUSH @SWR ON STACK
061542 010637 061652 MOV SP,$SAVR6 ::SAVE SP
061546 012737 061560 000024 MOV #SPWRUP,@#PWRVEC ::SET UP VECTOR
061554 000000 HALT
061556 000776 BR -2 ::HANG UP
```

```
*****
:POWER UP ROUTINE
061560 012737 061646 000024 $PWRUP: MOV $ILLUP,@#PWRVEC ::SET FOR FAST DOWN
061566 013706 061652 MOV $SAVR6,SP ::GET SP
061572 005037 061652 CLR $SAVR6 ::WAIT LOOP FOR THE TTY
061576 005237 061652 1$: INC $SAVR6 ::WAIT FOR THE INC
061602 001375 BNE 1$ ::OF WORD
061604 012677 117344 MOV (SP)+,@SWR ::POP STACK INTO @SWR
061610 012605 MOV (SP)+,R5 ::POP STACK INTO R5
061612 012604 MOV (SP)+,R4 ::POP STACK INTO R4
061614 012603 MOV (SP)+,R3 ::POP STACK INTO R3
061616 012602 MOV (SP)+,R2 ::POP STACK INTO R2
061620 012601 MOV (SP)+,R1 ::POP STACK INTO R1
061622 012600 MOV (SP)+,R0 ::POP STACK INTO R0
061624 012737 061506 000024 MOV #SPWRDN,@#PWRVEC ::SET UP THE POWER DOWN VECTOR
061632 012737 000340 000026 MOV #340,@#PWRVEC+2 ::PRIO:7
061640 104401 TYPE $POWER ::REPORT THE POWER FAILURE
061642 061654 $PWRMG: .WORD $POWER ::POWER FAIL MESSAGE POINTER
061644 000002 RTI
061646 000000 $ILLUP: HALT ::THE POWER UP SEQUENCE WAS STARTED
061650 000776 BR -2 ::BEFORE THE POWER DOWN WAS COMPLETE
061652 000000 $SAVR6: 0 ::PUT THE SP HERE
061654 015 012 120 $POWER: .ASCIZ <15><12>'POWER'
.EVEN
```

12 .SBTTL APT COMMUNICATIONS ROUTINE

```
*****
061664 112737 000001 062130 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
061672 112737 000001 062126 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
061700 000403 BR $ATYC
061702 112737 000001 062130 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
061710 $ATYC:
061710 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
061712 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
```

061714	105737	062126		TSTB	\$MFLG	:: SHOULD TYPE A MESSAGE?
061720	001450			BEQ	5\$:: IF NOT: BR
061722	122737	000001	001242	CMPB	#APTENV,\$ENV	:: OPERATING UNDER APT?
061730	001031			BNE	3\$:: IF NOT: BR
061732	132737	000100	001243	BITB	#APTPOOL,\$ENVM	:: SHOULD SPOOL MESSAGES?
061740	001425			BEQ	3\$:: IF NOT: BR
061742	017600	000004		MOV	@4(SP),R0	:: GET MESSAGE ADDR.
061746	062766	000002	000004	ADD	#2,4(SP)	:: BUMP RETURN ADDR.
061754	005737	001222		1\$: TST	\$MSGTYPE	:: SEE IF DONE W/ LAST XMISSION?
061760	001375			BNE	1\$:: IF NOT: WAIT
061762	010037	001236		MOV	R0,\$MSGAD	:: PUT ADDR IN MAILBOX
061766	105720			2\$: TSTB	(R0)+	:: FIND END OF MESSAGE
061770	001376			BNE	2\$	
061772	163700	001236		SUB	\$MSGAD,R0	:: SUB START OF MESSAGE
061776	006200			ASR	R0	:: GET MESSAGE LNTH IN WORDS
062000	010037	001240		MOV	R0,\$MSGGLT	:: PUT LENGTH IN MAILBOX
062004	012737	000004	001222	MOV	#4,\$MSGTYPE	:: TELL APT TO TAKE MSG.
062012	000413			BR	5\$	
062014	017637	000004	062040	3\$: MOV	@4(SP),4\$:: PUT MSG ADDR IN JSR LINKAGE
062022	062766	000002	000004	ADD	#2,4(SP)	:: BUMP RETURN ADDRESS
062030	013746	177776		MOV	177776,-(SP)	:: PUSH 177776 ON STACK
062034	004737	056532		JSR	PC,\$TYPE	:: CALL TYPE MACRO
062040	000000			4\$: .WORD	0	
062042				5\$:		
062042	105737	062130		10\$: TSTB	\$FFLG	:: SHOULD REPORT FATAL ERROR?
062046	001416			BEQ	12\$:: IF NOT: BR
062050	005737	001242		TST	\$ENV	:: RUNNING UNDER APT?
062054	001413			BEQ	12\$:: IF NOT: BR
062056	005737	001222		11\$: TST	\$MSGTYPE	:: FINISHED LAST MESSAGE?
062062	001375			BNE	11\$:: IF NOT: WAIT
062064	017637	000004	001224	MOV	@4(SP),\$FATAL	:: GET ERROR #
062072	062766	000002	000004	ADD	#2,4(SP)	:: BUMP RETURN ADDR.
062100	005237	001222		INC	\$MSGTYPE	:: TELL APT TO TAKE ERROR
062104	105037	062130		12\$: CLRB	\$FFLG	:: CLEAR FATAL FLAG
062110	105037	062127		CLRB	\$LFLG	:: CLEAR LOG FLAG
062114	105037	062126		CLRB	\$MFLG	:: CLEAR MESSAGE FLAG
062120	012601			MOV	(SP)+,R1	:: POP STACK INTO R1
062122	012600			MOV	(SP)+,R0	:: POP STACK INTO R0
062124	000207			RTS	PC	:: RETURN
062126	000			\$MFLG:	.BYTE	0
062127	000			\$LFLG:	.BYTE	0
062130	000			\$FFLG:	.BYTE	0
				.EVEN		
	000200			APTSIZE	=	200
	000001			APTENV	=	001
	000100			APTPOOL	=	100
	000040			APTCSUP	=	040


```
1          .SBTTL  CONSOLE MESSAGES
2
3 062132    075    000          EQUALS: .ASCIZ  @=@
4 062134    101    114    114  ALL:     .ASCIZ  @ALL@<CRLF>
5 062141    040    077    040  QUES:    .ASCIZ  @ ? @
6 062145    054    040    000  COMMA:  .ASCIZ  @, @
7 062150    200    124    131  MSHELP: .ASCIZ  <CRLF>@TYPE HELP TEXT (Y/N) ? @
8 062201
9 062201    200    103    110  UBUSQST: .ASCIZ  <CRLF>@CHANGE ADDRESSES (Y/N) ? @
10 062234   200    125    123  CNSLO0: .ASCIZ  <CRLF>@USE SAME DEVICES (Y/N) ? @
11 062267   200    102    125  CNSLO1: .ASCIZ  <CRLF>@BUS ADDRESS @
12 062305   040    114    111  CNSLO2: .ASCIZ  @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
13 062347   126    105    103  CNSLO3: .ASCIZ  @VECTOR ADDRESS @
14 062367   040    114    111  CNSLO4: .ASCIZ  @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
15 062423   102    122    040  CNSLO5: .ASCIZ  @BR LEVEL @
16 062435   040    114    111  CNSLO6: .ASCIZ  @ LIMITS - LO= 0, HI= 7@<CRLF><LF>
17 062466   200
18 062467   200    124    131  CNSLO7: .ASCII  <CRLF>
19 062554   200    101    116  .ASCIZ  <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
20 062631   200
21 062632   040    077    111  .ASCIZ  <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
22 062653   200    104    122  CNSLO8: .ASCII  <CRLF>
23 062667   104    122    111  CNSLO9: .ASCIZ  @ ?ILLEGAL INPUT@<CRLF>
24 062675   040    111    123  MSDRVS: .ASCIZ  <CRLF>/DRIVE(S): /
25 062715   040    116    117  MSGDRV: .ASCIZ  /DRIVE/
26 062732   040    116    117  LODEV:  .ASCIZ  / IS LOAD DEVICE/
27
28          .EVEN
29
```

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 062752
56
57 062752 020000

```
.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
```


58	062754	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	062756	132000	.WORD	ATA!OPI!IVC!IAE	:SEEK
60	062760	130000	.WORD	ATA!OPI!IVC	:RECALIBRATE
61	062762	020000	.WORD	OPI	:DRIVE CLEAR
62	062764	030000	.WORD	OPI!IVC	:RELEASE
63	062766	130000	.WORD	OPI!ATA!IVC	:OFFSET
64	062770	130000	.WORD	OPI!ATA!IVC	:RETURN TO CENTERLINE
65	062772	020000	.WORD	OPI	:READ IN PRESET
66	062774	020000	.WORD	OPI	:PACK ACKNOWLEDGE
67	062776	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	063000	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	063002	132000	.WORD	ATA!OPI!IVC!IAE	:SEARCH
70	063004	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	063006	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	063010	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	063012	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	063014	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	063016	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	063020	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	063022	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	063024	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	063026	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	063030	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	063032	037200	.WORD	OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	063034	037000	.WORD	OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	063036	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	063040	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	063042	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	063044	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	063046	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	063050	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)
89					

1
2
3 063052 001
4 063053 002
5 063054 004
6 063055 010
7 063056 020
8 063057 040
9 063060 100
10 063061 200
11

.SBTTL ATTENTION (ATA) TABLE

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

Line	Address	Value	Category	Count
1				
2				
3	063062		RGDTPT:	
4	063062		MIXED:	
5	063062	000000	.WORD	0.
6	063064	000001	.WORD	1.
7	063066	000003	.WORD	3.
8	063070	000007	.WORD	7.
9	063072	000017	.WORD	15.
10	063074	000037	.WORD	31.
11	063076	000077	.WORD	63.
12	063100	000177	.WORD	127.
13	063102	000377	.WORD	255.
14	063104	000777	.WORD	511.
15	063106	001777	.WORD	1023.
16	063110	003777	.WORD	2047.
17	063112	007777	.WORD	4095.
18	063114	017777	.WORD	8191.
19	063116	037777	.WORD	16383.
20	063120	077777	.WORD	32767.
21	063122	177777	ONES: .WORD	65535.
22	063124	177777	.WORD	65535.
23	063126	077777	.WORD	32767.
24	063130	037777	.WORD	16383.
25	063132	017777	.WORD	8191.
26	063134	007777	.WORD	4095.
27	063136	003777	.WORD	2047.
28	063140	001777	.WORD	1023.
29	063142	000777	.WORD	511.
30	063144	000377	.WORD	255.
31	063146	000177	.WORD	127.
32	063150	000077	.WORD	63.
33	063152	000037	.WORD	31.
34	063154	000017	.WORD	15.
35	063156	000007	.WORD	7.
36	063160	000003	.WORD	3.
37	063162	000001	.WORD	1.
38	063164	000000	ZEROS: .WORD	0.
39	063166	000000	.WORD	0.
40	063170	000001	.WORD	1.
41	063172	000002	.WORD	2.
42	063174	000004	.WORD	4.
43	063176	000010	.WORD	8.
44	063200	000020	.WORD	16.
45	063202	000040	.WORD	32.
46	063204	000100	.WORD	64.
47	063206	000200	.WORD	128.
48	063210	000400	.WORD	256.
49	063212	001000	.WORD	512.
50	063214	002000	.WORD	1024.
51	063216	004000	.WORD	2048.
52	063220	010000	.WORD	4096.
53	063222	020000	.WORD	8192.
54	063224	040000	.WORD	16384.
55	063226	100000	.WORD	32768.
56	063230	100000	.WORD	32768.
57	063232	040000	.WORD	16384.

58	063234	020000	.WORD	8192.
59	063236	010000	.WORD	4096.
60	063240	004000	.WORD	2048.
61	063242	002000	.WORD	1024.
62	063244	001000	.WORD	512.
63	063246	000400	.WORD	256.
64	063250	000200	.WORD	128.
65	063252	000100	.WORD	64.
66	063254	000040	.WORD	32.
67	063256	000020	.WORD	16.
68	063260	000010	.WORD	8.
69	063262	000004	.WORD	4.
70	063264	000002	.WORD	2.
71	063266	000001	.WORD	1.
72	063270	000000	.WORD	0.
73	063272	177777	.WORD	65535.
74	063274	177776	.WORD	65534.
75	063276	177774	.WORD	65532.
76	063300	177770	.WORD	65528.
77	063302	177760	.WORD	65520.
78	063304	177740	.WORD	65504.
79	063306	177700	.WORD	65472.
80	063310	177600	.WORD	65408.
81	063312	177400	.WORD	65280.
82	063314	177000	.WORD	65024.
83	063316	176000	.WORD	64512.
84	063320	174000	.WORD	63488.
85	063322	170000	.WORD	61440.
86	063324	160000	.WORD	57344.
87	063326	140000	.WORD	49152.
88	063330	100000	.WORD	32768.
89	063332	000000	.WORD	0.
90	063334	000000	.WORD	0.
91	063336	100000	.WORD	32768.
92	063340	140000	.WORD	49152.
93	063342	160000	.WORD	57344.
94	063344	170000	.WORD	61440.
95	063346	174000	.WORD	63488.
96	063350	176000	.WORD	64512.
97	063352	177000	.WORD	65024.
98	063354	177400	.WORD	65280.
99	063356	177600	.WORD	65408.
100	063360	177700	.WORD	65472.
101	063362	177740	.WORD	65504.
102	063364	177760	.WORD	65520.
103	063366	177770	.WORD	65528.
104	063370	177774	.WORD	65532.
105	063372	177776	.WORD	65534.
106	063374	177777	.WORD	65535.
107	063376	125252	.WORD	43690.
108	063400	152525	.WORD	43690./2
109	063402	125252	.WORD	43690.
110	063404	177777	.WORD	65535.
111	063406	177776	.WORD	65534.
112	063410	177775	.WORD	65533.
113	063412	177773	.WORD	65531.
114	063414	177767	.WORD	65527.

115	063416	177757	.WORD	65519.
116	063420	177737	.WORD	65503.
117	063422	177677	.WORD	65471.
118	063424	177577	.WORD	65407.
119	063426	177377	.WORD	65279.
120	063430	176777	.WORD	65023.
121	063432	175777	.WORD	64511.
122	063434	173777	.WORD	63487.
123	063436	167777	.WORD	61439.
124	063440	157777	.WORD	57343.
125	063442	137777	.WORD	49151.
126	063444	077777	.WORD	32767.
127	063446	077777	.WORD	32767.
128	063450	137777	.WORD	49151.
129	063452	157777	.WORD	57343.
130	063454	167777	.WORD	61439.
131	063456	173777	.WORD	63487.
132	063460	175777	.WORD	64511.
133	063462	176777	.WORD	65023.
134	063464	177377	.WORD	65279.
135	063466	177577	.WORD	65407.
136	063470	177677	.WORD	65471.
137	063472	177737	.WORD	65503.
138	063474	177757	.WORD	65519.
139	063476	177767	.WORD	65527.
140	063500	177773	.WORD	65531.
141	063502	177775	.WORD	65533.
142	063504	177776	.WORD	65534.
143	063506	177777	.WORD	65535.
144	063510			
145				
146				

ENRGDT:
.EVEN

				.SBTTL	ERROR MESSAGE TABLE	
1						
2						
3	063510	076634	071542	000000	EMT1:	.WORD EMS300,EMS1,0
4	063516	076652	076675	076722	EMT2:	.WORD EMS301,EMS302,EMS303,EMS1,EMS304
5	063530	101656	101061	101222		.WORD EMS511,EMS500,EMS501,EMS502,EMS503,0
6	063544	076652	076742	076675	EMT3:	.WORD EMS301,EMS306,EMS302
7	063552	101656	101377	101222		.WORD EMS511,EMS505,EMS501,EMS502,0
8	063564	076634	076675	076756	EMT4:	.WORD EMS300,EMS302,EMS307,EMS2
9	063574	101656	101247	101222		.WORD EMS511,EMS502,EMS501,EMS503,0
10	063606	076652	077017	077034	EMT5:	.WORD EMS301,EMS310,EMS311
11	063614	101656	101247	101222		.WORD EMS511,EMS502,EMS501,EMS503,EMS504
12	063626	077100	000000			.WORD EMS312,0
13	063632	076652	076742	077034	EMT6:	.WORD EMS301,EMS306,EMS311
14	063640	101656	101247	101222		.WORD EMS511,EMS502,EMS501,EMS503,EMS504,0
15	063654	076652	077136	076675	EMT7:	.WORD EMS301,EMS313,EMS302
16	063662	101656	101222	101247		.WORD EMS511,EMS501,EMS502,EMS504,EMS503,0
17	063676	077244	077265	077167	EMT10:	.WORD EMS316,EMS317,EMS314
18	063704	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
19	063714	077244	077265	077216	EMT11:	.WORD EMS316,EMS317,EMS315
20	063722	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
21	063732	077244	077305	077167	EMT12:	.WORD EMS316,EMS320,EMS314
22	063740	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
23	063750	077244	077305	077216	EMT13:	.WORD EMS316,EMS320,EMS315
24	063756	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
25	063766	077244	077325	077167	EMT14:	.WORD EMS316,EMS321,EMS314
26	063774	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
27	064004	077244	077325	077216	EMT15:	.WORD EMS316,EMS321,EMS315
28	064012	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
29	064022	077244	077345	077167	EMT16:	.WORD EMS316,EMS322,EMS314
30	064030	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
31	064040	077244	077345	077216	EMT17:	.WORD EMS316,EMS322,EMS315
32	064046	101656	101222	101247		.WORD EMS511,EMS501,EMS502,0
33	064056	076652	077404	076122	EMT20:	.WORD EMS301,EMS324,EMS250
34	064064	101656	101344			.WORD EMS511,EMS504
35	064070	077100	077435	000000		.WORD EMS312,EMS326,0
36	064076	076652	077365	076122	EMT21:	.WORD EMS301,EMS323,EMS250
37	064104	101656	101344			.WORD EMS511,EMS504
38	064110	077100	077424	000000		.WORD EMS312,EMS325,0
39	064116	076652	077136	076122	EMT22:	.WORD EMS301,EMS313,EMS250
40	064124	101656	101344	000000		.WORD EMS511,EMS504,0
41	064132	076652	077404	076160	EMT23:	.WORD EMS301,EMS324,EMS251
42	064140	101656	101222			.WORD EMS511,EMS501
43	064144	077100	077435	000000		.WORD EMS312,EMS326,0
44	064152	076652	077365	076160	EMT24:	.WORD EMS301,EMS323,EMS251
45	064160	101656	101222			.WORD EMS511,EMS501
46	064164	077100	077424	000000		.WORD EMS312,EMS325,0
47	064172	076652	077136	076160	EMT25:	.WORD EMS301,EMS313,EMS251
48	064200	101656	101222	000000		.WORD EMS511,EMS501,0
49	064206	076652	076742	076224	EMT26:	.WORD EMS301,EMS306,EMS252,EMS253,EMS327,EMS254
50	064222	101656	101317	101222		.WORD EMS511,EMS503,EMS501,EMS502
51	064232	077453	077167	000000		.WORD EMS330,EMS314,0
52	064240	076634	076224		EMT27:	.WORD EMS300,EMS252
53	064244	101656	101222	000000		.WORD EMS511,EMS501,0
54	064252	076634	076224		EMT30:	.WORD EMS300,EMS252
55	064256	101656	101222	101344		.WORD EMS511,EMS501,EMS504,0
56	064266	076634	076224		EMT31:	.WORD EMS300,EMS252
57	064272	101656	101222	101317		.WORD EMS511,EMS501,EMS503,0

58	064302	076652	077404	076224	EMT32:	.WORD	EMS301,EMS324,EMS252
59	064310	101656	101222	000000		.WORD	EMS511,EMS501,0
60	064316	076652	077404	076224	EMT33:	.WORD	EMS301,EMS324,EMS252
61	064324	101656	101222	101344		.WORD	EMS511,EMS501,EMS504,0
62	064334	076652	077404	076224	EMT34:	.WORD	EMS301,EMS324,EMS252
63	064342	101656	101222	101317		.WORD	EMS511,EMS501,EMS503,0
64	064352	076652	077365	076224	EMT35:	.WORD	EMS301,EMS323,EMS252
65	064360	101656	101222	000000		.WORD	EMS511,EMS501,0
66	064366	076652	077136	076224	EMT36:	.WORD	EMS301,EMS313,EMS252
67	064374	101656	101222	000000		.WORD	EMS511,EMS501,0
68	064402	076652	077404	076353	EMT37:	.WORD	EMS301,EMS324,EMS255
69	064410	077100	077435	000000		.WORD	EMS312,EMS326,0
70	064416	076652	077365	076353	EMT40:	.WORD	EMS301,EMS323,EMS255
71	064424	101656	101344			.WORD	EMS511,EMS504
72	064430	077100	077424	000000		.WORD	EMS312,EMS325,0
73	064436	076652	077136	076353	EMT41:	.WORD	EMS301,EMS313,EMS255
74	064444	101656	101344	000000		.WORD	EMS511,EMS504,0
75	064452	076652	077365	076122	EMT42:	.WORD	EMS301,EMS323,EMS250,EMS327,EMS255
76	064464	101656	101344	101222		.WORD	EMS511,EMS504,EMS501,EMS503,0
77	064476	076634	071662		EMT43:	.WORD	EMS300,EMS3
78	064502	101656	101222	000000		.WORD	EMS511,EMS501,0
79	064510	077472	071727	076257	EMT44:	.WORD	EMS331,EMS4,EMS253
80	064516	101656	101222	000000		.WORD	EMS511,EMS501,0
81	064524	076634	076257		EMT45:	.WORD	EMS300,EMS253
82	064530	101656	101222	101317		.WORD	EMS511,EMS501,EMS503,0
83	064540	076634	076257		EMT46:	.WORD	EMS300,EMS253
84	064544	101656	101222	101344		.WORD	EMS511,EMS501,EMS504,0
85	064554	076652	077404	076257	EMT47:	.WORD	EMS301,EMS324,EMS253
86	064562	101656	101222	101317		.WORD	EMS511,EMS501,EMS503
87	064570	077100	077435	000000		.WORD	EMS312,EMS326,0
88	064576	076652	077404	076257	EMT50:	.WORD	EMS301,EMS324,EMS253
89	064604	101656	101222	101344		.WORD	EMS511,EMS501,EMS504
90	064612	077100	077435	000000		.WORD	EMS312,EMS326,0
91	064620	076652	077365	076257	EMT51:	.WORD	EMS301,EMS323,EMS253
92	064626	101656	101222			.WORD	EMS511,EMS501
93	064632	077100	077424	000000		.WORD	EMS312,EMS325,0
94	064640	076652	077136	076257	EMT52:	.WORD	EMS301,EMS313,EMS253
95	064646	101656	101222	000000		.WORD	EMS511,EMS501,0
96	064654	077472	071727	076415	EMT53:	.WORD	EMS331,EMS4,EMS256
97	064662	101656	101222	000000		.WORD	EMS511,EMS501,0
98	064670	076652	077404	076415	EMT54:	.WORD	EMS301,EMS324,EMS256
99	064676	101656	101222			.WORD	EMS511,EMS501
100	064702	077100	077435	000000		.WORD	EMS312,EMS326,0
101	064710	076652	077365	076415	EMT55:	.WORD	EMS301,EMS323,EMS256
102	064716	101656	101222			.WORD	EMS511,EMS501
103	064722	077100	077424	000000		.WORD	EMS312,EMS325,0
104	064730	076652	077136	076415	EMT56:	.WORD	EMS301,EMS313,EMS256
105	064736	101656	101222	000000		.WORD	EMS511,EMS501,0
106	064744	076445	077522		EMT57:	.WORD	EMS257,EMS332
107	064750	101656	101452	101222		.WORD	EMS511,EMS506,EMS501,0
108	064760	071760	077540		EMT60:	.WORD	EMS5,EMS333
109	064764	101656	101513	101222		.WORD	EMS511,EMS507,EMS501,0
110	064774	076652	077404	076501	EMT61:	.WORD	EMS301,EMS324,EMS260
111	065002	101656	101222			.WORD	EMS511,EMS501
112	065006	077100	077435	000000		.WORD	EMS312,EMS326,0
113	065014	076652	077365	076501	EMT62:	.WORD	EMS301,EMS323,EMS260
114	065022	101656	101222			.WORD	EMS511,EMS501

115	065026	077100	077424	000000		.WORD	EMS312,EMS325,0
116	065034	076652	077136	076501	EMT63:	.WORD	EMS301,EMS313,EMS260
117	065042	101656	101222	000000		.WORD	EMS511,EMS501,0
118	065050	076634	072301		EMT64:	.WORD	EMS300,EMS12
119	065054	101656	101222	000000		.WORD	EMS511,EMS501,0
120	065062	072301	077564	077662	EMT65:	.WORD	EMS12,EMS335,EMS342
121	065070	101656	101222	000000		.WORD	EMS511,EMS501,0
122	065076	072301	077607	077662	EMT66:	.WORD	EMS12,EMS336,EMS342
123	065104	101656	101222	000000		.WORD	EMS511,EMS501,0
124	065112	076634	072030		EMT67:	.WORD	EMS300,EMS6
125	065116	101656	101222			.WORD	EMS511,EMS501
126	065122	072074	077540	000000		.WORD	EMS7,EMS333,0
127	065130	076634	072030	077445	EMT70:	.WORD	EMS300,EMS6,EMS327,EMS7
128	065140	101656	101222	101344		.WORD	EMS511,EMS501,EMS504,0
129	065150	072030	077564	077642	EMT71:	.WORD	EMS6,EMS335,EMS340,EMS10,EMS333,EMS342
130	065164	101656	101222	101247		.WORD	EMS511,EMS501,EMS502,0
131	065174	072030	077607	077642	EMT72:	.WORD	EMS6,EMS336,EMS340,EMS10,EMS334,EMS342
132	065210	101656	101222	101247		.WORD	EMS511,EMS501,EMS502,0
133	065220	076652	076501	076722	EMT73:	.WORD	EMS301,EMS260,EMS303,EMS11
134	065230	101656	101222	101247		.WORD	EMS511,EMS501,EMS502,0
135	065240	077714	077730	077662	EMT74:	.WORD	EMS343,EMS344,EMS342,0
136	065250	076634	072357		EMT75:	.WORD	EMS300,EMS13
137	065254	101656	101317	000000		.WORD	EMS511,EMS503,0
138	065262	100005	072357	077757	EMT76:	.WORD	EMS346,EMS13,EMS345
139	065270	101656	101317	000000		.WORD	EMS511,EMS503,0
140	065276	077626	072357	077757	EMT77:	.WORD	EMS337,EMS13,EMS345
141	065304	101656	101317	000000		.WORD	EMS511,EMS503,0
142	065312	076652	077136	076312	EMT100:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS13
143	065324	101656	101317	000000		.WORD	EMS511,EMS503,0
144	065332	100005	072426	077653	EMT101:	.WORD	EMS346,EMS14,EMS341,EMS15
145	065342	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
146	065352	077626	072426	077653	EMT102:	.WORD	EMS337,EMS14,EMS341,EMS15
147	065362	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
148	065372	076652	077136	076312	EMT103:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS15
149	065404	101656	101317			.WORD	EMS511,EMS503
150	065410	072426	077522	000000		.WORD	EMS14,EMS332,0
151	065416	100005	072632	077653	EMT104:	.WORD	EMS346,EMS17,EMS341,EMS16
152	065426	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
153	065436	077626	072632	077653	EMT105:	.WORD	EMS337,EMS17,EMS341,EMS16
154	065446	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
155	065456	076652	077136	076312	EMT106:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS16
156	065470	101656	101317			.WORD	EMS511,EMS503
157	065474	072632	077522	000000		.WORD	EMS17,EMS332,0
158	065502	100005	072673	077653	EMT107:	.WORD	EMS346,EMS20,EMS341,EMS21
159	065512	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
160	065522	072673	100051	072737	EMT110:	.WORD	EMS20,EMS351,EMS21,EMS350,EMS22,EMS315
161	065536	101656	101222	000000		.WORD	EMS511,EMS501,0
162	065544	072673	077554	100044	EMT111:	.WORD	EMS20,EMS334,EMS350,EMS22,EMS333
163	065556	101656	101222	000000		.WORD	EMS511,EMS501,0
164	065564	077626	072673	077653	EMT112:	.WORD	EMS337,EMS20,EMS341,EMS21
165	065574	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
166	065604	077626	072673	077653	EMT113:	.WORD	EMS337,EMS20,EMS341,EMS21,EMS350,EMS22,EMS334
167	065622	101656	101222	000000		.WORD	EMS511,EMS501,0
168	065630	072673	100067	072737	EMT114:	.WORD	EMS20,EMS352,EMS21,EMS350,EMS22,EMS333
169	065644	101656	101222	000000		.WORD	EMS511,EMS501,0
170	065652	076652	077136	076312	EMT115:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS21
171	065664	101656	101317			.WORD	EMS511,EMS503

172	065670	072673	077522	000000		.WORD	EMS20,EMS332,0
173	065676	100005	073063	077653	EMT116:	.WORD	EMS346,EMS23,EMS341,EMS24
174	065706	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
175	065716	077626	073063	077653	EMT117:	.WORD	EMS337,EMS23,EMS341,EMS24
176	065726	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
177	065736	076652	077136	076312	EMT120:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS24
178	065750	101656	101317			.WORD	EMS511,EMS503
179	065754	073063	077522	000000		.WORD	EMS23,EMS332,0
180	065762	100005	073220	077653	EMT121:	.WORD	EMS346,EMS25,EMS341,EMS26
181	065772	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
182	066002	077626	073220	077653	EMT122:	.WORD	EMS337,EMS25,EMS341,EMS26
183	066012	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
184	066022	076652	077136	076312	EMT123:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS26
185	066034	101656	101317			.WORD	EMS511,EMS503
186	066040	073220	077522	000000		.WORD	EMS25,EMS332,0
187	066046	076634	073355	076756	EMT124:	.WORD	EMS300,EMS27,EMS307,EMS2
188	066056	101656	101317	000000		.WORD	EMS511,EMS503,0
189	066064	077472	073355	100103	EMT125:	.WORD	EMS331,EMS27,EMS353
190	066072	101656	101317			.WORD	EMS511,EMS503
191	066076	073421	077216	000000		.WORD	EMS30,EMS315,0
192	066104	077626	073355	077653	EMT126:	.WORD	EMS337,EMS27,EMS341,EMS30
193	066114	101656	101317	000000		.WORD	EMS511,EMS503,0
194	066122	076652	077136	076312	EMT127:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS30
195	066134	101656	101317			.WORD	EMS511,EMS503
196	066140	073355	077522	000000		.WORD	EMS27,EMS332,0
197	066146	073501	100127	076122	EMT130:	.WORD	EMS31,EMS354,EMS250
198	066154	101656	101344	101317		.WORD	EMS511,EMS504,EMS503,0
199	066164	073552	100127	076122	EMT131:	.WORD	EMS32,EMS354,EMS250
200	066172	101656	101344	101317		.WORD	EMS511,EMS504,EMS503,0
201	066202	100162	073632	076122	EMT132:	.WORD	EMS355,EMS33,EMS250,EMS341,EMS30
202	066214	101656	101344	000000		.WORD	EMS511,EMS504,0
203	066222	100162	073662	076122	EMT133:	.WORD	EMS355,EMS34,EMS250,EMS341,EMS30
204	066234	101656	101344	000000		.WORD	EMS511,EMS504,0
205	066242	077472	071727	076353	EMT134:	.WORD	EMS331,EMS4,EMS255
206	066250	101656	101344	000000		.WORD	EMS511,EMS504,0
207	066256	073711	100224	100246	EMT135:	.WORD	EMS35,EMS357,EMS360,EMS15
208	066266	101656	101222	000000		.WORD	EMS511,EMS501,0
209	066274	076532	100322		EMT136:	.WORD	EMS261,EMS362
210	066300	101656	101317	000000		.WORD	EMS511,EMS503,0
211	066306	076634	073753	076756	EMT137:	.WORD	EMS300,EMS36,EMS307,EMS2
212	066316	101656	101222	000000		.WORD	EMS511,EMS501,0
213	066324	100162	074003	076353	EMT140:	.WORD	EMS355,EMS37,EMS255,EMS341,EMS30
214	066336	101656	101344	000000		.WORD	EMS511,EMS504,0
215	066344	100005	074035	077757	EMT141:	.WORD	EMS346,EMS40,EMS345
216	066352	101656	101344	000000		.WORD	EMS511,EMS504,0
217	066360	077626	074035	077653	EMT142:	.WORD	EMS337,EMS40,EMS341,EMS30
218	066370	101656	101344	000000		.WORD	EMS511,EMS504,0
219	066376	100343	077017	074113	EMT143:	.WORD	EMS363,EMS310,EMS41
220	066404	101656	101222	000000		.WORD	EMS511,EMS501,0
221	066412	077626	074113	077653	EMT144:	.WORD	EMS337,EMS41,EMS341,EMS252,EMS327,EMS253
222	066426	101656	101222	000000		.WORD	EMS511,EMS501,0
223	066434	074113	100127	100360	EMT145:	.WORD	EMS41,EMS354,EMS364,EMS252,EMS365,EMS253
224	066450	101656	101222	000000		.WORD	EMS511,EMS501,0
225	066456	076652	076742	073753	EMT146:	.WORD	EMS301,EMS306,EMS36
226	066464	101656	101222	101317		.WORD	EMS511,EMS501,EMS503,0
227	066474	100406	074161		EMT147:	.WORD	EMS366,EMS42
228	066500	101656	101317	000000		.WORD	EMS511,EMS503,0

229	066506	100431	100103	100401	EMT150:	.WORD	EMS367,EMS353,EMS365,EMS42,EMS354,EMS3
230	066522	101656	101317	000000		.WORD	EMS511,EMS503,0
231	066530	077626	073753		EMT151:	.WORD	EMS337,EMS36
232	066534	101656	101222	000000		.WORD	EMS511,EMS501,0
233	066542	074243	100127	073753	EMT152:	.WORD	EMS43,EMS354,EMS36
234	066550	101656	101222	000000		.WORD	EMS511,EMS501,0
235	066556	100431	100103	100401	EMT153:	.WORD	EMS367,EMS353,EMS365,EMS36,EMS370
236	066570	101656	101317	000000		.WORD	EMS511,EMS503,0
237	066576	100431	100103	100401	EMT154:	.WORD	EMS367,EMS353,EMS365,EMS36,EMS371
238	066610	101656	101317	000000		.WORD	EMS511,EMS503,0
239	066616	076634	074314	076756	EMT155:	.WORD	EMS300,EMS44,EMS307,EMS2
240	066626	101656	101317	000000		.WORD	EMS511,EMS503,0
241	066634	100431	100103	100401	EMT156:	.WORD	EMS367,EMS353,EMS365,EMS44,EMS354,EMS3
242	066650	101656	101317	000000		.WORD	EMS511,EMS503,0
243	066656	076634	074355	076756	EMT157:	.WORD	EMS300,EMS45,EMS307,EMS2
244	066666	101656	101317	000000		.WORD	EMS511,EMS503,0
245	066674	100431	100103	100401	EMT160:	.WORD	EMS367,EMS353,EMS365,EMS45,EMS354,EMS3
246	066710	101656	101317	101222		.WORD	EMS511,EMS503,EMS501
247	066716	073711	077540	000000		.WORD	EMS35,EMS333,0
248	066724	076634	074432	076756	EMT161:	.WORD	EMS300,EMS46,EMS307,EMS2
249	066734	101656	101317	000000		.WORD	EMS511,EMS503,0
250	066742	077626	074432	100103	EMT162:	.WORD	EMS337,EMS46,EMS353
251	066750	101656	101317	101222		.WORD	EMS511,EMS503,EMS501,0
252	066760	073711	077564	077626	EMT163:	.WORD	EMS35,EMS335,EMS337,EMS41,EMS334,EMS372
253	066774	101656	101222	000000		.WORD	EMS511,EMS501,0
254	067002	074514	077564	077626	EMT164:	.WORD	EMS47,EMS335,EMS337,EMS41,EMS335,EMS372
255	067016	101656	101222	000000		.WORD	EMS511,EMS501,0
256	067024	077626	073711	077445	EMT165:	.WORD	EMS337,EMS35,EMS327,EMS47
257	067034	101656	101222			.WORD	EMS511,EMS501
258	067040	074113	077540	100545		.WORD	EMS41,EMS333,EMS372,0
259	067050	076634	074514	076756	EMT166:	.WORD	EMS300,EMS47,EMS307,EMS2
260	067060	101656	101222	101317		.WORD	EMS511,EMS501,EMS503,0
261	067070	074554	077564	077642	EMT167:	.WORD	EMS50,EMS335,EMS340,EMS36,EMS333
262	067102	101656	101222	101317		.WORD	EMS511,EMS501,EMS503,0
263	067112	077626	073711		EMT170:	.WORD	EMS337,EMS35
264	067116	101656	101222	000000		.WORD	EMS511,EMS501,0
265	067124	074554	073662	071662	EMT171:	.WORD	EMS50,EMS34,EMS3
266	067132	101656	101222	000000		.WORD	EMS511,EMS501,0
267	067140	076652	076742	074623	EMT172:	.WORD	EMS301,EMS306,EMS51
268	067146	101656	101222	101344		.WORD	EMS511,EMS501,EMS504,0
269	067156	100431	100103	100401	EMT173:	.WORD	EMS367,EMS353,EMS365,EMS47,EMS354,EMS3
270	067172	101656	101222	000000		.WORD	EMS511,EMS501,0
271	067200	076634	076122	077445	EMT174:	.WORD	EMS300,EMS250,EMS327,EMS255,EMS327,EMS256
272	067214	077653	101755			.WORD	EMS341,EMS600
273	067220	101656	101222	000000		.WORD	EMS511,EMS501,0
274	067226	076634	076415	077653	EMT175:	.WORD	EMS300,EMS256,EMS341,EMS600
275	067236	101656	101222	000000		.WORD	EMS511,EMS501,0
276	067244	076634	076122	077653	EMT176:	.WORD	EMS300,EMS250,EMS341,EMS600
277	067254	101656	101344	000000		.WORD	EMS511,EMS504,0
278	067262	076634	076353	077653	EMT177:	.WORD	EMS300,EMS255,EMS341,EMS600
279	067272	101656	101344	000000		.WORD	EMS511,EMS504,0
280	067300	077626	074672	077653	EMT200:	.WORD	EMS337,EMS52,EMS341,EMS601
281	067310	101656	101222	000000		.WORD	EMS511,EMS501,0
282	067316	100005	074672	077653	EMT201:	.WORD	EMS346,EMS52,EMS341,EMS602
283	067326	101656	101222	000000		.WORD	EMS511,EMS501,0
284	067334	100005	074623	077653	EMT202:	.WORD	EMS346,EMS51,EMS341,EMS602
285	067344	101656	101222	000000		.WORD	EMS511,EMS501,0

286	067352	100005	074672	077757	EMT203: .WORD	EMS346,EMS52,EMS345,EMS373,EMS255
287	067364	101656	101344	101222	.WORD	EMS511,EMS504,EMS501,0
288	067374	100005	074672	077653	EMT204: .WORD	EMS346,EMS52,EMS341,EMS27
289	067404	101656	101344	101222	.WORD	EMS511,EMS504,EMS501,0
290	067414	074733	100127	071662	EMT205: .WORD	EMS53,EMS354,EMS3
291	067422	101656	101317	101247	.WORD	EMS511,EMS503,EMS502,EMS510,0
292	067434	077626	074774	100603	EMT206: .WORD	EMS337,EMS54,EMS374,EMS250,EMS327,EMS255
293	067450	077445	071662		.WORD	EMS327,EMS3
294	067454	101656	101344	101222	.WORD	EMS511,EMS504,EMS501,0
295	067464	074774	100127	071662	EMT207: .WORD	EMS54,EMS354,EMS3
296	067472	101656	101222	000000	.WORD	EMS511,EMS501,0
297	067500	074774	100127	100360	EMT210: .WORD	EMS54,EMS354,EMS364,EMS250
298	067510	101656	101344	000000	.WORD	EMS511,EMS504,0
299	067516	074774	100127	100360	EMT211: .WORD	EMS54,EMS354,EMS364,EMS255
300	067526	101656	101344	000000	.WORD	EMS511,EMS504,0
301	067534	077626	075051		EMT212: .WORD	EMS337,EMS55
302	067540	101656	101344	000000	.WORD	EMS511,EMS504,0
303	067546	075127	077554	077757	EMT213: .WORD	EMS56,EMS334,EMS345,EMS373,EMS262,EMS327,EMS251
304	067564	101656	101222		.WORD	EMS511,EMS501
305	067570	075127	077564	000000	.WORD	EMS56,EMS335,0
306	067576	077626	075127		EMT214: .WORD	EMS337,EMS56
307	067602	101656	101222	000000	.WORD	EMS511,EMS501,0
308	067610	075222	077540	100044	EMT215: .WORD	EMS57,EMS333,EMS350,EMS60,EMS334
309	067622	101656	101222	000000	.WORD	EMS511,EMS501,0
310	067630	100343	076742	071760	EMT216: .WORD	EMS363,EMS306,EMS5
311	067636	101656	101222	000000	.WORD	EMS511,EMS501,0
312	067644	100343	076742	075360	EMT217: .WORD	EMS363,EMS306,EMS61
313	067652	101656	101222	000000	.WORD	EMS511,EMS501,0
314	067660	075433	077540	100634	EMT220: .WORD	EMS62,EMS333,EMS375,EMS251
315	067670	101656	101222	000000	.WORD	EMS511,EMS501,0
316	067676	075433	077540	100650	EMT221: .WORD	EMS62,EMS333,EMS376,EMS262
317	067706	101656	101222	000000	.WORD	EMS511,EMS501,0
318	067714	075433	077540	100650	EMT222: .WORD	EMS62,EMS333,EMS376,EMS250
319	067724	101656	101222	000000	.WORD	EMS511,EMS501,0
320	067732	100005	075433	077653	EMT223: .WORD	EMS346,EMS62,EMS341,EMS603
321	067742	101656	101222	000000	.WORD	EMS511,EMS501,0
322	067750	100005	075514	100650	EMT224: .WORD	EMS346,EMS63,EMS376,EMS262
323	067760	101656	101222	101317	.WORD	EMS511,EMS501,EMS503,0
324	067770	075514	077540	100044	EMT225: .WORD	EMS63,EMS333,EMS350,EMS363,EMS310,EMS262
325	070004	101656	101222	000000	.WORD	EMS511,EMS501,0
326	070012	075514	100224	073753	EMT226: .WORD	EMS63,EMS357,EMS36,EMS372
327	070022	101656	101222	000000	.WORD	EMS511,EMS501,0
328	070030	075514	100204	100246	EMT227: .WORD	EMS63,EMS356,EMS360,EMS15
329	070040	101656	101222	000000	.WORD	EMS511,EMS501,0
330	070046	075514	100204	100274	EMT230: .WORD	EMS63,EMS356,EMS361,EMS15
331	070056	101656	101222	000000	.WORD	EMS511,EMS501,0
332	070064	075514	100204	074113	EMT231: .WORD	EMS63,EMS356,EMS41
333	070072	101656	101222	000000	.WORD	EMS511,EMS501,0
334	070100	074113	100664	077662	EMT232: .WORD	EMS41,EMS377,EMS342,EMS365,EMS63,EMS332
335	070114	101656	101222	000000	.WORD	EMS511,EMS501,0
336	070122	100431	100103	100401	EMT233: .WORD	EMS367,EMS353,EMS365,EMS63,EMS401
337	070134	101656	101317	101222	.WORD	EMS511,EMS503,EMS501,0
338	070144	072551	100664	100545	EMT234: .WORD	EMS16,EMS377,EMS372,EMS365,EMS64,EMS354,EMS3
339	070162	101656	101317	101222	.WORD	EMS511,EMS503,EMS501,0
340	070172	076634	075624	076756	EMT235: .WORD	EMS300,EMS65,EMS307,EMS2
341	070202	101656	101247	101317	.WORD	EMS511,EMS502,EMS503,0
342	070212	075127	100664	100650	EMT236: .WORD	EMS56,EMS377,EMS376,EMS252,EMS372,EMS350

343	070226	075624	100712		.WORD	EMS65,EMS401
344	070232	101656	101247	101317	.WORD	EMS511,EMS502,EMS503,EMS501,0
345	070244	076634	075665	076756	EMT237: .WORD	EMS300,EMS66,EMS307,EMS2
346	070254	101656	101222	101317	.WORD	EMS511,EMS501,EMS503,0
347	070264	072473	100675	100545	EMT240: .WORD	EMS15,EMS400,EMS372,EMS350,EMS66,EMS401
348	070300	101656	101317	101222	.WORD	EMS511,EMS503,EMS501,0
349	070310	075665	077607	077642	EMT241: .WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS604
350	070326	101656	101317	000000	.WORD	EMS511,EMS503,0
351	070334	100736	102107	100727	EMT242: .WORD	EMS403,EMS604,EMS402,EMS21,EMS377
352	070346	101656	101317		.WORD	EMS511,EMS503
353	070352	073753	077564	000000	.WORD	EMS36,EMS335,0
354	070360	075665	077607	077642	EMT243: .WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS604
355	070376	101656	101317	000000	.WORD	EMS511,EMS503,0
356	070404	075514	100712	101016	EMT244: .WORD	EMS63,EMS401,EMS405,EMS604
357	070414	101656	101317	000000	.WORD	EMS511,EMS503,0
358	070422	073753	100501	101016	EMT245: .WORD	EMS36,EMS370,EMS405,EMS604
359	070432	101656	101317	000000	.WORD	EMS511,EMS503,0
360	070440	100736	102107	100727	EMT246: .WORD	EMS403,EMS604,EMS402,EMS24,EMS377
361	070452	101656	101317		.WORD	EMS511,EMS503
362	070456	073753	077564	000000	.WORD	EMS36,EMS335,0
363	070464	075742	077522	101016	EMT247: .WORD	EMS67,EMS332,EMS405,EMS604
364	070474	101656	101317	000000	.WORD	EMS511,EMS503,0
365	070502	075665	077607	077642	EMT250: .WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS605
366	070520	101656	101317	000000	.WORD	EMS511,EMS503,0
367	070526	100736	102134	100727	EMT251: .WORD	EMS403,EMS605,EMS402,EMS21,EMS377
368	070540	101656	101317		.WORD	EMS511,EMS503
369	070544	073753	077564	000000	.WORD	EMS36,EMS335,0
370	070552	075665	077607	077642	EMT252: .WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS605
371	070570	101656	101317	000000	.WORD	EMS511,EMS503,0
372	070576	075514	100712	101016	EMT253: .WORD	EMS63,EMS401,EMS405,EMS605
373	070606	101656	101317	000000	.WORD	EMS511,EMS503,0
374	070614	073753	100501	101016	EMT254: .WORD	EMS36,EMS370,EMS405,EMS605
375	070624	101656	101317	000000	.WORD	EMS511,EMS503,0
376	070632	100736	102134	100727	EMT255: .WORD	EMS403,EMS605,EMS402,EMS24,EMS377
377	070644	101656	101317		.WORD	EMS511,EMS503
378	070650	073753	077564	000000	.WORD	EMS36,EMS335,0
379	070656	075742	077522	101016	EMT256: .WORD	EMS67,EMS332,EMS405,EMS605
380	070666	101656	101317	000000	.WORD	EMS511,EMS503,0
381	070674	075665	077607	077642	EMT257: .WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS606
382	070712	101656	101317	000000	.WORD	EMS511,EMS503,0
383	070720	100736	102152	100727	EMT260: .WORD	EMS403,EMS606,EMS402,EMS21,EMS377
384	070732	101656	101317		.WORD	EMS511,EMS503
385	070736	073753	077564	000000	.WORD	EMS36,EMS335,0
386	070744	075665	077607	077642	EMT261: .WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS606
387	070762	101656	101317	000000	.WORD	EMS511,EMS503,0
388	070770	075514	100712	101016	EMT262: .WORD	EMS63,EMS401,EMS405,EMS606
389	071000	101656	101317	000000	.WORD	EMS511,EMS503,0
390	071006	073753	100501	101016	EMT263: .WORD	EMS36,EMS370,EMS405,EMS606
391	071016	101656	101317	000000	.WORD	EMS511,EMS503,0
392	071024	100736	102152	100727	EMT264: .WORD	EMS403,EMS606,EMS402,EMS24,EMS377
393	071036	101656	101317		.WORD	EMS511,EMS503
394	071042	073753	077564	000000	.WORD	EMS36,EMS335,0
395	071050	076054	100712	101016	EMT265: .WORD	EMS70,EMS401,EMS405,EMS606
396	071060	101656	101317	000000	.WORD	EMS511,EMS503,0
397	071066	075742	077522	101016	EMT266: .WORD	EMS67,EMS332,EMS405,EMS606
398	071076	101656	101317	000000	.WORD	EMS511,EMS503,0
399	071104	075665	100204	101041	EMT267: .WORD	EMS66,EMS356,EMS407

400	071112	101656	101317	000000		.WORD	EMS511,EMS503,0
401	071120	075665	077607	077642	EMT270:	.WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS607
402	071136	101656	101317	000000		.WORD	EMS511,EMS503,0
403	071144	100736	102172	100727	EMT271:	.WORD	EMS403,EMS607,EMS402,EMS21,EMS377
404	071156	101656	101317			.WORD	EMS511,EMS503
405	071162	073355	077607	000000		.WORD	EMS27,EMS336,0
406	071170	073355	100501	101016	EMT272:	.WORD	EMS27,EMS370,EMS405,EMS607
407	071200	101656	101317	000000		.WORD	EMS511,EMS503,0
408	071206	073355	100516	101016	EMT273:	.WORD	EMS27,EMS371,EMS405,EMS607
409	071216	101656	101317	000000		.WORD	EMS511,EMS503,0
410	071224	073753	100777	101016	EMT274:	.WORD	EMS36,EMS404,EMS405,EMS607
411	071234	101656	101317	000000		.WORD	EMS511,EMS503,0
412	071242	074733	100712	101016	EMT275:	.WORD	EMS53,EMS401,EMS405,EMS607
413	071252	101656	101317	101247		.WORD	EMS511,EMS503,EMS502,0
414	071262	075742	077522	101016	EMT276:	.WORD	EMS67,EMS332,EMS405,EMS607
415	071272	101656	101317	000000		.WORD	EMS511,EMS503,0
416	071300	075665	077607	077642	EMT277:	.WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS607
417	071316	101656	101317	000000		.WORD	EMS511,EMS503,0
418	071324	100736	102172	100727	EMT300:	.WORD	EMS403,EMS607,EMS402,EMS24,EMS377
419	071336	101656	101317			.WORD	EMS511,EMS503
420	071342	073355	077607	000000		.WORD	EMS27,EMS336,0
421	071350	076054	100712	101016	EMT301:	.WORD	EMS70,EMS401,EMS405,EMS607
422	071360	101656	101317	000000		.WORD	EMS511,EMS503,0
423							

1	071366	102210	000000	EHT1:	.WORD	EH1,0
2	071372	102266	000000	EHT2:	.WORD	EH2,0
3	071376	102303	000000	EHT5:	.WORD	EH5,0
4	071402	102341	000000	EHT7:	.WORD	EH7,0
5	071406	102360	000000	EHT57:	.WORD	EH57,0
6	071412	102456	000000	EHT65:	.WORD	EH65,0
7	071416	102532	000000	EHT71:	.WORD	EH71,0
8	071422	102274	000000	EHT74:	.WORD	EH3,0
9	071426	102607	000000	EHT115:	.WORD	EH115,0
10	071432	102704	000000	EHT130:	.WORD	EH130,0
11	071436	103022	000000	EHT132:	.WORD	EH132,0
12	071442	103120	000000	EHT142:	.WORD	EH142,0
13	071446	103216	000000	EHT145:	.WORD	EH145,0
14	071452	103333	000000	EHT150:	.WORD	EH150,0
15	071456	103431	000000	ENT213:	.WORD	EH213,0
16	071462	103526	000000	EHT220:	.WORD	EH220,0
17						

1 071466 103564
2 071470 103574
3 071472 103600
4 071474 103606
5 071476 103620
6 071500 103630
7 071502 103574
8 071504 103640
9 071506 103652
10 071510 103640
11 071512 103666
12

EDT1: .WORD ED1
EDT2: .WORD ED2
EDT5: .WORD ED5
EDT57: .WORD ED57
EDT65: .WORD ED65
EDT71: .WORD ED71
EDT74: .WORD ED2
EDT115: .WORD ED115
EDT130: .WORD ED130
EDT132: .WORD ED115
EDT220: .WORD ED220

1	071514	103672	EFT1:	.WORD	EF1
2	071516	103675	EFT2:	.WORD	EF2
3	071520	103676	EFT5:	.WORD	EF5
4	071522	103700	EFT57:	.WORD	EF57
5	071524	103672	EFT65:	.WORD	EF1
6	071526	103672	EFT71:	.WORD	EF1
7	071530	103675	EFT74:	.WORD	EF2
8	071532	103700	EFT115:	.WORD	EF57
9	071534	103704	EFT130:	.WORD	EF130
10	071536	103700	EFT132:	.WORD	EF57
11	071540	103676	EFT220:	.WORD	EF5
12					

Line	Code	Hex 1	Hex 2	Hex 3	Label	Format	Description
1					.SBTTL		ERROR MESSAGE STRINGS
2							
3	071542	116	117	116	EMS1:	.ASCIZ	@NONEXISTENT DEVICE 'NED' (RMCS2,BIT 12) a
4	071613	103	117	116	EMS2:	.ASCIZ	@CONTROLLER CLEAR 'CLR' (RMCS2,BIT 05) a
5	071662	106	125	116	EMS3:	.ASCIZ	@FUNCTION CODE (RMCS1, BITS 01 - 05) a
6	071727	125	116	125	EMS4:	.ASCIZ	@UNUSED BIT POSITIONS OF a
7	071760	104	105	126	EMS5:	.ASCIZ	@DEVICE AVAILABLE 'DVA' (RMCS1, BIT 11) a
8	072030	120	101	122	EMS6:	.ASCIZ	@PARTIY ERROR 'PAR' (RMER1, BIT 03) a
9	072074	104	101	124	EMS7:	.ASCIZ	@DATA PARITY ERROR 'DPE' (RMER2, BIT 03) a
10	072145	120	101	122	EMS10:	.ASCIZ	@PARITY TEST 'PAT' (RMCS2, BIT 04) a
11	072210	115	101	123	EMS11:	.ASCII	@MASSBUS CONTROL BUS PARITY ERROR 'MCPE' a
12	072260	050	122	115		.ASCIZ	@(RMCS1, BIT 13) a
13	072301	111	114	114	EMS12:	.ASCIZ	@ILLEGAL REGISTER ERROR 'ILR' (RMER1, BIT 01) a
14	072357	104	111	101	EMS13:	.ASCIZ	@DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 00) a
15	072426	115	105	104	EMS14:	.ASCIZ	@MEDIUM ON LINE 'MOL' (RMDS, BIT 12) a
16	072473	115	101	111	EMS15:	.ASCIZ	@MAINTENANCE UNIT READY 'MUR' (RMMR1, BIT 09) a
17	072551	115	101	111	EMS16:	.ASCIZ	@MAINTENANCE WRITE PROTECT 'MWP' (RMMR1, BIT 03) a
18	072632	127	122	111	EMS17:	.ASCIZ	@WRITE LOCK 'WRL' (RMDS, BIT 11) a
19	072673	104	105	126	EMS20:	.ASCIZ	@DEVICE CHECK 'DVC' (RMER2, BIT 07) a
20	072737	115	101	111	EMS21:	.ASCIZ	@MAINTENANCE DRIVE FAULT 'MDF' (RMMR1, BIT 06) a
21	073016	125	116	123	EMS22:	.ASCIZ	@UNSAFE STATUS 'UNS' (RMER1, BIT 14) a
22	073063	123	105	105	EMS23:	.ASCIZ	@SEEK INCOMPLETE STATUS 'SKI' (RMER2, BIT 14) a
23	073141	115	101	111	EMS24:	.ASCIZ	@MAINTENANCE SEEK ERROR 'MSER' (RMMR1, BIT 07) a
24	073220	120	117	123	EMS25:	.ASCIZ	@POSITIONING IN PROGRESS 'PIP' (RMDS, BIT 13) a
25	073276	115	101	111	EMS26:	.ASCIZ	@MAINTENANCE ON CYLINDER 'MOC' (RMMR1, BIT 08) a
26	073355	105	116	104	EMS27:	.ASCIZ	@END OF BLOCK 'EBL' (RMMR1, BIT 13) a
27	073421	104	111	101	EMS30:	.ASCIZ	@DIAGNOSTIC END OF BLOCK 'DEBL' (RMMR1, BIT 13) a
28	073501	114	101	123	EMS31:	.ASCIZ	@LAST SECTOR STATUS 'LS' (RMMR1, BIT 02) a
29	073552	114	101	123	EMS32:	.ASCIZ	@LAST SECTOR/TRACK STATUS 'LST' (RMMR1, BIT 01) a
30	073632	123	105	103	EMS33:	.ASCIZ	@SECTOR ADDRESS BITS OF a
31	073662	124	122	101	EMS34:	.ASCIZ	@TRACK ADDRESS BITS OF a
32	073711	126	117	114	EMS35:	.ASCIZ	@VOLUME VALID 'VV' (RMDS, BIT 06) a
33	073753	107	117	040	EMS36:	.ASCIZ	@GO BIT (RMCS1, BIT 00) a
34	074003	103	131	114	EMS37:	.ASCIZ	@CYLINDER ADDRESS BITS OF a
35	074035	114	101	123	EMS40:	.ASCIZ	@LAST BLOCK TRANSFERRED, 'LBT' (RMDS, BIT 10) a
36	074113	103	117	115	EMS41:	.ASCIZ	@COMPOSITE ERROR 'ERR' (RMDS, BIT 14) a
37	074161	103	117	115	EMS42:	.ASCIZ	@COMMAND SEQUENCER TEST BIT 'TST' (RMMR2, BIT 12) a
38	074243	104	122	111	EMS43:	.ASCIZ	@DRIVE READY STATUS 'DRY' (RMDS, BIT 07) a
39	074314	103	117	116	EMS44:	.ASCIZ	@CONTINUE 'CONT' (RMMR1, BIT 06) a
40	074355	111	116	126	EMS45:	.ASCIZ	@INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) a
41	074432	114	117	123	EMS46:	.ASCIZ	@LOSS OF SYSTEM CLOCK ERROR 'LSC' (RMER2, BIT 11) a
42	074514	117	103	103	EMS47:	.ASCIZ	@OCCUPIED 'OCC' (RMMR1, BIT 15) a
43	074554	111	114	114	EMS50:	.ASCIZ	@ILLEGAL FUNCTION 'ILF' (RMER1, BIT 0) a
44	074623	117	106	106	EMS51:	.ASCIZ	@OFFSET DIRECTION 'OFD' (RMOF, BIT 07) a
45	074672	117	106	106	EMS52:	.ASCIZ	@OFFSET MODE 'OM' (RMDS, BIT 00) a
46	074733	122	125	116	EMS53:	.ASCIZ	@RUN AND GO 'RG' (RMMR1, BIT 14) a
47	074774	111	116	126	EMS54:	.ASCIZ	@INVALID ADDRESS ERROR 'IAE' (RMER1, BIT 10) a
48	075051	101	104	104	EMS55:	.ASCIZ	@ADDRESS OVERFLOW ERROR 'AOE' (RMER1, BIT 09) a
49	075127	122	105	107	EMS56:	.ASCII	@REGISTER MODIFICATION REFUSED ERROR a
50	075173	042	122	115		.ASCIZ	@'RMR' (RMER1, BIT 02) a
51	075222	104	122	111	EMS57:	.ASCIZ	@DRIVE REQUEST REQUIRED STATUS 'DRQ' (RMDT, BIT 11) a
52	075306	120	122	117	EMS60:	.ASCIZ	@PROGRAMMABLE STATUS 'PGM' (RMDS, BIT 09) a
53	075360	104	122	111	EMS61:	.ASCIZ	@DRIVE PRESENT STATUS 'DPR' (RMDS, BIT 08) a
54	075433	120	117	122	EMS62:	.ASCIZ	@PORT REQUEST FLOP 'RQA,RQB' (RMMR2, BITS 15,14) a
55	075514	101	124	124	EMS63:	.ASCIZ	@ATTENTION 'ATA' (RMDS, BIT 15) a
56	075554	127	122	111	EMS64:	.ASCIZ	@WRITE LOCK ERROR 'WLE' (RMER1, BIT 11) a
57	075624	105	130	103	EMS65:	.ASCIZ	@EXCEPTION 'REX' (RMMR1, BIT 12) a

58	075665	111	116	126	EMS66:	.ASCIZ	@INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) @
59	075742	124	101	107	EMS67:	.ASCIZ	@TAG BUS (RMMR2, BITS 00-09) OR TAG CONTROL @
60	076016	114	111	116		.ASCIZ	@LINES (RMMR2, BITS 10,11,13) @
61	076054	123	105	101	EMS70:	.ASCIZ	@SEARCH ENABLE 'ESRC' (RMMR1, BIT 11) @
62							
63	076122	104	111	123	EMS250:	.ASCIZ	@DISK ADDRESS REGISTER (RMDA) @
64	076160	103	117	116	EMS251:	.ASCIZ	@CONTROL STATUS REGISTER #1 (RMCS1) @
65	076224	105	122	122	EMS252:	.ASCIZ	@ERROR REGISTER #1 (RMER1) @
66	076257	105	122	122	EMS253:	.ASCIZ	@ERROR REGISTER #2 (RMER2) @
67	076312	115	101	111	EMS254:	.ASCIZ	@MAINTENANCE REGISTER #1 (RMMR1) @
68	076353	104	105	123	EMS255:	.ASCIZ	@DESIRED CYLINDER REGISTER (RMDC) @
69	076415	117	106	106	EMS256:	.ASCIZ	@OFFSET REGISTER (RMOF) @
70	076445	104	122	111	EMS257:	.ASCIZ	@DRIVE TYPE REGISTER (RMDT) @
71	076501	110	117	114	EMS260:	.ASCIZ	@HOLDING REGISTER (RMHR) @
72	076532	123	105	122	EMS261:	.ASCIZ	@SERIAL NUMBER REGISTER (RMSN) @
73	076571	101	124	124	EMS262:	.ASCIZ	@ATTENTION SUMMARY REGISTER (RMAS) @
74							
75	076634	103	101	116	EMS300:	.ASCIZ	@CANNOT CLEAR @
76	076652	103	101	116	EMS301:	.ASCIZ	@CANNOT WRITE/READ @
77	076675	101	116	131	EMS302:	.ASCIZ	@ANY DEVICE REGISTER @
78	076722	127	111	124	EMS303:	.ASCIZ	@WITHOUT @
79	076733	105	122	122	EMS304:	.ASCIZ	@ERROR @
80	076742	101	040	117	EMS306:	.ASCIZ	@A ONE FROM @
81	076756	125	123	111	EMS307:	.ASCIZ	@USING MASSBUS INITIALIZE, I.E., @
82	077017	101	040	132	EMS310:	.ASCIZ	@A ZERO FROM @
83	077034	105	126	105	EMS311:	.ASCIZ	@EVERY DEVICE REGISTER BIT POSITION @
84	077100	124	110	105	EMS312:	.ASCIZ	@THE FOLLOWING BITS ARE STUCK @
85	077136	101	040	123	EMS313:	.ASCIZ	@A SHIFTING ONE BIT FROM @
86	077167	101	120	120	EMS314:	.ASCIZ	@APPEARS STUCK AT ZERO @
87	077216	101	120	120	EMS315:	.ASCIZ	@APPEARS STUCK AT ONE @
88	077244	122	105	107	EMS316:	.ASCIZ	@REGISTER SELECT @
89	077265	061	040	050	EMS317:	.ASCIZ	@1 (1,2,4,8,16) @
90	077305	062	040	050	EMS320:	.ASCIZ	@2 (1,2,4,8,16) @
91	077325	064	040	050	EMS321:	.ASCIZ	@4 (1,2,4,8,16) @
92	077345	070	040	050	EMS322:	.ASCIZ	@8 (1,2,4,8,16) @
93	077365	101	114	114	EMS323:	.ASCIZ	@ALL ONES FROM @
94	077404	101	114	114	EMS324:	.ASCIZ	@ALL ZEROS FROM @
95	077424	101	124	040	EMS325:	.ASCIZ	@AT ZERO @
96	077435	101	124	040	EMS326:	.ASCIZ	@AT ONE @
97	077445	054	040	117	EMS327:	.ASCIZ	@, OR @
98	077453	015	012	103	EMS330:	.ASCIZ	<CR><LF>@CS MBA CLRL @
99	077472	103	101	116	EMS331:	.ASCIZ	@CANNOT READ ZEROS FROM @
100	077522	111	123	040	EMS332:	.ASCIZ	@IS INCORRECT @
101	077540	111	123	040	EMS333:	.ASCIZ	@IS NOT SET @
102	077554	111	123	040	EMS334:	.ASCIZ	@IS SET @
103	077564	123	110	117	EMS335:	.ASCIZ	@SHOULD NOT BE SET @
104	077607	123	110	117	EMS336:	.ASCIZ	@SHOULD BE SET @
105	077626	103	101	116	EMS337:	.ASCIZ	@CANNOT SET @
106	077642	102	105	103	EMS340:	.ASCIZ	@BECAUSE @
107	077653	125	123	111	EMS341:	.ASCIZ	@USING @
108	077662	104	125	122	EMS342:	.ASCIZ	@DURING REGISTER TRANSFER @
109	077714	125	116	105	EMS343:	.ASCIZ	@UNEXPECTED @
110	077730	102	125	123	EMS344:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
111	077757	102	131	040	EMS345:	.ASCIZ	@BY REGISTER TRANSFER @
112	100005	103	101	116	EMS346:	.ASCIZ	@CANNOT RESET @
113	100023	127	111	124	EMS347:	.ASCIZ	@WITHOUT SETTING @
114	100044	102	125	124	EMS350:	.ASCIZ	@BUT @

115	100051	127	101	123	EMS351:	.ASCIZ	@WAS RESET BY @
116	100067	127	101	123	EMS352:	.ASCIZ	@WAS SET BY @
117	100103	111	116	040	EMS353:	.ASCIZ	@IN DIAGNOSTIC MODE @
118	100127	111	123	040	EMS354:	.ASCIZ	@IS INCORRECT ACCORDING TO @
119	100162	103	101	116	EMS355:	.ASCIZ	@CANNOT INCREMENT @
120	100204	127	101	123	EMS356:	.ASCIZ	@WAS NOT SET BY @
121	100224	127	101	123	EMS357:	.ASCIZ	@WAS NOT RESET BY @
122	100246	060	040	124	EMS360:	.ASCIZ	@0 TO 1 TRANSITION OF @
123	100274	061	040	124	EMS361:	.ASCIZ	@1 TO 0 TRANSITION OF @
124	100322	111	123	040	EMS362:	.ASCIZ	@IS INCONSISTENT @
125	100343	103	101	116	EMS363:	.ASCIZ	@CANNOT READ @
126	100360	124	105	123	EMS364:	.ASCIZ	@TEST PATTERN IN @
127	100401	101	116	104	EMS365:	.ASCIZ	@AND @
128	100406	103	101	116	EMS366:	.ASCIZ	@CANNOT INITIALIZE @
129	100431	124	110	105	EMS367:	.ASCIZ	@THE COMMAND SEQUENCER HAS BEEN CLOCKED @
130	100501	122	105	123	EMS370:	.ASCIZ	@RESET EARLY @
131	100516	104	111	104	EMS371:	.ASCIZ	@DID NOT RESET ON TIME @
132	100545	104	125	122	EMS372:	.ASCIZ	@DURING COMMAND EXECUTION @
133	100577	124	117	040	EMS373:	.ASCIZ	@TO @
134	100603	127	111	124	EMS374:	.ASCIZ	@WITH ANY COMBINATION OF @
135	100634	102	131	040	EMS375:	.ASCIZ	@BY READING @
136	100650	102	131	040	EMS376:	.ASCIZ	@BY WRITING @
137	100664	127	101	123	EMS377:	.ASCIZ	@WAS SET @
138	100675	127	101	123	EMS400:	.ASCIZ	@WAS NOT SET @
139	100712	104	111	104	EMS401:	.ASCIZ	@DID NOT SET @
140	100727	127	110	111	EMS402:	.ASCIZ	@WHILE @
141	100736	103	117	115	EMS403:	.ASCIZ	@COMMAND SEQUENCER DID NOT ABORT @
142	100777	127	101	123	EMS404:	.ASCIZ	@WAS NOT RESET @
143	101016	104	125	122	EMS405:	.ASCIZ	@DURING @
144	101026	127	101	123	EMS406:	.ASCIZ	@WAS RESET @
145	101041	123	105	101	EMS407:	.ASCIZ	@SEARCH TIMEOUT @
146							
147	101061	011	104	105	EMS500:	.ASCII	@ DEVICE IS NON-EXISTENT,@<CR><LF>
148	101113	011	104	105		.ASCII	@ DEVICE IS SWITCHED TO OTHER PORT@<CR><LF>
149	101156	011	124	122		.ASCIZ	@ TRANSCEIVER ENABLE SWITCH IS OFF@<CR><LF>
150	101222	011	111	106	EMS501:	.ASCIZ	@ IF MODULE, M7686,@<CR><LF>
151	101247	011	115	101	EMS502:	.ASCIZ	@ MASSBUS TRANSCEIVER, M5922 OR M5923 @<CR><LF>
152	101317	011	103	123	EMS503:	.ASCIZ	@ CS MODULE, M7684,@<CR><LF>
153	101344	011	104	123	EMS504:	.ASCIZ	@ DS MODULE, M8685/M7685,@<CR><LF>
154	101377	011	104	105	EMS505:	.ASCIZ	@ DEVICE IS SWITCHED TO A/B PORT POSITION@<CR><LF>
155	101452	011	104	105	EMS506:	.ASCIZ	@ DEVICE IS NOT AN RM05/3/2, OR@<CR><LF>
156	101513	011	104	105	EMS507:	.ASCIZ	@ DEVICE IS SWITCHED TO PROGRAMMABLE PORT POSITION, OR@<CR><LF>
157	101603	011	101	123	EMS510:	.ASCIZ	@ ASSUMING THE RH CONTROLLER HAS NO FAULT@<CR><LF>
158	101656	015	012	011	EMS511:	.ASCII	<CR><LF>@ PROBABLE FAULT(S):@<CR><LF>
159	101705	011	050	116		.ASCIZ	@ (NOT INCLUDING CABLES OR CONNECTORS)@<CR><LF>
160							
161	101755	122	105	101	EMS600:	.ASCIZ	@READ IN PRESET COMMAND @
162	102005	117	106	106	EMS601:	.ASCIZ	@OFFSET COMMAND @
163	102025	122	105	124	EMS602:	.ASCIZ	@RETURN TO CENTER CENTER COMMAND @
164	102066	122	105	114	EMS603:	.ASCIZ	@RELEASE COMMAND @
165	102107	122	105	103	EMS604:	.ASCIZ	@RECALIBRATE COMMAND @
166	102134	123	105	105	EMS605:	.ASCIZ	@SEEK COMMAND @
167	102152	123	105	101	EMS606:	.ASCIZ	@SEARCH COMMAND @
168	102172	104	101	124	EMS607:	.ASCIZ	@DATA COMMAND @
169							

1	102210	105	130	120	EH1:	.ASCII	@EXPCTD	RECEVD	REGSTR@<CRLF>
2	102237	123	124	101		.ASCIZ	@STATUS	STATUS	ADRESS@
3	102266	040	102	101	EH2:	.ASCII	@BASE@<CRLF>		
4	102274	101	104	122	EH3:	.ASCIZ	@ADDRESS@		
5	102303	105	130	120	EH5:	.ASCII	@EXPCTD	STUCK@<CRLF>	
6	102322	122	105	123		.ASCIZ	@RESULT	BIT(S)@	
7	102341	105	130	120	EH7:	.ASCII	@EXPCTD	RECEVD@	
8	102360	123	116	107	EH57:	.ASCII	@SNGPRT	DULPRT	RECEVD DRVTYP@<CRLF>
9	102417	104	122	126		.ASCIZ	@DRVTYP	DRVTYP	DRVTYP REGADR@
10	102456	105	130	120	EH65:	.ASCII	@EXPCTD	RECEVD	TEST@<CRLF>
11	102503	123	124	101		.ASCIZ	@STATUS	STATUS	REGSTR@
12	102532	105	130	120	EH71:	.ASCII	@EXPCTD	RECEVD	TEST@<CRLF>
13	102560	123	124	101		.ASCIZ	@STATUS	STATUS	PATTRN@
14	102607	105	130	120	EH115:	.ASCII	@EXPCTD	RECEVD	REGSTR TEST@<CRLF>
15	102645	123	124	101		.ASCIZ	@STATUS	STATUS	ADDRESS PATTRN@
16	102704	105	130	120	EH130:	.ASCII	@EXPCTD	RECEVD	REGSTR TEST OFFSET@<CRLF>
17	102753	123	124	101		.ASCIZ	@STATUS	STATUS	ADDRESS PATTRN REGSTR@
18	103022	105	130	120	EH132:	.ASCII	@EXPCTD	ACTUAL	REGSTR OFFSET@<CRLF>
19	103061	103	117	125		.ASCIZ	@COUNT	COUNT	ADDRESS REGSTR@
20	103120	105	130	120	EH142:	.ASCII	@EXPCTD	RECEVD	REGSTR OFFSET@<CRLF>
21	103157	123	124	101		.ASCIZ	@STATUS	STATUS	ADDRESS REGSTR@
22	103216	105	130	120	EH145:	.ASCII	@EXPCTD	ACTUAL	REGSTR RMER1 RMER2@<CRLF>
23	103264	103	115	120		.ASCIZ	@CMPERR	CMPERR	ADDRESS PATTRN PATTRN@
24									
25	103333	105	130	120	EH150:	.ASCII	@EXPCTD	ACTUAL	REGSTR FUNCTION@<CRLF>
26	103373	122	105	123		.ASCIZ	@RESULT	RESULT	ADDRESS CODE@
27	103431	105	130	120	EH213:	.ASCII	@EXPCTD	ACTUAL	STATUS TEST@<CRLF>
28	103467	122	105	123		.ASCIZ	@RESULT	RESULT	ADDRESS REGSTR@
29	103526	101	103	124	EH220:	.ASCII	@ACTUAL	REGSTR@<CRLF>	
30	103545	122	105	123		.ASCIZ	@RESULT	ADDRESS@	
31									
32					.EVEN				

1	103564	001140	001142	001136	ED1:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,0
2	103574	001136	000000		ED2:	.WORD	\$BDADR,0
3	103600	001140	001142	000000	ED5:	.WORD	\$GDDAT,\$BDDAT,0
4	103606	001174	001176	001142	ED57:	.WORD	\$TMP0,\$TMP1,\$BDDAT,\$BDADR,0
5	103620	001140	001142	001174	ED65:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,0
6	103630	001140	001142	001444	ED71:	.WORD	\$GDDAT,\$BDDAT,\$RMHRO,0
7	103640	001140	001142	001136	ED115:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,\$TMP0,0
8	103652	001140	001142	001136	ED130:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,\$TMP0,\$TMP1,0
9	103666	001142	001136		ED220:	.WORD	\$BDDAT,\$BDADR
10							
11	103672	000	000	000	EF1:	.BYTE	0,0,0
12	103675	000			EF2:	.BYTE	0
13	103676	000	000		EF5:	.BYTE	0,0
14	103700	000	000	000	EF57:	.BYTE	0,0,0,0
15	103704	000	000	000	EF130:	.BYTE	0,0,0,0,0
16							
17					.EVEN		

ERROR MESSAGE STRINGS

58	106131	124	065	067	.ASCII	@T57	RIP/RMOF TEST@<CRLF>
59	106153	124	066	060	.ASCII	@T60	RMDA/RMDC/RIP TEST@<CRLF>
60	106202	124	066	061	.ASCII	@T61	OFFSET COMMAND TEST@<CRLF>
61	106232	124	066	062	.ASCII	@T62	RETURN TO CENTER TEST@<CRLF>
62	106264	124	066	063	.ASCII	@T63	RMDC CLEAR OFFSET TEST@<CRLF>
63	106317	124	066	064	.ASCII	@T64	EBL CLEAR OFFSET TEST@<CRLF>
64	106351	124	066	065	.ASCII	@T65	RUN AND GO TEST@<CRLF>
65	106375	124	066	066	.ASCII	@T66	SET IAE TEST@<CRLF>
66	106416	124	066	067	.ASCII	@T67	SEARCH, SEEK, READ, WRITE TEST@<CRLF>
67	106461	124	067	060	.ASCII	@T70	INVALID TRACK/SECTOR TEST@<CRLF>
68	106517	124	067	061	.ASCII	@T71	INVALID CYLINDER TEST@<CRLF>
69	106551	124	067	062	.ASCII	@T72	SET AOE TEST@<CRLF>
70	106572	124	067	063	.ASCII	@T73	SET RMR TEST@<CRLF>
71	106613	124	067	064	.ASCII	@T74	PGM STATUS CHECK@<CRLF>
72	106640	124	067	065	.ASCII	@T75	DVA/DPR STATUS CHECK@<CRLF>
73	106671	124	067	066	.ASCII	@T76	PORT REQUEST TEST, PART 1@<CRLF>
74	106727	124	067	067	.ASCII	@T77	PORT REQUEST TEST, PART 2@<CRLF>
75	106765	124	061	060	.ASCII	@T100	PORT REQUEST TEST, PART 3@<CRLF>
76	107024	124	061	060	.ASCII	@T101	RELEASE TEST@<CRLF>
77	107046	124	061	060	.ASCII	@T102	WRITE ATA TEST@<CRLF>
78	107072	124	061	060	.ASCII	@T103	RESET ATA BY GO TEST@<CRLF>
79	107124	124	061	060	.ASCII	@T104	UNIT READY ATA TEST@<CRLF>
80	107155	124	061	060	.ASCII	@T105	ERROR ATA TEST@<CRLF>
81	107201	124	061	060	.ASCII	@T106	REGISTER TRANSFER ATA TEST@<CRLF>
82	107241	124	061	060	.ASCII	@T107	P SET ATA TEST@<CRLF>
83	107265	124	061	061	.ASCII	@T110	SET WLE TEST@<CRLF>
84	107307	124	061	061	.ASCII	@T111	EXCEPTION TEST@<CRLF>
85	107333	124	061	061	.ASCII	@T112	RECALIBRATE TEST@<CRLF>
86	107361	124	061	061	.ASCII	@T113	SEEK TEST@<CRLF>
87	107400	124	061	061	.ASCII	@T114	SEARCH TEST@<CRLF>
88	107421	124	061	061	.ASCII	@T115	SEARCH TIMEOUT TEST@<CRLF>
89	107452	124	061	061	.ASCII	@T116	DATA COMMAND TESTS (1)@<CRLF>
90	107506	124	061	061	.ASCII	@T117	DATA COMMAND TESTS (2)@<CRLF>
91	107542	124	061	062	.ASCII	@T120	DATA COMMAND TESTS (3)@<CRLF>
92	107576	200			.ASCII	<CRLF>	
93	107577	117	120	105	.ASCII	@OPERATIONAL SWITCH SETTINGS@<CRLF>	
94	107633	055	055	055	.ASCII	@-----@<CRLF>	
95	107667	123	127	111	.ASCII	@SWITCH	USE@<CRLF>
96	107704	055	055	055	.ASCII	@-----@<CRLF>	
97	107741	040	040	061	.ASCII	@ 15	HALT ON ERROR@<CRLF>
98	107765	040	040	061	.ASCII	@ 14	LOOP ON TEST@<CRLF>
99	110010	040	040	061	.ASCII	@ 13	INHIBIT ERROR TYPEOUTS@<CRLF>
100	110045	040	040	061	.ASCII	@ 12	@<CRLF>
101	110054	040	040	061	.ASCII	@ 11	INHIBIT ITERATIONS@<CRLF>
102	110105	040	040	061	.ASCII	@ 10	BELL ON ERROR@<CRLF>
103	110131	040	040	040	.ASCII	@ 9	LOOP ON ERROR@<CRLF>
104	110155	040	040	040	.ASCII	@ 8	LOOP ON TEST IN SWR<7:0>@<CRLF>
105	110214	040	040	040	.ASCII	@ 7	TN128@<CRLF>
106	110230	040	040	040	.ASCII	@ 6	TN64@<CRLF>
107	110243	040	040	040	.ASCII	@ 5	TN32@<CRLF>
108	110256	040	040	040	.ASCII	@ 4	TN16@<CRLF>
109	110271	040	040	040	.ASCII	@ 3	TN8@<CRLF>
110	110303	040	040	040	.ASCII	@ 2	TN4@<CRLF>
111	110315	040	040	040	.ASCII	@ 1	TN2@<CRLF>
112	110327	040	040	040	.ASCII	@ 0	TN1@<CRLF>
113							
114	000200			.END		200	

ABASE = 176700
 ACDW1 = 000000
 ACDW2 = 000000
 ACPUOP = 000000
 ADDW0 = 000000
 ADDW1 = 000000
 ADDW10 = 000000
 ADDW11 = 000000
 ADDW12 = 000000
 ADDW13 = 000000
 ADDW14 = 000000
 ADDW15 = 000000
 ADDW2 = 000000
 ADDW3 = 000000
 ADDW4 = 000000
 ADDW5 = 000000
 ADDW6 = 000000
 ADDW7 = 000000
 ADDW8 = 000000
 ADDW9 = 000000
 ADEVCT = 000000
 ADEVM = 000000
 ADR = 000001
 AENV = 000000
 AENVM = 000000
 AFATAL = 000000
 ALL = 062134
 AMADR1 = 000000
 AMADR2 = 000000
 AMADR3 = 000000
 AMADR4 = 000000
 AMAMS1 = 000000
 AMAMS2 = 000000
 AMAMS3 = 000000
 AMAMS4 = 000000
 AMSGAD = 000000
 AMGLG = 000000
 AMSGTY = 000000
 AMTYP1 = 000000
 AMTYP2 = 000000
 AMTYP3 = 000000
 AMTYP4 = 000000
 AOE = 001000
 APASS = 000000
 APE = 100000
 APRIOR = 000000
 APTCSU = 000040
 APTENV = 000001
 APTSIZ = 000200
 APTSPO = 000100
 ASWREG = 000000
 ATA = 100000
 ATESTN = 000000
 ATNSK = 000377
 ATNTBL = 063052
 AUNIT = 000000
 AUSWR = 000000

AVECT1 = 120254
 AVECT2 = 000000
 A16 = 000400
 A17 = 001000
 BADTMO = 004546
 BAI = 000010
 BB00 = 000001
 BB01 = 000002
 BB02 = 000004
 BB03 = 000010
 BB04 = 000020
 BB05 = 000040
 BB06 = 000100
 BB07 = 000200
 BB08 = 000400
 BB09 = 001000
 BIT0 = 000001
 BIT00 = 000001
 BIT01 = 000002
 BIT02 = 000004
 BIT03 = 000010
 BIT04 = 000020
 BIT05 = 000040
 BIT06 = 000100
 BIT07 = 000200
 BIT08 = 000400
 BIT09 = 001000
 BIT1 = 000002
 BIT10 = 002000
 BIT11 = 004000
 BIT12 = 010000
 BIT13 = 020000
 BIT14 = 040000
 BIT15 = 100000
 BIT2 = 000004
 BIT3 = 000010
 BIT4 = 000020
 BIT5 = 000040
 BIT6 = 000100
 BIT7 = 000200
 BIT8 = 000400
 BIT9 = 001000
 BOTADR = 054672
 BOTFLG = 054674
 BPTVEC = 000014
 BSE = 100000
 BUFFER = 103712
 BUFONE = 103712
 BUFTWO = 104716
 CC = 004000
 CH = 002000
 CHRCNT = 054675
 CKSWR = 104410
 CLOCK = 001532
 CLR = 000040
 CMNSTA = 007000
 CNSL00 = 062234

CNSL01 = 062267
 CNSL02 = 062305
 CNSL03 = 062347
 CNSL04 = 062367
 CNSL05 = 062423
 CNSL06 = 062435
 CNSL07 = 062466
 CNSL08 = 062631
 CNSL09 = 062632
 CNTCLR = 055614
 COMMA = 062145
 CONT = 000100
 CR = 000015
 CRLF = 000200
 CYLSK = 001777
 DBCK = 100000
 DBEN = 040000
 DBL = 002000
 DCK = 100000
 DDISP = 177570
 DEBL = 020000
 DISPLA = 001156
 DISPRE = 000174
 DLT = 100000
 DMD = 000001
 DPE = 000010
 DPEHI = 040000
 DPELO = 020000
 DPR = 000400
 DRQ = 004000
 DRVCLR = 000010
 DRY = 000200
 DSWR = 177570
 DTE = 010000
 DTO = 010000
 DULPRT = 024024
 DVA = 004000
 DVC = 000200
 EBL = 020000
 ECH = 000100
 ECI = 004000
 ECRC = 001000
 EDT1 = 071466
 EDT115 = 071504
 EDT130 = 071506
 EDT132 = 071510
 EDT2 = 071470
 EDT220 = 071512
 EDT5 = 071472
 EDT57 = 071474
 EDT65 = 071476
 EDT71 = 071500
 EDT74 = 071502
 ED1 = 103564
 ED115 = 103640
 ED130 = 103652
 ED2 = 103574

ED220 = 103666
 ED5 = 103600
 ED57 = 103606
 ED65 = 103620
 ED71 = 103630
 EECC = 000020
 EFT1 = 071514
 EFT115 = 071532
 EFT130 = 071534
 EFT132 = 071536
 EFT2 = 071516
 EFT220 = 071540
 EFT5 = 071520
 EFT57 = 071522
 EFT65 = 071524
 EFT71 = 071526
 EFT74 = 071530
 EF1 = 103672
 EF130 = 103704
 EF2 = 103675
 EF5 = 103676
 EF57 = 103700
 EHT1 = 071366
 EHT115 = 071426
 EHT130 = 071432
 EHT132 = 071436
 EHT142 = 071442
 EHT145 = 071446
 EHT150 = 071452
 EHT2 = 071372
 EHT213 = 071456
 EHT220 = 071462
 EHT5 = 071376
 EHT57 = 071406
 EHT65 = 071412
 EHT7 = 071402
 EHT71 = 071416
 EHT74 = 071422
 EH1 = 102210
 EH115 = 102607
 EH130 = 102704
 EH132 = 103022
 EH142 = 103120
 EH145 = 103216
 EH150 = 103333
 EH2 = 102266
 EH213 = 103431
 EH220 = 103526
 EH3 = 102274
 EH5 = 102303
 EH57 = 102360
 EH65 = 102456
 EH7 = 102341
 EH71 = 102532
 EMS1 = 071542
 EMS10 = 072145
 EMS11 = 072210

EMS12 = 072301
 EMS13 = 072357
 EMS14 = 072426
 EMS15 = 072473
 EMS16 = 072551
 EMS17 = 072632
 EMS2 = 071613
 EMS20 = 072673
 EMS21 = 072737
 EMS22 = 073016
 EMS23 = 073063
 EMS24 = 073141
 EMS25 = 073220
 EMS250 = 076122
 EMS251 = 076160
 EMS252 = 076224
 EMS253 = 076257
 EMS254 = 076312
 EMS255 = 076353
 EMS256 = 076415
 EMS257 = 076445
 EMS26 = 073276
 EMS260 = 076501
 EMS261 = 076532
 EMS262 = 076571
 EMS27 = 073355
 EMS3 = 071662
 EMS30 = 073421
 EMS300 = 076634
 EMS301 = 076652
 EMS302 = 076675
 EMS303 = 076722
 EMS304 = 076733
 EMS306 = 076742
 EMS307 = 076756
 EMS31 = 073501
 EMS310 = 077017
 EMS311 = 077034
 EMS312 = 077100
 EMS313 = 077136
 EMS314 = 077167
 EMS315 = 077216
 EMS316 = 077244
 EMS317 = 077265
 EMS32 = 073552
 EMS320 = 077305
 EMS321 = 077325
 EMS322 = 077345
 EMS323 = 077365
 EMS324 = 077404
 EMS325 = 077424
 EMS326 = 077435
 EMS327 = 077445
 EMS33 = 073632
 EMS330 = 077453
 EMS331 = 077472
 EMS332 = 077522

EMS333	077540	EMS47	074514	EMT116	065676	EMT20	064056	EMT262	070770
EMS334	077554	EMS5	071760	EMT117	065716	EMT200	067300	EMT263	071006
EMS335	077564	EMS50	074554	EMT12	063732	EMT201	067316	EMT264	071024
EMS336	077607	EMS500	101061	EMT120	065736	EMT202	067334	EMT265	071050
EMS337	077626	EMS501	101222	EMT121	065762	EMT203	067352	EMT266	071066
EMS34	073662	EMS502	101247	EMT122	066002	EMT204	067374	EMT267	071104
EMS340	077642	EMS503	101317	EMT123	066022	EMT205	067414	EMT27	064240
EMS341	077653	EMS504	101344	EMT124	066046	EMT206	067434	EMT270	071120
EMS342	077662	EMS505	101377	EMT125	066064	EMT207	067464	EMT271	071144
EMS343	077714	EMS506	101452	EMT126	066104	EMT21	064076	EMT272	071170
EMS344	077730	EMS507	101513	EMT127	066122	EMT210	067500	EMT273	071206
EMS345	077757	EMS51	074623	EMT13	063750	EMT211	067516	EMT274	071224
EMS346	100005	EMS510	101603	EMT130	066146	EMT212	067534	EMT275	071242
EMS347	100023	EMS511	101656	EMT131	066164	EMT213	067546	EMT276	071262
EMS35	073711	EMS52	074672	EMT132	066202	EMT214	067576	EMT277	071300
EMS350	100044	EMS53	074733	EMT133	066222	EMT215	067610	EMT3	063544
EMS351	100051	EMS54	074774	EMT134	066242	EMT216	067630	EMT30	064252
EMS352	100067	EMS55	075051	EMT135	066256	EMT217	067644	EMT300	071324
EMS353	100103	EMS56	075127	EMT136	066274	EMT22	064116	EMT301	071350
EMS354	100127	EMS57	075222	EMT137	066306	EMT220	067660	EMT31	064266
EMS355	100162	EMS6	072030	EMT14	063766	EMT221	067676	EMT32	064302
EMS356	100204	EMS60	075306	EMT140	066324	EMT222	067714	EMT33	064316
EMS357	100224	EMS600	101755	EMT141	066344	EMT223	067732	EMT34	064334
EMS36	073753	EMS601	102005	EMT142	066360	EMT224	067750	EMT35	064352
EMS360	100246	EMS602	102025	EMT143	066376	EMT225	067770	EMT36	064366
EMS361	100274	EMS603	102066	EMT144	066412	EMT226	070012	EMT37	064402
EMS362	100322	EMS604	102107	EMT145	066434	EMT227	070030	EMT4	063564
EMS363	100343	EMS605	102134	EMT146	066456	EMT23	064132	EMT40	064416
EMS364	100360	EMS606	102152	EMT147	066474	EMT230	070046	EMT41	064436
EMS365	100401	EMS607	102172	EMT15	064004	EMT231	070064	EMT42	064452
EMS366	100406	EMS61	075360	EMT150	066506	EMT232	070100	EMT43	064476
EMS367	100431	EMS62	075433	EMT151	066530	EMT233	070122	EMT44	064510
EMS37	074003	EMS63	075514	EMT152	066542	EMT234	070144	EMT45	064524
EMS370	100501	EMS64	075554	EMT153	066556	EMT235	070172	EMT46	064540
EMS371	100516	EMS65	075624	EMT154	066576	EMT236	070212	EMT47	064554
EMS372	100545	EMS66	075665	EMT155	066616	EMT237	070244	EMT5	063606
EMS373	100577	EMS67	075742	EMT156	066634	EMT24	064152	EMT50	064576
EMS374	100603	EMS7	072074	EMT157	066656	EMT240	070264	EMT51	064620
EMS375	100634	EMS70	076054	EMT16	064022	EMT241	070310	EMT52	064640
EMS376	100650	EMTVEC=	000030	EMT160	066674	EMT242	070334	EMT53	064654
EMS377	100664	EMT1	063510	EMT161	066724	EMT243	070360	EMT54	064670
EMS4	071727	EMT10	063676	EMT162	066742	EMT244	070404	EMT55	064710
EMS40	074035	EMT100	065312	EMT163	066760	EMT245	070422	EMT56	064730
EMS400	100675	EMT101	065332	EMT164	067002	EMT246	070440	EMT57	064744
EMS401	100712	EMT102	065352	EMT165	067024	EMT247	070464	EMT6	063632
EMS402	100727	EMT103	065372	EMT166	067050	EMT25	064172	EMT60	064760
EMS403	100736	EMT104	065416	EMT167	067070	EMT250	070502	EMT61	064774
EMS404	100777	EMT105	065436	EMT17	064040	EMT251	070526	EMT62	065014
EMS405	101016	EMT106	065456	EMT170	067112	EMT252	070552	EMT63	065034
EMS406	101026	EMT107	065502	EMT171	067124	EMT253	070576	EMT64	065050
EMS407	101041	EMT11	063714	EMT172	067140	EMT254	070614	EMT65	065062
EMS41	074113	EMT110	065522	EMT173	067156	EMT255	070632	EMT66	065076
EMS42	074161	EMT111	065544	EMT174	067200	EMT256	070656	EMT67	065112
EMS43	074243	EMT112	065564	EMT175	067226	EMT257	070674	EMT7	063654
EMS44	074314	EMT113	065604	EMT176	067244	EMT26	064206	EMT70	065130
EMS45	074355	EMT114	065630	EMT177	067262	EMT260	070720	EMT71	065150
EMS46	074432	EMT115	065652	EMT2	063516	EMT261	070744	EMT72	065174

EMT73 065220
 EMT74 065240
 EMT75 065250
 EMT76 065262
 EMT77 065276
 ENRGDT 063510
 EQUALS 062132
 ERR = 040000
 ERRNMB 054670
 ERROR = 104000
 ERRTP 054046
 ERRVEC= 000004
 ERTY00 054676
 ERTY01 054704
 ERTY02 054714
 ERTY03 054723
 ERTY04 054731
 ERTY05 054734
 ESRC = 004000
 FER = 000020
 FMT16 = 010000
 FNCDTB 062752
 FNCMSK= 000077
 FO = 000002
 F1 = 000004
 F2 = 000010
 F3 = 000020
 F4 = 000040
 GETBUF 001332
 GO = 000001
 GTSWR = 104407
 HCE = 000200
 HCI = 002000
 HCRC = 000400
 HELP 103712
 HT = 000011
 IAE = 002000
 IDXMSK= 000077
 IE = 000100
 ILF = 000001
 ILF02 = 000002
 ILF24 = 000024
 ILF26 = 000026
 ILF30 = 000030
 ILF32 = 000032
 ILF34 = 000034
 ILF36 = 000036
 ILF40 = 000040
 ILF42 = 000042
 ILF44 = 000044
 ILF46 = 000046
 ILF54 = 000054
 ILF56 = 000056
 ILF64 = 000064
 ILF66 = 000066
 ILF74 = 000074
 ILF76 = 000076

ILR = 000002
 ILRG50= 000050
 ILRG52= 000052
 ILRG54= 000054
 ILRG56= 000056
 ILRG60= 000060
 ILRG62= 000062
 ILRG64= 000064
 ILRG66= 000066
 ILRG70= 000070
 ILRG72= 000072
 ILRG74= 000074
 ILRG76= 000076
 IOTVEC= 000020
 IPCK0 = 000001
 IPCK1 = 000002
 IPCK2 = 000004
 IPCK3 = 000010
 IR = 000100
 IVC = 010000
 LBC = 002000
 LBT = 002000
 LCLOCK 055236
 LCOUNT 055306
 LF = 000012
 LODEV 062675
 LS = 000004
 LSC = 004000
 LST = 000002
 LSTOP 055350
 LSTRK 001330
 MCLK = 004000
 MCPE = 020000
 MDF = 000100
 MDPE = 000400
 MI = 000004
 MIXED 063062
 MOC = 000400
 MOH = 020000
 MUL = 010000
 MRD = 002000
 MR1AAA= 051401
 MS = 000040
 MSC = 000002
 MSDRVS 062653
 MSE = 100000
 MSER = 000200
 MSGDRV 062667
 MSHELP 062150
 MUR = 001000
 MWD = 000010
 MWP = 000010
 MXF = 001000
 NDTMSK= 115760
 NED = 010000
 NEM = 004000
 NOP = 000000

NOTAVL 062732
 NOTPRS 062715
 NSA = 100000
 OCC = 100000
 OFD = 000200
 OFFSET= 000014
 OM = 000001
 ONES 063122
 OPE = 020000
 OPI = 020000
 OR = 000200
 PACACK= 000022
 PAKACK= 000022
 PAR = 000010
 PAT = 000020
 PCLOCK 055220
 PCOUNT 055306
 PDA = 000400
 PGE = 002000
 PGM = 001000
 PHA = 000200
 PIP = 020000
 PIRQ = 177772
 PIRQVE= 000240
 PLCLK 055244
 PLFS = 002000
 PLSTP 055354
 PRO = 000000
 PR1 = 000040
 PR2 = 000100
 PR3 = 000140
 PR4 = 000200
 PR5 = 000240
 PR6 = 000300
 PR7 = 000340
 PS = 177776
 PSEL = 002000
 PSTOP 055342
 PSW = 177776
 PUTBUF 001406
 PWRVEC= 000024
 QUES 062141
 RD = 000070
 RDCHR = 104411
 RDLIN = 104412
 RDOCT = 104413
 RDY = 000200
 READY 007152
 RECAL = 000006
 RESPEG= 104415
 RESVEC= 000010
 REX = 010000
 RG = 040000
 RGDTP 063062
 RH = 000072
 RIP = 000020
 RELEASE= 000012

RMAS = 000016
 RMASI 001350
 RMASO 001424
 RMBA = 000004
 RMBAE = 000050
 RMBAEI 001402
 RMBAEO 001456
 RMBAI 001336
 RMBAO 001412
 RMCS1 = 000000
 RMCS1I 001332
 RMCS10 001406
 RMCS2 = 000010
 RMCS2I 001342
 RMCS20 001416
 RMCS3 = 000052
 RMCS3I 001404
 RMCS30 001460
 RMDA = 000006
 RMDAI 001340
 RMDAO 001414
 RMDB = 000022
 RMDBI 001354
 RMDBO 001430
 RMDC = 000034
 RMDCI 001366
 RMDCO 001442
 RMDS = 000012
 RMDSI 001344
 RMDSO 001420
 RMDT = 000026
 RMDTI 001360
 RMDTO 001434
 RMEC1 = 000044
 RMEC1I 001376
 RMEC10 001452
 RMEC2 = 000046
 RMEC2I 001400
 RMEC20 001454
 RMER1 = 000014
 RMER1I 001346
 RMER10 001422
 RMER2 = 000042
 RMER2I 001374
 RMER20 001450
 RMHR = 000036
 RMHRI 001370
 RMHRO 001444
 RMLA = 000020
 RMLAI 001352
 RMLAO 001426
 RMMR1 = 000024
 RMMR1I 001356
 RMMR10 001432
 RMMR2 = 000040
 RMMR2I 001372
 RMMR20 001446

RMOF = 000032
 RMOFI 001364
 RMOFO 001440
 RMR = 000004
 RMSN = 000030
 RMSNI 001362
 RMSNO 001436
 RMWC = 000002
 RMWCI 001334
 RMWCO 001410
 RQA = 100000
 RQB = 040000
 RTC = 000016
 R6 = %000006
 R7 = %000007
 SADMSK= 000377
 SAVREG= 104414
 SA1 = 000001
 SA16 = 000020
 SA2 = 000002
 SA4 = 000004
 SA8 = 000010
 SC = 100000
 SCOPE = 000004
 SCIMSK= 003700
 SCO = 000100
 SC1 = 000200
 SC2 = 000400
 SC3 = 001000
 SC4 = 002000
 SEARCH= 000030
 SEEK = 000004
 SETOM 055512
 SETVV 055370
 SHUT 055642
 SIZCLK 054760
 SKI = 040000
 SNGPRT= 020024
 STACK = 001100
 STANDA 006074
 START 004626
 STKLMT= 177774
 STOP 001534
 SWR 001154
 SWREG 000176
 SWO = 000001
 SWO0 = 000001
 SWO1 = 000002
 SWO2 = 000004
 SWO3 = 000010
 SWO4 = 000020
 SWO5 = 000040
 SWO6 = 000100
 SWO7 = 000200
 SWO8 = 000400
 SWO9 = 001000
 SW1 = 000002

SW10 = 002000	TST16 014672	TYPDS = 104405	SDDW1 001310	SMAMS1 001252
SW11 = 004000	TST17 015404	TYPE = 104401	SDDW2 001312	SMAMS2 001256
SW12 = 010000	TST2 007616	TYPOC = 104402	SDDW3 001314	SMAMS3 001262
SW13 = 020000	TST20 015506	TYPON = 104404	SDDW4 001316	SMAMS4 001266
SW14 = 040000	TST21 015772	TYPOS = 104403	SDDW5 001320	SMBADR 001102
SW15 = 100000	TST22 016110	UBUSQS 062201	SDDW6 001322	SMFLG 062126
SW2 = 000004	TST23 016236	UNS = 040000	SDDW7 001324	SMNEW 061264
SW3 = 000010	TST24 016510	UNTMSK= 000007	SDEVCT 001232	SMSGAD 001236
SW4 = 000020	TST25 017010	UPE = 020000	SDEV M 001300	SMSGLG 001240
SW5 = 000040	TST26 017432	USE = 040000	SDOAGN 054036	SMSGTY 001222
SW6 = 000100	TST27 017526	UJ = 000001	SDTBL 056264	SMSWR 061253
SW7 = 000200	TST3 010000	U1 = 000002	SENDAD 054026	SMTYP1 001253
SW8 = 000400	TST30 020014	U2 = 000004	SENDCT 053672	SMTYP2 001257
SW9 = 001000	TST31 020330	VV = 000100	SENULL 054042	SMTYP3 001263
TADMSK= 177400	TST32 020620	WATCH 001530	SENV 001242	SMTYP4 001267
TAG = 020000	TST33 021336	WC = 000040	SENV M 001243	SMXCNT 057376
TAGADR= 001114	TST34 021650	WCD = 000050	SEOP 053636	SNLL 001170
TAP = 040000	TST35 022142	WCE = 040000	SEOPCT 053664	SNWTST= 000001
TA1 = 000400	TST36 022464	WCEHI = 010000	SEOSP 053600	SOCNT 056526
TA16 = 010000	TST37 023024	WCELO = 004000	SERFLG 001117	SOMODE 056530
TA2 = 001000	TST4 010150	WCF = 000040	SERMAX 001131	SOVER 057362
TA4 = 002000	TST40 023500	WCH = 000052	SERROR 057640	SPASS 001230
TAB = 004000	TST41 023764	WD = 000060	SERRPC 001132	SPASTM 001106
TBITVE= 000014	TST42 024246	WH = 000062	SERRTB 001536	SPOWER 061654
TIME 001526	TST43 024650	WLE = 004000	SERTTL 001126	SPSW 001524
TKVEC = 000060	TST44 025020	WRL = 004000	SESCAP 001210	SPWRDN 061506
TPVEC = 000064	TST45 025400	XNUDC = 176000	SETABL 001242	SPWRMG 061642
TRAPVE= 000034	TST46 026110	XNUER2= 001567	SETEND 001326	SPWRUP 061560
TRE = 040000	TST47 026404	XNUOF = 161577	SFATAL 001224	SQUES 001216
TRTVEC= 000014	TST5 010300	XSIZ 005666	SFFLG 062130	SRDCHR 060700
TST = 010000	TST50 027006	XXDP 001326	SFILLC 001172	SRDLIN 060770
TSTNMB 054666	TST51 027242	ZEROS 063164	SFILLS 001171	SRDOCT 061276
TSTQUE 001462	TST52 027450	\$APTHD 001100	\$GDADR 001134	SRDSZ = 000010
TST1 007306	TST53 030302	\$ATYC 061710	\$GDDAT 001140	\$RESRE 055746
TST10 012572	TST54 030554	\$ATY1 061664	\$GET42 054016	\$RMO2 054740
TST100 035462	TST55 030756	\$ATY3 061672	\$GTSWR 060426	\$RMO3 054745
TST101 035614	TST56 031170	\$ATY4 061702	\$HD = 000000	\$RMO5 054752
TST102 036004	TST57 031364	\$AUTOB 001150	\$HIBTS 001100	\$RTNAD 054040
TST103 036206	TST6 011364	\$BASE 001276	\$HIOCT 061376	\$SAVRE 055710
TST104 036334	TST60 031544	\$BDADR 001136	\$ICNT 001120	\$SAVR6 061652
TST105 036540	TST61 031762	\$BDDAT 001142	\$ILLUP 061646	\$SCOPE 057066
TST106 036670	TST62 032130	\$BELL 001212	\$INTAG 001151	\$SETUP= 000137
TST107 037064	TST63 032354	\$BIN 056056	\$ITEMB 001130	\$STUP = 177777
TST11 012656	TST64 032470	\$CDW1 001302	\$LF 001220	\$SVLAD 057326
TST110 037262	TST65 032656	\$CDW2 001304	\$LFLG 062127	\$SVPC = 000204
TST111 037540	TST66 033106	\$CHARC 057062	\$LLCSR 001516	\$SWR = 167400
TST112 040044	TST67 033324	\$CKSWR 060336	\$LLVEC 001520	\$SWREG 001244
TST113 041436	TST7 012434	\$CMTAG 001114	\$LPADR 001122	\$SWRMK= 000000
TST114 043164	TST70 033552	\$CM3 = 000000	\$LPCSB 001510	\$SWOBT 057400
TST115 045344	TST71 034022	\$CM4 = 000005	\$LPCSR 001506	\$TESTN 001226
TST116 045724	TST72 034230	\$CNTLC 061234	\$LPERR 001124	\$TIMES 001206
TST117 050104	TST73 034520	\$CNTLG 061246	\$LPVEC 001512	\$TKB 001162
TST12 013116	TST74 034746	\$CNTLU 061241	\$MADR1 001254	\$TKCNT 060040
TST120 052372	TST75 035054	\$CPUOP 001250	\$MADR2 001260	\$TKINT 060050
TST13 013464	TST76 035200	\$CRLF 001217	\$MADR3 001264	\$TKQEN= 060047
TST14 014246	TST77 035330	\$DBLK 056274	\$MADR4 001270	\$TKQIN 060042
TST15 014346	TYPBN = 104406	\$DDW0 001306	\$MAIL 001222	\$TKQOU 060044

SYMBOL TABLE

\$TKQSR	060046	\$TN	= 000121	\$TSTM	001104	\$TYPOC	056330	\$XOFF	= 000023
\$TKS	001160	\$TPB	001166	\$TSTNM	001116	\$TYPON	056344	\$XON	= 000021
\$TKSRV	060120	\$TPFLG	001173	\$TITYIN	061224	\$TYPOS	056304	\$XTSTR	057100
\$TMP0	001174	\$TPS	001164	\$TYPBN	056004	\$UNIT	001234	\$\$GET4=	000000
\$TMP1	001176	\$TRAP	061400	\$TYPDS	056060	\$UNITM	001110	\$\$SW08=	000121
\$TMP2	001200	\$TRAP2	061440	\$TYPE	056532	\$USWR	001246	\$OFILL	056527
\$TMP3	001202	\$TRP	= 000016	\$TYPEC	056744	\$VECT1	001272	\$.X	= 001100
\$TMP4	001204	\$TRPAD	061452	\$TYPEX	057064	\$VECT2	001274		

. ABS. 110342 000
 000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 55280 WORDS (216 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 69 PAGES
CZRMPA.BIC,CZRMPA/C=CZRMPA.DOC,CZRMPA,SYSMAC/M

SDDW6

5-0A

F 6

SEQ 0276

SLF

5-0#

19-5

19-5

19-7

19-7

19-8

19-8

H

6

19-8

SEQ 0278

\$LFLG	19-12#	19-12*													
\$LLCSR	6-0#	15-26	15-55*	15-83*											
\$LLVEC	6-0#	15-17	15-17*	15-18*	15-29*	15-30*	15-42	15-42*	15-43*						
\$LPADR	5-0#	8-19*	19-6	19-6	19-6	19-6*	19-6*	19-6*							
\$LPCSB	6-0#	15-51*													
\$LPCSR	6-0#	15-12	15-52*	15-80*											
\$LPERR	5-0#	8-19*	12-Y83*	12-Z21*	12-Z52*	12-Z94*	12-[51*	12-\50*	12-\91*	12-J25*	12-J70*	12-^30*	12- 41*	12- 85*	
	12-19*	12-81*	12-a41*	12-b12*	12-d14*	12-e06*	12-e65*	12-f37*	12-h20*	12-i00*	12-i64*	12-j54*	19-8	19-8	
	19-6	19-6*	19-7												
\$LPVEC	6-0#	15-15*	15-16*	15-31	15-31*	15-32*	15-40	15-40*	15-41*						
\$MADR1	5-0#														
\$MADR2	5-0#														
\$MADR3	5-0#														
\$MADR4	5-0#														
\$MAIL	4-991	4-991	5-0#	8-19	8-25	10-27	12-1	12-73	12-106	12-137	12-159	12-329	12-521	12-548	
	12-559	12-612	12-679	12-783	12-796	12-866	12-975	12-998	12-:51	12-:79	12-:99	12-;43	12-<05	12-=00	
	12-=17	12-=75	12->40	12-?01	12-a03	12-a64	12-A24	12-B06	12-B74	12-C76	12-D29	12-D82	12-E62	12-E97	
	12-G18	12-H68	12-I43	12-J82	12-K28	12-K62	12-L94	12-M54	12-M90	12-N38	12-N77	12-007	12-047	12-073	
	12-P17	12-P38	12-P65	12-Q09	12-Q86	12-R34	12-R84	12-S21	12-S71	12-T28	12-T46	12-T69	12-T98	12-U27	
	12-U57	12-U94	12-V32	12-V52	12-V85	12-W06	12-W55	12-X00	12-X70	12-Y47	12-\09	12-_00	12-c06	12-c74	
	12-g76	12-k50	19-5	19-6	19-7										
\$MAMS1	5-0#														
\$MAMS2	5-0#														
\$MAMS3	5-0#														
\$MAMS4	5-0#														
\$MBADR	4-991#														
\$MFLG	19-12	19-12#	19-12*	19-12*											
\$MNEW	19-8	19-8#													
\$MSGAD	5-0#	19-12	19-12*												
\$MSGLG	5-0#	19-12*													
\$MSGTY	5-0#	19-12	19-12	19-12*	19-12*										
\$MSWR	19-8	19-8#													
\$MTYP1	5-0#														
\$MTYP2	5-0#														
\$MTYP3	5-0#														
\$MTYP4	5-0#														
\$MXCNT	19-6	19-6	19-6	19-6#											
\$NULL	5-0#	19-5	19-5	19-5											
\$NWTST	12-1	12-1	12-1#	12-1#	12-73	12-73	12-73#	12-73#	12-106	12-106	12-106#	12-106#	12-137	12-137	
	12-137#	12-137#	12-159	12-159	12-159#	12-159#	12-329	12-329	12-329#	12-329#	12-521	12-521	12-521#	12-521#	
	12-548	12-548	12-548#	12-548#	12-559	12-559	12-559#	12-559#	12-612	12-612	12-612#	12-612#	12-679	12-679	
	12-679#	12-679#	12-783	12-783	12-783#	12-783#	12-796	12-796	12-796#	12-796#	12-866	12-866	12-866#	12-866#	
	12-975	12-975	12-975#	12-975#	12-998	12-998	12-998#	12-998#	12-:51	12-:51	12-:51#	12-:51#	12-:79	12-:79	
	12-:79#	12-:79#	12-:99	12-:99	12-:99#	12-:99#	12-:43	12-:43	12-:43#	12-:43#	12-<05	12-<05	12-<05#	12-<05#	
	12-=00	12-=00	12-=00#	12-=00#	12-=17	12-=17	12-=17#	12-=17#	12-=75	12-=75	12-=75#	12-=75#	12->40	12->40	
	12->40#	12->40#	12-?01	12-?01	12-?01#	12-?01#	12-a03	12-a03	12-a03#	12-a03#	12-a64	12-a64	12-a64#	12-a64#	
	12-A24	12-A24	12-A24#	12-A24#	12-B06	12-B06	12-B06#	12-B06#	12-B74	12-B74	12-B74#	12-B74#	12-C76	12-C76	
	12-C76#	12-C76#	12-D29	12-D29	12-D29#	12-D29#	12-D82	12-D82	12-D82#	12-D82#	12-E62	12-E62	12-E62#	12-E62#	
	12-E97	12-E97	12-E97#	12-E97#	12-G18	12-G18	12-G18#	12-G18#	12-H68	12-H68	12-H68#	12-H68#	12-I43	12-I43	
	12-I43#	12-I43#	12-J82	12-J82	12-J82#	12-J82#	12-K28	12-K28	12-K28#	12-K28#	12-K62	12-K62	12-K62#	12-K62#	
	12-L94	12-L94	12-L94#	12-L94#	12-M54	12-M54	12-M54#	12-M54#	12-M90	12-M90	12-M90#	12-M90#	12-N38	12-N38	
	12-N38#	12-N38#	12-N77	12-N77	12-N77#	12-N77#	12-007	12-007	12-007#	12-007#	12-047	12-047	12-047#	12-047#	
	12-073	12-073	12-073#	12-073#	12-P17	12-P17	12-P17#	12-P17#	12-P38	12-P38	12-P38#	12-P38#	12-P65	12-P65	
	12-P65#	12-P65#	12-Q09	12-Q09	12-Q09#	12-Q09#	12-Q86	12-Q86	12-Q86#	12-Q86#	12-R34	12-R34	12-R34#	12-R34#	
	12-R84	12-R84	12-R84#	12-R84#	12-S21	12-S21	12-S21#	12-S21#	12-S71	12-S71	12-S71#	12-S71#	12-T28	12-T28	
	12-T28#	12-T28#	12-T46	12-T46	12-T46#	12-T46#	12-T69	12-T69	12-T69#	12-T69#	12-T98	12-T98	12-T98#	12-T98#	

12-U27 12-U27 12-U27# 12-U27# 12-U57 12-U57 12-U57^J#⁶ 12-U57# 12-U94 12-U94 12-U94# 12-U94# 12-V32 12-V32
SEQ 0280

	12-V32#	12-V32#	12-V52	12-V52	12-V52#	12-V52#	12-V85	12-V85	12-V85#	12-V85#	12-W06	12-W06	12-W06#	12-W06#
	12-W55	12-W55	12-W55#	12-W55#	12-X00	12-X00	12-X00#	12-X00#	12-X70	12-X70	12-X70#	12-X70#	12-Y47	12-Y47
	12-Y47#	12-Y47#	12-\09	12-\09	12-\09#	12-\09#	12-\00	12-\00	12-\00#	12-\00#	12-c06	12-c06	12-c06#	12-c06#
	12-c74	12-c74	12-c74#	12-c74#	12-g76	12-g76	12-g76#	12-g76#	12-K50	12-K50	12-k50#	12-k50#		
\$SOCNT	19-4#	19-4*	19-4*											
\$SOMODE	19-4	19-4#	19-4*	19-4*	19-4*	19-4*								
\$SOVER	19-6	19-6	19-6	19-6	19-6#									
\$SPASS	5-0#	8-19*	13-20	13-20	13-20	13-20*	13-20*	19-6	19-6	19-6				
\$SPASTM	4-991#													
\$SPOWER	19-11	19-11#												
\$SPSW	6-0#	15-59*	15-86											
\$SPWRDN	8-19	19-11	19-11#											
\$SPWRMG	19-11#													
\$SPWRUP	19-11	19-11#												
\$SQUES	5-0#	19-5	19-5	19-7	19-7	19-8	19-8	19-8	19-8					
\$SR2A	19-10													
\$SRDCHR	19-8#	19-10	19-10											
\$SRDEC	19-10													
\$SRDLIN	19-8#	19-10	19-10											
\$SRDOCT	19-9#	19-10	19-10											
\$SRDSZ	19-8	19-8#												
\$SRESRE	19-1#	19-10												
\$SRM02	14-31	14-164#												
\$SRM03	14-27	14-165#												
\$SRM05	14-35	14-166#												
\$SRTNAD	13-20#													
\$SAVR6	19-11	19-11#	19-11*	19-11*	19-11*									
\$SAVRE	19-1#	19-10	19-10											
\$SCOPE	8-19	19-6#												
\$SETUP	4-984	4-984	4-984	4-984	4-984	4-984	4-984#	4-984#	4-984#	4-984#	4-984#	4-984#	4-984#	8-19
	8-19	8-19	8-19	8-19	8-19	8-19	8-19	8-19	8-19	8-19	8-19	8-25	8-25	8-25
	10-27	13-20	13-20	19-6	19-7	19-7	19-7	19-7	19-8	19-8	19-8	19-8	19-8	8-25
\$STUP	4-984	4-984	4-984	4-984	4-984	4-984	4-984#	4-984#	4-984#	4-984#	4-984#	4-984#	4-984#	4-984#
	4-984#	4-984#	4-984#	4-984#										
\$SVLAD	19-6	19-6#												
\$SVPC	4-988	4-988#												
\$SWOBT	19-6	19-6#												
\$SWR	4-673#	4-684	4-685	4-685	4-685	4-685	4-685	4-685	4-685	4-685	5-0	5-0	5-0	8-19
	8-19	8-19	8-19	8-19	12-1	12-73	12-106	12-137	12-159	12-329	12-521	12-548	12-559	12-612
	12-679	12-783	12-796	12-866	12-975	12-998	12-:51	12-:79	12-:99	12-:43	12-<05	12-=00	12-=17	12-=75
	12->40	12-?01	12-a03	12-a64	12-A24	12-B06	12-B74	12-C76	12-D29	12-D82	12-E62	12-E97	12-G18	12-H68
	12-I43	12-J82	12-K28	12-K62	12-L94	12-M54	12-M90	12-N38	12-N77	12-O07	12-O47	12-O73	12-P17	12-P38
	12-P65	12-Q09	12-Q86	12-R34	12-R84	12-S21	12-S71	12-T28	12-T46	12-T69	12-T98	12-U27	12-U57	12-U94
	12-V32	12-V52	12-V85	12-W06	12-W55	12-X00	12-X70	12-Y47	12-\09	12-\00	12-c06	12-c74	12-g76	12-k50
	13-20	13-20	13-20	13-20	13-20	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6
	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6
	19-7	19-7	19-7	19-7	19-7	19-7	19-7	19-7	19-7	19-11				
\$SWREG	5-0#	8-19												
\$SWRMK	4-685	4-685	4-685	4-685	4-685	4-685	4-685	4-685	4-685	4-685	19-6	19-6	19-6	19-6
	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6	19-6
\$TESTN	5-0#	12-1*	12-73*	12-106*	12-137*	12-159*	12-329*	12-521*	12-548*	12-559*	12-612*	12-679*	12-783*	12-796*
	12-866*	12-975*	12-998*	12-:51*	12-:79*	12-:99*	12-:43*	12-<05*	12-=00*	12-=17*	12-=75*	12->40*	12-?01*	12-a03*
	12-a64*	12-A24*	12-B06*	12-B74*	12-C76*	12-D29*	12-D82*	12-E62*	12-E97*	12-G18*	12-H68*	12-I43*	12-J82*	12-K28*
	12-K62*	12-L94*	12-M54*	12-M90*	12-N38*	12-N77*	12-O07*	12-O47*	12-O73*	12-P17*	12-P38*	12-P65*	12-Q09*	12-Q86*
	12-R34*	12-R84*	12-S21*	12-S71*	12-T28*	12-T46*	12-T69*	12-T98*	12-U27*	12-U57*	12-U94*	12-V32*	12-V52*	12-V85*
	12-W06*	12-W55*	12-X00*	12-X70*	12-Y47*	12-\09*	12-\00*	12-c06*	12-c74*	12-g76*	12-k50*	14-44	19-6*	

\$TIMES 5-0# 8-19* 10-38* 13-20* 19-6 19-6 19-6^L 6 19-6* 19-6*

SEQ 0282

STSTNM	5-0#	10-37*	13-20*	19-6	19-6	19-6	19-6	19-6 ^{N 6}	19-6	19-6*	19-6*	19-7	19-7	19-7	SEQ 0284
--------	------	--------	--------	------	------	------	------	---------------------	------	-------	-------	------	------	------	----------

\$TTYIN	19-8	19-8	19-8	19-8	19-8	19-8#								
\$TYPBN	19-2#	19-10	19-10											
\$TYPDS	19-3#	19-10	19-10											
\$TYPE	19-5#	19-10	19-10	19-12										
\$TYPEC	19-5	19-5	19-5	19-5#	19-8									
\$TYPEX	19-5	19-5	19-5#											
\$TYPOC	19-4#	19-10	19-10											
\$TYPON	19-4	19-4#	19-10											
\$TYPOS	19-4#	19-10												
\$UNIT	5-0#	10-41*	14-23											
\$UNITM	4-991#													
\$USWR	5-0#													
\$VECT1	5-0#	9-69	9-81*	9-85	9-107*									
\$VECT2	5-0#													
\$XOFF	19-5	19-5												
\$XON	19-5	19-5												
\$XTSTR	19-6#													
.\$ASTA	19-12	19-12												
.\$X	4-991	4-991#												
A16	4-930#	12-<83												
A17	4-929#	12-<83												
ABASE	4-981#	5-0	5-0											
ACDW1	5-0	5-0												
ACDW2	5-0	5-0												
ACPUOP	5-0	5-0												
ADDW0	5-0	5-0												
ADDW1	5-0	5-0												
ADDW10	5-0													
ADDW11	5-0													
ADDW12	5-0													
ADDW13	5-0													
ADDW14	5-0													
ADDW15	5-0													
ADDW2	5-0	5-0												
ADDW3	5-0	5-0												
ADDW4	5-0	5-0												
ADDW5	5-0	5-0												
ADDW6	5-0	5-0												
ADDW7	5-0	5-0												
ADDW8	5-0													
ADDW9	5-0													
ADEVCT	5-0	5-0												
ADEVMT	5-0	5-0												
ADR	12-1	12-1#	12-1#	12-73	12-73#	12-73#	12-106	12-106#	12-106#	12-137	12-137#	12-137#	12-159	12-159#
	12-159#	12-329	12-329#	12-329#	12-521	12-521#	12-521#	12-548	12-548#	12-548#	12-559	12-559#	12-559#	12-612
	12-612#	12-612#	12-679	12-679#	12-679#	12-783	12-783#	12-783#	12-796	12-796#	12-796#	12-866	12-866#	12-866#
	12-975	12-975#	12-975#	12-998	12-998#	12-998#	12-:51	12-:51#	12-:51#	12-:79	12-:79#	12-:79#	12-:99	12-:99#
	12-:99#	12-:43	12-:43#	12-:43#	12-<05	12-<05#	12-<05#	12-=00	12-=00#	12-=00#	12-=17	12-=17#	12-=17#	12-=75
	12-=75#	12-=75#	12->40	12->40#	12->40#	12-?01	12-?01#	12-?01#	12-a03	12-a03#	12-a03#	12-a64	12-a64#	12-a64#
	12-A24	12-A24#	12-A24#	12-B06	12-B06#	12-B06#	12-B74	12-B74#	12-B74#	12-C76	12-C76#	12-C76#	12-D29	12-D29#
	12-D29#	12-D82	12-D82#	12-D82#	12-E62	12-E62#	12-E62#	12-E97	12-E97#	12-E97#	12-G18	12-G18#	12-G18#	12-H68
	12-H68#	12-H68#	12-I43	12-I43#	12-I43#	12-J82	12-J82#	12-J82#	12-K28	12-K28#	12-K28#	12-K62	12-K62#	12-K62#
	12-L94	12-L94#	12-L94#	12-M54	12-M54#	12-M54#	12-M90	12-M90#	12-M90#	12-N38	12-N38#	12-N38#	12-N77	12-N77#
	12-N77#	12-007	12-007#	12-007#	12-047	12-047#	12-047#	12-073	12-073#	12-073#	12-P17	12-P17#	12-P17#	12-P38
	12-P38#	12-P38#	12-P65	12-P65#	12-P65#	12-P65#	12-Q09	12-Q09#	12-Q09#	12-Q86	12-Q86#	12-Q86#	12-R34	12-R34#
	12-R84	12-R84#	12-R84#	12-S21	12-S21#	12-S21#	12-S71	12-S71#	12-S71#	12-T28	12-T28#	12-T28#	12-T46	12-T46#

	12-U94#	12-U94#	12-V32	12-V32#	12-V32#	12-V52	12-V52#	12-V52#	12-V85	12-V85#	12-V85#	12-W06	12-W06#	12-W06#
	12-W55	12-W55#	12-W55#	12-X00	12-X00#	12-X00#	12-X70	12-X70#	12-X70#	12-Y47	12-Y47#	12-Y47#	12-Y09	12-Y09#
	12-109#	12-00	12-00#	12-00#	12-c06	12-c06#	12-c06#	12-c74	12-c74#	12-c74#	12-g76	12-g76#	12-g76#	12-k50
	12-k50#	12-k50#												
AENV	5-0	5-0												
AENVM	5-0	5-0												
AFATAL	5-0	5-0												
ALL	9-121	20-4#												
AMADR1	5-0	5-0												
AMADR2	5-0	5-0												
AMADR3	5-0	5-0												
AMADR4	5-0	5-0												
AMAMS1	5-0	5-0												
AMAMS2	5-0	5-0												
AMAMS3	5-0	5-0												
AMAMS4	5-0	5-0												
AMSGAD	5-0	5-0												
AMSGLG	5-0	5-0												
AMSGTY	5-0	5-0												
AMTYP1	5-0	5-0												
AMTYP2	5-0	5-0												
AMTYP3	5-0	5-0												
AMTYP4	5-0	5-0												
AOE	4-776#	4-787	12-697	12-726	12-N18	12-S56	12-S60	21-77	21-78	21-81	21-82	21-85	21-86	
APASS	5-0	5-0												
APE	4-959#													
APRIOR	5-0													
APTCSU	19-5	19-12#												
APTENV	19-5	19-7	19-12	19-12#										
APTSIZ	8-19	19-12#												
APTSP0	19-5	19-12	19-12#											
ASWREG	5-0	5-0												
ATA	4-756#	12-V12	12-V43	12-V64	12-V68	12-V76	12-V80	12-V97	12-V99	12-W22	12-W45	12-W77	12-W81	12-Z87
	12-289	12-J63	12-J65	12-74	12-76	21-58	21-59	21-60	21-63	21-64	21-67	21-68	21-69	21-70
	21-71	21-72	21-73	21-74	21-75	21-76	21-79	21-80	21-83	21-84	21-87	21-88		
ATESTN	5-0	5-0												
ATNMSK	4-793#	12-V07												
ATNTBL	8-74	8-93	9-149	10-11	12-V06	22-3#								
AUNIT	5-0	5-0												
AUSWR	5-0	5-0												
AVECT1	4-982#	5-0	5-0											
AVECT2	5-0	5-0												
BADTMO	8-3#	8-21												
BAI	4-948#													
BB00	4-877#													
BB01	4-877#													
BB02	4-877#	12-L82												
BB03	4-877#	12-L78	12-L81											
BB04	4-877#													
BB05	4-877#													
BB06	4-877#	12-[93	12-[94	12-[95	12-[96	12-[97	12-[98	12-[99	12-100	12-101	12-102			
BB07	4-877#													
BB08	4-877#													
BB09	4-877#													
BIT0	4-688#	12-211	12-529	12-531	12-534	12-536	12-577	12-641	12-724	12-732	12-813	12-821	12-834	12-887
	12-895	12-903	12-916	12-924	12-936	12-;14	12-;57	12-;67	12-;40	12-;89	12->54	12-?18	12-@18	12-@79

DDISP 4-688# 5-0 8-19

1 7

SEQ 0292

ED2

26-2

26-7

30-2#

K 7

SEQ 0294



EH132 25-11 29-18#

M 7

SEQ 0296

EMS343 24-135 28-109#

D 8

SEQ 0300

EMS344	24-135	28-110#												
EMS345	24-138	24-140	24-215	24-286	24-303	28-111#								
EMS346	24-138	24-144	24-151	24-158	24-173	24-180	24-215	24-282	24-284	24-286	24-288	24-320	24-322	28-112#
EMS347	24-142	24-148	24-155	24-170	24-177	24-184	24-194	28-113#						
EMS35	24-207	24-247	24-252	24-256	24-263	28-32#								
EMS350	24-160	24-162	24-166	24-168	24-308	24-324	24-342	24-347	28-114#					
EMS351	24-160	28-115#												
EMS352	24-168	28-116#												
EMS353	24-189	24-229	24-235	24-237	24-241	24-245	24-250	24-269	24-336	28-117#				
EMS354	24-197	24-199	24-223	24-229	24-233	24-241	24-245	24-269	24-290	24-295	24-297	24-299	24-338	28-118#
EMS355	24-201	24-203	24-213	28-119#										
EMS356	24-328	24-330	24-332	24-399	28-120#									
EMS357	24-207	24-326	28-121#											
EMS36	24-211	24-225	24-231	24-233	24-235	24-237	24-261	24-326	24-353	24-358	24-362	24-369	24-374	24-378
	24-385	24-390	24-394	24-410	28-33#									
EMS360	24-207	24-328	28-122#											
EMS361	24-330	28-123#												
EMS362	24-209	28-124#												
EMS363	24-219	24-310	24-312	24-324	28-125#									
EMS364	24-223	24-297	24-299	28-126#										
EMS365	24-223	24-229	24-235	24-237	24-241	24-245	24-269	24-334	24-336	24-338	28-127#			
EMS366	24-227	28-128#												
EMS367	24-229	24-235	24-237	24-241	24-245	24-269	24-336	28-129#						
EMS37	24-213	28-34#												
EMS370	24-235	24-358	24-374	24-390	24-406	28-130#								
EMS371	24-237	24-408	28-131#											
EMS372	24-252	24-254	24-258	24-326	24-338	24-342	24-347	28-132#						
EMS373	24-286	24-303	28-133#											
EMS374	24-292	28-134#												
EMS375	24-314	28-135#												
EMS376	24-316	24-318	24-322	24-342	28-136#									
EMS377	24-334	24-338	24-342	24-351	24-360	24-367	24-376	24-383	24-392	24-403	24-418	28-137#		
EMS4	24-79	24-96	24-205	28-6#										
EMS40	24-215	24-217	28-35#											
EMS400	24-347	28-138#												
EMS401	24-336	24-343	24-347	24-356	24-372	24-388	24-395	24-412	24-421	28-139#				
EMS402	24-351	24-360	24-367	24-376	24-383	24-392	24-403	24-418	28-140#					
EMS403	24-351	24-360	24-367	24-376	24-383	24-392	24-403	24-418	28-141#					
EMS404	24-354	24-370	24-386	24-410	24-416	28-142#								
EMS405	24-349	24-354	24-356	24-358	24-363	24-365	24-370	24-372	24-374	24-379	24-381	24-386	24-388	24-390
	24-395	24-397	24-401	24-406	24-408	24-410	24-412	24-414	24-416	24-421	28-143#			
EMS406	24-349	24-365	24-381	24-401	28-144#									
EMS407	24-399	28-145#												
EMS41	24-219	24-221	24-223	24-252	24-254	24-258	24-332	24-334	28-36#					
EMS42	24-227	24-229	28-37#											
EMS43	24-233	28-38#												
EMS44	24-239	24-241	28-39#											
EMS45	24-243	24-245	28-40#											
EMS46	24-248	24-250	28-41#											
EMS47	24-254	24-256	24-259	24-269	28-42#									
EMS5	24-108	24-310	28-7#											
EMS50	24-261	24-265	28-43#											
EMS500	24-5	28-147#												
EMS501	24-5	24-7	24-9	24-11	24-14	24-16	24-18	24-20	24-22	24-24	24-26	24-28	24-30	24-32
	24-42	24-45	24-48	24-50	24-53	24-55	24-57	24-59	24-61	24-63	24-65	24-67	24-76	24-78
	24-80	24-82	24-84	24-86	24-89	24-92	24-95	24-97	24-99	24-102	24-105	24-107	24-109	24-111

24-114 24-117 24-119 24-121 24-123 24-125 24-128^{F 8} 24-130 24-132 24-134 24-145 24-147 24-152 24-154
SEQ 0302

EMS65 24-340 24-343 28-57#

H 8

SEQ 0304

EMT155 7-328 24-239#

J 8

SEQ 0306

EMT156	7-331	24-241#
EMT157	7-334	24-243#
EMT16	7-43	24-29#
EMT160	7-337	24-245#
EMT161	7-340	24-248#
EMT162	7-343	24-250#
EMT163	7-346	24-252#
EMT164	7-349	24-254#
EMT165	7-352	24-256#
EMT166	7-355	24-259#
EMT167	7-358	24-261#
EMT17	7-46	24-31#
EMT170	7-361	24-263#
EMT171	7-364	24-265#
EMT172	7-367	24-267#
EMT173	7-370	24-269#
EMT174	7-373	24-271#
EMT175	7-376	24-274#
EMT176	7-379	24-276#
EMT177	7-382	24-278#
EMT2	7-6	24-4#
EMT20	7-49	24-33#
EMT200	7-385	24-280#
EMT201	7-388	24-282#
EMT202	7-391	24-284#
EMT203	7-394	24-286#
EMT204	7-397	24-288#
EMT205	7-400	24-290#
EMT206	7-403	24-292#
EMT207	7-406	24-295#
EMT21	7-52	24-36#
EMT210	7-409	24-297#
EMT211	7-412	24-299#
EMT212	7-415	24-301#
EMT213	7-418	24-303#
EMT214	7-421	24-306#
EMT215	7-424	24-308#
EMT216	7-427	24-310#
EMT217	7-430	24-312#
EMT22	7-55	24-39#
EMT220	7-433	24-314#
EMT221	7-436	24-316#
EMT222	7-439	24-318#
EMT223	7-442	24-320#
EMT224	7-445	24-322#
EMT225	7-448	24-324#
EMT226	7-451	24-326#
EMT227	7-454	24-328#
EMT23	7-58	24-41#
EMT230	7-457	24-330#
EMT231	7-460	24-332#
EMT232	7-464	24-334#
EMT233	7-467	24-336#
EMT234	7-470	24-338#
EMT235	7-473	24-340#
EMT236	7-476	24-342#

EMT237 7-479 7-482 24-345#

L 8

SEQ 0308

EMT24	7-61	24-44#
EMT240	24-347#	
EMT241	7-485	24-349#
EMT242	7-488	24-351#
EMT243	7-492	24-354#
EMT244	7-495	24-356#
EMT245	7-498	24-358#
EMT246	7-501	24-360#
EMT247	7-504	24-363#
EMT25	7-64	24-47#
EMT250	7-508	24-365#
EMT251	7-511	24-367#
EMT252	7-515	24-370#
EMT253	7-518	24-372#
EMT254	7-521	24-374#
EMT255	7-524	24-376#
EMT256	7-527	24-379#
EMT257	7-530	24-381#
EMT26	7-67	24-49#
EMT260	7-533	24-383#
EMT261	7-537	24-386#
EMT262	7-540	24-388#
EMT263	7-543	24-390#
EMT264	7-546	24-392#
EMT265	7-549	24-395#
EMT266	7-552	24-397#
EMT267	7-555	24-399#
EMT27	7-70	24-52#
EMT270	7-558	24-401#
EMT271	7-561	24-403#
EMT272	7-564	24-406#
EMT273	7-567	24-408#
EMT274	7-570	24-410#
EMT275	7-573	24-412#
EMT276	7-576	24-414#
EMT277	7-580	24-416#
EMT3	7-9	24-6#
EMT30	7-73	24-54#
EMT300	7-583	24-418#
EMT301	7-586	24-421#
EMT31	7-76	24-56#
EMT32	7-79	24-58#
EMT33	7-82	24-60#
EMT34	7-85	24-62#
EMT35	7-88	24-64#
EMT36	7-91	24-66#
EMT37	7-94	24-68#
EMT4	7-12	24-8#
EMT40	7-97	24-70#
EMT41	7-100	24-73#
EMT42	7-103	24-75#
EMT43	7-106	24-77#
EMT44	7-109	24-79#
EMT45	7-112	24-81#
EMT46	7-115	24-83#
EMT47	7-118	24-85#

EMTS

7-15

24-10#

N 8

SEQ 0310

EMT50	7-121	24-88#																		
EMT51	7-124	24-91#																		
EMT52	7-127	24-94#																		
EMT53	7-130	24-96#																		
EMT54	7-133	24-98#																		
EMT55	7-136	24-101#																		
EMT56	7-139	24-104#																		
EMT57	7-142	24-106#																		
EMT6	7-18	24-13#																		
EMT60	7-145	24-108#																		
EMT61	7-148	24-110#																		
EMT62	7-151	24-113#																		
EMT63	7-154	24-116#																		
EMT64	7-157	24-118#																		
EMT65	7-160	24-120#																		
EMT66	7-163	24-122#																		
EMT67	7-166	24-124#																		
EMT7	7-22	24-15#																		
EMT70	7-169	24-127#																		
EMT71	7-172	24-129#																		
EMT72	7-175	24-131#																		
EMT73	7-178	24-133#																		
EMT74	7-181	24-135#																		
EMT75	7-184	24-136#																		
EMT76	7-187	24-138#																		
EMT77	7-190	24-140#																		
EMTVEC	4-688#	8-19*	8-19*																	
ENRGDT	23-144#																			
EQUALS	20-3#																			
ERR	4-757#	12-D94	12-D98	12-E04	12-E20	12-E45	12-E46	12-L61												
ERRNMB	14-47*	14-48*	14-51	14-55	14-60	14-152#														
ERROR	4-688#																			
ERRTYP	14-15#	19-7																		
ERRVEC	4-688#	8-19	8-19*	8-19*	8-21*	8-22*	12-<26	12-<27	12-<28*	12-<29*	12-<35*	12-<36*	12-<40*	12-<41*						
	12-<67	12-<68	12-<69*	12-<70*	12-<88*	12-<89*	12-<96*	12-<97*	15-6	15-7	15-8*	15-9*	15-25*	15-45*						
	15-46*	19-6	19-6*	19-6*	19-6*															
ERTY00	14-22	14-157#																		
ERTY01	14-45	14-158#																		
ERTY02	14-50	14-159#																		
ERTY03	14-52	14-160#																		
ERTY04	14-144	14-161#																		
ERTY05	14-38	14-162#																		
ESRC	4-829#	12-a84	12-a91	12-c44	12-c49	12-k09	12-k14													
F0	4-699#																			
F1	4-698#																			
F2	4-697#																			
F3	4-696#																			
F4	4-695#																			
FER	4-781#	4-787	12-697	12-726																
FMT16	4-855#	12-872	12-822	12-841	12-861	12-863	12-C63	12-C65	12-D22	12-D24	12-D75	12-D77	12-E34	12-P50						
	12-S64	12-S66																		
FNCDTB	12-K12	12-M73	12-N18	12-R16	21-55#															
FNCMSK	4-701#	12-E84	12-F26	12-G39	12-G67	12-H90	12-I04	12-I69												
GETBUF	6-0#																			
GNS	4-986	4-986	8-6	8-25	8-39	8-50	8-56	8-57	10-24	13-20	13-20	18-17	19-10	19-10						
	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10						

19-10 19-10 19-10 19-10 19-10 19-10 19-10^c 9 19-10 19-10 19-10

SEQ 0312

IPCK3 4-966#

E 9

SEQ 0314

IR	4-945#	12-13	12-34											
IVC	4-884#	12-892	12-913	12-J96	12-K11	12-K13	21-58	21-59	21-60	21-62	21-63	21-64	21-67	21-68
	21-69	21-70	21-71	21-72	21-73	21-74	21-75	21-76	21-77	21-78	21-79	21-80	21-81	21-82
	21-83	21-84	21-85	21-86	21-87	21-88								
LBC	4-886#	12-900	12-921											
LBT	4-761#	12-D51	12-D69	12-D71										
LCLOCK	15-27	15-55#												
LCOUNT	15-29	15-69#												
LF	4-688#	14-79	19-5	19-5	20-14	20-16	28-98	28-147	28-148	28-149	28-150	28-151	28-152	28-153
	28-154	28-155	28-156	28-157	28-158	28-158	28-159							
LODEV	8-76	20-24#												
LS	4-838#	12-B20	12-B30											
LSC	4-885#	12-892	12-913	12-K40	12-K55	12-K57								
LST	4-839#	12-B39	12-B49											
LSTOP	15-28	15-83#												
LSTRK	6-0#	10-45*	10-53*	12-B12	12-C55	12-C82	12-D35	12-P40	12-R36	12-S24	12-g18	12-g20	12-g23	12-k57
MCLK	4-812#	12-d50	12-d76	12-f20	12-h88	12-j41	12-k33							
MCPE	4-927#	12-:68												
MDF	4-817#	12-?40	12-?81	12-Z06	12-Z10	12-Z11	12-\76	12-\80	12-\81	12- 67	12- 71	12- 72	12-a97	12-b01
	12-b02	12-d38	12-d42	12-d43	12-d50	12-d51	12-d68	12-d69	12-d76	12-d77	12-d95	12-d96	12-f08	12-f12
	12-f13	12-f20	12-f21	12-h88	12-h89	12-j29	12-j33	12-j34	12-j41	12-j42	12-k21	12-k25	12-k26	12-k33
	12-k34													
MDPE	4-943#													
MI	4-820#	12- 46	12-a67											
MIXED	23-4#													
MOC	4-815#	4-841	12-a73	12-A04	12-Z61	12-Z68	12-Z69	12-Z75	12-Z80	12-Z81	12-[03	12-[11	12-[12	12-[60
	12-[85	12-J35	12-J44	12-J45	12-J51	12-J56	12-J57	12-J80	12-J90	12-J91	12-^41	12-^69	12- 30	12- 39
	12- 40	12- 49	12- 54	12- 55	12- 61	12- 62	12- 63	12- 67	12- 68	12- 91	12-a01	12-a02	12-a51	12-a60
	12-a61	12-a70	12-a75	12-a76	12-a86	12-a87	12-a97	12-b01	12-b02	12-b23	12-b39	12-b40	12-b65	12-b66
	12-c13	12-c22	12-c23	12-c29	12-c33	12-c34	12-c41	12-c42	12-c57	12-c83	12-c92	12-c93	12-c98	12-d02
	12-d03	12-d23	12-d32	12-d33	12-d38	12-d42	12-d43	12-d50	12-d51	12-d68	12-d69	12-d76	12-d77	12-d95
	12-d96	12-e15	12-e24	12-e25	12-e42	12-e43	12-e75	12-f02	12-f03	12-f08	12-f12	12-f13	12-f20	12-f21
	12-f47	12-f71	12-f72	12-f78	12-f89	12-f90	12-g06	12-g07	12-g85	12-h09	12-h10	12-h29	12-h53	12-h54
	12-h88	12-h89	12-i17	12-i39	12-i40	12-i47	12-i51	12-i52	12-i79	12-j01	12-j02	12-j09	12-j13	12-j14
	12-j41	12-j42	12-j63	12-j87	12-j88	12-j98	12-j99	12-k06	12-k07	12-k21	12-k25	12-k26	12-k33	12-k34
	12-k98	12-l38	12-l39	12-l52	12-l94	12-l95	12-m09	12-m27	12-m28					
MOH	4-847#													
MOL	4-759#	12-=85	12-=96	12-=98	12->17	12->21								
MRTAAA	4-841#													
MRD	4-813#													
MS	4-818#	12- 62												
MSC	4-821#	12- 61	12- 62	12- 63										
MSDRVS	9-116	20-22#												
MSE	4-893#													
MSER	4-816#	12-a24	12-a44	12-[36	12-[40	12-[41	12-^15	12-^19	12-^20	12-a26	12-a30	12-a31	12-h76	12-h80
	12-h81													
MSGDRV	8-95	20-23#												
MSHELP	9-27	20-7#												
MUR	4-814#	4-841	12-=94	12->19	12-E71	12-F07	12-F33	12-F34	12-G27	12-G64	12-G65	12-H77	12-I01	12-I02
	12-153	12-182	12-183	12-J92	12-K08	12-K09	12-K72	12-K73	12-K93	12-K94	12-L14	12-L15	12-L54	12-M04
	12-M21	12-M22	12-M63	12-M67	12-M68	12-N01	12-N09	12-N10	12-N43	12-N56	12-N57	12-N86	12-N94	12-N95
	12-014	12-023	12-024	12-058	12-061	12-062	12-091	12-094	12-095	12-P54	12-P56	12-P57	12-Q23	12-Q33
	12-034	12-R01	12-R08	12-R09	12-R50	12-R54	12-R55	12-R95	12-R98	12-R99	12-S33	12-S45	12-S46	12-S51
	12-S52	12-S81	12-S89	12-U66	12-U69	12-U70	12-V01	12-V38	12-V62	12-W64	12-W70	12-W71	12-X10	12-Y04
	12-Y07	12-Y08	12-Y54	12-Y61	12-Y62	12-Y92	12-Z00	12-Z01	12-Z06	12-Z10	12-Z11	12-Z30	12-Z38	12-Z39
	12-Z61	12-Z68	12-Z69	12-Z74	12-Z75	12-Z80	12-Z81	12-[03	12-[11	12-[12	12-[17	12-[22	12-[23	12-[36

12-[40 12-[41 12-[60 12-[84 12-[85 12-\19 12-\28^G 9 12-\29 12-\160 12-\170 12-\171 12-\176 12-\180 12-\181
SEQ 0316

PR5

4-688# 15-63

I 9

SEQ 0318

12-412 12-467* 12-471 12-484* 12-488 12-503* 12-570*^{K 9} 12-571* 12-581* 12-582* 12-597* 12-598* 12-:84* 12-:86*
SEQ 0320

	12-:88	12-:05*	12-:06*	12-:07	12-:12	12-:16*	12-:17*	12-:18	12-:23	12-:30*	12-:31*	12-:32	12-:37	12-B17*
	12-B77	12-B94*	12-C04	12-C36*	12-C46	12-C98*	12-D47*	12-N48*	12-N62	12-017*	12-029	12-033	12-P51*	12-Q18*
	12-Q96*	12-R46*	12-R92*	12-S37*	12-T20	12-U42*	12-W44	12-Y02*	12-\20*	12-\61*	12-J02*	12-J36*	12-J81*	12-^42*
	12- 11*	12- 52*	12- 96*	12- 31*	12- 92*	12-a52*	12-b24*	12-c17*	12-c86*	12-d26*	12-e18*	12-e78*	12-f50*	12-g88*
RMDAI	12-F33*	12-T11*	12-T73*	12-j66*	12-k86*									
	6-0#	12-86*	12-95	12-181*	12-194*	12-200	12-230*	12-243	12-270*	12-279	12-390*	12-393	12-412*	12-415
RMDAO	12-471*	12-473	12-488*	12-491	12-:88*	12-:90								
	6-0#	12-C25*	12-C36	12-C58*	12-C60*	12-C82*	12-C83*	12-C98	12-D25*	12-D35*	12-D36*	12-D47	12-D78*	12-P40*
	12-P41*	12-P51	12-R36*	12-R37*	12-R46	12-R66	12-R72	12-R74*	12-R75	12-R77*	12-R79*	12-R80	12-S24*	12-S25*
	12-S37	12-S67*	12-k57*	12-k58*	12-k86	12-L87								
RMDB	4-977#	12-60												
RMDBI	6-0#													
RMDBO	6-0#													
RMDC	4-908#	12-80*	12-87	12-176*	12-184	12-218*	12-224*	12-232	12-265*	12-272	12-369*	12-373	12-429*	12-433
	12-447*	12-453	12-485*	12-499*	12-505	12-:85*	12-:87*	12-:89	12-:49	12-:52*	12-:53	12-:61*	12-:62*	12-:63
	12-:71*	12-:72*	12-:73	12-:89*	12-:90*	12-:93	12-C80	12-C87*	12-D06	12-D46*	12-N49*	12-N65	12-018*	12-038
	12-042	12-055*	12-P29*	12-Q19*	12-Q97*	12-R45*	12-R91*	12-S36*	12-Y03*	12-\21*	12-\62*	12-J03*	12-J37*	12-J82*
	12-^43*	12- 12*	12- 53*	12- 97*	12- 32*	12- 93*	12-a53*	12-b25*	12-c16*	12-c87*	12-d27*	12-e19*	12-e79*	12-f51*
	12-g89*	12-F32*	12-T10*	12-T74*	12-j67*	12-k87*								
RMDCI	6-0#	12-87*	12-97*	12-184*	12-191*	12-197*	12-203	12-232*	12-239*	12-245	12-272*	12-289	12-373*	12-376*
	12-377	12-433*	12-437*	12-438	12-453*	12-459*	12-460	12-505*	12-509	12-510	12-:89*	12-:92*	12-:93	
RMDCO	6-0#	12-R86*	12-R91	12-S10	12-S16*	12-S17	12-k56*	12-k87	12-L34					
RMDS	4-900#	8-84	12-469*	12-501*	12-=79	12-=84	12-=95	12->07	12->16	12->44	12->50	12->60	12->77	12-@69
	12-a74	12-a84	12-A01	12-D32	12-D50	12-D68	12-D86	12-D93	12-E03	12-E19	12-E44	12-G51	12-G56	12-G91
	12-G94	12-K76	12-K83	12-K97	12-L04	12-L59	12-L60	12-L65	12-L97	12-M07	12-M25	12-050	12-067	12-076
	12-P00	12-P20	12-P30	12-P43	12-P58	12-T33	12-T42	12-T59	12-T64	12-T87	12-U16	12-U45	12-V11	12-V15
	12-V42	12-V47	12-V63	12-V67	12-V75	12-V79	12-V96	12-W01	12-W21	12-W28	12-W76	12-W80	12-Z86	12-Z91
	12-J62	12-J67	12- 73	12- 78	16-20	16-24	17-18	17-23						
RMDSI	6-0#													
RMDSO	6-0#													
RMDT	4-905#	10-49	12-448*	12-523	12-543	12-T32	12-U75	14-25						
RMDTI	6-0#													
RMDTO	6-0#													
RMEC1	4-912#	12-450*												
RMEC1I	6-0#													
RMEC1O	6-0#													
RMEC2	4-913#	12-62	12-411*	12-<42										
RMEC2I	6-0#													
RMEC2O	6-0#													
RMER1	4-901#	12-112*	12-120	12-142*	12-147	12-167*	12-174*	12-182	12-225*	12-233	12-266*	12-273	12-368*	12-372
	12-428*	12-432	12-445*	12-451	12-483*	12-487	12-502*	12-684*	12-687	12-693	12-701	12-709	12-714*	12-715*
	12-716	12-722	12-730	12-739	12-745*	12-746*	12-747	12-752	12-764*	12-765	12-776	12-:11	12-:16	12-:28
	12-<13	12-<18	12-<48	12-?13*	12-?23	12-?28	12-?56	12-?61	12-B92*	12-D63*	12-D91*	12-E01*	12-E16*	12-E18*
	12-E40*	12-E72*	12-F08*	12-G28*	12-H78*	12-I54*	12-J93*	12-K37*	12-K69*	12-K90*	12-L11*	12-L55*	12-L82	12-L86
	12-M05*	12-M71	12-M78	12-N44*	12-N87*	12-O15*	12-O16*	12-P77*	12-Q39	12-R14	12-R21	12-R60	12-R65	12-S04
	12-S09	12-S55	12-S59	12-S74	12-S82*	12-S92	12-U99*	12-V36*	12-V57*	12-V90*	12-V95*	12-W14*	12-X26	12-X32
	12-Y13*	12-Y55*	12-Y75	12-Y80	12-Y93*	12-Z31*	12-Z44	12-Z48	12-Z62*	12-[04*	12-[61*	12-\22*	12-\42	12-\47
	12-\63*	12-J04*	12-J17	12-J21	12-J38*	12-J83*	12-^44*	12- 13*	12- 33	12- 38	12- 54*	12- 98*	12- 11	12- 15
	12- 33*	12- 94*	12-a54*	12-b26*	12-c14*	12-c59	12-c67	12-c84*	12-d06	12-d11	12-d24*	12-e16*	12-e76*	12-f48*
	12-g86*	12-h13	12-h18	12-h30*	12-i18*	12-i57	12-i62	12-i80*	12-j64*	12-k99*	16-17*	17-15*		
RMER1I	6-0#	12-120*	12-128	12-147*	12-150*	12-182*	12-195*	12-201	12-233*	12-246	12-273*	12-295	12-372*	12-374
	12-432*	12-434*	12-435	12-451*	12-454	12-487*	12-489	12-687*	12-688	12-696	12-704	12-716*	12-717	12-725
	12-734													
RMER1O	6-0#	12-K64*	12-K69	12-K79	12-K90	12-L00	12-L11	12-L21	12-L31	12-L40*				
RMER2	4-911#	12-113*	12-121	12-143*	12-148	12-169*	12-177*	12-185	12-226*	12-234	12-267*	12-274	12-386*	12-392
	12-408*	12-414	12-449*	12-468*	12-472	12-486*	12-500*	12-506	12-870	12-876*	12-880	12-907*	12-908*	12-909

12-928* 12-929* 12-930 12-956* 12-957 12-:12 12-?06^M 12-?12*⁹ 12-?14 12-?41 12-?45 12-?76* 12-?78 12-?87
SEQ 0322

RMOF 4-907# 12-81* 12-88 12-168* 12-175* 12-183 12-217*^{B 10} 12-223* 12-231 12-264* 12-271 12-385* 12-391 12-407*
SEQ 0324

	12-413	12-446*	12-452	12-470*	12-498*	12-504	12-786*	12-787	12-792	12-800	12-804*	12-805*	12-806	12-825*
	12-826*	12-827	12-852*	12-853*	12-854	12-879*	12-955*	12-B16*	12-B95*	12-C35*	12-C96*	12-D48*	12-E43*	12-N50*
	12-N68	12-N80	12-N89*	12-000	12-083*	12-P08	12-P13	12-P50*	12-Q20*	12-Q98*	12-R47*	12-S34*	12-k88*	
RMOF I	6-0#	12-88*	12-99*	12-183*	12-190*	12-196*	12-202	12-231*	12-238*	12-244	12-271*	12-283	12-391*	12-395*
	12-396	12-413*	12-417*	12-418	12-452*	12-456*	12-457	12-504*	12-507*	12-806*	12-809	12-817	12-827*	12-830
	12-838													
RMOFO	6-0#	12-872*	12-879	12-955	12-B11*	12-B16	12-B22	12-B34	12-B41	12-B61	12-B63*	12-B78*	12-B95	12-B96
	12-C35	12-C63	12-C65*	12-C81*	12-C96	12-C97	12-D22	12-D24*	12-D34*	12-D48	12-D49	12-D75	12-D77*	12-E34*
	12-E43	12-S23*	12-S34	12-S35	12-S64	12-S66*	12-k60*	12-k70*	12-k88	12-L79				
RMR	4-783#	12-689	12-718	12-S93	12-S98	12-T21	12-Y42							
RMSN	4-906#	12-388*	12-431*	12-979	12-983	12-987								
RMSNI	6-0#													
RMSNO	6-0#													
RMWC	4-974#	12-52	12-S38*	12-k94*										
RMWC I	6-0#													
RMWCO	6-0#	12-k62*	12-k94											
RQA	4-869#	12-T80	12-T91	12-U09	12-U20	12-U38	12-U49	12-U77	12-U81	12-[68	12-^51	12-b48	12-e54	12-f99
	12-L32	12-L70												
RQB	4-870#	12-T80	12-T91	12-U09	12-U20	12-U38	12-U49	12-U77	12-U83	12-[68	12-^51	12-b48	12-e54	12-f99
	12-L32	12-L70												
RTC	4-711#	12-H21	12-J40	12-092	12-W96									
SA1	4-748#													
SA16	4-744#													
SA2	4-747#													
SA4	4-746#													
SAB	4-745#													
SADMSK	4-752#													
SAVREG	14-16	19-10#												
SC	4-925#													
SCO	4-801#													
SC1	4-800#													
SC2	4-799#													
SC3	4-798#													
SC4	4-797#													
SCOPE	4-688#	12-1	12-73	12-106	12-137	12-159	12-329	12-521	12-548	12-559	12-612	12-679	12-783	12-796
	12-866	12-975	12-998	12-:51	12-:79	12-:99	12-:43	12-<05	12-=00	12-=17	12-=75	12->40	12-?01	12-a03
	12-a64	12-A24	12-B06	12-B74	12-C76	12-D29	12-D82	12-E62	12-E97	12-G18	12-H68	12-I43	12-J82	12-K28
	12-K62	12-L94	12-M54	12-M90	12-N38	12-N77	12-O07	12-O47	12-O73	12-P17	12-P38	12-P65	12-Q09	12-Q86
	12-R34	12-R84	12-S21	12-S71	12-T28	12-T46	12-T69	12-T98	12-U27	12-U57	12-U94	12-V32	12-V52	12-V85
	12-W06	12-W55	12-X00	12-X70	12-Y47	12-\09	12-_00	12-c06	12-c74	12-g76	12-k50	13-9		
SCTMSK	4-803#													
SEARCH	4-717#	12-F97	12-I33	12-J60	12-Q57	12-R51	12-R96	12-_15	12-_56	12-'00	12-'35	12-'96	12-a56	12-b28
	12-c18													
SEEK	4-706#	12-H12	12-J20	12-Q60	12-\24	12-\65	12-J06	12-J40	12-J85	12-^46				
SETOM	12-084	12-P25	12-P48	12-i12	12-i75	12-k89	17-13#							
SETVV	12-M58	12-M96	12-053	12-078	12-088	12-P22	12-P45	12-Q13	12-Q90	12-R39	12-R88	12-S30	12-T71	12-U00
	12-U29	12-U61	12-W59	12-X04	12-X97	12-Y49	12-Y87	12-Z25	12-Z56	12-Z98	12-[55	12-\13	12-\54	12-\95
	12-J29	12-J74	12-^35	12- 04	12- 45	12- 89	12-'24	12-'85	12-a45	12-b17	12-c08	12-c78	12-d18	12-e10
	12-e70	12-f42	12-g80	12-H24	12-T05	12-T68	12-j58	12-k81	16-13#					
SHUT	18-17#	19-8	19-8											
SIZCLK	10-21	15-5#												
SKI	4-882#	12-152	12-958	12-960	12-a14	12-a26	12-a28	12-a42	12-a46					
SNGPRT	4-850#	12-524	12-529	12-534	12-540									
STACK	4-688#	8-19	12-1	12-73	12-106	12-137	12-159	12-329	12-521	12-548	12-559	12-612	12-679	12-783
	12-796	12-866	12-975	12-998	12-:51	12-:79	12-:99	12-:43	12-<05	12-=00	12-=17	12-=75	12->40	12-?01
	12-a03	12-a64	12-A24	12-B06	12-B74	12-C76	12-D29	12-D82	12-E62	12-E97	12-G18	12-H68	12-I43	12-J82

12-K28 12-K62 12-L94 12-M54 12-M90 12-N38 12-N77^{D 10} 12-007 12-047 12-073 12-P17 12-P38 12-P65 12-Q09
SEQ 0326

	12-Q86	12-R34	12-R84	12-S21	12-S71	12-T28	12-T46	12-T69	12-T98	12-U27	12-U57	12-U94	12-V32	12-V52
	12-V85	12-W06	12-W55	12-X00	12-X70	12-Y47	12-Y09	12-_00	12-c06	12-c74	12-g76	12-k50		
STANDA	8-63	9-3#												
START	4-986	8-14#	10-34	18-22										
STKLMT	4-688#													
STOP	6-0#	12-K50	12-P91	12-P96	12-Y20	12-Y30	12-c64	12-c70	12-e89	12-e95	12-f61	12-f67	12-g99	12-h05
	12-h43	12-h49	12-i29	12-i35	12-i91	12-i97	12-j77	12-j83	12-l10	12-l16	15-14*	15-28*		
SW0	4-688#													
SW00	4-688	4-688#												
SW01	4-688	4-688#												
SW02	4-688	4-688#												
SW03	4-688	4-688#												
SW04	4-688	4-688#												
SW05	4-688	4-688#												
SW06	4-688	4-688#												
SW07	4-688	4-688#												
SW08	4-688	4-688#												
SW09	4-688	4-688#												
SW1	4-688#													
SW10	4-688#													
SW11	4-688#													
SW12	4-688#													
SW13	4-688#	14-17												
SW14	4-688#													
SW15	4-688#													
SW2	4-688#													
SW3	4-688#													
SW4	4-688#													
SW5	4-688#													
SW6	4-688#													
SW7	4-688#													
SW8	4-688#													
SW9	4-688#													
SWR	5-0#	8-19	8-19	8-19*	8-19*	8-19*	8-25	10-27	14-17	19-6	19-6	19-6	19-6	19-6
	19-7	19-7	19-7	19-7	19-8	19-8	19-8*	19-11	19-11*					
SWREG	4-986#	8-19	8-25	10-27	19-8	19-8	19-8							
TA1	4-741#	12-C26	12-g17											
TA16	4-737#	10-53	12-g18											
TA2	4-740#	10-53												
TA4	4-739#	10-45												
TA8	4-738#													
TADMSK	4-751#													
TAG	4-871#	12-[95	12-[96	12-[97	12-[98	12-[99	12-\00	12-^86	12-^87	12-^88	12-^89	12-^90	12-^91	12-b86
	12-b87	12-b89	12-b90	12-b91	12-b92	12-e54	12-g39	12-g40	12-g42	12-g43	12-g44	12-g45	12-m48	12-m49
	12-m50	12-m51	12-m52	12-m53	12-m59	12-m60	12-m61	12-m62	12-m63	12-m64	12-m91	12-m92	12-m93	12-m94
	12-m95													
TAGADR	4-992#	5-0												
TAP	4-846#													
TBITVE	4-688#													
TIME	6-0#	15-57*	15-70*	15-72*										
TKVEC	4-688#	19-8*	19-8*											
TPVEC	4-688#													
TRAPVE	4-688#	8-19*	8-19*											
TRE	4-926#													
TRTVEC	4-688#													
TST	4-872#	12-F14	12-F16	12-F17	12-F40	12-F67	12-F75	12-F83	12-F91	12-F99	12-G07	12-[68	12-^51	12-b48

12-f99 12-L32 12-L70

F 10

SEQ 0328

TST1	12-1#	19-6
TST10	12-548#	19-6
TST100	12-U27#	19-6
TST101	12-U57#	19-6
TST102	12-U94#	19-6
TST103	12-V32#	19-6
TST104	12-V52#	19-6
TST105	12-V85#	19-6
TST106	12-W06#	19-6
TST107	12-W55#	19-6
TST11	12-559#	19-6
TST110	12-X00#	19-6
TST111	12-X70#	19-6
TST112	12-Y47#	19-6
TST113	12-\09#	19-6
TST114	12- 00#	19-6
TST115	12-c06#	19-6
TST116	12-c74#	19-6
TST117	12-g76#	19-6
TST12	12-612#	19-6
TST120	12-k50#	19-6
TST13	12-679#	19-6
TST14	12-783#	19-6
TST15	12-796#	19-6
TST16	12-866#	19-6
TST17	12-975#	19-6
TST2	12-73#	19-6
TST20	12-998#	19-6
TST21	12-:51#	19-6
TST22	12-:79#	19-6
TST23	12-:99#	19-6
TST24	12-:43#	19-6
TST25	12-<05#	19-6
TST26	12-=00#	19-6
TST27	12-=17#	19-6
TST3	12-106#	19-6
TST30	12-=75#	19-6
TST31	12->40#	19-6
TST32	12-?01#	19-6
TST33	12-@03#	19-6
TST34	12-@64#	19-6
TST35	12-A24#	19-6
TST36	12-B06#	19-6
TST37	12-B74#	19-6
TST4	12-137#	19-6
TST40	12-C76#	19-6
TST41	12-D29#	19-6
TST42	12-D82#	19-6
TST43	12-E62#	19-6
TST44	12-E97#	19-6
TST45	12-G18#	19-6
TST46	12-H68#	19-6
TST47	12-I43#	19-6
TST5	12-159#	19-6
TST50	12-J82#	19-6
TST51	12-K28#	19-6

TST52 12-K62# 19-6

H 10

SEQ 0330

WCE

4-937# 21-77 21-78

J 10

SEQ 0332

GETMR! 4-254# 12-149 12--25 12--35 12--46 12--61 12-A33^{N 10} 12-A44 12-A54 12-A71 12-B18 12-I58 12-I84 12-L18
SEQ 0336

	12-L70	12-N15	12-P85	12-X89	12-Y15	12-a83	12-c43	12-d52	12-d83	12-e30	12-e84	12-f22	12-f56	12-g94
GETMR2	12-h38	12-h90	12-i24	12-i86	12-j43	12-j72	12-k08	12-k37	12-l05					
	4-262#	12-F13	12-F39	12-T79	12-T90	12-U08	12-U19	12-U37	12-U48	12-U76	12-[67	12-^50	12-b47	12-e46
GETOF	12-f98	12-L21	12-L56	12-m13										
	4-270#	12-88	12-183	12-231	12-271	12-391	12-413	12-452	12-504	12-787	12-806	12-827	12-854	12-N68
GETPRI	12-000	12-P08												
GETSN	4-688#	15-58												
GETSWR	4-222#	12-983	12-987											
GETWC	4-688#	8-25	8-25#	10-27										
GETX	4-134#													
MSG	4-118#													
	4-8#	12-1	12-73	12-106	12-137	12-159	12-324#	12-329	12-518#	12-521	12-548	12-559	12-612	12-679
	12-783	12-796	12-866	12-975	12-998	12-:51	12-:79	12-:99	12-:43	12-<05	12-=00	12-=17	12-=75	12->40
	12-?01	12-a03	12-a64	12-A24	12-A93#	12-B06	12-B71#	12-B74	12-C76	12-D29	12-D82	12-E62	12-E97	12-G18
	12-H68	12-I43	12-J82	12-K28	12-K62	12-L94	12-M54	12-M90	12-N38	12-N77	12-O07	12-O47	12-O73	12-P17
	12-P38	12-P65	12-Q09	12-Q86	12-R34	12-R84	12-S21	12-S71	12-T28	12-T46	12-T69	12-T98	12-U27	12-U57
	12-U94	12-V32	12-V52	12-V85	12-W06	12-W55	12-X00	12-X70	12-Y47	12-\09	12-_00	12-c06	12-c74	12-g76
	12-k50													
MULT	4-688#													
NEWTST	4-688#	12-1	12-73	12-106	12-137	12-159	12-329	12-521	12-548	12-559	12-612	12-679	12-783	12-796
	12-866	12-975	12-998	12-:51	12-:79	12-:99	12-:43	12-<05	12-=00	12-=17	12-=75	12->40	12-?01	12-a03
	12-a64	12-A24	12-B06	12-B74	12-C76	12-D29	12-D82	12-E62	12-E97	12-G18	12-H68	12-I43	12-J82	12-K28
	12-K62	12-L94	12-M54	12-M90	12-N38	12-N77	12-O07	12-O47	12-O73	12-P17	12-P38	12-P65	12-Q09	12-Q86
	12-R34	12-R84	12-S21	12-S71	12-T28	12-T46	12-T69	12-T98	12-U27	12-U57	12-U94	12-V32	12-V52	12-V85
	12-W06	12-W55	12-X00	12-X70	12-Y47	12-\09	12- 00	12-c06	12-c74	12-g76	12-k50			
NWTST	4-504#	12-1	12-73	12-106	12-137	12-159	12-329	12-521	12-548	12-559	12-612	12-679	12-783	12-796
	12-866	12-975	12-998	12-:51	12-:79	12-:99	12-:43	12-<05	12-=00	12-=17	12-=75	12->40	12-?01	12-a03
	12-a64	12-A24	12-B06	12-B74	12-C76	12-D29	12-D82	12-E62	12-E97	12-G18	12-H68	12-I43	12-J82	12-K28
	12-K62	12-L94	12-M54	12-M90	12-N38	12-N77	12-O07	12-O47	12-O73	12-P17	12-P38	12-P65	12-Q09	12-Q86
	12-R34	12-R84	12-S21	12-S71	12-T28	12-T46	12-T69	12-T98	12-U27	12-U57	12-U94	12-V32	12-V52	12-V85
	12-W06	12-W55	12-X00	12-X70	12-Y47	12-\09	12- 00	12-c06	12-c74	12-g76	12-k50			
POP	4-688#	12-<35	12-<36	12-<40	12-<41	12-<96	12-<97	15-45	15-46	18-12	19-1	19-3	19-9	19-11
	19-11	19-12	19-12											
PUSH	4-688#	12-<26	12-<27	12-<67	12-<68	15-6	15-7	18-8	19-1	19-3	19-9	19-11	19-11	19-12
	19-12	19-12												
PUTAS	4-466#	12-387	12-U13	12-V08	12-V59	12-V73	12-V92	12-W15						
PUTBA	4-330#	12-S39	12-k95											
PUTBAE	4-474#													
PUTCS1	4-315#	12-78	12-111	12-165	12-172	12-221	12-430	12-619	12-632	12-633	12-645	12-646	12-665	12-666
	12-=06	12-B98	12-C39	12-D01	12-D65	12-E78	12-F27	12-G35	12-H86	12-I70	12-K07	12-K46	12-K70	12-K91
	12-L12	12-L81	12-M18	12-M66	12-N04	12-N53	12-N92	12-O21	12-O59	12-O92	12-P55	12-P84	12-Q27	12-R04
	12-R51	12-R96	12-S40	12-S86	12-T76	12-U05	12-U34	12-U67	12-V41	12-W67	12-X16	12-Y05	12-Y57	12-Y95
	12-Z33	12-Z64	12-[06	12-[63	12-\24	12-\65	12-[06	12-[40	12-[85	12-^46	12- 15	12- 56	12-'00	12-'35
	12-'96	12-a56	12-b28	12-c18	12-c88	12-d28	12-e20	12-e80	12-f52	12-g90	12-H34	12-I20	12-i82	12-j68
	12-l01	16-19	17-17											
PUTCS2	4-346#	12-:26												
PUTDA	4-370#	12-79	12-166	12-173	12-216	12-222	12-263	12-384	12-406	12-467	12-484	12-503	12-570	12-571
	12-581	12-582	12-597	12-598	12-:84	12-:86	12-:05	12-:06	12-:16	12-:17	12-:30	12-:31	12-B17	12-B94
	12-C36	12-C98	12-D47	12-N48	12-O17	12-P51	12-Q18	12-Q96	12-R46	12-R92	12-S37	12-U42	12-Y02	12-\20
	12-\61	12-J02	12-J36	12-J81	12-^42	12- 11	12- 52	12- 96	12-'31	12-'92	12-a52	12-b24	12-c17	12-c86
	12-d26	12-e18	12-e78	12-f50	12-g88	12-H33	12-I11	12-I73	12-j66	12-k86				
PUTDB	4-338#													
PUTDC	4-378#	12-80	12-176	12-218	12-224	12-265	12-369	12-429	12-447	12-485	12-499	12-:85	12-:87	12-:52
	12-:61	12-:62	12-:71	12-:72	12-:89	12-:90	12-C87	12-D46	12-N49	12-O18	12-O55	12-P29	12-Q19	12-Q97
	12-R45	12-R91	12-S36	12-Y03	12-\21	12-\62	12-[03	12-[37	12-[82	12-^43	12- 12	12- 53	12- 97	12-'32
	12-'93	12-a53	12-b25	12-c16	12-c87	12-d27	12-e19	12-e79	12-f51	12-g89	12-H32	12-I10	12-I74	12-j67

12-k87

C 11

SEQ 0338

SETLFS	4-649#													
SETOM	4-615#	12-084	12-P25	12-P48	12-i12	12-i75	12-k89							
SETPRI	4-688#	8-23	10-40	15-63	15-86	19-8								
SETTRA	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10	19-10#
SETUP	4-688#	8-19												
SETVV	4-604#	12-M58	12-M96	12-053	12-078	12-088	12-P22	12-P45	12-Q13	12-Q90	12-R39	12-R88	12-S30	12-T71
	12-U00	12-U29	12-U61	12-W59	12-X04	12-X97	12-Y49	12-Y87	12-Z25	12-Z56	12-Z98	12-[55	12-\13	12-\54
	12-\95	12-J29	12-J74	12-^35	12- 04	12- 45	12- 89	12- '24	12- '85	12-a45	12-b17	12-c08	12-c78	12-d18
	12-e10	12-e70	12-f42	12-g80	12-H24	12-I05	12-I68	12-j58	12-k81					
SKIP	4-688#													
SLASH	4-688#													
STARS	4-688#	4-988	4-991	4-991	4-991	5-0	5-0	5-0	12-1	12-1	12-73	12-73	12-106	12-106
	12-137	12-137	12-159	12-159	12-329	12-329	12-521	12-521	12-548	12-548	12-559	12-559	12-612	12-612
	12-679	12-679	12-783	12-783	12-796	12-796	12-866	12-866	12-975	12-975	12-998	12-998	12-:51	12-:51
	12-:79	12-:79	12-:99	12-:99	12-:43	12-:43	12-<05	12-<05	12-=00	12-=00	12-=17	12-=17	12-=75	12-=75
	12->40	12->40	12-?01	12-?01	12-@03	12-@03	12-@64	12-@64	12-A24	12-A24	12-B06	12-B06	12-B74	12-B74
	12-C76	12-C76	12-D29	12-D29	12-D82	12-D82	12-E62	12-E62	12-E97	12-E97	12-G18	12-G18	12-H68	12-H68
	12-I43	12-I43	12-J82	12-J82	12-K28	12-K28	12-K62	12-K62	12-L45	12-L47	12-L94	12-L94	12-M54	12-M54
	12-M90	12-M90	12-N38	12-N38	12-N77	12-N77	12-O07	12-O07	12-O47	12-O47	12-O73	12-O73	12-P17	12-P17
	12-P38	12-P38	12-P65	12-P65	12-Q09	12-Q09	12-Q86	12-Q86	12-R34	12-R34	12-R84	12-R84	12-S21	12-S21
	12-S71	12-S71	12-T28	12-T28	12-T46	12-T46	12-T69	12-T69	12-T98	12-T98	12-U27	12-U27	12-U57	12-U57
	12-U94	12-U94	12-V32	12-V32	12-V52	12-V52	12-V85	12-V85	12-W06	12-W06	12-W55	12-W55	12-X00	12-X00
	12-X70	12-X70	12-Y47	12-Y47	12-\09	12-\09	12- 00	12- 00	12-b30	12-b44	12-c06	12-c06	12-c74	12-c74
	12-e48	12-e56	12-f80	12-f94	12-g76	12-g76	12-K50	12-K50	12-k76	12-k78	12-L23	12-L30	12-L58	12-L68
	12-m15	12-m23	13-20	14-2	19-1	19-2	19-3	19-4	19-5	19-6	19-7	19-8	19-8	19-8
	19-8	19-8	19-9	19-10	19-11	19-11	19-12							
SWRSU	4-688#	8-19	8-19#											
TAGS	4-54#	5-0												
TRMTRP	19-10#													
TYPBIN	4-688#													
TYPDEC	4-688#	13-20	13-20											
TYPNAM	4-676#	4-688#	8-25											
TYPNUM	4-688#													
TYPOCS	4-688#	8-96	14-23	14-46	14-51	14-53								
TYPOCT	4-688#	9-56	19-8											
TYPTXT	4-688#	8-6	8-39	8-50	8-56	8-57	10-24	13-20	13-20	18-17				