

RM80

DSKLS TEST PT 2
CZRNCAO

AH-T109A-MC
FICHE 1 OF 1

JUL 1982
COPYRIGHT © 1982
MADE IN USA



The main body of the document is a large, dark grid containing numerous small, illegible tables or data points. The content is too faint to transcribe accurately, but it appears to be a structured dataset or a series of test results organized in rows and columns.

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

.REM \

IDENTIFICATION

PRODUCT CODE: AC-T108A-MC
PRODUCT NAME: CZRNCAO RM80 DISKLESS TEST, PT 2
PRODUCT DATE: APRIL 1, 1982
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1982 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

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

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

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM80 DISKLESS DIAGNOSTIC IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL AND DIAGNOSTIC MEANS TO VERIFY THE OPERABILITY OF THE RM80 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 RM80 MASSBUS ADAPTER;

TO RESOLVE HARDWARE FAILURES IN RM80 TO A FIELD REPLACEABLE MODULE OR MODULES.

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM80 DISK SUBSYSTEM, EXCLUDING THE STORAGE MODULE DISK DRIVE AND THE 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 RM80 DISKLESS DIAGNOSTIC:

- PDP-11/70 PROCESSOR
- 12K MEMORY
- KW11-L OR KW11-P CLOCK
- PROGRAM LOADING DEVICE
- TERMINAL
- RH70 CONTROLLER
- 1 TO 8 RM80 DISK DRIVES

2.2 MEDIA REQUIREMENTS

NONE

2.3 PREREQUISITE PROGRAMS

RM80 DISKLESS TEST, PART 1

3.0 OPERATING PROCEDURE

3.1 LOADING

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

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

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

3.2 SWITCH OPTIONS

THE FOLLOWING SWITCH OPTIONS ARE 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 THE OCTAL NUMBER OF THE TEST WHICH THE PROGRAM WILL LOOP ON.

3.3 STARTING

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

3.4 HALTING

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

3.5 RESTARTING

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

4.0 OPERATOR INTERFACE

4.1 PROGRAM ID

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

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

4.2 CONSOLE DIALOGUE

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

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

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

EXAMPLE 1

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

EXAMPLE 2

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

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

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

```
'UNIT STATUS:  
0 ONLINE RM80  
1 LOAD DEVICE  
2 OFFLINE RM80  
3 NOT PRESENT  
4 NOT PRESENT  
5 NOT AN RM80  
6 NOT PRESENT  
7 NOT PRESENT'
```

THE ABOVE UNIT STATUS SHOWS THAT DRIVE 0 & 2 WILL BE TESTED, WHILE DRIVES 1, & 3 - 7 WILL NOT BE TESTED.

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

```
'DRIVE(S) TO BE TESTED, 0, 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

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

'DRIVE(S) TO BE TESTED, NONE'

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

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

'DRIVE 0
DRIVE 2'

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

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

4.3 PROGRESS REPORTS

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

'END OF PASS 1'

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

'TOTAL ERRORS SINCE LAST REPORT 0'

4.4 PERFORMANCE REPORT

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

4.5 PROGRAM HALTS

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

4.6 ERROR REPORTS

THE RM80 DISKLESS DIAGNOSTIC PROVIDES COMPREHENSIVE ERROR REPORTS INTENDED TO (1) AID IN FAULT RESOLUTION AND (2) MINIMIZE

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

REFERENCES TO PROGRAM LISTINGS.

THE FIRST LINE OF THE ERROR REPORT CONTAINS THE NUMBER OF THE UNIT (DRIVE) 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.

THE FOLLOWING PRINTOUT IS AN ERROR MESSAGE IN THIS PROGRAM:

```
DRV# 0 - RM80, TEST# 23, ERR# 61, PC=017724
INCORRECT ECC PATTERN GENERATED DURING WRITE
PROBABLE FAULT(S):
(NOT INCLUDING CABLES OR CONNECTORS)
IF MODULE, M8685,
```

```
EXPECTED   ECC      RECEIVED   ECC
13117     175154    017677    003402
```

4.7 EXECUTION TIME

PASS 1 OF THE PROGRAM TAKES ABOUT 2 SECONDS. PASS 2 AND SUBSEQUENT PASSES TAKE 12 SECONDS.

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM80 DISKLESS DIAGNOSTIC IS EXECUTABLE ON A PDP-11/70 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET.

5.2 DUAL PORT CONFIGURATIONS

THE RM80 DISKLESS DIAGNOSTIC IS NOT EXECUTABLE ON RM80 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 RM80 DISKLESS DIAGNOSTIC.

5.4 MEMORY MANAGEMENT HARDWARE

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

MEMORY MANAGEMENT HARDWARE WILL NOT BE USED DURING THE RM80 DISKLESS DIAGNOSTIC.

5.5 ACT11, APT11 COMPATIBILITY

THE RM80 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 RM80 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 'RM80 DISKLES DIAGNOSTIC' CAN BE EXECUTED USING AN RH70 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.

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

TEST 1 TRANSFER TEST**PURPOSE:**

TO VERIFY THAT THE RM80 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

TEST 2 CTOD TEST**PURPOSE:**

TO VERIFY THAT DATA CAN BE TRANSFERRED TO AND FROM THE RM80 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

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

TEST 3 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

TEST 4 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

TEST 5 TRISTATE TRANSFER TEST

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

PURPOSE:

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

TEST 6 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

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

MANNER; SELECT LINE 16 IS EXPLICITLY TESTED IN THE "ILR TEST".

PROBABLE FAULT:

- 1. IF MODULE
- 2. ASYNCHRONOUS MASSBUS MODULE

TEST 7 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 PORT OR DUAL PORT RM80 DRIVE

PROBABLE FAULT:

- 1. IF MOULE

TEST 10 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

TEST 11 SEARCH TIMEOUT TEST

PURPOSE:

TO VERIFY THAT THE SEARCH TIMEOUT ONE SHOT SETS 'OPI', EXCEPT WHEN 'SEARCH TO DISABLE' IS ACTIVE.

PROCEDURE:

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

WITH SEARCH TIMEOUT DISABLED, THE TEST EXECUTES A DATA COMMAND TO THE POINT WHERE 'P ENABLE SEARCH' IS ASSERTED. AFTER WAITING A SUFFICIENT PERIOD AND VERIFYING THAT OPI IS NOT SET, THE TEST ENABLES SEARCH TIMEOUT AND VERIFIES THAT OPI SETS.

PROBABLE FAULT:

1. CS MODULE

NOTE: IT IS ASSUMED THAT THE 'SET OPI TEST' HAS ALREADY PASSED, THUS MAKING THE IF MODULE AN IMPROBABLE FAULT.

TEST 12 SET DTE TEST

PURPOSE:

IN ADDITION TO VERIFYING THAT 'DRIVE TIMING ERROR' CAN BE SET BY THE CS MODULE, THIS TEST ALSO VERIFIES

* THAT 'MAINTENANCE SECTOR COMPARE' IS NOT STUCK AT ONE OR ZERO.

* THAT 'ENABLE SEARCH' IS NOT STUCK AT ONE OR ZERO.

PROCEDURE:

(1) INITIALIZE AND VERIFY THAT 'DTE' IS RESET, THEN SET MAINTENANCE INDEX PULSE AND VERIFY THAT DTE IS STILL RESET. THIS TEST WILL INSURE THAT THE SECTOR COMPARE FLOP IS NOT STUCK AT ONE.

(2) EXECUTE A DATA COMMAND IN MAINTENANCE MODE TO THE POINT WHERE SEARCH IS ENABLED, I.E., P EN SEARCH H IS ACTIVE. SET AND RESET THE SECTOR PULSE TO SET 'CS3 EN SEARCH' FLOP, AND CLOCK 'CSS SECTOR COMPARE' FLOP WHICH SHOULD NOT SET. SET SECTOR PULSE AND VERIFY THAT DTE IS RESET. THIS TEST FAILS IF J INPUT TO SECTOR COMPARE FLOP IS STUCK AT ONE.

REPEAT, BUT SET MAINTENANCE SECTOR COMPARE AND VERIFY THAT DTE SETS. THIS TEST FAILS IF MAINTENANCE SECTOR COMPARE IS STUCK AT ZERO.

PROBABLE FAULT:

1. CS MODULE
2. DS MODULE

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

TEST 13 FORMAT CHANGE TEST**PURPOSE:**

TO VERIFY THAT A CHANGE IN FORMAT INHIBITS SEARCH ENABLE UNTIL THE NEXT INDEX PULSE.

PROCEDURE:

THE TEST WILL USE 'DTE' FOR VISIBILITY OF 'CS3 EN SEARCH H'.

THE FOLLOWING STEPS ARE EXECUTED:

(1) INITIALIZE AND SET THE FORMAT TO 18 BIT MODE TO SET 'CS3 FMT CHANGE' FLOP.

(2) EXECUTE A DATA COMMAND TO THE POINT WHERE SEARCH IS ENABLED BY THE SEQUENCER.

(3) SET 'MAINTENANCE SECTOR COMPARE', THEN SET 'MAINTENANCE SECTOR PULSE' TO CLOCK 'CS3 EN SEARCH' FLOP WHICH SHOULD NOT SET BECAUSE OF THE FORMAT CHANGE.

(4) RESET SECTOR PULSE TO CLOCK 'CSS SECTOR COMPARE' FLOP WHICH WILL NOT SET IF 'CS3 EN SEARCH H' IS INACTIVE.

(5) SET SECTOR PULSE AND VERIFY THAT DRIVE TIMING ERROR IS RESET.

(6) SET AND RESET INDEX PULSE TO CLEAR THE FORMAT CHANGE FLOP.

(7) SET AND RESET SECTOR PULSE TO SET 'CS3 EN SEARCH' FLOP AND 'CSS SECTOR COMPARE' FLOP.

(8) SET SECTOR COMPARE AND VERIFY THAT DTE IS SET.

REPEAT THE TEST WITH A FORMAT CHANGE FROM 18 BIT MODE TO 16 BIT MODE.

PROBABLE FAULT:

1. CS MODULE

TEST 14 PROM STROBE TEST**PURPOSE:**

TO VERIFY THAT WORD CLOCK AND PROM STROBE CAN BE MANIPULATED

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

IN MAINTENANCE MODE.

PROCEDURE:

INITIALIZE AND SET 16 BIT MODE, THEN SEQUENCE THE MAINTENANCE CLOCK UNTIL PROM STROBE SETS. ISSUE -- MORE MAINTENANCE CLOCK PULSES AND VERIFY THAT PROM STROBE RESETS.

PROBABLE FAULT:

1. CS MODULE
2. DS MODULE

TEST 15 SYNC WORD COUNT INHIBIT TEST

PURPOSE:

TO VERIFY THE FOLLOWING DURING READ COMMAND:

- * THAT "CS4 P LFS" (LOOKING FOR SYNC) GOES ACTIVE.
- * THAT "LOOKING FOR SYNC" INHIBITS THE WORD COUNT

PROCEDURE:

A READ COMMAND IS SETUP AND EXECUTED TO THE POINT WHERE "LOOKING FOR SYNC" SHOULD BE ACTIVE, WITH THE PROGRAM VERIFYING THE TRANSITION OF THE SIGNAL. THE PROGRAM THEN SUPPLIES A SERIES OF BIT CLOCKS AND VERIFIES THAT "PROM STROBE" NEVER GOES ACTIVE.

PROBABLE FAULT:

1. CS MODULE IF LFS FAILS.
2. DS MODULE IF PROM STROBE FAILS.

TEST 16 SYNC DETECTION TEST

PURPOSE:

TO VERIFY THAT THE APPROPRIATE SYNC PATTERN IS RECOGNIZED.

PROCEDURE:

THE TEST EXECUTES A READ COMMAND IN MAINTENANCE MODE, USING BIT 10 OF THE MAINTENANCE REGISTER (RMMR1) TO SIMULATE

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

THE SYNC BIT STREAM, AND USING PROM STROBE TO DETERMINE IF THE SYNC PATTERN HAS BEEN DETECTED.

THE SYNC PATTERN IS 00011001, WITH THE LEFT MOST BIT REPRESENTING THE LAST BIT OF THE STREAM.

PROBABLE FAULT:

- 1. DS MODULE
- 2. CS MODULE

TEST 17 ABORT SYNC DETECTION TEST

PURPOSE:

TO VERIFY THAT 'WORD COUNT INHIBIT' IS RESET IF A 'DRIVE TIMING ERROR' OCCURS DURING SYNC DETECTION.

PROCEDURE:

A READ COMMAND IS INITIATED IN MAINTENANCE MODE. WHEN 'LOOKING FOR SYNC' GOES ACTIVE, THE TEST FORCES A DRIVE TIMING ERROR AND USES PROM STROBE TO VERIFY THAT 'WORD COUNT INHIBIT' HAS BEEN RESET.

PROBABLE FAULT:

- 1. DS MODULE

TEST 20 SYNC GENERATION TEST

PURPOSE:

TO VERIFY THAT THE APPROPRIATE SYNC PATTERN IS GENERATED DURING A FORMAT OPERATION.

PROCEDURE:

THE TEST EXECUTES A WRITE HEADER AND DATA COMMAND IN MAINTENANCE MODE, AND VERIFIES THAT THE CORRECT SYNC PATTERN IS GENERATED BY MONITORING WRITE DATA AT THE MAINTENANCE REGISTER (RMMR1).

PROBABLE FAULT:

- 1. DS 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

TEST 21 WRITE HEADER TEST**PURPOSE:**

TEST THE OPERATION OF (1) THE DATA BUFFER AND SHIFT REGISTER AS WELL AS (2) THE ECC GENERATION DURING WRITE.

PROCEDURE:

A WRITE HEADER AND DATA COMMAND IS EXECUTED IN MAINTENANCE MODE. THE TEST VERIFIES HEADER WORDS ONE AND TWO AND THE CRC CHARACTER USING THE WRITE DATA BIT OF THE MAINTENANCE REGISTER.

THE TEST USES CYLINDER 560., TRACK 13., SECTOR 30. AND 16 BIT FORMAT, WHICH CORRESPOND TO THE FOLLOWING:

HEADER:

WORD 1 - 1101001000110000

WORD 2 - 0000110100011110

PROBABLE FAULT:

1. DS MODULE OR MASSBUS MODULE

TEST 22 HEADER COMPARE TEST**PURPOSE:**

TO CHECK THE OPERATION OF (1) THE SHIFT REGISTER AND DATA BUFFER AS WELL AS THE (2) CRC GENERATOR DURING READ.

PROCEDURE:

THE TEST EXECUTES A READ HEADER AND DATA COMMAND IN MAINTENANCE MODE USING BIT 10 OF THE MAINTENANCE REGISTER (RMMR1) TO SIMULATE THE DATA BITS FOR THE FIRST AND SECOND HEADER WORDS AS WELL AS THE CRC CHARACTER. THE CRC CHARACTER IS IS FAULTED, AND THE TEST VERIFIES THAT A CRC ERROR IS DETECTED. ADDITIONALLY, THE TEST VERIFIES THAT HEADER WORDS ONE AND TWO ARE CORRECTLY TRANSFERRED TO MEMORY. THE TEST USES THE SAME HEADER AS IN THE PREVIOUS TEST EXCEPT THAT BIT 15 IS INVERTED.

PROBABLE FAULT:

1. DS OR IF MODULE IF CRC ERROR NOT DETECTED;

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

2. DS OR MASSBUS MODULE IF DATA INCORRECT

TEST 23 ECC GENERATION TEST

PURPOSE:

TO CHECK ECC OPERATION DURING WRITE.

PROCEDURE:

A WRITE DATA COMMAND IS EXECUTED IN MAINTENANCE MODE. ALL ONES DATA FIELD IS USED, AND THE TEST VERIFIES THE ECC CHARACTER VIA THE WRITE DATA BIT OF THE MAINTENANCE REGISTER. THE DATA FIELD IS NOT VERIFIED.

PROBABLE FAULT:

1. DS MODULE

TEST 24 ECC DETECTION TEST

PURPOSE:

TO CHECK THE ECC GENERATION DURING READ.

PROCEDURE:

THE TEST EXECUTES A READ DATA COMMAND IN MAINTENANCE MODE, AN ALL ONES DATA FIELD IS USED, HOWEVER THE LAST DATA WORD IS FAULTED.

THE TEST VERIFIES (1) THAT AN ECC ERROR IS DETECTED AND THAT (2) THE POSITION AND PATTERN REGISTERS ARE VALID.

PROBABLE FAULT:

1. DS MODULE

1
695
696

:*LAST REVISION 07-AUG-81

.TITLE CZRNC AO RM80 DSKLS PT2
:*COPYRIGHT (C) 1982
:*DIGITAL EQUIPMENT CORPORATION
:*COLORADO SPGS., CO. 80919

:*PROGRAM BY MIKE LEAVITT

:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81

697

.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>
7	TN128
6	TN64
5	TN32
4	TN16
3	TN8
2	TN4
1	TN2
0	TN1

698

.SBTTL BASIC DEFINITIONS

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

001100
104000
000004

STACK = 1100

ERROR = EMT

SCOPE = IOT

::BASIC DEFINITION OF ERROR CALL

::BASIC DEFINITION OF SCOPE CALL

:*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

HT = 11

LF = 12

CR = 15

CRLF = 200

PS = 177776

PSW=PS

STKLMT = 177774

PIRQ = 177772

DSWR = 177570

DDISP = 177570

::CODE FOR HORIZONTAL TAB

::CODE FOR LINE FEED

::CODE FOR CARRIAGE RETURN

::CODE FOR CARRIAGE RETURN-LINE FEED

::PROCESSOR STATUS WORD

::STACK LIMIT REGISTER

::PROGRAM INTERRUPT REQUEST REGISTER

::HARDWARE SWITCH REGISTER

::HARDWARE DISPLAY REGISTER

:*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003
000004

R0 = %0

R1 = %1

R2 = %2

R3 = %3

R4 = %4

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::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
    
```

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

.SBTTL RM80 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
    
```

732	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
733	000050	WCD = 000050	:WRITE CHECK DATA COMMAND
734	000052	WCH = 000052	:WRITE CHECK HEADER AND DATA
735	000054	ILF54 = 000054	:ILLEGAL COMMAND
736	000056	ILF56 = 000056	:ILLEGAL COMMAND
737	000060	WD = 000060	:WRITE DATA COMMAND
738	000062	WH = 000062	:WRITE HEADER AND DATA COMMAND
739	000064	ILF64 = 000064	:ILLEGAL COMMAND
740	000066	ILF66 = 000066	:ILLEGAL COMMAND
741	000070	RD = 000070	:READ DATA COMMAND
742	000072	RH = 000072	:READ HEADER AND DATA COMMAND
743	000074	ILF74 = 000074	:ILLEGAL COMMAND
744	000076	ILF76 = 000076	:ILLEGAL COMMAND
745			
746		:*RMDA DISK ADDRESS REGISTER	
747			
748		:TRACK ADDRESS DEFINITIONS	
749	004000	TAB = BIT11	:TRACK ADDRESS 8.
750	002000	TA4 = BIT10	:TRACK ADDRESS 4
751	001000	TA2 = BIT09	:TRACK ADDRESS 2
752	000400	TA1 = BIT08	:TRACK ADDRESS 1
753			
754		:SECTOR ADDRESS DEFINITIONS	
755	000020	SA16 = BIT04	:SECTOR ADDRESS 16.
756	000010	SA8 = BIT03	:SECTOR ADDRESS 8.
757	000004	SA4 = BIT02	:SECTOR ADDRESS 4
758	000002	SA2 = BIT01	:SECTOR ADDRESS 2
759	000001	SA1 = BIT00	:SECTOR ADDRESS 1
760			
761		:TRACK & SECTOR MASKS	
762	177400	TADMSK = 177400	:TRACK ADDRESS MASK
763	000377	SADMSK = 000377	:SECTOR ADDRESS MASK
764			
765		:*RMDS DRIVE STATUS REGISTER	
766			
767	100000	ATA = BIT15	:ATTENTION ACTIVE
768	040000	ERR = BIT14	:COMPOSITE ERROR
769	020000	PIP = BIT13	:POSITIONING IN PROGRESS
770	010000	MOL = BIT12	:MEDIUM ON LINE
771	004000	WRL = BIT11	:WRITE LOCK
772	002000	LBT = BIT10	:LAST BLOCK TRANSFERRED
773	001000	PGM = BIT09	:PROGRAMMABLE
774	000400	DPR = BIT08	:DRIVE PRESENT
775	000200	DRY = BIT07	:DRIVE READY
776	000100	VV = BIT06	:VOLUME VALID
777	000001	OM = BIT00	:OFFSET MODE ACTIVE
778			
779		:*RMER1 ERROR REGISTER #1	
780			
781	100000	DCK = BIT15	:DATA CHECK ERROR

782	040000	UNS	= BIT14	:DRIVE UNSAFE
783	020000	OPI	= BIT13	:OPERATION INCOMPLETE
784	010000	DTE	= BIT12	:DRIVE TIMING ERROR
785	004000	WLE	= BIT11	:WRITE LOCK ERROR
786	002000	IAE	= BIT10	:INVALID ADDRESS ERROR
787	001000	AOE	= BIT09	:ADDRESS OVERFLOW ERROR
788	000400	HCRC	= BIT08	:HEADER CRC ERROR
789	000200	HCE	= BIT07	:HEADER COMPARE ERROR
790	000100	ECH	= BIT06	:ECC 'HARD' ERROR
791	000040	WCF	= BIT05	:WRITE CLOCK FAILURE
792	000020	FER	= BIT04	:FORMAT ERROR
793	000010	PAR	= BIT03	:PARITY ERROR
794	000004	RMR	= BIT02	:REGISTER MODIFICATION REFUSED
795	000002	ILR	= BIT01	:ILLEGAL REGISTER
796	000001	ILF	= BIT00	:ILLEGAL FUNCTION
797				
798	115760	NDTMSK	= DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER	
799			;'NDTMSK' IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA	
800			:COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS	
801				
802				
803				
804	000377	ATNMSK	= 377	:MASK FOR ATTENTION BITS
805				
806				
807				
808	002000	SC4	= BIT10	:SECTOR COUNT = 16
809	001000	SC3	= BIT09	:SECTOR COUNT = 8
810	000400	SC2	= BIT08	:SECTOR COUNT = 4
811	000200	SC1	= BIT07	:SECTOR COUNT = 2
812	000100	SC0	= BIT06	:SECTOR COUNT = 1
813				
814	003700	SCTMSK	= 003700	:SECTOR COUNT MASK
815				
816				
817				
818				
819	100000	DBCK	= BIT15	:DEBUG CLOCK
820	040000	DBEN	= BIT14	:DEBUG CLOCK ENABLE
821	020000	DEBL	= BIT13	:DIAGNOSTIC END OF BLOCK
822	010000	MSEN	= BIT12	:SEARCH TIMEOUT ENABLE
823	004000	MCLK	= BIT11	:MAINTENANCE CLOCK
824	002000	MRD	= BIT10	:READ DATA
825	001000	MUR	= BIT09	:UNIT READY
826	000400	MOC	= BIT08	:ON CYLINDER
827	000200	MSER	= BIT07	:SEEK ERROR
828	000100	MDF	= BIT06	:DRIVE FAULT
829	000040	MS	= BIT05	:SECTOR PULSE
830	000010	MWP	= BIT03	:WRITE PROTECT
831	000004	MI	= BIT02	:INDEX PULSE
832	000002	MSC	= BIT01	:SECTOR COMPARE
833	000001	DMD	= BIT00	:DIAGNOSTIC MODE
834				
835				
836	100000	OCC	= BIT15	:OCCUPIED
837	040000	RG	= BIT14	:RUN AND GO
838	020000	EBL	= BIT13	:END OF BLOCK

839	010000	REX	= BIT12	:EXCEPTION
840	004000	ESRC	= BIT11	:ENABLE SEARCH
841	002000	PLFS	= BIT10	:LOOKING FOR SYNC
842	001000	ECRC	= BIT09	:ENABLE CRC OUT
843	000400	PDA	= BIT08	:DATA AREA
844	000200	PHA	= BIT07	:HEADER AREA
845	000100	CONT	= BIT06	:CONTINUE
846	000040	WC	= BIT05	:WORD CLOCK
847	000020	EECC	= BIT04	:ENABLE ECC OUT
848	000010	MWD	= BIT03	:WRITE DATA BIT
849	000004	LS	= BIT02	:LAST SECTOR
850	000002	LST	= BIT01	:LAST SECTOR AND TRACK
851	000001	DMD	= BIT00	:DIAGNOSTIC MODE
852	051401	MR1AAA	= DMD!MUR!DBEN!MOC!MSEN	
853				
854		:*RMDT DRIVE TYPE REGISTER		
855				
856	100000	NSA	= BIT15	:NOT SECTOR ADDRESSED = 0
857	040000	TAP	= BIT14	:TAPE DRIVE = 0
858	020000	MOH	= BIT13	:MOVING HEAD = 1
859	004000	DRQ	= BIT11	:DRIVE REQUEST REQUIRED
860				
861	020026	SNGPRT	= 020026	:SINGLE PORT DRIVE TYPE
862	024026	DULPRT	= 024026	:DUAL PORT DRIVE TYPE
863				
864		:*RMOF OFFSET REGISTER		
865				
866	010000	FMT16	= BIT12	:16 BIT WORD FORMAT
867	004000	ECI	= BIT11	:ECC INHIBIT
868	002000	HCI	= BIT10	:HEADER COMPARE INHIBIT
869	001000	SSEI	= BIT09	:SKIP SECTOR ERROR INHIBIT
870	000200	OFD	= BIT07	:OFFSET FORWARD
871	160577	XNUOF	= 160577	:UNUSED BITS OF RMOF
872				
873		:*RMDC DESIRED CYLINDER ADDRESS REGISTER		
874				
875	001777	CYLSK	= 001777	:MASK FOR CYLINDER ADDRESS
876	176000	XNUDC	= 176000	:UNUSED BITS OF RMDC
877				
878		:*RMR2 MAINTENANCE REGISTER #2		
879				
880		:READ ONLY BITS		
881	100000	RQA	= BIT15	:PORT A REQUEST
882	040000	RQB	= BIT14	:PORT B REQUEST
883	020000	TAG	= BIT13	:TAG CONTROL
884	010000	TST	= BIT12	:COMMAND SEQUENCE TEST BIT
885	004000	CC	= BIT11	:CONTROL OR CYLINDER TAG
886	002000	CH	= BIT10	:CONTROL OR HEAD TAG
889	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

```

890          000001          BB00    = BIT00          ;TAG BUS
891          ;*RMER2 ERROR REGISTER 2
892
893          100000          BSE     = BIT15          ;BAD SECTOR ERROR
894          040000          SKI     = BIT14          ;SEEK INCOMPLETE
895          020000          OPE     = BIT13          ;OPERATOR PLUG ERROR
896          010000          IVC     = BIT12          ;INVALID COMMAND ERROR
897          004000          LSC     = BIT11          ;LOSS OF SYSTEM CLOCK
898          002000          LBC     = BIT10          ;LOSS OF BIT CLOCK
899          000200          DVC     = BIT07          ;DEVICE CHECK
900          000040          SSE     = BIT05          ;SKIP SECTOR ERROR
901          000010          DPE     = BIT03          ;DATA PARITY ERROR
902          001527          XNUER2  = 001527        ;UNUSED BITS OF RMER2
903
904          .SBTTL PROGRAM MNEMONICS
905
906          100000          MSE     = BIT15          ;MANUFACTURING DETECTED SECTOR ERROR
907          040000          USE     = BIT14          ;USER DETECTED SECTOR ERROR
908          020000          SSF     = BIT13          ;SKIP SECTOR FAILURE
909
910          .SBTTL RM80 REGISTER INDEX VALUES
911
912          000000          RMCS1   = 00          ;CONTROL STATUS REGISTER #1
913          000006          RMDA    = 06          ;DISK ADDRESS REGISTER
914          000012          RMDS    = 12          ;DRIVE STATUS REGISTER
915          000014          RMER1   = 14          ;ERROR REGISTER #1
916          000016          RMAS    = 16          ;ATTENTION SUMMARY REGISTER
917          000020          RMLA    = 20          ;LOOK AHEAD REGISTER
918          000024          RMMR1   = 24          ;MAINTENANCE REGISTER
919          000026          RMDT    = 26          ;DRIVE TYPE REGISTER
920          000030          RMSN    = 30          ;SERIAL NUMBER REGISTER
921          000032          RMOF    = 32          ;OFFSET REGISTER
922          000034          RMDC    = 34          ;DESIRED CYLINDER REGISTER
923          000036          RMHR    = 36          ;HOLDING REGISTER
924          000040          RMMR2   = 40          ;MAINTENANCE REGISTER #2
925          000042          RMER2   = 42          ;ERROR REGISTER #2
926          000044          RMEC1   = 44          ;ECC POSITION REGISTER
927          000046          RMEC2   = 46          ;ECC PATTERN REGISTER
930          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
931
932
933          000077          IDXMSK  = 77          ;MASK FOR REGISTER INDEX NUMBER
934
935          .SBTTL RH CONTROLLER REGISTER BIT DEFINITIONS
936

```

```

937          ;*RMCS1 CONTROL STATUS REGISTER #1
938
939          100000      SC      = BIT15      ;SPECIAL CONDITION-READ ONLY
940          040000      TRE      = BIT14      ;TRANSFER ERROR
941          020000      MCPE     = BIT13      ;MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
942          002000      PSEL     = BIT10      ;PORT B SELECT
943          001000      A17      = BIT09      ;ADDRESS EXTENSION
944          000400      A16      = BIT08      ;ADDRESS EXTENSION
945          000200      RDY      = BIT07      ;READY-READ ONLY
946          000100      IE       = BIT06      ;INTERRUPT ENABLE
947
948          ;*RMCS2 RH CONTROL STATUS REGISTER #2
949
950          100000      DLT       = BIT15      ;DATA LATE-READ ONLY
951          040000      WCE       = BIT14      ;WRITE CHECK ERROR-READ ONLY
952          020000      UPE       = BIT13      ;UNIBUS PARITY ERROR
953          010000      NED       = BIT12      ;NONEXISTANT DRIVE-READ ONLY
954          004000      NEM       = BIT11      ;NONEXISTANT MEMORY-READ ONLY
955          002000      PGE       = BIT10      ;PROGRAM ERROR-READ ONLY
956          001000      MXF       = BIT09      ;MISSED TRANSFER
957          000400      MDPE      = BIT08      ;MASSBUS DATA BUS PARITY ERROR-READ ONLY
958          000200      OR        = BIT07      ;OUTPUT READY-READ ONLY
959          000100      IR        = BIT06      ;INPUT READY-READ ONLY
960          000040      CLR       = BIT05      ;CONTROLLER CLEAR
961          000020      PAT       = BIT04      ;PARITY TEST
962          000010      BAI       = BIT03      ;UNIBUS ADDRESS INCREMENT INHIBIT
963          000004      U2        = BIT02      ;UNIT SELECT
964          000002      U1        = BIT01      ;UNIT SELECT
965          000001      U0        = BIT00      ;UNIT SELECT
966
967          ;UNIT SELECT MASK
968
969          000007      UNTMSK    = 7          ;UNIT SELECT MASK
970
971          ;*RMCS3 RH70 CONTROL STATUS REGISTER #3
972
973          100000      APE       = BIT15      ;ADDRESS PARITY ERROR
974          040000      DPEHI     = BIT14      ;DATA PARITY ERROR HIGH WORD
975          020000      DPELO    = BIT13      ;DATA PARITY ERROR LOW WORD
976          010000      WCEHI     = BIT12      ;WRITE CHECK ERROR HIGH WORD
977          004000      WCELO    = BIT11      ;WRITE CHECK ERROR LOW WORD
978          002000      DBL       = BIT10      ;DOUBLE WORD TRANSFER
979          000100      IE        = BIT06      ;INTERRUPT ENABLE
980          000010      IPCK3     = BIT03      ;INVERT PARITY CHECK
981          000004      IPCK2     = BIT02      ;INVERT PARITY CHECK
982          000002      IPCK1     = BIT01      ;INVERT PARITY CHECK
983          000001      IPCK0     = BIT00      ;INVERT PARITY CHECK
984
985          .SBTTL  RH CONTROLLER REGISTER INDEX VALUES
986
987          000000      RMCS1     = 00        ;CONTROL, STATUS REGISTER #1
988          000002      RMC      = 02        ;WORD COUNT REGISTER
989          000004      RMA      = 04        ;BUS ADDRESS REGISTER
990          000010      RMCS2     = 10        ;CONTROL, STATUS REGISTER #2
991          000022      RMB      = 22        ;DATA BUFFER
992          000050      RMAE      = 50        ;BUS ADDRESS EXTENSION
993          000052      RMCS3     = 52        ;CONTROL, STATUS REGISTER #3

```

994
995
996
997

176700
120254

ABASE = 176700
AVECT1 = 120254

:UNIBUS ADDRESS
:UNIBUS VECTOR ADDRESS AND PRIORITY

1

.SBTTL TRAP CATCHER

000000

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

000174 000174
 000176 000000
 000176 000000

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

.SBTTL STARTING ADDRESS(ES)

000200 000137 002732

JMP @#START ;:JUMP TO STARTING ADDRESS OF PROGRAM

000204 000137 002722

JMP @#START1 ;CHANGE RH/RM BUS ADDRESS

2
 3
 4
 5

.SBTTL ACT11 HOOKS

:::*****

:::HOOKS REQUIRED BY ACT11

000046 000210
 000046 000046
 000052 021354
 000052 000052
 000052 000000
 000052 000210

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

11
 12
 13

001100

. =1100
 .SBTTL APT PARAMETER BLOCK

:::*****

:::SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

:::*****

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

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

:::*****

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

001100
 001100 000000
 001102 001222
 001104 000002
 001106 000002
 001110 000002
 001112 000042
 14 001114

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

0

.SBTTL COMMON TAGS

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

001114	001114			SCMTAG: .WORD	0	::START OF COMMON TAGS
001114	000000			STSTNM: .BYTE	0	::CONTAINS THE TEST NUMBER
001116	000			SERFLG: .BYTE	0	::CONTAINS ERROR FLAG
001117	000			SICNT: .WORD	0	::CONTAINS SUBTEST ITERATION COUNT
001120	000000			SLPADR: .WORD	0	::CONTAINS SCOPE LOOP ADDRESS
001122	000000			SLPERR: .WORD	0	::CONTAINS SCOPE RETURN FOR ERRORS
001124	000000			SERTTL: .WORD	0	::CONTAINS TOTAL ERRORS DETECTED
001126	000000			SITEMB: .BYTE	0	::CONTAINS ITEM CONTROL BYTE
001130	000			SERMAX: .BYTE	1	::CONTAINS MAX. ERRORS PER TEST
001131	001			SERRPC: .WORD	0	::CONTAINS PC OF LAST ERROR INSTRUCTION
001132	000000			SGDADR: .WORD	0	::CONTAINS ADDRESS OF 'GOOD' DATA
001134	000000			SBDADR: .WORD	0	::CONTAINS ADDRESS OF 'BAD' DATA
001136	000000			SGDDAT: .WORD	0	::CONTAINS 'GOOD' DATA
001140	000000			SBDDAT: .WORD	0	::CONTAINS 'BAD' DATA
001142	000000				0	::RESERVED--NOT TO BE USED
001144	000000				0	
001146	000000				0	
001150	000			SAUTOB: .BYTE	0	::AUTOMATIC MODE INDICATOR
001151	000			SINTAG: .BYTE	0	::INTERRUPT MODE INDICATOR
001152	000000				0	
001154	177570			SWR: .WORD	DSWR	::ADDRESS OF SWITCH REGISTER
001156	177570			DISPLAY: .WORD	DDISP	::ADDRESS OF DISPLAY REGISTER
001160	177560			STKS: 177560		::TTY KBD STATUS
001162	177562			STKB: 177562		::TTY KBD BUFFER
001164	177564			STPS: 177564		::TTY PRINTER STATUS REG. ADDRESS
001166	177566			STPB: 177566		::TTY PRINTER BUFFER REG. ADDRESS
001170	000			SNULL: .BYTE	0	::CONTAINS NULL CHARACTER FOR FILLS
001171	002			SFILLS: .BYTE	2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			SFILLC: .BYTE	12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000			STPFLG: .BYTE	0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000			STMP0: .WORD	0	::USER DEFINED
001176	000000			STMP1: .WORD	0	::USER DEFINED
001200	000000			STMP2: .WORD	0	::USER DEFINED
001202	000000			STMP3: .WORD	0	::USER DEFINED
001204	000000			STMP4: .WORD	0	::USER DEFINED
001206	000000			STIMES: 0		::MAX. NUMBER OF ITERATIONS
001210	000000			SESCAPE: 0		::ESCAPE ON ERROR ADDRESS
001212	207	377	377	SBELL: .ASCIZ	<207><377><377>	::CODE FOR BELL
001216	077			SQUES: .ASCII	/?/	::QUESTION MARK
001217	015			SCRLF: .ASCII	<15>	::CARRIAGE RETURN
001220	012	000		SLF: .ASCIZ	<12>	::LINE FEED

.SBTTL APT MAILBOX-ETABLE

001222				SMAIL: .WORD	AMSGTY	::APT MAILBOX
001222	000000			SMSGTY: .WORD	AFATAL	::MESSAGE TYPE CODE
001224	000000			SFATAL: .WORD	ATESTN	::FATAL ERROR NUMBER
001226	000000			STESTN: .WORD		::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			

.SBTTL USER DEFINED TAGS

001326 000000
 001330 000000
 001332 000000

AUTSIZ: .WORD 0 ;ALLOW AUTO DRIVE SIZING = 0, USE MANUALLY INPUT DRIVES = 1
 CHGADR: .WORD 0 ;CHANGE RH/RM BUS ADDRESS = -1, NO CHANGE = 0
 XXDP: .WORD 0 ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
 ;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
 ;'XXDP' DEVICE CODE FOR THE RM80.

001334 000
 001335 015

LSTRK: .BYTE 0 ;LO BYTE = 0
 .BYTE 13. ;HI BYTE, CONTAINS LAST TRACK ADDRESS OF UNIT
 ;UNDER TEST. RM80 = 13.

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

001336

GETBUF:

001336 000000
 001340 000000
 001342 000000
 001344 000000
 001346 000000
 001350 000000
 001352 000000
 001354 000000
 001356 000000
 001360 000000
 001362 000000
 001364 000000
 001366 000000
 001370 000000
 001372 000000
 001374 000000
 001376 000000
 001400 000000
 001402 000000
 001404 000000
 001406 000000
 001410 000000

:REGISTER INPUT BUFFER

RMCS1I: .WORD 0 ;CONTROL, STATUS REGISTER #1
 RMWCI: .WORD 0 ;WORD COUNT REGISTER
 RMBAI: .WORD 0 ;BUS ADDRESS REGISTER
 RMDAI: .WORD 0 ;DISK ADDRESS REGISTER
 RMCS2I: .WORD 0 ;CONTROL, STATUS REGISTER #2
 RMDSI: .WORD 0 ;DRIVE STATUS REGISTER
 RMER1I: .WORD 0 ;ERROR REGISTER #1
 RMAI: .WORD 0 ;ATTENTION SUMMARY REGISTER
 RMLAI: .WORD 0 ;LOOK AHEAD REGISTER
 RMDBI: .WORD 0 ;DATA BUFFER
 RMMR1I: .WORD 0 ;MAINTENANCE REGISTER #1
 RMDTI: .WORD 0 ;DRIVE TYPE REGISTER
 RMSNI: .WORD 0 ;SERIAL NUMBER REGISTER
 RMOFI: .WORD 0 ;OFFSET REGISTER
 RMDCI: .WORD 0 ;DESIRED CYLINDER REGISTER
 RMR1I: .WORD 0 ;HOLDING REGISTER
 RMMR2I: .WORD 0 ;MAINTENANCE REGISTER #2
 RMER2I: .WORD 0 ;ERROR REGISTER #2
 RMEC1I: .WORD 0 ;ECC POSITION REGISTER
 RMEC2I: .WORD 0 ;ECC PATTERN REGISTER
 RMBAEI: .WORD 0 ;BUS ADDRESS EXTENSION REGISTER
 RMCS3I: .WORD 0 ;CONTROL, STATUS REGISTER #3

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

001412

PUTBUF:

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

:REGISTER OUTPUT BUFFER

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

001440	000000	RMDTO: .WORD	0	:DRIVE TYPE REGISTER
001442	000000	RMSNO: .WORD	0	:SERIAL NUMBER REGISTER
001444	000000	RMOFO: .WORD	0	:OFFSET REGISTER
001446	000000	RMDCO: .WORD	0	:DESIRED CYLINDER REGISTER
001450	000000	RMHRO: .WORD	0	:HOLDING REGISTER
001452	000000	RMPR20: .WORD	0	:MAINTENANCE REGISTER #2
001454	000000	RMER20: .WORD	0	:ERROR REGISTER #2
001456	000000	RMEC10: .WORD	0	:ECC POSITION REGISTER
001460	000000	RMEC20: .WORD	0	:ECC PATTERN REGISTER
001462	000000	RMBAE0: .WORD	0	:BUS ADDRESS EXTENSION REGISTER
001464	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.

001466	000000	TSTQUE: .WORD	0	:CONTAINS DEVICE POINTER
001470		.BLKW	8.	:TEST QUE FOR DEVICES UNDER TEST
001510	000000	.WORD	0	:TABLE TERMINATOR GOES HERE WHEN :ALL 8. DEVICES ARE UNDER TEST.
001512	172540	SLPCSR: .WORD	172540	:KW11-P CONTROL + STATUS REGISTER
001514	172542	SLPCSB: .WORD	172542	:KW11-P COUNT SET BUFFER
001516	000104	SLPVEC: .WORD	104	:KW11-P INTERRUPT VECTOR
001520	000106	.WORD	106	
001522	177546	SLLCSR: .WORD	177546	:KW11-L CONTROL + STATUS REGISTER
001524	000100	SLLVEC: .WORD	100	:KW11-L INTERRUPT VECTOR
001526	000102	.WORD	102	
001530	000000	SPSW: .WORD		:STORAGE FOR PRIORITY
001532	000000	TIME: .WORD		:STORAGE FOR ELAPSED TIME
001534	000000	WATCH: .WORD		:STORAGE FOR REMAINING TIME
001536	000000	CLOCK: .WORD		:ADDRESS OF START CLOCK SUB
001540	000000	STOPCL: .WORD		:ADDRESS OF STOP CLOCK SUB

:PUT TAGS HERE

.SBTTL ERROR POINTER TABLE

:*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 :*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 :*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 :*NOTE1: IF SITEMB 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

001542

SERRTB:

1
2
3

:ERROR 1 CANNOT CLEAR NED STATUS

001542 032244
 001544 040152
 001546 040266
 001550 040322

EMT1
 EHT1
 EDT1
 EFT1

4
5
6

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

001552 032252
 001554 040156
 001556 040270
 001560 040324

EMT2
 EHT2
 EDT2
 EFT2

7
8
9

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

001562 032300
 001564 000000
 001566 000000
 001570 000000

EMT3
 0
 0
 0

10
11
12

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

001572 032320
 001574 000000
 001576 000000
 001600 000000

EMT4
 0
 0
 0

13
14
15

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

001602 032342
 001604 040162
 001606 040272
 001610 040326

EMT5
 EHT5
 EDT5
 EFT5

16
17

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

18	001612	032366	EMT6	
	001614	040162	EHT5	
	001616	040272	EDT5	
	001620	040326	EFT5	
19				
20			:ERROR 7	CANNOT WRITE/READ SHIFTING ONE BIT TO ALL BIT POSITIONS
21			:	OF DEVICE REGISTERS
22				
	001622	032410	EMT7	
	001624	040166	EHT7	
	001626	040272	EDT5	
	001630	040326	EFT5	
23				
24			:ERROR 10	REGISTER SELECT 1 APPEARS S-A-0
25				
	001632	032432	EMT10	
	001634	000000	0	
	001636	000000	0	
	001640	000000	0	
26				
27			:ERROR 11	REGISTER SELECT 1 APPEARS S-A-1
28				
	001642	032450	EMT11	
	001644	000000	0	
	001646	000000	0	
	001650	000000	0	
29				
30			:ERROR 12	REGISTER SELECT 2 APPEARS S-A-0
31				
	001652	032466	EMT12	
	001654	000000	0	
	001656	000000	0	
	001660	000000	0	
32				
33			:ERROR 13	REGISTER SELECT 2 APPEARS S-A-1
34				
	001662	032504	EMT13	
	001664	000000	0	
	001666	000000	0	
	001670	000000	0	
35				
36			:ERROR 14	REGISTER SELECT 4 APPEARS S-A-0
37				
	001672	032522	EMT14	
	001674	000000	0	
	001676	000000	0	
	001700	000000	0	
38				
39			:ERROR 15	REGISTER SELECT 4 APPEARS S-A-1

40	001702	032540		EMT15
	001704	000000		0
	001706	000000		0
	001710	000000		0
41			:ERROR 16	REGISTER SELECT 8 APPEARS S-A-0
42				
43	001712	032556		EMT16
	001714	000000		0
	001716	000000		0
	001720	000000		0
44			:ERROR 17	REGISTER SELECT 8 APPEARS S-A-1
45				
46	001722	032574		EMT17
	001724	000000		0
	001726	000000		0
	001730	000000		0
47			:ERROR 20	OPI SET WITH SEACH TIMOUT DISABLED
48				
49	001732	032612		EMT20
	001734	040152		EHT1
	001736	040266		EDT1
	001740	040322		EFT1
50			:ERROR 21	OPI SET WITH SECTOR PULSE,SECTOR COMPARE WAS RESET
51				
52	001742	032634		EMT21
	001744	040152		EHT1
	001746	040266		EDT1
	001750	040322		EFT1
53			:ERROR 22	DTE SET WITH INDEX PULSE,SEARCH WAS NOT ENABLED
54				
55	001752	032656		EMT22
	001754	040152		EHT1
	001756	040266		EDT1
	001760	040322		EFT1
56			:ERROR 23	DTE SET WITH SECTOR PULSE,SECTOR COMPARE WAS RESET
57				
58	001762	032702		EMT23
	001764	040152		EHT1
	001766	040266		EDT1
	001770	040322		EFT1
59			:ERROR 24	DTE DID NOT SET WITH SECTOR PULSE,SECTOR COMPARE WAS SET
60				
61				

	001772	032726	EMT24	
	001774	040152	EHT1	
	001776	040266	EDT1	
	002000	040322	EFT1	
62				
63			:ERROR 25	DTE SET WITH SECTOR PULSE DURING FORMAT CHANGE
64				
	002002	032752	EMT25	
	002004	040152	EHT1	
	002006	040266	EDT1	
	002010	040322	EFT1	
65				
66			:ERROR 26	MBA CLR L IS STUCK ACTIVE
67				
	002012	032772	EMT26	
	002014	000000	0	
	002016	000000	0	
	002020	000000	0	
68				
69			:ERROR 27	COULD NOT SET DTE AFTER FORMAT CHANGE
70				
	002022	033024	EMT27	
	002024	040152	EHT1	
	002026	040266	EDT1	
	002030	040322	EFT1	
71				
72			:ERROR 30	CANNOT SET PROM STROBE WITH BIT CLOCK
73				
	002032	033044	EMT30	
	002034	040152	EHT1	
	002036	040266	EDT1	
	002040	040322	EFT1	
74				
75			:ERROR 31	CANNOT CLEAR RMER1,DTE
76				
	002042	033064	EMT31	
	002044	040152	EHT1	
	002046	040266	EDT1	
	002050	040322	EFT1	
77				
78			:ERROR 32	PROM STROBE RESET EARLY
79				
	002052	033100	EMT32	
	002054	040152	EHT1	
	002056	040266	EDT1	
	002060	040322	EFT1	
80				
81			:ERROR 33	PROM STROBE SET EARLY
82				
	002062	033112	EMT33	

	002064	040152	EHT1	
	002066	040266	EDT1	
	002070	040322	EFT1	
83				
84				:ERROR 34 LOOKING FOR SYNC SET EARLY
85				
	002072	033124	EMT34	
	002074	040152	EHT1	
	002076	040266	EDT1	
	002100	040322	EFT1	
86				
87				:ERROR 35 LOOKING FOR SYNC DID NOT SET
88				
	002102	033136	EMT35	
	002104	040152	EHT1	
	002106	040266	EDT1	
	002110	040322	EFT1	
89				
90				:ERROR 36 PROM STROBE SET WHILE LOOKING FOR SYNC
91				
	002112	033150	EMT36	
	002114	040152	EHT1	
	002116	040266	EDT1	
	002120	040322	EFT1	
92				
93				:ERROR 37 SYNC DETECTED WITH WRONG PATTERN
94				
	002122	033170	EMT37	
	002124	040226	EHT115	
	002126	040312	EDT115	
	002130	040340	EFT115	
95				
96				:ERROR 40 SYNC NOT DETECTED
97				
	002132	033220	EMT40	
	002134	040152	EHT1	
	002136	040266	EDT1	
	002140	040322	EFT1	
98				
99				:ERROR 41 DRIVE TIMING ERROR DID NOT CLEAR LOOKING FOR SYNC
100				
	002142	033242	EMT41	
	002144	040152	EHT1	
	002146	040266	EDT1	
	002150	040322	EFT1	
101				
102				:ERROR 42 WRITE GATE DID NOT COME ON OR RESET EARLY
103				
	002152	033270	EMT42	
	002154	040152	EHT1	

	002156	040266	EDT1	
	002160	040322	EFT1	
104				
105			:ERROR 43	INCORRECT SYNC PATTERN DURING HEADER
106				
	002162	033306	EMT43	
	002164	000000	0	
	002166	000000	0	
	002170	000000	0	
107				
108			:ERROR 44	INCORRECT SYNC PATTERN DURING HEADER
109				
	002172	033306	EMT43	
	002174	040166	EHT7	
	002176	040272	EDT5	
	002200	040326	EFT5	
110				
111			:ERROR 45	HEADER AREA DID NOT COME ON OR RESET EARLY
112				
	002202	033340	EMT45	
	002204	040152	EHT1	
	002206	040266	EDT1	
	002210	040322	EFT1	
113				
114			:ERROR 46	CRC ENABLE DID NOT SET
115				
	002212	033354	EMT46	
	002214	040152	EHT1	
	002216	040266	EDT1	
	002220	040322	EFT1	
116				
117			:ERROR 47	INCORRECT HEADER GENERATED DURING WRITE
118				
	002222	033370	EMT47	
	002224	040172	EHT47	
	002226	040274	EDT47	
	002230	040322	EFT1	
119				
120			:ERROR 50	READ GATE INCORRECT DURING HEADER AREA
121				
	002232	033406	EMT50	
	002234	040152	EHT1	
	002236	040266	EDT1	
	002240	040322	EFT1	
122				
123			:ERROR 51	UNEXPECTED HEADER ERROR DURING DIAGNOSTIC MODE
124				
	002242	033424	EMT51	
	002244	040152	EHT1	
	002246	040266	EDT1	

	002250	040322	EFT1	
125				
126			:ERROR 52	INCORRECT HEADER READ IN DIAGNOSTIC MODE
127				
	002252	033440	EMT52	
	002254	040176	EHT52	
	002256	040276	EDT52	
	002260	040330	EFT57	
128				
129			:ERROR 53	INCORRECT TAG BUS DURING DATA COMMAND
130				
	002262	040030	EMT276	
	002264	040152	EHT1	
	002266	040266	EDT1	
	002270	040322	EFT1	
131				
132			:ERROR 54	DATA TIMING SEQUENCER CONTROLS INCORRECT DURING DATA COMMAND
133				
	002272	033472	EMT54	
	002274	040152	EHT1	
	002276	040266	EDT1	
	002300	040322	EFT1	
134				
135			:ERROR 55	DATA AREA DID NOT COME ON OR RESET EARLY
136				
	002302	033510	EMT55	
	002304	040152	EHT1	
	002306	040266	EDT1	
	002310	040322	EFT1	
137				
138			:ERROR 56	ECC ENABLE DID NOT SET
139				
	002312	033526	EMT56	
	002314	040152	EHT1	
	002316	040266	EDT1	
	002320	040322	EFT1	
140				
141			:ERROR 57	DEVICE IS NOT AN RM80
142				
	002322	033542	EMT57	
	002324	040202	EHT57	
	002326	040300	EDT57	
	002330	040330	EFT57	
143				
144			:ERROR 60	DEVICE AVAILABLE IS NOT SET
145				
	002332	033556	EMT60	
	002334	040152	EHT1	
	002336	040266	EDT1	
	002340	040322	EFT1	

146				
147			:ERROR 61	INCORRECT ECC PATTERN GENERATED DURING WRITE
148				
	002342	033572	EMT61	
	002344	040206	EHT61	
	002346	040302	EDT61	
	002350	040330	EFT57	
149				
150			:ERROR 62	CANNOT CLEAR PROM STROBE WITH BIT CLOCK
151				
	002352	033610	EMT62	
	002354	040152	EHT1	
	002356	040266	EDT1	
	002360	040322	EFT1	
152				
153			:ERROR 63	DATA AREA DID NOT RESET
154				
	002362	033626	EMT63	
	002364	040152	EHT1	
	002366	040266	EDT1	
	002370	040322	EFT1	
155				
156			:ERROR 64	UNEXPECTED ECC ERROR IN DIAGNOSTIC MODE
157				
	002372	033640	EMT64	
	002374	040152	EHT1	
	002376	040266	EDT1	
	002400	040322	EFT1	
158				
159			:ERROR 65	INCORRECT DATA TRANSFERRED TO MEMORY
160				
	002402	033654	EMT65	
	002404	040152	EHT1	
	002406	040266	EDT1	
	002410	040322	EFT1	
161				
162			:ERROR 66	
163				
	002412	033666	EMT66	
	002414	000000	0	
	002416	000000	0	
	002420	000000	0	
164				
165			:ERROR 67	
166				
	002422	033702	EMT67	
	002424	000000	0	
	002426	000000	0	
	002430	000000	0	

167
168 :ERROR 70
169
002432 033720 EMT70
002434 000000 0
002436 000000 0
002440 000000 0

170
171 :ERROR 71
172
002442 033740 EMT71
002444 000000 0
002446 000000 0
002450 000000 0

173
174 :ERROR 72
175
002452 033764 EMT72
002454 000000 0
002456 000000 0
002460 000000 0

176
177 :ERROR 73
178
002462 034010 EMT73
002464 000000 0
002466 000000 0
002470 000000 0

179
180 :ERROR 74
181
002472 034030 EMT74
002474 000000 0
002476 000000 0
002500 000000 0

182
183 :ERROR 75
184
002502 034040 EMT75
002504 000000 0
002506 000000 0
002510 000000 0

185
186 :ERROR 76
187
002512 034052 EMT76
002514 000000 0
002516 000000 0
002520 000000 0

188

189		:ERROR 77	
190	002522 034066	EMT77	
	002524 000000	0	
	002526 000000	0	
	002530 000000	0	
191		:ERROR 100	CANT SET VOLUME VALID
192		EMT170	
193	002532 035674	EHT150	
	002534 040252	EDT115	
	002536 040312	EFT115	
	002540 040340		
194		:ERROR 101	RUN AND GO WONT SET
195		EMT275	
196	002542 040010	EHT1	
	002544 040152	EDT1	
	002546 040266	EFT1	
	002550 040322		
197		:ERROR 102	SEARCH ENABLE WONT SET
198		EMT102	
199	002552 034142	EHT1	
	002554 040152	EDT1	
	002556 040266	EFT1	
	002560 040322		
200		:ERROR 103	OCCUPIED IS INCORRECT FOR FUNCTION CODE
201		EMT173	
202	002562 035740	EHT150	
	002564 040252	EDT115	
	002566 040312	EFT115	
	002570 040340		
203		:ERROR 104	READ IN PRESET DIDNT CLEAR RMDA, RMDC OR RMOF
204		EMT174	
205	002572 035762	0	
	002574 000000	0	
	002576 000000	0	
	002600 000000		
206		:ERROR 105	READ IN PRESET DIDNT CLEAR RMOF
207		EMT175	
208	002602 036010	EHT1	
	002604 040152	EDT1	
	002606 040266	EFT1	
	002610 040322		
209		:ERROR 106	READ IN PRESET DIDNT CLEAR RMDA
210			

211

002612 036026
002614 040152
002616 040266
002620 040322

EMT176
EHT1
EDT1
EFT1

212

213

214

002622 040134
002624 040152
002626 040266
002630 040322

:ERROR 107 READ IN PRESET DIDNT CLEAR RMDC

EMT302
EHT1
EDT1
EFT1

215

216

217

002632 036046
002634 040152
002636 040266
002640 040322

:ERROR 110 CANT SET OFFSET MODE BY OFFSET COMMAND

EMT200
EHT1
EDT1
EFT1

218

219

:PUT ERROR TABLE HERE

```

1      ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3      002642 011600
4      002644 005740
5      002646 022626
6      002650 104401 002656
       002654 000417
       002714
7      002714 010046
8      002716 104402
9      002720 000240
10
11
12
13      .SBTTL START OF PROGRAM
14      002722 012737 177777 001330 START1: MOV    #-1,CHGADR    ;CHANGE RH/RM BUS ADDRESS
15      002730 000402                BR      START2
16
17      002732 005037 001330 START:  CLR    CHGADR    ;NO CHANGE IN ADDRESS
18      002736 000240 START2: NOP
19      002740 005227 000000        INC     #0          ;TTY LOOP, WAIT FOR INCREMENT
20      002744 001375                BNE    #-4         ;OF WORD
21      002746 000005                RESET        ;RESET THE WORLD
22
23      .SBTTL INITIALIZE THE COMMON TAGS
       ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
       MOV    #SCMTAG,R6          ;;FIRST LOCATION TO BE CLEARED
       CLR    (R6)+              ;;CLEAR MEMORY LOCATION
       CMP    #SWR,R6            ;;DONE?
       BNE    -6                 ;;LOOP BACK IF NO
       MOV    #STACK,SP          ;;SETUP THE STACK POINTER
       ;;INITIALIZE A FEW VECTORS
       MOV    #SCOPE,@IOTVEC     ;;IOT VECTOR FOR SCOPE ROUTINE
       MOV    #340,@IOTVEC+2     ;;LEVEL 7
       MOV    #ERROR,@EMTVEC     ;;EMT VECTOR FOR ERROR ROUTINE
       MOV    #340,@EMTVEC+2     ;;LEVEL 7
       MOV    #TRAP,@TRAPVEC     ;;TRAP VECTOR FOR TRAP CALLS
       MOV    #340,@TRAPVEC+2    ;;LEVEL 7
       MOV    #SPURDN,@PURVEC    ;;POWER FAILURE VECTOR
       MOV    #340,@PURVEC+2    ;;LEVEL 7
       MOV    SENDCT,SEOPCT     ;;SETUP END-OF-PROGRAM COUNTER
       CLR    STINES            ;;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    @ERRVEC,-(SP)     ;;SAVE ERROR VECTOR
       MOV    #64,$@ERRVEC      ;;SET UP ERROR VECTOR
       MOV    #DSWR,$SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
       MOV    #DDISP,$DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
       CMP    #-1,$SWR          ;;TRY TO REFERENCE HARDWARE SWR
       BNE    66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
       ;;AND THE HARDWARE SWR IS NOT = -1
       BR     65$              ;;BRANCH IF NO TIMEOUT
    
```

```

003150 012716 003156      64$:  MOV    #65$, (SP)      ;;SET UP FOR TRAP RETURN
003154 000002
003156 012737 000176 001154 65$:  MOV    #SWREG, SWR      ;;POINT TO SOFTWARE SWR
003164 012737 000174 001156      MOV    #DISPRÉG, DISPLAY
003172 012637 000004      66$:  MOV    (SP)+, @#ERRVEC    ;;RESTORE ERROR VECTOR

003176 005037 001230      CLR    $PASS              ;;CLEAR PASS COUNT
003202 132737 000200 001243      BITB   #APTSIZE, $ENVM    ;;TEST USER SIZE UNDER APT
003210 001403      BEQ    67$                ;;YES, USE NON-APT SWITCH
003212 012737 001244 001154      MOV    #SSWREG, SWR      ;;NO, USE APT SWITCH REGISTER
003220

24      67$:  ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 003220 012737 002642 000004      MOV    #BADTMO, ERRVEC   ;;SETUP FOR UNEXPECTED TIMEOUT
26 003226 012737 000300 000006      MOV    #PR6, ERRVEC+2    ;;LEVEL 6
27
28      .SBTTL  TYPE PROGRAM NAME
      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
003234 005227 177777      INC    #-1                ;;FIRST TIME?
003240 001032      BNE    68$                ;;BRANCH IF NO
003242 022737 021354 000042      CMP    #SENDAD, @#42     ;;ACT-11?
003250 001426      BEQ    68$                ;;BRANCH IF YES
003252 104401 003260      TYPE   ,69$              ;;TYPE ASCIZ STRING
003256 000423      BR     68$                ;;GET OVER THE ASCIZ
      ;;69$: .ASCIZ <CRLF>@CZRNCAD - RMB0 DISKLESS TEST, PT 2@<CRLF>
003326      68$:  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
003326 005737 000042      TST    @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
003332 001012      BNE    70$                ;;BRANCH IF YES
003334 123727 001242 000001      CMPB   $ENV, #1          ;;ARE WE RUNNING UNDER APT?
003342 001406      BEQ    70$                ;;BRANCH IF YES
003344 023727 001154 000176      CMP    SWR, #SWREG       ;;SOFTWARE SWITCH REG SELECTED?
003352 001005      BNE    71$                ;;BRANCH IF NO
003354 104407      GTSWR                      ;;GET SOFT-SWR SETTINGS
003356 000403      BR     71$
003360 112737 000001 001150 70$:  MOVB   #1, $AUTOB        ;;SET AUTO-MODE INDICATOR
003366      71$:

29      ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
30      ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
31
32
33 003366 005037 001332      CLR    XXDP              ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
34 003372 122737 000016 000041      CMPB   #16, @#41         ;;LOADED FROM AN RMB0 ?
35 003400 001121      BNE    3$                ;;BR IF NOT
36 003402 013737 000040 001332      MOV    @#40, XXDP        ;;GET DEVICE INDICATOR AND NUMBER
37 003410 122737 000007 001332      CMPB   #7, XXDP         ;;IS IT A VALID NUMBER ?
38 003416 103002      BHIS   1$                ;;YES
39 003420 105037 001332      CLRB   XXDP              ;;NO, DEFAULT TO DRIVE 0
40 003424 005737 000042      1$:  TST    @#42              ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
41 003430 001425      BEQ    2$                ;;BR IF NEITHER
42 003432 104401 003440      TYPE   ,73$             ;;TYPE ASCIZ STRING
003436 000412      BR     72$              ;;GET OVER THE ASCIZ
      ;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
003464      72$:
43 003464 005046      CLR    -(SP)             ;;CLEAR WORD ON STACK
44 003466 113716 001332      MOVB   XXDP, (SP)        ;;GET DRIVE ADDRESS
45 003472 104403      TYPOS                      ;;TYPE THE ADDRESS
46 003474 001      .BYTE  1                ;;ONLY 1 CHARACTER

```

```

47 003475      000      .BYTE 0      ;SUPRESS LEADING ZEROS
48 003476 104401 001217  TYPE $CRLF  ;CR-LF
49 003502 000460      BR 3$      ;GET NUMBER OF DRIVES
50
51 003504 005227 177777 2$: INC #1      ;FIRST TIME THRU HERE ?
52 003510 001055      BNE 3$      ;NO
53 003512 104401 003520  TYPE 75$    ;:TYPE ASCIZ STRING
    003516 000410      BR 74$      ;:GET OVER THE ASCIZ
    ;:75$: .ASCIZ <CRLF>/TO TEST DRIVE /
    74$:
54 003540 005046      CLR -(SP)    ;CLEAR WORD ON STACK
55 003542 113716 001332  MOVB XXDP,(SP) ;GET DRIVE ADDRESS
56 003546 104403      TYPOS      ;TYPE DRIVE ADDRESS
57 003550      001      .BYTE 1      ;ONLY 1 CHARACTER
58 003551      000      .BYTE 0      ;SUPPRESS LEADING ZEROS
59 003552 104401 003560  TYPE 76$    ;:TYPE ASCIZ STRING
    003556 000432      BR 3$      ;:GET OVER THE ASCIZ
    ;:76$: .ASCIZ /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
    3$:
63
64 ;CHECK FOR AUTO MODE OR STANDALONE MODE
65 003644 005037 001326  CLR AUTSIZ  ;LET AUTO DRIVE SIZING OCCUR
66 003650 005737 000042  TST #42    ;RUNNING IN AUTO MODE ?
67 003654 001537      BEQ STANDALONE ;BR IF NO
68 003656 012737 000377 001300  MOV #377,$DEV  ;SET DEVICE MAP FOR ALL DRIVES
69
70 ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
71 003664      XSIZ:
72 003664 132737 000200 001243  BITB #BIT7,$ENVM ;SIZING ALLOWED ?
73 003672 001124      BNE 12$      ;NO
74
75 003674 005001      CLR R1      ;START FROM DRIVE 0
76 003676 013700 001276  MOV $BASE,R0 ;LOAD THE BASE ADDRESS
77 003702 104401 031770  TYPE ,SYSTAT ;TYPE 'UNIT STATUS:'
78
79 003706 136137 032234 001300 1$: BITB ATNTBL(R1),$DEV  ;IS DEVICE PRESENT IN MAP ?
80 003714 001507      BEQ 11$      ;BR IF NO
81 003716 104401 001217  TYPE $CRLF  ;CR-LF
82 003722 010146      MOV R1,-(SP) ;:SAVE R1 FOR TYPEOUT
    003724 104403      TYPOS      ;:GO TYPE--OCTAL ASCII
    003726      002      .BYTE 2      ;:TYPE 2 DIGIT(S)
    003727      000      .BYTE 0      ;:SUPPRESS LEADING ZEROS
83 003730 104401 032126  TYPE ,BLNKS4 ;TYPE 4 BLANKS
84
85 003734 012760 000040 000010  MOV #CLR,RMCS2(R0) ;CLEAR MASS BUS
86 003742 010160 000010  MOV R1,RMCS2(R0) ;LOAD THE DRIVE ADDRESS
87 003746 005760 000012  TST RMD5(R0) ;ACCESS DRIVE REGISTER
88 003752 032760 010000 000010  BIT #MED,RMCS2(R0) ;IS DRIVE PRESENT ?
89 003760 001027      BNE 3$      ;BR IF NO
90 003762 032760 004000 000000  BIT #DVA,RMCS1(R0) ;IS DRIVE AVAILABLE ?
91 003770 001426      BEQ 4$      ;BR IF NO
92 003772 012737 032006 004132  MOV #SRM80,10$ ;ASSUME RM80 DEVICE
93 004000 022760 020026 000026  CMP #20026,RMDT(R0) ;SINGLE PORT RM80 ?
94 004006 001407      BEQ 2$      ;BR IF YES
95 004010 022760 024026 000026  CMP #24026,RMDT(R0) ;DUAL PORT RM80 ?
96 004016 001403      BEQ 2$      ;BR IF YES
97 004020 104401 032013  TYPE ,NOTRM  ;DRIVE NOT AN RM80
    
```

98	004024	000443			BR	11\$:CHECK NEXT DRIVE
99	004026	032760	010000	000012	2\$: BIT	#MOL,RMDS(R0)	:IS MEDIUM ON LINE ?
100	004034	001415			BEQ	6\$:BR IF NO
101	004036	000417			BR	7\$	
102							
103	004040	104401	032045		3\$: TYPE	NOTPRS	:DRIVE NOT PRESENT
104	004044	000402			BR	5\$:CHECK NEXT DRIVE
105							
106	004046	104401	032062		4\$: TYPE	NOTAVL	:DRIVE NOT AVAILABLE
107	004052	005737	001326		5\$: TST	AUTSIZ	:AUTO SIZING ON ?
108	004056	001026			BNE	11\$:BR IF NO
109	004060	146137	032234	001300	BICB	ATNTBL(R1),SDEVN	:CLEAR DEVICE FROM BIT MAP
110	004066	000422			BR	11\$:CHECK NEXT DRIVE
111							
112	004070	104401	032101		6\$: TYPE	UNTOFF	:DRIVE OFFLINE
113	004074	000413			BR	9\$:PRINT DRIVE TYPE
114							
115	004076	005737	001332		7\$: TST	XXDP	:LOADED FROM RM80 ?
116	004102	001406			BEQ	8\$:NO
117	004104	123701	001332		CMPB	XXDP,R1	:IS THIS THE DRIVE ?
118	004110	001360			BNE	5\$:BR IF NO
119	004112	104401	032030		TYPE	LODEV	:DRIVE IS LOAD DEVICE
120	004116	000755			BR	5\$	
121							
122	004120	104401	032112		8\$: TYPE	UNTON	:DRIVE ONLINE
123	004124	104401	032130		9\$: TYPE	BLNKS2	:TYPE 2 BLANKS
124	004130	104401			TYPE		:PRINT DRIVE TYPE
125	004132	000000			10\$: .WORD	0	:MESSAGE ADDRESS HERE
126							
127	004134	005201			11\$: INC	R1	:INCREMENT THE DRIVE ADDRESS
128	004136	020127	000007		CMP	R1,#7	:ALL DRIVES ARE CHECKED ?
129	004142	003661			BLE	1\$:BRANCH IF NOT
130							
131	004144	104401	001217		12\$: TYPE	SCRFL	:CR-LF
132	004150	000137	004672		JMP	CMNSTART	:JUMP TO COMMON START


```

1      .SBTTL  STANDALONE INPUT ROUTINES
2
3      004154
4      004154  004737  027224
5
6      004160  005227  177777
7      004164  001023
8
9      ;SEE IF OPERATOR WANTS HELP TEXT
10
11     004166  104401  031360
12     004172  104411
13     004174  012637  001176
14     004200  123727  001176  000131
15     004206  001005
16     004210  104401  001176
17     004214  104401  054420
18     004220  000414
19     004222  104401  032122
20     004226  104401  001217
21     004232  000407
22
23     ;SEE IF USER WANTS TO CHANGE UNIBUS ADDRESS
24     004234
25     004234  005737  001330
26     004240  001475
27     004242  005037  001330
28     004246  104401  001217
29
30     ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
31     004252
32     004252  104401  031411
33     004256  013746  001276
34     004262  104402
35     004264  104401  032130
36     004270  104413
37     004272  012637  001176
38     004276  001412
39     004300  022737  160000  001176
40     004310  104401  031421
41     004314  000756
42     004316  013737  001176  001276  4$:
43
44     004324  104401  031463  5$:
45     004330  005046
46     004332  113716  001272
47     004336  104402
48     004340  104401  032130
49     004344  104413
50     004346  012637  001176
51     004352  001430
52     004354  022737  001000  001176
53     004362  101003
54     004364  104401  031472
55     004370  000755
56     004372  113700  001272  6$:

      STANDALONE:
      JSR      PC,$TKINT      ;INITIALIZE CONSOLE
      INC      #-1           ;FIRST TIME THRU HERE ?
      BNE      2$           ;BR IF NO

      ;SEE IF OPERATOR WANTS HELP TEXT
      TYPE     ,MSHELP      ;WANT HELP ?
      RDCHR    ;GET RESPONSE
      MOV      (SP)+,$TMP1  ;SAVE AND ECHO RESPONSE
      CMPB    $TMP1,#'Y    ;WAS IT A YES RESPONSE ?
      BNE      1$          ;NO
      TYPE     , $TMP1      ;TYPE 'Y'
      TYPE     ,HELP       ;YES - TYPE HELP TEXT
      BR      3$
1$:      TYPE     ,N        ;TYPE 'N'
      TYPE     , $CRLF     ;CR-LF
      BR      3$

      ;SEE IF USER WANTS TO CHANGE UNIBUS ADDRESS
2$:      TST      CHGADR     ;CHANGE RH/RM BUS ADDRESS ?
      BEQ      7$          ;BR IF NO
      CLR      CHGADR     ;NO CHANGE NEXT TIME
      TYPE     , $CRLF     ;CR-LF

      ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
3$:      TYPE     ,CNSLO1   ;TYPE CURRENT BUS ADDRESS
      MOV      $BASE,-(SP) ;SAVE $BASE FOR TYPEOUT
      TYPOC   ;GO TYPE--OCTAL ASCII(ALL DIGITS)
      TYPE     ,BLNKS2     ;TYPE 2 BLANKS
      RDOCT   ;GET NEW BUS ADDRESS
      MOV      (SP)+,$TMP1 ;CARRIAGE RETURN ?
      BEQ      5$          ;YES-SKIP TO NEXT ENTRY
      CMP      #160000,$TMP1 ;BASE ADDRESS IN I/O PAGE ?
      BLOS    4$          ;YES
      TYPE     ,CNSLO2     ;TYPE WARNING MESSAGE
      BR      5$          ;TRY AGAIN
4$:      MOV      $TMP1,$BASE ;STORE NEW BUS ADDRESS

5$:      TYPE     ,CNSLO3   ;
      CLR      -(SP)      ;
      MOVB    $VECT1,(SP) ;GET CURRENT VECTOR ADDRESS
      TYPOC   ;TYPE 2 BLANKS
      RDOCT   ;GET NEW VECTOR ADDRESS
      MOV      (SP)+,$TMP1 ;CARRIAGE RETURN?
      BEQ      7$          ;YES-SKIP TO NEXT ENTRY
      CMP      #1000,$TMP1 ;VECTOR ADDRESS < 1000 ?
      BHI     6$          ;YES!!
      TYPE     ,CNSLO4     ;TYPE WARNING MESSAGE
      BR      5$          ;RETRY
6$:      MOVB    $VECT1,R0  ;GET CURRENT VECTOR ADDRESS
    
```

```

57 004376 042700 177400          BIC      #^C<377>,R0      :CLEAR SIGN EXTENSION
58 004402 005720                   TST      (R0)+          :INCREMENT R0 BY 2
59 004404 010060 177776          MOV      R0,-2(R0)      :SETUP TRAP CATCHER ADDRESS AND
60 004410 005010                   CLR      (R0)          :PUT HALT IN TRAP ADDRESS +2
61 004412 013700 001176          MOV      STMP1,R0      :GET NEW VECTOR ADDRESS
62 004416 012720 021754          MOV      #IRP,(R0)+    :GET ADDRESS FOR INTERRUPT VECTOR
63 004422 113710 001273          MOVVB   $VECT1+1,(R0)  :GET PRIORITY LEVEL
64 004426 113737 001176 001272  MOVVB   STMP1,$VECT1   :STORE NEW VECTOR ADDRESS
65
66                                :DIALOGUE TO INPUT DEVICE NUMBERS
67 004434 005227 177777          7$:    INC      #-1      :FIRST TIME THRU ?
68 004440 001002                   BNE     8$             :BR IF NO
69 004442 104401 031526          TYPE   .CNSLO7        :TYPE INPUT INSTRUCTIONS
70 004446 104401 001217          8$:    TYPE   .SCRFLF   :CR-LF
71 004452 005037 001300          9$:    CLR      $DEVN   :CLEAR DEVICE MAP
72 004456 104401 031746          TYPE   .MSDRVS        :TYPE 'DRIVE(S): '
73 004462 104411                   RDCHR
74 004464 012637 001176          MOV     (SP)+,STMP1    :GET RESPONSE
75 004470 023727 001176 000101  CMP     STMP1,#'A      :IS INPUT 'A' ?
76 004476 001007                   BNE     10$           :NO
77 004500 104401 031344          TYPE   .ALL           :YES, TYPE 'ALL' AND GO
78 004504 012737 000377 001300  MOV     #377,$DEVN    :SET DEVICE MAP FOR ALL DRIVES
79 004512 000137 003664          JMP     XSIZ          :AUTO SIZE
80
81 004516 023727 001176 000015 10$:   CMP     STMP1,#CR     :CARRIAGE RETURN ?
82 004524 001436                   BEQ     12$           :YES
83 004526 104401 001176          TYPE   .STMP1         :ECHO RESPONSE
84 004532 023727 001176 000060  CMP     STMP1,#'0     :NUMBER < 0 ?
85 004540 002430                   BLT     12$           :YES
86 004542 023727 001176 000067  CMP     STMP1,#'7     :NUMBER > 7 ?
87 004550 003427                   BLE     13$           :NO
88 004552 000423                   BR      12$           :ILLEGAL INPUT
89
90 004554 104411                   11$:  RDCHR
91 004556 012637 001176          MOV     (SP)+,STMP1    :GET RESPONSE
92 004562 023727 001176 000015  CMP     STMP1,#CR     :CARRIAGE RETURN ?
93 004570 001432                   BEQ     14$           :YES
94 004572 104401 031355          TYPE   .COMMA         :TYPE ','
95 004576 104401 001176          TYPE   .STMP1         :ECHO RESPONSE
96 004602 023727 001176 000060  CMP     STMP1,#'0     :NUMBER < 0 ?
97 004610 002404                   BLT     12$           :YES
98 004612 023727 001176 000067  CMP     STMP1,#'7     :NUMBER > 7 ?
99 004620 003403                   BLE     13$           :NO
100 004622 104401 031670          12$:  TYPE   .CNSLO8    :TYPE "?ILLEGAL INPUT"
101 004626 000711                   BR      9$            :RETRY
102
103 004630 013701 001176          13$:  MOV     STMP1,R1      :R1 = DRIVE NUMBER
104 004634 042701 177770          BIC     #^C7,R1
105 004640 156137 032234 001300  BISB   ATNTBL(R1),$DEVN :SET DEVICE IN MAP
106 004646 122737 000377 001300  CMPB   #377,$DEVN    :DONE ?
107 004654 101337                   BHI     11$           :NO
108 004656 005237 001326          14$:  INC     AUTSIZ      :DO NOT AUTO SIZE WHEN TYPING DRIVE STATUS
109 004662 104401 001217          TYPE   .SCRFLF       :CR-LF
110 004666 000137 003664          JMP     XSIZ          :GO SIZE DEVICES
    
```

```

1
2 004672
3 004672 104401 031712
4 004676 013700 001300
5 004702 001004
6 004704 104401 031355
7 004710 104401 031741
8 004714 012701 001470
9 004720 010137 001466
10 004724 012702 000001
11 004730 005003
12 004732 030200
13 004734 001413
14 004736 104401 031355
15 004742 010311
16 004744 010346
    004746 104403
    004750 001
    004751 000
17 004752 116361 032234 000001
18 004760 062701 000002
19 004764 006302
20 004766 105702
21 004770 001402
22 004772 005203
23 004774 000756
24 004776 005011
25 005000 104401 001217
26
27
28 005004 004737 021374
29 005010 000425
30 005012 104401 005020
    005016 000413
    005046
31 005046 005737 000042
32 005052 001002
33 005054 000137 002732
34 005060 000137 021344
35 005064 000413
36
37 005066 000240
38 005070 105737 001300
39 005074 001007
40 005076 005737 000042
41 005102 001002
42 005104 000137 002732
43 005110 000137 021344
44 005114
45
46 005114 105037 001116
47 005120 005037 001206
48 005124 004737 027224
49 005130 012746 000000
    005134 012746 005142
    005140 000002

:ASSEMBLE TEST QUE FROM DEVICE MAP
CMNSTART:
TYPE      DRIVES      ;TYPE 'DRIVE(S) TO BE TESTED'
MOV       $DEVN,R0    ;R0 = DEVICE MAP
BNE       1$          ;BR IF DRIVES TO TEST
TYPE      ,COMMA      ;TYPE ' '
TYPE      ,NONE       ;TYPE 'NONE'
1$: MOV     #TSTQUE+2,R1 ;R1 = ADDRESS OF FIRST ENTRY IN QUE
MOV       R1,TSTQUE   ;INITIALIZE ENTRY POINTER
MOV       #1,R2        ;R2 = DEVICE POINTER
CLR       R3           ;R3 = DEVICE NUMBER
2$: BIT    R2,R0       ;IS THIS DEVICE IN MAP ?
BEQ       3$          ;NO !!
TYPE      ,COMMA      ;TYPE ' '
MOV       R3,(R1)     ;YES - ENTER DEVICE NUMBER IN QUE
MOV       R3,-(SP)    ;SAVE R3 FOR TYPEOUT
TYPOS    ;GO TYPE--OCTAL ASCII
.BYTE    1            ;TYPE 1 DIGIT(S)
.BYTE    0            ;SUPPRESS LEADING ZEROS
MOVB     ATNTBL(R3),1(R1) ;ENTER ATTENTION BIT IN QUE
ADD      #2,R1        ;ADVANCE ENTRY POINTER
3$: ASL    R2           ;ADVANCE DEVICE POINTER
TSTB     R2           ;DONE ALL DEVICES ?
BEQ      4$          ;YES
INC      R3           ;ADVANCE DEVICE NUMBER
BR       2$          ;ENTER NEXT DEVICE
4$: CLR   (R1)        ;TERMINATE TEST QUE
TYPE     ,SCRLF      ;TYPE CR-LF

:SIZE FOR CLOCK
JSR     PC,SIZCLK    ;SEE IF CLOCK PRESENT
BR      6$          ;YES - CLOCK IS PRESENT
TYPE    ,65$        ;TYPE ASCII STRING
BR      64$        ;GET OVER THE ASCII
65$: .ASCIZ <CRLF>/NO 'L' OR 'P' CLOCK/
64$:
TST     #42         ;ANY MONITOR PRESENT ?
BNE     5$          ;BR IF YES
JMP     START      ;JUMP TO START
5$: JMP  $GET42     ;RETURN CONTROL TO MONITOR
6$: BR   READY1

READY: NOP          ;READY TO START TEST
TSTB   $DEVN       ;ANY DRIVES IN MAP ?
BNE    2$          ;BR IF YES
TST    #42         ;ANY MONITOR PRESENT ?
BNE    1$          ;BR IF YES
JMP    START      ;JUMP TO START
1$: JMP  $GET42     ;RETURN CONTROL TO MONITOR
2$:

READY1: CLRB       ;RESET TEST NUMBER
CLR     $TIMES     ;INITIALIZE NUMBER OF ITERATIONS
JSR    PC,$TKINT  ;INITIALIZE TTY
MOV    #PRO,-(SP) ;PUT NEW PS ON STACK
MOV    #64$,-(SP) ;PUT NEW PC ON STACK
RTI    ;POP NEW PC AND PS
    
```

```

005142
50 005142 117737 174320 001234 64$:      MOV      @TSTQUE,$UNIT      ;LOAD DRIVE NUMBER
51
52                                     ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND SET LAST TRACK ADDRESS
53 005150 013700 001276      MOV      $BASE,R0          ;R0 = UNIBUS ADDRESS
54 005154 012760 000040 000010  MOV      #CLR,RMCS2(R0)    ;CLEAR MASSBUS
55 005162 117760 174300 000010  MOV      @TSTQUE,RMCS2(R0) ;SELECT DEVICE UNDER TEST
56 005170 012737 006400 001334  MOV      #TAB!TA4!TA1,LSTRK ;SET LAST TRACK = 13.
57
58                                     ;TYPE DRIVE NUMBER TO BE TESTED($UNIT)
59 005176 104401 001217      TYPE     ,SCRLF            ;CR-LF
60 005202 105737 001300      TSTB    $DEVN             ;ANY DRIVES IN MAP ?
61 005206 001406              BEQ     1$                ;BR IF NO
62 005210 104401 031762      TYPE     ,MSGDRV          ;TYPE 'DRIVE'
63 005214 013746 001234      MOV     $UNIT,-(SP)       ;;SAVE $UNIT FOR TYPEOUT
                                ;;TYPE DRIVE NUMBER
                                ;;GO TYPE--OCTAL ASCII
                                ;;TYPE 2 DIGIT(S)
                                ;;SUPPRESS LEADING ZEROS
                                ;THESE TWO LOOPS ARE ADDED TO
                                ;WAIT FOR TTY
        005220 104403      TYPOS
        005222      .BYTE 2
        005223      .BYTE 0
64 005224 005004 1$:      CLR     R4
65 005226 005304      DEC    R4
66 005230 001376      BNE    .-2
67 005232 005304      DEC    R4
68 005234 001376      BNE    .-2
    
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

.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

:STMP0-STMP4 TEMPORARY STORAGE, NOT SAVED BY SUBROUTINES
:SGDDAT,SBDDAT EXPECTED AND RECEIVED STATUS FOR ERROR TYPEOUT
:SGDADR,SBDAADR ADDRESS OF EXPECTED AND RECEIVED STATUS IF APPLICABLE,
ALSO THE ADDRESS OF A REGISTER ERROR
:STSTN = TEST NUMBER
:SUNIT = 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
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

 :*TEST 1 TRANSFER TEST

005236 000004
 005236 000240
 005240 012706 001100
 005242 013700 001276
 005246 013701 001466
 005252 012737 000001 001226
 005256 012737 000001 001226

MOV #0,R2 ;R2 = REGISTER INDEX

:CLEAR THE MASSBUS AND VERIFY THAT NONEXISTANT DEVICE ERROR IS RESET
 10\$:

005270 004737 022230
 005270 016037 000010 001142
 005274 032737 010000 001142
 005302 001417
 005310 111137 001140
 005312 042737 177770 001140
 005316 052737 000100 001140
 005324 010037 001136
 005332 062737 000010 001136
 005336 104001
 005344 000475

:READ THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE READ
 DOES NOT SET 'NED' ERROR
 20\$:

005350 010003
 005350 060203
 005352 011304
 005354 032760 010000 000010
 005356 001470

005366 004737 022230
 005372 016037 000010 001142
 005400 032737 010000 001142
 005406 001417
 005410 111137 001140
 005414 042737 177770 001140
 005422 052737 000100 001140
 005430 010037 001136
 005434 062737 000010 001136
 005442 104001
 005444 000436

:WRITE THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE WRITE
 DOES NOT SET 'NED' ERROR
 30\$:

005446 012713 000000
 005446 032760 010000 000010
 005452 001432

: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 40$:
51 005462 062702 000002 ADD #2,R2 :ADVANCE TO NEXT REGISTER
52 005466 022702 000002 CMP #RMWC,R2 :IS THIS RMWC??
53 005472 001773 BEQ 40$ :YES - TRY NEXT REGISTER
54 005474 022702 000004 CMP #RMBA,R2 :IS THIS RMBA??
55 005500 001770 BEQ 40$ :YES - TRY NEXT REGISTER
56 005502 022702 000010 CMP #RMCS2,R2 :IS THIS RMCS2??
57 005506 001765 BEQ 40$ :YES - TRY ANOTHER REGISTER
58 005510 022702 000016 CMP #RMAS,R2 :IS THIS RMAS ??
59 005514 001762 BEQ 40$ :YES - TRY ANOTHER REGISTER
60 005516 022702 000022 CMP #RMDB,R2 :IS THIS RMDB??
61 005522 001757 BEQ 40$ :YES - TRY ANOTHER REGISTER
62 005524 022702 000046 CMP #RMEC2,R2 :IS THIS A LEGAL REGISTER
63 005530 103257 BHIS 10$ :YES - TRY THIS REGISTER
64
65 :GOT 'NONEXISTENT DEVICE' ERROR FOR EVERY REMOTE REGISTER ADDRESS
66 50$:
67 005532 013737 001276 001136 MOV $BASE,$BADDR :STORE BASE ADDRESS
68 005540 104002 EMT 2
69 005542 000137 021120 60$: JMP $EOSP :GO SELECT NEXT DEVICE
70
71 005546 70$:
72
73 :*****
74 :*TEST 2 CTOD TEST
75 :*****
76 TST2:
77 SCOPE :SCOPE CALL
78 005546 000004 NOP
79 005550 000240 MOV #STACK,SP :LOAD THE STACK POINTER
80 005552 012706 001100 MOV $BASE,R0 :R0 = UNIBUS ADDRESS
81 005556 013700 001276 MOV TSTQUE,R1 :R1 = POINTER TO DEVICE
82 005562 013701 001466 MOV #2,$TESTN :SET TEST NUMBER IN APT MAIL BOX
83 005566 012737 000002 001226 JSR PC,CNTCLR :GO CLEAR CONTROLLER
84
85 :WRITE ONES IN REMOTE REGISTERS
86 005600 012760 000076 000000 MOV #ILF76,RMCS1(R0) :LOAD RMCS1
87 005606 012760 177777 000006 MOV #-1,RMDA(R0) :LOAD RMDA
88 005614 012760 001777 000034 MOV #CYLMASK,RMDC(R0) :LOAD RMDC
89 005622 012760 017200 000032 MOV #^CXNUOF,RMOF(R0) :LOAD RMOF
90 005630 012760 017200 000032 MOV #^CXNUOF,RMOF(R0) :LOAD RMOF AGAIN TO SET SSEI
91
92 :READ REMOTE REGISTERS TWICE
93 10$:
94 005636 012702 000001 MOV #1,R2
95 005642 016037 000000 001336 MOV RMCS1(R0),RMCS1I :STORE RMCS1 IN INPUT BUFFER
96 005650 016037 000006 001344 MOV RMDA(R0),RMDAI :STORE RMDA IN INPUT BUFFER
97 005656 016037 000034 001372 MOV RMDC(R0),RMDCI :STORE RMDC IN INPUT BUFFER
98 005664 016037 000032 001370 MOV RMOF(R0),RMOFI :STORE RMOF IN INPUT BUFFER
99 DEC R2
100 BPL 10$
101
102 :SEE IF ANY ONE BITS CAME BACK

```

T2 CTOD TEST

93 005676 042737 177701 001336
 94 005704 001014
 95 005706 005737 001344
 96 005712 001011
 97 005714 042737 176000 001372
 98 005722 001005
 99 005724 042737 160577 001370
 100 005732 001001
 101
 102
 103 005734 104003
 104 005736
 105
 106

BIC #*CILF76,RMCS1I ;IS RMCS1 0??
 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 A ONE FROM ANY REMOTE REGISTER
 EMT 3

20\$:

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

::*****

TST3:

005736
 005736 000004
 005740 000240
 005742 012706 001100
 005746 013700 001276
 005752 013701 001466
 005756 012737 000003 001226
 107
 108 005764 004737 022230
 109
 110
 111 005770 012760 000076 000000
 112 005776 012760 177777 000014
 113 006004 012760 177777 000042
 114
 115
 116 006012 004737 022230
 117
 118
 119 006016 016037 000000 001336
 120 006024 016037 000014 001352
 121 006032 016037 000042 001400
 122
 123
 124 006040 052737 177701 001336
 125 006046 052737 001527 001400
 126 006054 022737 177777 001336
 127 006062 001011
 128 006064 022737 177777 001352
 129 006072 001005
 130 006074 022737 177777 001400
 131 006102 001001
 132
 133
 134 006104 104004
 135 006106
 136
 137

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
 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
 ;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
 ;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\$

;NONE OF THE BITS WERE CLEARED

EMT 4

10\$:

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


```

006106
006106 000004
006110 000240
006112 012706 001100
006116 013700 001276
006122 013701 001466
006126 012737 000004 001226
138
139 006134 004737 022230
140
141
142 006140 012760 177777 000014
143 006146 012760 177777 000042
144 006154 012760 000001 000024
145
146
147 006162 016037 000014 001352
148 006170 016037 000042 001400
149 006176 016037 000024 001362
150 006204 042737 040000 001352
151 006212 001011
152 006214 042737 040200 001400
153 006222 001005
154 006224 032737 000001 001362
155 006232 001001
156 006234 104026
157 006236
158
159
    
```

```

*****
TST4:
SCOPE                               ;SCOPE CALL
NOP
MOV #STACK,SP                       ;LOAD THE STACK POINTER
MOV SBASE,R0                         ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1                        ;R1 = POINTER TO DEVICE
MOV #4,$TESTN                        ;;SET TEST NUMBER IN APT MAIL BOX
JSR PC,CNTCLR                        ;GO CLEAR CONTROLLER

;WRITE ONES IN TEST REGISTERS
MOV #-1,RMER1(R0)                   ;LOAD RMER1
MOV #-1,RMER2(R0)                   ;LOAD RMER2
MOV #DMD,RMMR1(R0)                  ;LOAD RMMR1

;READ TEST REGISTERS AND SEE IF ANY BITS ARE ON
MOV RMER1(R0),RMER1I                 ;STORE RMER1 IN INPUT BUFFER
MOV RMER2(R0),RMER2I                 ;STORE RMER2 IN INPUT BUFFER
MOV RMMR1(R0),RMMR1I                 ;STORE RMMR1 IN INPUT BUFFER
BIC #UNS,RMER1I                      ;DONT ACCEPT UNSAFE
BNE 10$                               ;BRANCH IF ANY OTHER BITS ON
BIC #SKI!DVC,RMER2I                  ;DONT ACCEPT SKI OR DVC
BNE 10$                               ;BRANCH IF ANY OTHER BITS ON
BIT #DMD,RMMR1I                      ;BRANCH IF DMD IS ON
BNE 10$
EMT 26
    
```

```

10$:
*****
;*TEST 5 TRISTATE TRANSFER TEST
*****
    
```

```

006236
006236 000004
006240 000240
006242 012706 001100
006246 013700 001276
006252 013701 001466
006256 012737 000005 001226
160
161 006264 005002
162 006266 004737 022230
163
164
165 006272 012760 000076 000000
166 006300 012760 177777 000006
167 006306 012760 177777 000014
168 006314 012760 177777 000032
    006322 012760 177777 000032
169 006330 012760 177777 000042
170
171
172 006336 012760 000000 000000
173 006344 012760 000000 000006
174 006352 012760 000000 000014
175 006360 012760 000000 000032
    
```

```

*****
TST5:
SCOPE                               ;SCOPE CALL
NOP
MOV #STACK,SP                       ;LOAD THE STACK POINTER
MOV SBASE,R0                         ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1                        ;R1 = POINTER TO DEVICE
MOV #5,$TESTN                        ;;SET TEST NUMBER IN APT MAIL BOX
CLR R2                               ;CLEAR ERROR FLAGS
JSR PC,CNTCLR                        ;GO CLEAR CONTROLLER

;WRITE ONES IN SELECTED REGISTERS
MOV #ILF76,RMCS1(R0)                ;LOAD RMCS1
MOV #-1,RMDA(R0)                     ;LOAD RMDA
MOV #-1,RMER1(R0)                    ;LOAD RMER1
MOV #-1,RMOF(R0)                     ;LOAD RMOF
MOV #-1,RMOF(R0)                     ;LOAD RMOF AGAIN TO SET SSEI
MOV #-1,RMER2(R0)                    ;LOAD RMER2

;WRITE ZEROS IN SELECTED REGISTERS
MOV #0,RMCS1(R0)                     ;LOAD RMCS1
MOV #0,RMDA(R0)                       ;LOAD RMDA
MOV #0,RMER1(R0)                      ;LOAD RMER1
MOV #0,RMOF(R0)                       ;LOAD RMOF
    
```

```

T5 TRISTATE TRANSFER TEST

176 006366 012760 000000 000034      MOV    #0,RMDC(R0)      ;LOAD RMDC
177 006374 012760 000000 000042      MCV    #0,RMER2(R0)    ;LOAD RMER2
178
179                                     ;READ BACK ALL REGISTERS
180 006402 016037 000000 001336      MOV    RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
181 006410 016037 000006 001344      MOV    RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
182 006416 016037 000014 001352      MOV    RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
183 006424 016037 000032 001370      MOV    RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
184 006432 016037 000034 001372      MOV    RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
185 006440 016037 000042 001400      MOV    RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
186
187                                     ;CHECK EACH REGISTER CONTENT FOR ZERO BITS WRITTEN & READ
188 006446 012702 177777              MOV    #-1,R2          ;ACCUMULATE ZEROS IN R2
189 006452 052737 177701 001336      BIS    #^C1LF76,RMCS1I ;SET ALL BITS NOT WRITTEN
190 006460 052737 160577 001370      BIS    #XNUOF,RMOFI
191 006466 052737 176000 001372      BIS    #XNUDC,RMDCI
192 006474 052737 001527 001400      BIS    #XNUER2,RMER2I
193 006502 005137 001336              COM    RMCS1I          ;COMPLEMENT REGISTER CONTENTS
194 006506 005137 001344              COM    RMDAI
195 006512 005137 001352              COM    RMER1I
196 006516 005137 001370              COM    RMOFI
197 006522 005137 001372              COM    RMDCI
198 006526 005137 001400              COM    RMER2I
199 006532 043702 001336              BIC    RMCS1I,R2      ;ACCUMULATE ALL ZERO BITS
200 006536 043702 001344              BIC    RMDAI,R2
201 006542 043702 001352              BIC    RMER1I,R2
202 006546 043702 001370              BIC    RMOFI,R2
203 006552 043702 001372              BIC    RMDCI,R2
204 006556 043702 001400              BIC    RMER2I,R2
205 006562 001407                      BEQ    10$            ;BRANCH IF EACH BIT IS ZERO
206
207                                     ;ONE OR MORE BIT POSITIONS ARE NOT ZERO
208 006564 010237 001142              MOV    R2,$BDDAT      ;SAVE RESULT FOR TYPE
209 006570 005037 001140              CLR    $GDDAT         ;LOAD EXPECTED RESULT
210 006574 104005                      EMT    5
211 006576 052702 000001              BIS    #BIT0,R2      ;SET ERROR FLAG
212
213 006602 004737 022230              10$: JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
214
215                                     ;PRESET SELECTED REGISTERS TO ZEROS
216                                     ;:(ASSUME RMCS1, RMER1, RMER2 WERE CLEARED BY INIT)
217 006606 012760 000000 000006      MOV    #0,RMDA(R0)    ;LOAD RMDA
218 006614 012760 000000 000032      MOV    #0,RMOF(R0)    ;LOAD RMOF
219 006622 012760 000000 000034      MOV    #0,RMDC(R0)    ;LOAD RMDC
220
221                                     ;WRITE ONES IN SELECTED REGISTERS
222 006630 012760 000076 000000      MOV    #1LF76,RMCS1(R0) ;LOAD RMCS1
223 006636 012760 177777 000006      MOV    #-1,RMDA(R0)   ;LOAD RMDA
224 006644 012760 017200 000032      MOV    #^CXNUOF,RMOF(R0) ;LOAD RMOF
225 006652 012760 017200 000032      MOV    #^CXNUOF,RMOF(R0) ;LOAD RMOF AGAIN TO SET SSEI
226 006660 012760 001777 000034      MOV    #^CXNUDC,RMDC(R0) ;LOAD RMDC
227 006666 012760 177777 000014      MOV    #-1,RMER1(R0)  ;LOAD RMER1
228 006674 012760 176250 000042      MOV    #^CXNUER2,RMER2(R0) ;LOAD RMER2
229
230 006702 016037 000000 001336      ;READ ALL REGISTERS
230 006702 016037 000000 001336      MOV    RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER

```

```

T5 TRISTATE TRANSFER TEST

231 006710 016037 000006 001344      MOV      RMDA(R0),RMDAI      ;STORE RMDA IN INPUT BUFFER
232 006716 016037 000032 001370      MOV      RMOF(R0),RMOFI      ;STORE RMOF IN INPUT BUFFER
233 006724 016037 000034 001372      MOV      RMDC(R0),RMDCI      ;STORE RMDC IN INPUT BUFFER
234 006732 016037 000014 001352      MOV      RMER1(R0),RMER1I     ;STORE RMER1 IN INPUT BUFFER
235 006740 016037 000042 001400      MOV      RMER2(R0),RMER2I     ;STORE RMER2 IN INPUT BUFFER
236
237                                     ;CHECK EACH REGISTER CONTENT FOR ONE BITS WRITTEN & READ
238 006746 042737 177701 001336      BIC      #^CILF76,RMCS1I     ;CLEAR ALL BITS NOT WRITTEN
239 006754 042737 160577 001370      BIC      #XNUOF,RMOFI
240 006762 042737 176000 001372      BIC      #XNUDC,RMDCI
241 006770 042737 001527 001400      BIC      #XNUER2,RMER2I
242 006776 005002                                     CLR      R2                   ;ACCUMULATE ONES IN R2
243 007000 053702 001336      BIS      RMCS1I,R2           ;ACCUMULATE ALL ONE BITS
244 007004 053702 001344      BIS      RMDAI,R2
245 007010 053702 001370      BIS      RMOFI,R2
246 007014 053702 001372      BIS      RMDCI,R2
247 007020 053702 001352      BIS      RMER1I,R2
248 007024 053702 001400      BIS      RMER2I,R2
249 007030 022702 177777      CMP      #-1,R2             ;SEE IF EACH BIT POSITION WAS ONE
250 007034 001410      BEQ      20$                ;BRANCH IF NONE STUCK
251
252                                     ;ONE OR MORE BIT POSITIONS ARE NOT ONE
253 007036 010237 001142      MOV      R2,$BDDAT          ;SAVE RESULT FOR TYPE
254 007042 012737 177777 001140      MOV      #-1,$GDDAT         ;EXPECTED RESULT
255 007050 104006      EMT      6
256 007052 052702 000002      BIS      #BIT1,R2           ;SET ERROR FLAG
257 007056      20$:
258 007056 005702      TST      R2                 ;ANY ERRORS DETECTED ??
259 007060 001130      BNE      30$                ;YES - DONT DO BIT TEST
260 007062 012702 000001      MOV      #1,R2              ;R2=BIT POSITION
261
262 007066      25$:
263 007066 004737 022230      JSR      PC,CNTCLR          ;GO CLEAR CONTROLLER
264
265                                     ;WRITE THE BIT PATTERN IN SELECTED DEVICE REGISTERS
265 007072 010260 000006      MOV      R2,RMDA(R0)        ;LOAD RMDA
266 007076 010260 000032      MOV      R2,RMOF(R0)        ;LOAD RMOF
267 007102 010260 000032      MOV      R2,RMOF(R0)        ;LOAD RMOF AGAIN TO SET SSEI
267 007106 010260 000034      MOV      R2,RMDC(R0)        ;LOAD RMDC
268 007112 010260 000014      MOV      R2,RMER1(R0)       ;LOAD RMER1
269 007116 010260 000042      MOV      R2,RMER2(R0)       ;LOAD RMER2
270
271                                     ;READ BACK THE REGISTERS
272 007122 016037 000006 001344      MOV      RMDA(R0),RMDAI     ;STORE RMDA IN INPUT BUFFER
273 007130 016037 000032 001370      MOV      RMOF(R0),RMOFI     ;STORE RMOF IN INPUT BUFFER
274 007136 016037 000034 001372      MOV      RMDC(R0),RMDCI     ;STORE RMDC IN INPUT BUFFER
275 007144 016037 000014 001352      MOV      RMER1(R0),RMER1I   ;STORE RMER1 IN INPUT BUFFER
276 007152 016037 000042 001400      MOV      RMER2(R0),RMER2I   ;STORE RMER2 IN INPUT BUFFER
277
278                                     ;CHECK REGISTER CONTENTS FOR CORRECT PATTERN
279 007160 005003      CLR      R3                 ;R3=ACCUMULATED ONE BIT
280 007162 012704 177777      MOV      #-1,R4             ;R4=ACCUMULATED ZERO BITS
281 007166 013705 001344      MOV      RMDAI,R5           ;GET ANY GOOD BITS FROM RMDA
282 007172 050503      BIS      R5,R3
283 007174 005105      COM      R5
284 007176 040504      BIC      R5,R4
285 007200 013705 001370      MOV      RMOFI,R5           ;GET GOOD BITS FROM RMOF

```

```

286 007204 042705 160577      BIC    #XNUOF,R5
287 007210 050503      BIS    R5,R3
288 007212 005105      COM    R5
289 007214 042705 160577      BIC    #XNUOF,R5
290 007220 040504      BIC    R5,R4
291 007222 013705 001372      MOV    RMDCI,R5          ;GET GOOD BITS FROM RMDC
292 007226 042705 176000      BIC    #XNUDC,R5
293 007232 050503      BIS    R5,R3
294 007234 005105      COM    R5
295 007236 042705 176000      BIC    #XNUDC,R5
296 007242 040504      BIC    R5,R4
297 007244 013705 001352      MOV    RMER1I,R5        ;GET GOOD BITS FROM RMER1
298 007250 050503      BIS    R5,R3
299 007252 005105      COM    R5
300 007254 040504      BIC    R5,R4
301 007256 013705 001400      MOV    RMER2I,R5        ;GET GOOD BITS FROM RMER2
302 007262 042705 001527      BIC    #XNUER2,R5
303 007266 050503      BIS    R5,R3
304 007270 005105      COM    R5
305 007272 042705 001527      BIC    #XNUER2,R5
306 007276 040504      BIC    R5,R4
307 007300 010205      MOV    R2,R5            ;RESET ALL ONES IN R3 EXCEPT
308 007302 005105      COM    R5                ;FOR THE TEST BIT
309 007304 040503      BIC    R5,R3
310 007306 040204      BIC    R2,R4            ;RESET TEST BIT IN R4
311 007310 050403      BIS    R4,R3            ;COMBINE ACCUMULATED 1'S + 0'S
312 007312 020302      CMP    R3,R2            ;IS PATTERN OK??
313 007314 001406      BEQ    26$              ;YES!!
314 007316 010237 001140      MOV    R2,$GDDAT        ;SAVE TEST PATTERN
315 007322 010337 001142      MOV    R3,$BDDAT        ;SAVE RESULT
316 007326 104007      EMT    ?
317 007330 000404      BR     30$              ;SKIP TO NEXT

```

:ADVANCE R2 TO THE NEXT PATTERN AND REPEAT TEST

```

318
319
320 007332 26$:
321 007332 006302      ASL    R2                ;SHIFT THE BIT
322 007334 001402      BEQ    30$              ;EXIT IF DONE
323 007336 000137 007066      JMP    25$
324 007342 30$:

```

 :*TEST 6 REGISTER SELECT TEST

 TST6:

```

007342      SCOPE                ;SCOPE CALL
007342 000004      NOP
007344 000240      MOV    #STACK,SP        ;LOAD THE STACK POINTER
007346 012706 001100      MOV    $BASE,R0         ;R0 = UNIBUS ADDRESS
007352 013700 001276      MOV    TSTQUE,R1        ;R1 = POINTER TO DEVICE
007356 013701 001466      MOV    #6,$TESTN        ;:SET TEST NUMBER IN APT MAIL BOX
007362 012737 000006 001226

```

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

```

327
328
329
330
331
332
: REGISTER NAME REG SEL (16,8,4,2,1)

```

333	:		
334	:	RMCS1	00000
335	:	RMDS	00001
336	:	RMER1	00010
337	:	RMMR1	00011
338	:	RMAS	00100
339	:	RMDA	00101
340	:	RMDT	00110
341	:	RMLA	00111
342	:	RMSN	01000
343	:	RMOF	01001
344	:	RMDC	01010
345	:	RMHR	01011
346	:	RMMR2	01100
347	:	RMER2	01101
348	:	RMEC1	01110
349	:	RMEC2	01111

: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.

360 007370 005002
 361 007372 012703 177777

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

:TEST REG SEL 1 FOR S-A-0

365 007376 004737 022230
 366 007402 010260 000014
 367 007406 010260 000034
 368 007412 010360 000024
 369 007416 010360 000036
 370 007422 016037 000014 001352
 371 007430 016037 000034 001372
 372 007436 020337 001352
 373 007442 001007
 374 007444 052737 176000 001372
 375 007452 020337 001372
 376 007456 001001
 377 007460 104010

JSR PC,CNTCLR :GO CLEAR CONTROLLER
 MOV R2,RMER1(RO) :LOAD RMER1
 MOV R2,RMDC(RO) :LOAD RMDC
 MOV R3,RMMR1(RO) :LOAD RMMR1
 MOV R3,RMHR(RO) :LOAD RMHR
 MOV RMER1(RO),RMER1I :STORE RMER1 IN INPUT BUFFER
 MOV RMDC(RO),RMDCI :STORE RMDC IN INPUT BUFFER
 CMP R3,RMER1I
 BNE 10\$
 BIS #XNDC,RMDCI
 CMP R3,RMDCI
 BNE 10\$
 ENT 10

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

380 007462
 381 007462 004737 022230
 382 007466 010260 000006
 383 007472 010260 000032
 007476 010260 000032
 384 007502 010260 000042
 385 007506 010360 000016
 386 007512 010360 000030
 387 007516 010360 000040
 388 007522 016037 000006 001344

JSR PC,CNTCLR :GO CLEAR CONTROLLER
 MOV R2,RMDA(RO) :LOAD RMDA
 MOV R2,RMOF(RO) :LOAD RMOF
 MOV R2,RMOF(RO) :LOAD RMOF AGAIN TO SET SSEI
 MOV R2,RMER2(RO) :LOAD RMER2
 MOV R3,RMAS(RO) :LOAD RMAS
 MOV R3,RMSN(RO) :LOAD RMSN
 MOV R3,RMMR2(RO) :LOAD RMMR2
 MOV RMDA(RO),RMDAI :STORE RMDA IN INPUT BUFFER

```

389 007530 016037 000032 001370      MOV      RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
390 007536 016037 000042 001400      MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
391 007544 020337 001344      CMP      R3,RMDAI
392 007550 001015      BNE      20$
393 007552 052737 160577 001370      BIS      #XNUOF,RMOFI
394 007560 020337 001370      CMP      R3,RMOFI
395 007564 001007      BNE      20$
396 007566 052737 001527 001400      BIS      #XNUER2,RMER2I
397 007574 020337 001400      CMP      R3,RMER2I
398 007600 001001      BNE      20$
399 007602 104011      EMT      11
    
```

:TEST REG SEL 2 FOR S-A-0
 20\$:

```

402 007604      JSR      PC,CNTCLR ;GO CLEAR CONTROLLER
403 007604 004737 022230      MOV      R2,RMDA(R0) ;LOAD RMDA
404 007610 010260 000006      MOV      R2,RMOF(R0) ;LOAD RMOF
405 007614 010260 000032      MOV      R2,RMOF(R0) ;LOAD RMOF AGAIN TO SET SSEI
      007620 010260 000032      MOV      R2,RMER2(R0) ;LOAD RMER2
406 007624 010260 000042      MOV      R3,RMLA(R0) ;LOAD RMLA
407 007630 010360 000020      MOV      R3,RMHR(R0) ;LOAD RMHR
408 007634 010360 000036      MOV      R3,RMEC2(R0) ;LOAD RMEC2
409 007640 010360 000046      MOV      RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
410 007644 016037 000006 001344      MOV      RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
411 007652 016037 000032 001370      MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
412 007660 016037 000042 001400      CMP      R3,RMDAI
413 007666 020337 001344      BNE      30$
414 007672 001015      BIS      #XNUOF,RMOFI
415 007674 052737 160577 001370      CMP      R3,RMOFI
416 007702 020337 001370      BNE      30$
417 007706 001007      BIS      #XNUER2,RMER2I
418 007710 052737 001527 001400      CMP      R3,RMER2I
419 007716 020337 001400      BNE      30$
420 007722 001001      EMT      12
421 007724 104012
    
```

:TEST REG SEL 2 FOR S-A-1
 30\$:

```

424 007726      JSR      PC,CNTCLR ;GO CLEAR CONTROLLER
425 007726 004737 022230      MOV      R2,RMER1(R0) ;LOAD RMER1
426 007732 010260 000014      MOV      R2,RMDC(R0) ;LOAD RMDC
427 007736 010260 000034      MOV      #ILF76,RMCS1(R0) ;LOAD RMCS1
428 007742 012760 000076 000000      MOV      R3,RMSN(R0) ;LOAD RMSN
429 007750 010360 000030      MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
430 007754 016037 000014 001352      MOV      RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
431 007762 016037 000034 001372      BIS      #CILF76,RMER1I
432 007770 052737 177701 001352      CMP      R3,RMER1I
433 007776 020337 001352      BNE      40$
434 010002 001007      BIS      #XNUDC,RMDCI
435 010004 052737 176000 001372      CMP      R3,RMDCI
436 010012 020337 001372      BNE      40$
437 010016 001001      EMT      13
438 010020 104013
    
```

:TEST REG SEL 4 FOR S-A-0
 40\$:

```

440      JSR      PC,CNTCLR ;GO CLEAR CONTROLLER
441 010022      MOV      R2,RMER1(R0) ;LOAD RMER1
442 010022 004737 022230      MOV      R2,RMOF(R0) ;LOAD RMOF
443 010026 010260 000014
444 010032 010260 000032
    
```

```

445 010036 010260 000032      MOV      R2,RMOF(R0)      ;LOAD RMOF AGAIN TO SET SSEI
446 010042 010260 000034      MOV      R2,RMDC(R0)      ;LOAD RMDC
447 010046 010360 000026      MOV      R3,RMDT(R0)      ;LOAD RMDT
448 010052 010360 000042      MOV      R3,RMER2(R0)     ;LOAD RMER2
449 010056 010360 000044      MOV      R3,RMEC1(R0)     ;LOAD RMEC1
450 010062 016037 000014 001352  MOV      RMER1(R0),RMER1I  ;STORE RMER1 IN INPUT BUFFER
451 010070 016037 000032 001370  MOV      RMOF(R0),RMOFI   ;STORE RMOF IN INPUT BUFFER
452 010076 016037 000034 001372  MOV      RMDC(R0),RMDCI   ;STORE RMDC IN INPUT BUFFER
453 010104 020337 001352      CMP      R3,RMER1I
454 010110 001015      BNE      50$
455 010112 052737 160577 001370  BIS      #XNUOF,RMOFI
456 010120 020337 001370      CMP      R3,RMOFI
457 010124 001007      BNE      50$
458 010126 052737 176000 001372  BIS      #XNUDC,RMDCI
459 010134 020337 001372      CMP      R3,RMDCI
460 010140 001001      BNE      50$
461 010142 104014      EMT      14
462
463 010144      ;TEST REG SEL 4 FOR S-A-1
464 010144 004737 022230 50$:      JSR      PC,CNTCLR        ;GO CLEAR CONTROLLER
465 010150 010260 000006      MOV      R2,RMDA(R0)      ;LOAD RMDA
466 010154 010260 000042      MOV      R2,RMER2(R0)     ;LOAD RMER2
467 010160 010360 000012      MOV      R3,RMDS(R0)      ;LOAD RMDS
468 010164 010360 000032      MOV      R3,RMOF(R0)      ;LOAD RMOF
469 010170 010360 000032      MOV      R3,RMOF(R0)      ;LOAD RMOF AGAIN TO SET SSEI
470 010174 016037 000006 001344  MOV      RMDA(R0),RMDAI   ;STORE RMDA IN INPUT BUFFER
471 010202 016037 000042 001400  MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
472 010210 020337 001344      CMP      R3,RMDAI
473 010214 001007      BNE      60$
474 010216 052737 001527 001400  BIS      #XNUER2,RMER2I
475 010224 020337 001400      CMP      R3,RMER2I
476 010230 001001      BNE      60$
477 010232 104015      EMT      15
478
479 010234      ;TEST REG SEL 3 FOR S-A-0
480 010234 004737 022230 60$:      JSR      PC,CNTCLR        ;GO CLEAR CONTROLLER
481 010240 010260 000014      MOV      R2,RMER1(R0)     ;LOAD RMER1
482 010244 010260 000006      MOV      R2,RMDA(R0)      ;LOAD RMDA
483 010250 010360 000034      MOV      R3,RMDC(R0)      ;LOAD RMDC
484 010254 010360 000042      MOV      R3,RMER2(R0)     ;LOAD RMER2
485 010260 016037 000014 001352  MOV      RMER1(R0),RMER1I  ;STORE RMER1 IN INPUT BUFFER
486 010266 016037 000006 001344  MOV      RMDA(R0),RMDAI   ;STORE RMDA IN INPUT BUFFER
487 010274 020337 001352      CMP      R3,RMER1I
488 010300 001004      BNE      70$
489 010302 020337 001344      CMP      R3,RMDAI
490 010306 001001      BNE      70$
491 010310 104016      EMT      16
492
493 010312      ;TEST REG SEL 8 FOR S-A-1
494 010312 004737 022230 70$:      JSR      PC,CNTCLR        ;GO CLEAR CONTROLLER
495 010316 010260 000032      MOV      R2,RMOF(R0)      ;LOAD RMOF
496 010322 010260 000032      MOV      R2,RMOF(R0)      ;LOAD RMOF AGAIN TO SET SSEI
497 010326 010260 000034      MOV      R2,RMDC(R0)      ;LOAD RMDC
498 010332 010260 000042      MOV      R2,RMER2(R0)     ;LOAD RMER2
    
```

499	010336	010360	000012		MOV	R3,RMDS(RO)	:LOAD RMDS
500	010342	010360	000014		MOV	R3,RMER1(RO)	:LOAD RMER1
501	010346	010360	000006		MOV	R3,RMDA(RO)	:LOAD RMDA
502	010352	016037	000032	001370	MOV	RMOF(RO),RMOFI	:STORE RMOF IN INPUT BUFFER
503	010360	016037	000034	001372	MOV	RMDC(RO),RMDCI	:STORE RMDC IN INPUT BUFFER
504	010366	016037	000042	001400	MOV	RMER2(RO),RMER2I	:STORE RMER2 IN INPUT BUFFER
505	010374	052737	160577	001370	BIS	#XNUOF,RMOFI	
506	010402	001015			BNE	80\$	
507	010404	022737	176000	001372	CMP	#XNUDC,RMDCI	
508	010412	020337	001372		CMP	R3,RMDCI	
509	010416	001007			BNE	80\$	
510	010420	052737	001527	001400	BIS	#XNUER2,RMER2I	
511	010426	020337	001400		CMP	R3,RMER2I	
512	010432	001001			BNE	80\$	
513	010434	104017			EMT	17	

80\$:

:REGISTER SELECT 16 IS TESTED BY THE ILR TEST

::*****
:*TEST 7 DRIVE TYPE TEST

::*****
TST7:

	010436						
	010436	000004			SCOPE		:SCOPE CALL
	010440	000240			NOP		
	010442	012706	001100		MOV	#STACK,SP	:LOAD THE STACK POINTER
	010446	013700	001276		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	010452	013701	001466		MOV	TSTQUE,R1	:R1 = POINTER TO DEVICE
	010456	012737	000007	001226	MOV	#7,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
519							
520	010464	016037	000026	001142	MOV	RMDT(RO),SBDDAT	:STORE RMDT AT SBDDAT
521	010472	022737	020026	001142	CMP	#SNGPRT,\$SBDDAT	:SINGLE PORT RM80??
522	010500	001414			BEQ	10\$:YES!!
523	010502	022737	024026	001142	CMP	#DULPRT,\$SBDDAT	:DUAL PORT RM80??
524	010510	001410			BEQ	10\$:YES!!
525	010512	010037	001136		MOV	R0,\$BDADR	:LOAD BAD ADDRESS
526	010516	062737	000026	001136	ADD	#RMDT,\$BDADR	
527	010524	104057			EMT	57	
528	010526	000137	021120		JMP	\$EOSP	:GO TO NEXT DEVICE

10\$:

::*****
:*TEST 10 DEVICE AVAILABLE TEST

::*****
TST10:

	010532						
	010532	000004			SCOPE		:SCOPE CALL
	010534	000240			NOP		
	010536	012706	001100		MOV	#STACK,SP	:LOAD THE STACK POINTER
	010542	013700	001276		MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	010546	013701	001466		MOV	TSTQUE,R1	:R1 = POINTER TO DEVICE
	010552	012737	000010	001226	MOV	#10,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
532							
533	010560	004737	022230		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
534	010564	016037	000000	001142	MOV	RMCS1(RO),SBDDAT	:STORE RMCS1 AT SBDDAT
535	010572	042737	173777	001142	BIC	#^CDVA,\$SBDDAT	:CLEAR ALL BUT DVA


```

536 010600 001006          BNE      10$          ;BRANCH IF DVA SET
537 010602 012737 004000 001140    MOV      #DVA,$GDDAT ;SETUP EXPECTED
538 010610 010037 001136          MOV      R0,$BDADR   ;SETUP REG ADDRESS
539 010614 104060          EMT      60
540 010616          10$:
541
542          ::*****
          ;*TEST 11          SEARCH TIMEOUT TEST
          ::*****
          TST11:
          010616          SCOPE          ;SCOPE CALL
          010616 000004          NOP
          010620 000240          MOV      #STACK,SP ;LOAD THE STACK POINTER
          010622 012706 001100    MOV      $BASE,R0   ;R0 = UNIBUS ADDRESS
          010626 013700 001276    MOV      TSTQUE,R1  ;R1 = POINTER TO DEVICE
          010632 013701 001466    MOV      #11,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
          010636 012737 000011 001226
          543          ;LOAD REGISTER OUTPUT BUFFER WITH COMMAND PARAMETERS
          544          MOV      #0,RMDAO ;SECTOR=0=TRACK
          545 010644 012737 000000 001420    MOV      #0,RMDCO ;CYLINDER=0
          546 010652 012737 000000 001446    MOV      #FMT16,RMOFO ;16 BIT FORMAT
          547 010660 012737 010000 001444    MOV      #BUFFER,RMBAO ;STARTING BUFFER ADDRESS
          548 010666 012737 054420 001416    MOV      #-1,RMWCO ;WORD COUNT
          549 010674 012737 177777 001414    MOV      #WD!GO,RMCS10 ;WRITE DATA COMMAND
          550 010702 012737 000061 001412
          551          JSR      PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE
          552 010710 004737 022264          BR      10$          ;SEARCH IS ENABLED USING SUBROUTINE.
          553          EMT      10$          ;BRANCH TO 10$ IF NO ERROR
          554          010714 000402          BR      50$          ;SLOC REST PF TEST
          555          010716 104000
          556          010720 000462
          557          ;START THE CLOCK AND WAIT FOR 100 MS
          558          10$:
          559          MOV      #100, WATCH ;SET WATCHDOG TIMER VALUE
          560          JSR      PC,@CLOCK ;START THE CLOCK
          561          20$:
          562          TST      WATCH
          563          BNE      20$
          564          JSR      PC,@STOPCL ;STOP THE CLOCK
          565          ;VERIFY THAT OPI IS NOT SET (SEARCH TIMEOUT IS DISABLED)
          566          MOV      RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
          567          MOV      R0,$BDADR ;SET UP FOR ERROR MSG
          568          ADD      #RMER1,$BDADR
          569          BIC      #^COPI,$BDDAT
          570          BEQ      30$          ;BRANCH IF NO ERROR
          571          CLR      $GDDAT
          572          EMT      20
          573          BR      50$          ;DISABLED
          574          ;SKIP
          575          ;ENABLE SEARCH TIMEOUT, THEN WAIT 100 MS
          576          30$:
          577          MOV      #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1
          578          MOV      #100, WATCH ;SET WATCHDOG TIMER VALUE
          579          JSR      PC,@CLOCK ;START THE CLOCK
          580          40$:
          581          TST      WATCH
  
```

```

578 011032 001375          BNE      40$
579 011034 004777 170500   JSR      PC,@STOPCL      ;STOP THE CLOCK
580
581                                     :OPI SHOULD NOW BE SET (SEARCH TIMEOUT IS ENABLED)
582 011040 016037 000014 001142   MOV      RMER1(RO),SBDDAT ;STORE RMER1 AT SBDDAT
583 011046 042737 157777 001142   BIC      #^COPI,SBDDAT
584 011054 001004          BNE      50$
585 011056 012737 020000 001140   MOV      #OPI,$GDDAT
586 011064 104021          EMT      21
587 011066          50$:
588
589                                     :*****
: *TEST 12          SET DTE TEST
:*****
: TST12:
011066          SCOPE          ;SCOPE CALL
011066 000004          NOP
011070 000240          MOV      #STACK,SP      ;LOAD THE STACK POINTER
011072 012706 001100   MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
011076 013700 001276   MOV      TSTQUE,R1     ;R1 = POINTER TO DEVICE
011102 013701 001466   MOV      #12,$TESTN    ;:SET TEST NUMBER IN APT MAIL BOX
011106 012737 000012 001226
590
591 011114 010037 001136   MOV      R0,$BDADR     ;SETUP ERROR MSG
592 011120 062737 000014 001136   ADD      #RMER1,$BDADR
593 011126 005037 001140   CLR      $GDDAT
594
595                                     :INITILAIZE AND VERIFY THAT DRIVE TIMING ERROR IS RESET
596 011132 004737 022230   JSR      PC,CNTCLR     ;GO CLEAR CONTROLLER
597
598 011136 016037 000014 001142   MOV      RMER1(RO),SBDDAT ;STORE RMER1 AT SBDDAT
599 011144 042737 167777 001142   BIC      #^CDTE,SBDDAT
600 011152 001402          BEQ      10$           ;BRANCH IF DTE=0
601 011154 104031          EMT      31
602 011156 000517          BR       50$           ;SKIP REST OF TEST
603
604                                     :SET MAINTENANCE INDEX PULSE AND VERIFY DTE REMAINS RESET
605 011160          10$:
606 011160 012760 000001 000024   MOV      #DMD,RMR1(RO) ;LOAD RMR1
607 011166 012760 000005 000024   MOV      #DMD!MI,RMR1(RO) ;LOAD RMR1
608 011174 016037 000014 001142   MOV      RMER1(RO),SBDDAT ;STORE RMER1 AT SBDDAT
609 011202 042737 167777 001142   BIC      #^CDTE,SBDDAT
610 011210 001402          BEQ      20$
611 011212 104000          EMT
612 011214 000500          BR       50$           ;COMPARE SHOULD BE RESET
613
614                                     :EXECUTE DUMMY DATA COMMAND TO ENABLE SEARCH
615 011216          20$:
616 011216 012737 000000 001420   MOV      #0,RMDAO
617 011224 012737 000005 001446   MOV      #5,RMDCO
618 011232 012737 010000 001444   MOV      #FMT16,RMOFO
619 011240 012737 054420 001416   MOV      #BUFFER,RMBAO
620 011246 012737 177777 001414   MOV      #-1,RMWCO
621 011254 012737 000061 001412   MOV      #WD!GO,RMCS10
622
623 011262 004737 022264          JSR      PC,ENBSCH     ;EXECUTE DATA COMMAND TO POINT WHERE
:SEARCH IS ENABLED USING SUBROUTINE.

```

```

011266 000402 BR 30$ ;BRANCH TO 30$ IF NO ERROR
011270 104000 EMT
624 011272 000451 BR 50$
625
626 ;WITH SEARCH ENABLED, SET AND RESET SECTOR PULSE TO SET ENABLE
627 ;SEARCH FLOP.
628 011274 30$:
629 011274 012760 051441 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MS,RMMR1(R0) ;LOAD RMMR1
630 011302 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
631
632 ;SET SECTOR PULSE AND VERIFY DTE DOES NOT SET
633 ;:
634 ;:
635 ;:
636 011310 016037 000014 001142 MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
637 011316 042737 167777 001142 BIC #^CDTE,SBDDAT
638 011324 001402 BEQ 40$
639 011326 104023 EMT 23
640 011330 001432 BEQ 50$ ;COMPARE SHOULD BE RESET
641
642 ;FORCE SECTOR COMPARE
643 011332 40$:
644 011332 012760 051403 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(R0) ;LOAD RMMR1
645 011340 012760 051443 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC!MS,RMMR1(R0) ;LOAD RMMR1
646 011346 012760 051403 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(R0) ;LOAD RMMR1
647
648 ;SET SECTOR PULSE AND VERIFY DTE SETS
649 011354 012760 051441 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MS,RMMR1(R0) ;LOAD RMMR1
650 011362 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
651 011370 016037 000014 001142 MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
652 011376 042737 167777 001142 BIC #^CDTE,SBDDAT
653 011404 001004 BNE 50$
654 011406 012737 010000 001142 MOV #DTE,SBDDAT
655 011414 104024 EMT 24
656 ;SECTOR COMPARE SET
657 011416 50$:
658
659 ;*****
; *TEST 13 FORMAT CHANGE TEST
;*****
TST13:
011416 000004 SCOPE ;SCOPE CALL
011416 000240 NOP
011420 000240
011422 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
011426 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
011432 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
011436 012737 000013 001226 MOV #13,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
660
661 011444 012702 011774 MOV #50$,R2 ;R2=TABLE POINTER
662
663 ;INITIALIZE AND SET THE FORMAT BIT, USE INDEX PULSE TO CLEAR FORMAT CHANGE
664 011450 10$:
665 011450 004737 022230 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
666 011454 011260 000032 MOV (R2),RMOF(R0) ;LOAD FORMAT MODE IN RMOF
667 011460 012760 000001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
668 011466 012760 000005 000024 MOV #DMD!MI,RMMR1(R0) ;LOAD RMMR1
  
```

```

669
670 ;SETUP AND EXECUTE DUMMY DATA COMMAND USING OPPOSITE FORMAT
671 011474 012737 000000 001420 MOV #0,RMDAO
672 011502 012737 000005 001446 MOV #5,RMDCO
673 011510 016237 000002 001444 MOV 2(R2),RMOFC
674 011516 012737 054420 001416 MOV #BUFFER,RMBAO
675 011524 012737 177777 001414 MOV #-1,RMWCO
676 011532 012737 000061 001412 MOV #WD!GO,RMCS10
677
678 011540 004737 022264 JSR PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE
;SEARCH IS ENABLED USING SUBROUTINE.
011544 000402 BR 20$ ;BRANCH TO 20$ IF NO ERROR
011546 104000 EMT
679 011550 000514 BR 60$
680
681 ;FORMAT CHANGE FLOP SHOULD BE SET - VERIFY BY TRYING TO FORCE A
682 ;DRIVE TIMING ERROR WHICH SHOULD NOT SET.
683 011552 20$:
684 011552 012760 051403 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(R0) ;LOAD RMMR1
685 011560 012760 051443 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC!MS,RMMR1(R0) ;LOAD RMMR1
686 011566 012760 051403 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(R0) ;LOAD RMMR1
687 011574 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
688
689 ;VERIFY THAT DRIVE TIMING ERROR DIDNT SET
690 011602 016037 000014 001142 MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
691 011610 042737 167777 001142 BIC #^CDTE,SBDDAT
692 011616 001416 BEQ 30$
693 011620 010037 001136 MOV R0,SBADR ;SETUP ERROR MESSAGE
694 011624 062737 000014 001136 ADD #RMER1,SBADR
695 011632 005037 001140 CLR $GDDAT
696 011636 011237 001174 MOV (R2),STMP0
697 011642 016237 000002 001176 MOV 2(R2),STMP1
698 011650 104025 EMT
699 011652 000453 BR 60$ ;A FORMAT CHANGE
700
701 ;CLEAR THE FORMAT CHANGE FLOP W/INDEX PULSE
702 011654 30$:
703 011654 012760 051405 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MI,RMMR1(R0) ;LOAD RMMR1
704
705 ;ENABLE SEARCH AND FORCE SECTOR COMPARE
706 011662 012760 051403 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(R0) ;LOAD RMMR1
707 011670 012760 051443 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC!MS,RMMR1(R0) ;LOAD RMMR1
708 011676 012760 051403 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(R0) ;LOAD RMMR1
709
710 ;SET DTE W/ANOTHER SECTOR PULSE - VERIFY DTE IS SET
711 011704 012760 051441 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MS,RMMR1(R0) ;LOAD RMMR1
712 011712 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
713 011720 016037 000014 001142 MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
714 011726 042737 167777 001142 BIC #^CDTE,SBDDAT
715 011734 001012 BNE 40$
716 011736 010037 001136 MOV R0,SBADR ;SETUP ERROR MESSAGE
717 011742 062737 000014 001136 ADD #RMER1,SBADR
718 011750 012737 010000 001140 MOV #DTE,$GDDAT
719 011756 104027 EMT
720 011760 000410 BR 60$
721
722 ;DO TEST W/18 TO 16 FORMAT CHANGE DURING SECOND EXECUTION
  
```

```

723 011762          40$:
724 011762 022702 011774      CMP      #50$,R2
725 011766 001005          BNE      60$      ;BRANCH IF DONE TEST
726 011770 005722          TST      (R2)+    ;MOVE POINTER
727 011772 000626          BR       10$      ;REPEAT TEST
728
729 011774 010000          50$:      .WORD    FMT16    ;16-18 FORMAT CHANGE
730 011776 000000          .WORD    0        ;18-16 FORMAT CHANGE
731 012000 010000          .WORD    FMT16
732 012002          60$:          ;END OF TEST
733
734
::*****
:*TEST 14      PROM STROBE TEST
*****
TST14:
012002          SCOPE          ;SCOPE CALL
012002 000004          NOP
012004 000240          MOV      #STACK,SP ;LOAD THE STACK POINTER
012006 012706 001100          MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
012012 013700 001276          MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
012016 013701 001466          MOV      #14,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
012022 012737 000014 001226
735
736 012030 004737 022230          JSR      PC,CNTCLR ;GO CLEAR CONTROLLER
737 012034 010037 001136          MOV      R0,$BDADR ;SETUP ERROR MESSAGE
738 012040 062737 000024 001136          ADD      #RMMR1,$BDADR
739
740          ;SET DIAGNOSTIC MODE AND TRY TO RESET PROM STROBE
741 012046 012760 000001 000024          MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
742 012054 012702 000021          MOV      #17.,R2      ;R2=MAXIMUM # CLOCK PULSES
743 012060          5$:
744 012060 012760 004001 000024          MOV      #DMD!MCLK,RMMR1(R0) ;LOAD RMMR1
745 012066 012760 000001 000024          MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
746 012074 016037 000024 001142          MOV      RMMR1(R0),$BDDAT ;STORE RMMR1 AT SBDDAT
747 012102 042737 177737 001142          BIC      #^CWC,$BDDAT
748 012110 001406          BEQ      6$
749 012112 005302          DEC      R2
750 012114 001361          BNE      5$
751 012116 005037 001140          CLR      $GDDAT
752 012122 104000          EMT
753 012124 000501          BR       60$
754 012126 012702 000021          6$:      MOV      #17.,R2
755 012132          10$:
756 012132 012760 004001 000024          MOV      #DMD!MCLK,RMMR1(R0) ;LOAD RMMR1
757 012140 012760 000001 000024          MOV      #DMD,RMMR1(R0) ;LOAD RMMR1
758 012146 016037 000024 001142          MOV      RMMR1(R0),$BDDAT ;STORE RMMR1 AT SBDDAT
759 012154 042737 177737 001142          BIC      #^CWC,$BDDAT
760 012162 001007          BNE      20$      ;EXIT LOOP WHEN PROM STROBE ON
761 012164 005302          DEC      R2
762 012166 001361          BNE      10$
763 012170 012737 000040 001140          MOV      #WC,$GDDAT
764 012176 104030          EMT
765 012200 000453          BR       60$
766
767 012202 012702 000003          20$:      MOV      #3.,R2
;VERIFY PROM STROBE IS ON FOR 3 BIT CLOCKS

```

```

768 012206          25$: MOV #DMD!MCLK,RMMR1(R0) ;LOAD RMMR1
    012206 012760 004001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
769 012214 012760 000001 000024 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
770 012222 016037 000024 001142 BIC #^CWC,SBDDAT
771 012230 042737 177737 001142 BNE 30$ ;BRANCH IF PROM STORBE DID NOT
772 012236 001005 ;DROP EARLY
773
774 012240 012737 000040 001140 MOV #WC,$GDDAT
775 012246 104032 EMT 32
776 012250 000427 BR 60$
777 012252 005302 30$: DEC R2
778 012254 001354 BNE 25$
779
780 ;VERIFY PROM STROBE IS OFF FOR 8 BIT CLOCKS
781 012256 012702 000010 MOV #8.,R2
782 012262
    012262 012760 004001 000024 40$: MOV #DMD!MCLK,RMMR1(R0) ;LOAD RMMR1
783 012270 012760 000001 000024 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
784 012276 016037 000024 001142 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
785 012304 042737 177737 001142 BIC #^CWC,SBDDAT
786 012312 001404 BEQ 50$ ;BRANCH IF PROM STROBE
787 012314 005037 001140 CLR $GDDAT ;DID NOT SET EARLY
788 012320 104033 EMT 33
789 012322 000402 BR 60$
790 012324 005302 50$: DEC R2
791 012326 001355 BNE 40$
792 012330 60$:
793
794
    ;*****
    ;*TEST 15 SYNC WORD COUNT INHIBIT TEST
    ;*****
    TST15:
    SCOPE ;SCOPE CALL
    NOP
    MOV #STACK,SP ;LOAD THE STACK POINTER
    MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
    MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
    MOV #15,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

795
796 ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES
797 012356 012737 000005 001446 MOV #5,RMDCO ;CYLINDER=5
798 012364 012737 000000 001420 MOV #0,RMDAO ;SECTOR=0, TRACK=0
799 012372 012737 010000 001444 MOV #FMT16,RMFO ;16 BIT FORMAT
800 012400 012737 054420 001416 MOV #BUFFER,RMBAO ;STARTING BUFFER ADDRESS
801 012406 012737 177777 001414 MOV #-1,RMWCO ;WORD COUNT
802 012414 012737 000061 001412 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
803
804 012422 004737 022264 JSR PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE
    ;SEARCH IS ENABLED USING SUBROUTINE.
    012426 000402 BR 10$ ;BRANCH TO 10$ IF NO ERROR
    012430 104000 EMT
805 012432 000562 BR 120$ ;SKIP IF FAILURE
806 012434 10$:
807 012434 004737 023136 JSR PC,SCTCMP ;FORCE SECTOR COMPARE USING SUBROUTINE
    012440 000402 BR 20$ ;BRANCH TO 20$ IF NO ERROR
    012442 104000 EMT
    
```

```

808 012444 000555          BR      120$
809
810          ;CAN NOW STEP DATA TIMING SEQUENCER WITH SECTOR COMPARE SET
811
812          ;STEP THE SEQUENCER TO LOCATION 10(10) WHILE CHECKING 'PLFS'
813 012446 012702 000000 20$:  MOV      #0,R2          ;R2=SEQUENCER ADDRESS
814
815          ;PULSE MAINTENANCE CLOCK UNTIL PROM STROBE SETS
816 012452          30$:
817 012452 012760 055401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(R0) ;LOAD RMMR1
818 012460 012760 051401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
819
820          ;SEE IF PROM STROBE IS SET
821 012466 016003 000024  MOV      RMMR1(R0),R3 ;STORE RMMR1 AT R3
822 012472 032703 000040  BIT      #WC,R3
823 012476 001765          BEQ      30$          ;ISSUE NEXT BIT CLOCK
824 012500 005202          INC      R2          ;INCREMENT SEQUENCER ADDRESS
825
826          ;CHECK 'LOOKING FOR SYNC' - SHOULD BE ZERO UNTIL LOCATION 11
827 012502 010337 001142  MOV      R3,$BDDAT
828 012506 042737 175777 001142  BIC      #^CPLFS,$BDDAT
829 012514 001413          BEQ      50$          ;BRANCH IF PLFS IS ZERO
830 012516 005037 001140  CLR      $GDDAT
831 012522 010037 001136  MOV      R0,$BDADR
832 012526 062737 000024 001136  ADD      #RMMR1,$BDADR
833 012534 010237 001174  MOV      R2,$STMP0
834 012540 104034          EMT      34
835 012542 000516          BR      120$          ;SKIP REST OF TEST
836
837          ;EXIT LOOP IF SEQUENCER NOW AT LOCATION 10
838 012544          50$:
839 012544 022702 000012  CMP      #10.,R2
840 012550 001414          BEQ      70$
841
842          ;PULSE MAINTENANCE CLOCK UNTIL PROM STROBE RESETS
843 012552          60$:
844 012552 012760 055401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(R0) ;LOAD RMMR1
845 012560 012760 051401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
846
847          ;SEE IF PROM STROBE IS RESET
848 012566 016003 000024  MOV      RMMR1(R0),R3 ;STORE RMMR1 AT R3
849 012572 032703 000040  BIT      #WC,R3
850 012576 001365          BNE      60$
851 012600 000724          BR      30$          ;GO STEP SEQUENCER TO NEXT LOC
852
853          ;THE NEXT PROM STROBE SHOULD SET 'PLFS' AT LOCATION 11(10) OF THE
854          ;DATA TIMING SEQUENCER.
855 012602          70$:
856 012602 012702 000020  MOV      #16.,R2          ;R2=NUMBER OF BIT CLOCKS
857
858          ;ISSUE 16 BIT CLOCKS TO GET NEXT PROM STROBE
859 012606          80$:
860 012606 012760 055401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(R0) ;LOAD RMMR1
861 012614 012760 051401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
862 012622 005302          DEC      R2
863 012624 001370          BNE      80$
864

```

```

865 ;VERIFY THAT 'PLFS' IS NOW SET
866 012626 016037 000024 001142 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
867 012634 042737 175777 001142 BIC #^CPLFS,SBDDAT
868 012642 001012 BNE 90$ ;BRANCH IF PLFS IS SET
869 012644 012737 002030 001140 MOV #PLFS,$GDDAT ;SETUP ERROR MESSAGE
870 012652 010037 001136 MOV R0,SBADR
871 012656 062737 000024 001136 ADD #RMMR1,SBADR
872 012664 104035 EMT 35
873 012666 000444 BR 120$ ;SKIP REST OF TEST
874
875 ;ISSUE 3 MORE BIT CLOCKS TO RESET PROM STROBE
876 012670 90$:
877 012670 012702 000003 MOV #3,R2
878 012674 95$:
879 012674 012760 055401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(R0) ;LOAD RMMR1
880 012702 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
881 012710 005302 DEC R2
882 012712 001370 BNE 95$
883 ;WITH 'LOOKING FOR SYNC SET, FURTHER BIT CLOCKS SHOULD NOT SET
884 ;PROM STROBE
885 012714 012702 000040 MOV #32.,R2 ;R2=NUMBER OF BIT CLOCKS
886
887 ;PULSE BIT CLOCK AND VERIFY PROM STROBE DOES NOT SET
888 012720 100$:
889 012720 012760 055401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(R0) ;LOAD RMMR1
890 012726 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
891 012734 016003 000024 MOV RMMR1(R0),R3 ;STORE RMMR1 AT R3
892 012740 042703 177737 BIC #^CWC,R3
893 012744 001413 BEQ 110$ ;BRANCH IF PROM STROBE IS 0
894 012746 005037 001140 CLR $GDDAT ;SETUP ERROR MESSAGE
895 012752 010337 001142 MOV R3,SBDDAT
896 012756 010037 001136 MOV R0,SBADR
897 012762 062737 000024 001136 ADD #RMMR1,SBADR
898 012770 104036 EMT 36
899 012772 000402 BR 120$
900 012774 005302 110$: DEC R2 ;CONTINUE FOR 32 BIT CLOCKS
901 012776 001350 BNE 100$
902
903 013000 120$: ;END OF TEST
904
905 ;*****
;*TEST 16 SYNC DETECTION TEST
;*****
TST16:
013000 SCOPE ;SCOPE CALL
013000 000004 NOP
013002 000240 MOV #STACK,SP ;LOAD THE STACK POINTER
013004 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
013010 013700 001276 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
013014 013701 001466 MOV #16,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
013020 012737 000016 001226

906 ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES
907 MOV #0,RMDAO ;SECTOR 0,TRACK 0
908 013026 012737 000000 001420 MOV #5,RMDCO ;CYLINDER 5
909 013034 012737 000005 001446 MOV #FMT16,RMOFO ;16 BIT MODE
910 013042 012737 010000 001444
  
```



```

T16 SYNC DETECTION TEST
911 013050 012737 054420 001416      MOV    #BUFFER,RMBAO    ;BUFFER ADDRESS
912 013056 012737 177777 001414      MOV    #-1,RMWC0       ;CORD COUNT
913 013064 012737 000071 001412      MOV    #RD!GO,RMCS10   ;READ DATA COMMAND
914
915 013072 004737 022264                JSR    PC,ENBSCH        ;EXECUTE DATA COMMAND TO POINT WHERE
                                ;SEARCH IS ENABLED USING SUBROUTINE.
                                ;BRANCH TO 10$ IF NO ERROR
                                10$:
                                BR    10$
                                100$:
                                BR    100$
916 013076 000402
013100 104000
013102 000550
917 013104
918 013104 004737 023136                JSR    PC,SCTCMP        ;FORCE SECTOR COMPARE USING SUBROUTINE
                                ;BRANCH TO 20$ IF NO ERROR
                                20$:
                                BR    20$
                                200$:
                                BR    200$
919 013114 000543
920 013116
921 013116 004737 023244                JSR    PC,SETLFS        ;SET 'LOOKING FOR SYNC' USING SUBROUTINE
                                ;BRANCH TO 30$ IF NO ERROR
                                30$:
                                BR    30$
                                300$:
                                BR    300$
922 013126 000536
923
924
925
926 013130
927 013130 012702 000020                ;CLOCK ALL ONES SYNC PATTERN AND VERIFY SYNC IS NOT DETECTED
                                ;(USING PROM STORBE AS INDICATION)
                                30$:
                                MOV    #16.,R2          ;NUMBER OF ONE BITS
                                MOV    R0,$BDDADR        ;SETUP ERROR MESSAGE
                                ADD    #RMR1,$BDDADR
                                CLR    $GDDAT
                                MOV    #MR1AAA!MRD,RMR1(R0) ;LOAD RMR1
                                40$:
                                MOV    #MR1AAA!MRD!MCLK,RMR1(R0) ;LOAD RMR1
                                MOV    #MR1AAA!MRD,RMR1(R0) ;LOAD RMR1
                                MOV    RMR1(R0),$BDDAT    ;STORE RMR1 AT $BDDAT
                                BIC    #^CWC,$BDDAT
                                BEQ    50$
                                MOV    #177777,$TMP0
                                EMT    37
                                BR    100$
928 013134 010037 001136
929 013140 062737 000024 001136
930 013146 005037 001140
931 013152 012760 053401 000024
932 013160
933 013160 012760 057401 000024
934 013166 012760 053401 000024
935 013174 016037 000024 001142
936 013202 042737 177737 001142
937 013210 001405
938 013212 012737 177777 001174
939 013220 104037
940 013222 000500
941 013224
942 013224 005302
943 013226 001354
944 013230 012702 000020
945
946
947 013234
948 013234 012760 055401 000024
949 013242 012760 051401 000024
950 013250 016037 000024 001142
951 013256 042737 177737 001142
952 013264 001404
953 013266 005037 001174
954 013272 104037
955 013274 000453
956 013276
957 013276 005302
958 013300 001355
959 013302 012702 000040
960 013306 012703 000031
                                ;CLOCK ALL ZERO SYNC PATTERN AND VERIFY SYNC IS NOT DETECTED
                                60$:
                                MOV    #MR1AAA!MCLK,RMR1(R0) ;LOAD RMR1
                                MOV    #MR1AAA,RMR1(R0) ;LOAD RMR1
                                MOV    RMR1(R0),$BDDAT    ;STORE RMR1 AT $BDDAT
                                BIC    #^CWC,$BDDAT    ;MAKE SURE SYNC NOT DETECTED
                                BEQ    70$
                                CLR    $TMP0
                                EMT    37
                                BR    100$
                                70$:
                                DEC    R2              ;REPEAT FOR 16 BITS
                                BNE    60$
                                MOV    #32.,R2          ;R2=NUMBER OF BITS
                                MOV    #000031,R3        ;R3=SYNC PATTERN FOR LEFT SHIFT

```

```

961
962
963
964 013312 012737 051401 001436 80$: MOV #MR1AAA,RMMR10 ;GENERATE VALUE OF RMMR1
965 013320 000241 CLC ;CLEAR THE CARR
966 013322 006003 ROR R3 ;SHIFT RIGHT
967 013324 103003 BCC 90$ ;BRANCH IF C BIT CLEAR
968 013326 052737 002000 001436 90$: BIS #MRD,RMMR10 ;SET MRD IF PATTERN BIT SETS
969 013334 013760 001436 000024 MOV RMMR10,RMMR1(R0) ;LOAD RMMR1
970 013342 052737 004000 001436 BIS #MCLK,RMMR10 ;SET THE MAINTENANCE DATA CLK
971 013350 013760 001436 000024 MOV RMMR10,RMMR1(R0) ;LOAD RMMR1
972 013356 042737 004000 001436 BIC #MCLK,RMMR10 ;RESET BIT CLOCK
973 013364 013760 001436 000024 MOV RMMR10,RMMR1(R0) ;LOAD RMMR1
974 013372 016037 000024 001142 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
975 013400 042737 177737 001142 BIC #CWC,SBDDAT
976 013406 001006 ONE 100$ ;BRANCH IF PROM STROBE IS SET
977 013410 005302 DEC R2 ;CONTINUE FOR 16 BIT CLOCKS
978 013412 001337 BNE 80$
979 013414 012737 000040 001140 MOV #WC,$GDDAT
980 013422 104040 EMT 40
981 013424 100$: ;END OF TEST
982
983
*****
;*TEST 17 ABORT SYNC DETECTION TEST
*****
TST17:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV SBASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #17,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

984
985 ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES
986 013452 012737 000000 001420 MOV #0,RMDAO ;SECTOR 0, TRACK 0
987 013460 012737 000005 001446 MOV #5,RMDCO ;CYLINDER 5
988 013466 012737 010000 001444 MOV #FMT16,RMFO ;16 BIT FORMAT
989 013474 012737 054420 001416 MOV #BUFFER,RMBAO ;STARTING BUFFER ADDRESS
990 013502 012737 177777 001414 MOV #-1,RMWC ;WORD COUNT
991 013510 012737 000071 001412 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
992
993 013516 004737 022264 JSR PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE
;SEARCH IS ENABLED USING SUBROUTINE.
;BRANCH TO 10$ IF NO ERROR
013522 000402 BR 10$
013524 104000 EMT
994 013526 000447 BR 50$
995 013530 10$:
996 013530 004737 023136 JSR PC,SCTCMP ;FORCE SECTOR COMPARE USING SUBROUTINE
;BRANCH TO 20$ IF NO ERROR
013534 000402 BR 20$
013536 104000 EMT
997 013540 000442 BR 50$
998 013542 20$:
999 013542 004737 023244 JSR PC,SETLFS ;SET 'LOOKING FOR SYNC' USING SUBROUTINE
;BRANCH TO 30$ IF NO ERROR
013546 000402 BR 30$
013550 104000 EMT
  
```

```

1000 013552 000435          BR      50$
1001
1002          ;LOOKING FOR SYNC INHIBITS WORD CLOCK, HOWEVER, 'DRIVE TIMING ERROR'
1003          ;SHOULD RESET 'LFS WORD COUNT INHIBIT' FLOP
1004          ;FORCE DRIVE TIMING ERROR
1005 013554
1006 013554 012760 051441 000024 30$:  MOV      #MR1AAA!MS,RMMR1(R0)      ;LOAD RMMR1
1007
1008          ;PULSE BIT CLOCK AND VERIFY PROM STROBE SETS
1009 013562 012702 000020 40$:  MOV      #16.,R2                ;R2, NUMBER OF BIT CLOCKS
1010 013566
1011 013566 012760 055441 000024  MOV      #MR1AAA!MS!MCLK,RMMR1(R0)      ;LOAD RMMR1
1012 013574 012760 051441 000024  MOV      #MR1AAA!MS,RMMR1(R0)      ;LOAD RMMR1
1013 013602 016037 000024 001142  MOV      RMMR1(R0),SBDDAT           ;STORE RMMR1 AT SBDDAT
1014 013610 042737 177737 001142  BIC      #^CWC,SBDDAT
1015 013616 001013          BNE      50$
1016 013620 005302          DEC      R2
1017 013622 001361          BNE      40$
1018 013624 012737 000040 001140  MOV      #WC,$GDDAT
1019 013632 010037 001136          MOV      R0,$BDADR
1020 013636 062737 000024 001136  ADD      #RMMR1,$BDADR
1021 013644 104041          EMT      41
1022 013646 50$:
1023
1024
1025          ;*****
          ;*TEST 20          SYNC GENERATION 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
1026
1027          ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES
1028 013674 012737 000000 001420  MOV      #0,RMDAO          ;SECTOR 0, TRACK 0
1029 013702 012737 000005 001446  MOV      #5,RMDCO          ;CYL 5
1030 013710 012737 010000 001444  MOV      #FMT16,RMOFO          ;16 BIT MODE
1031 013716 012737 054420 001416  MOV      #BUFFER,RMBAO          ;BUFFER ADDRESS
1032 013724 012737 177776 001414  MOV      #-2,RMWCO          ;WORD COUNT
1033 013732 012737 000063 001412  MOV      #WH!GO,RMCS10          ;WRITE HEAD AND DATA COMMAND
1034
1035 013740 004737 022264          JSR      PC,ENBSCH          ;EXECUTE DATA COMMAND TO POINT WHERE
          ;SEARCH IS ENABLED USING SUBROUTINE.
          ;BRANCH TO 10$ IF NO ERROR
          BR      10$
          EMT
          JMP      160$
1036 013750 000137 014340 10$:
1037 013754
1038 013754 004737 023136          JSR      PC,SCTCMP          ;FORCE SECTOR COMPARE USING SUBROUTINE
          BR      20$          ;BRANCH TO 20$ IF NO ERROR
          EMT
          BR      160$
1039 013764 000565
1040
1041          ;WRITE PROM OF DATA TIMING SEQUENCER SHOULD NOW BE ENABLED
  
```

```

1042
1043
1044 013766
1045 013766 016037 000040 001142
1046 013774 042737 177776 001142
1047 014002 001011
1048 014004 005037 001140
1049 014010 010037 001136
1050 014014 062737 000040 001136
1051 014022 104042
1052 014024 000545
1053
1054
1055
1056 014026
1057 014026 012702 000016
1058 014032
1059 014032 016037 000040 001142
1060 014040 032737 000001 001142
1061 014046 001014
1062 014050 005302
1063 014052 001367
1064 014054 010037 001136
1065 014060 062737 000040 001136
1066 014066 012737 000001 001140
1067 014074 104042
1068 014076 000520
1069 014100
1070
1071 014100
1072 014100 012760 055401 000024
1073 014106 012760 051401 000024
1074 014114 016003 000024
1075 014120 032703 000040
1076 014124 001765
1077
1078
1079 014126 005302
1080 014130 001414
1081
1082
1083 014132
1084 014132 012760 055401 000024
1085 014140 012760 051401 000024
1086 014146 016003 000024
1087 014152 032703 000040
1088 014156 001725
1089 014160 000764
1090
1091
1092
1093
1094 014162
1095 014162 012702 000020
1096 014166 005003
1097 014170
    014170 016004 000024
    
```

```

;FIRST, VERIFY THAT WRITE GATE IS ON, INDICATING THAT
;SECTOR COMPARE SET
20$:
MOV RMMR2(R0),SBDDAT ;STORE RMMR2 AT SBDDAT
BIC #^CBB00,SBDDAT ;BUS BIT 0 IS WRITE GATE
BNE 30$
CLR $GDDAT
MOV R0,SBDADR
ADD #RMMR2,SBDADR
EMT 42
BR 160$ ;FORMAT-CS FAILURE

;STEP THE DATA TIMING SEQUENCER TO LOCATION 16 AND VERIFY WRITE
;GATE STAYS ON
30$:
MOV #14.,R2 ;MAXMUM NUMBER OF CLOCK
40$:
MOV RMMR2(R0),SBDDAT ;STORE RMMR2 AT SBDDAT
BIT #BB00,SBDDAT
BNE 50$ ;BRANCH IF WRITE GATE ON
DEC R2
BNE 40$
MOV R0,SBDADR
ADD #RMMR2,SBDADR
MOV #BB00,$GDDAT
EMT 42
BR 160$ ;FORMAT - CS FAILURE

50$:
60$:
MOV #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
MOV #MR1AAA,RMMR1(R0) ;LOAD RMMR1
MOV RMMR1(R0),R3 ;STORE RMMR1 AT R3
BIT #WC,R3
BEQ 60$

;PROM STROBE CAME ON-DECREMENT PROM STROBE COUNT
DEC R2
BEQ 80$

;WAIT FOR PROM STROBE TO GO OFF, THEN REPEAT LOOP
70$:
MOV #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
MOV #MR1AAA,RMMR1(R0) ;LOAD RMMR1
MOV RMMR1(R0),R3 ;STORE RMMR1 AT R3
BIT #WC,R3
BEQ 40$
BR 70$

;VERIFY HEADER SYNC GENERATION
;WRITE DATA BIT IS INVERTED AT MAINTENANCE REGISTER
;FIRST,COUNT NUMBER OF ZERO BITS UNTIL FIRST ONE BIT
80$:
MOV #16.,R2 ;MAX TIMES THRU LOOP
CLR R3
90$:
MOV RMMR1(R0),R4 ;STORE RMMR1 AT R4
    
```

```

1098 014174 032704 000010          BIT    #MWD,R4
1099 014200 001414          BEQ    110$      ;JUMP OUT OF LOOP WITH FIRST 1
1100 014202 005203          INC    R3       ;INCREMENT ZERO COUNT
1101 014204 005302          DEC    R2       ;DECREMENT MAX LOOP COUNT
1102 014206 001002          BNE    100$
1103 014210 104043          EMT    43
1104 014212 000452          BR     160$      ;HEADER SYNC ,CANT GET FIRST 1
1105 014214
1106 014214 012760 055401 000024 100$:  MOV    #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1107 014222 012760 051401 000024  MOV    #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1108 014230 000757          BR     90$
1109
1110          ;MAKE SURE THERE WERE AT LEAST 8 ZERO BITS IN HEADER SYNC
1111 014232          110$:
1112 014232 020327 000010          CMP    R3,#8.
1113 014236 103002          BHIS  120$
1114 014240 104043          EMT    43
1115 014242 000436          BR     160$      ;HEADER SYNC
1116
1117          ;SAMPLE AND STORE THE REST OF THE HEADER SYNC
1118 014244          120$:
1119 014244 012702 000010          MOV    #8.,R2    ;NUMBER OF SAMPLES
1120 014250 005003          CLR    R3       ;HEADER SYNC
1121 014252          130$:
1122 014252 016004 000024          MOV    RMMR1(R0),R4 ;STORE RMMR1 AT R4
1123 014256 006303          ASL    R3
1124 014260 032704 000010          BIT    #MWD,R4    ;SHIFT SAMPLE,IF SYNC BIT IS 1
1125 014264 001002          BNE    140$      ;THEN SET SAMPLE INPUT
1126 014266 052703 000001          BIS    #BIT0,R3
1127 014272 005302          140$:  DEC    R2
1128 014274 001407          BEQ    150$
1129 014276 012760 055401 000024  MOV    #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1130 014304 012760 051401 000024  MOV    #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1131 014312 000757          BR     130$      ;DO LOOP TIL R2=0
1132
1133          ;VERIFY THE SYNC BIT STREAM IS 0000000010011000
1134 014314          150$:
1135 014314 012737 000230 001140          MOV    #000230,$GDDAT
1136 014322 010337 001142          MOV    R3,$BDDAT
1137 014326 023737 001140 001142          CMP    $GDDAT,$BDDAT
1138 014334 001401          BEQ    160$
1139 014336 104044          EMT    44
1140 014340          160$:
1141
1142          ;*****
          ;*TEST 21      WRITE HEADER TEST
          ;*****
          TST21:
014340          SCOPE          ;SCOPE CALL
014340 000004          NOP
014342 000240          MOV    #STACK,SP ;LOAD THE STACK POINTER
014344 012706 001100          MOV    $BASE,R0   ;R0 = UNIBUS ADDRESS
014350 013700 001276          MOV    TSTQUE,R1  ;R1 = POINTER TO DEVICE
014354 013701 001466          MOV    #21,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
014360 012737 000021 001226
1143
1144          ;SETUP THE REGISTER_OUTPUT BUFFER FOR SUBROUTINES
    
```

```

1145 014366 012737 001060 001446      MOV      #560.,RMDCO      ;LAST CYLINDER
1146 014374 013737 001334 001420      MOV      LSTRK,RMDAO     ;SET LAST TRACK AND
1147 014402 112737 000036 001420      MOVVB   #30.,RMDAO     ;LAST SECTOR
1148 014410 012737 010000 001444      MOV      #FMT16,RMOFO   ;IN 16 BIT FORMAT
1149 014416 012702 054420      MOV      #BUFFER,R2     ;BUFFER ADDRESS
1150 014422 010237 001416      MOV      R2,RMBAO      ;BUFFER STARTS
1151                                     ;STORE FIRST HEADER WORD
1152 014426 012712 150000      MOV      #150000,(R2)   ;SET MF/UF BITS GOOD, 16 BIT FORMAT AND
1153 014432 052722 001060      BIS      #560.,(R2)+   ;LAST CYLINDER
1154                                     ;STORE SECOND HEADER WORD
1155 014436 013712 001334      MOV      LSTRK,(R2)    ;SET LAST TRACK ADDRESS AND
1156 014442 112712 000036      MOVVB   #30.,(R2)     ;SECTOR ADDRESS.
1157 014446 012737 177776 001414      MOV      #-2,RMWCO     ;2 WORDS (2'S COMP)
1158 014454 012737 000063 001412      MOV      #WH!GO,RMCS10 ;WRITE HEADER COMMAND
1159
1160 014462 004737 022264      JSR      PC,ENBSCH     ;EXECUTE DATA COMMAND TO POINT WHERE
                                ;SEARCH IS ENABLED USING SUBROUTINE.
                                ;BRANCH TO 10$ IF NO ERROR
                                10$:
                                BR      10$
                                EMT
                                JMP      160$
1161 014472 000137 015274
1162 014476
1163 014476 004737 023136      JSR      PC,SCTCMP     ;FORCE SECTOR COMPARE USING SUBROUTINE
                                ;BRANCH TO 20$ IF NO ERROR
                                20$:
                                BR      20$
                                EMT
                                JMP      160$
1164 014506 000137 015274
1165
1166                                     ;STEP THE DATA TIMING SEQUENCER UNTIL 'HEADER AREA' COMES ON
                                20$:
1167 014512
1168 014512 012702 000017      MOV      #15.,R2       ;ALLOW 15 MORE PROM STORBES
1169 014516 012704 000041      MOV      #33.,R4      ;ALLOW 33 BIT CLOCKS PER PROM STROBE
1170
1171                                     ;PULSE BIT CLOCK UNTIL PROM STROBE IS ON
                                40$:
1172 014522
1173 014522 012760 055401 000024      MOV      #R1AAA!MCLK,RMPR1(R0) ;LOAD RMPR1
1174 014530 012760 051401 000024      MOV      #R1AAA,RMPR1(R0)   ;LOAD RMPR1
1175 014536 016003 000024      MOV      RMPR1(R0),R3     ;STORE RMPR1 AT R3
1176 014542 032703 000040      BIT      #VC,R3
1177 014546 001004      BNE      50$
1178 014550 005304      DEC      R4
1179 014552 001363      BNE      40$
1180 014554 000137 015274      JMP      160$          ;COUNT EXHAUSTED
1181
1182                                     ;SEE IF HEADER AREA CAME ON
                                50$:
1183 014560
1184 014560 032703 000200      BIT      #PHA,R3
1185 014564 001062      BNE      80$
1186 014566 005302      DEC      R2
1187 014570 001015      BNE      60$
1188
1189                                     ;ERROR-HEADER AREA NEVER CAME ON
1190 014572 012737 000200 001140      MOV      #PHA,$GDDAT     ;SETUP ERROR MESSAGE
1191 014600 005037 001142      CLR      $BDDAT
1192 014604 010037 001136      MOV      R0,$BDADR
1193 014610 062737 000024 001136      ADD      #RMPR1,$BDADR
1194 014616 104045      EMT      45
1195 014620 000137 015274      JMP      160$          ;SKIP REST OF TEST
1196

```

```

1197                                     :WAIT FOR PROM STROBE TO GO OFF
1198 014624                                     60$:
1199 014624 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0)  ;LOAD RMMR1
1200 014632 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1201 014640 016003 000024              MOV      RMMR1(R0),R3           ;STORE RMMR1 AT R3
1202 014644 032703 000040              BIT      #WC,R3
1203 014650 001404                      BEQ      70$
1204 014652 005304                      DEC      R4
1205 014654 001363                      BNE     60$
1206 014656 000137 015274              JMP      160$                  ;COUNT EXHAUSTED
1207
1208                                     :VERIFY THE TAG BUS ONCE EVERY PROM STROBE
1209 014662                                     70$:
1210 014662 016037 000040 001142      MOV      RMMR2(R0),SBDDAT      ;STORE RMMR2 AT SBDDAT
1211 014670 042737 176000 001142      BIC      #^C1777,SBDDAT
1212 014676 022737 000001 001142      CMP      #BB00,SBDDAT        ;WRITE GATE SHOULD BE ON
1213 014704 001704                      BEQ      30$                  ;ALL ELSE OFF
1214 014706 010037 001136              MOV      R0,SBDADR
1215 014712 062737 000040 001136      ADD      #RMMR2,SBDADR
1216 014720 012737 000001 001140      MOV      #BB00,$GDDAT
1217 014726 104042                      EMT     42
1218 014730 000561                      BR      160$
1219
1220                                     :HEADER AREA IS NOW ON CAN START SAMPLE
1221                                     :DATA AFTER 5 MORE BIT CLOCKS
1222 014732                                     80$:
1223 014732 012702 000005              MOV      #5,R2
1224 014736                                     81$:
1225 014736 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1226 014744 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0)    ;LOAD RMMR1
1227 014752 005302                      DEC      R2
1228 014754 001370                      BNE     81$
1229
1230                                     :SAMPLE AND STORE THE TWO HEADER WORDS
1231 014756 005037 001174              CLR      $TMP0
1232 014762 005037 001176              CLR      $TMP1
1233 014766 012702 001174              MOV      #TMP0,R2             ;R2 NUMBER OF WORDS/ADDRESS
1234 014772 012703 000020      90$:  MOV      #16.,R3           ;NUMBER OF BITS
1235 015000 005004                      CLR      R4                   ;WORD STORAGE
1236 015000 016005 000024      100$: MOV      RMMR1(R0),R5        ;STORE RMMR1 AT R5
1237 015004 006304                      ASL      R4                   ;SHIFT WORD SAMPLE
1238 015006 032705 000010              BIT      #WD,R5              ;SET BIT 0 IF WRITE BIT ON
1239 015012 001002                      BNE     110$
1240 015014 052704 000001              BIS      #BIT0,R4
1241 015020                                     110$:
1242 015020 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1243 015026 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0)    ;LOAD RMMR1
1244 015034 005303                      DEC      R3
1245 015036 001360                      BNE     100$
1246 015040 010422                      MOV      R4,(R2)+            ;STORE WORD SAMPLE
1247 015042 022702 001200      CMP      #TMP0+4,R2          ;DONE ?
1248 015046 101415                      BLOS    120$                 ;YES
1249
1250                                     :HEADER AREA SHOULD STILL BE ON FOR SECOND HEADER WORD
1251 015050 042705 177577              BIC      #^C<PHA>,R5
1251 015054 001346                      BNE     90$                  ;BRANCH IF ON
    
```

```

1252 015056 010037 001136      MOV      R0,$BDADR
1253 015062 062737 000024 001136      ADD      #RMMR1,$BDADR
1254 015070 012737 000200 001140      MOV      #PHA,$GDDAT
1255 015076 104045      EMT      45
1256 015100 000475      BR       160$
1257
1258      ;WAIT UNTIL ECRC SETS
1259 015102      120$:
1260 015102 005003      CLR      R3      ;CHECK NUMBER BITS DIFFER BET PHA AND ECRC
1261
1262      125$:
1263 015104 016005 000024      MOV      RMMR1(R0),R5 ;STORE RMMR1 AT R5
1264 015110 042705 176777      BIC      #^C<ECRC>,R5 ;CHECK OUT THE REST BITS
1265 015114 001026      BNE      127$      ;BRANCH IF ECRC ON
1266 015116 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1267 015124 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0) ;LOAD RMMR1
1268 015132 005203      INC      R3      ;TOTAL NUMBER OF BIT DIFF
1269 015134 022703 000020      CMP      #16.,R3 ;OVER ONE WORD ?
1270 015140 101361      BHI      125$      ;BRANCH IF NOT
1271 015142 010037 001136      MOV      R0,$BDADR
1272 015146 062737 000024 001136      ADD      #RMMR1,$BDADR
1273 015154 012737 001000 001140      MOV      #ECRC,$GDDAT
1274 015162 005037 001142      CLR      $BDDAT ;CLEAR THE EXP
1275 015166 104046      EMT      46
1276 015170 000441      BR       160$      ;EXIT
1277
1278      ;SAMPLE AND STORE CRC WORD
1279 015172      127$:
1280 015176 012703 000020      MOV      #16.,R3
1281 015200      130$:
1282 015200 016005 000024      MOV      RMMR1(R0),R5 ;STORE RMMR1 AT R5
1283 015204 006304      ASL      R4      ;SHIFT WORD SAMPLE
1284 015206 032705 000010      BIT      #MWD,R5 ;SET BIT 0 IF WRITE BIT ON
1285 015212 001002      BNE      140$
1286 015214 052704 000001      BIS      #BIT0,R4
1287 015220      140$:
1288 015222 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1289 015226 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0) ;LOAD RMMR1
1290 015234 005303      DEC      R3
1291 015236 001360      BNE      130$
1292 015240 010422      MOV      R4,(R2)+
1293
1294      ;VERIFY 3 HEADER WORDS
1295      ;NOTE THAT DATA WAS STORED IN THE REVERSE
1296      ;ORDER FROM WHICH IT WAS WRITTEN
1297 015242 022737 006113 001174      CMP      #006113,$TMP0 ;CORRECT FIPST HDR WRD ?
1298 015250 001010      BNE      150$      ;NO !!
1299 015252 022737 074260 001176      CMP      #074260,$TMP1 ;CORRECT SECOND HDR WRD ?
1300 015260 001004      BNE      150$      ;NO !!
1301 015262 022737 163072 001200      CMP      #163072,$TMP2 ;CORRECT HEADER CRC ?
1302 015270 001401      BEQ      160$      ;YES !!
1303 015272      150$:
1304 015272 104047      EMT      47
1305      160$:
1306      ;END OF SUB TEST
1307
1308      ;:*****
    
```


;*TEST 22 HEADER COMPARE TEST

```

015274
015274 000004
015276 000240
015300 012706 001100
015304 013700 001276
015310 013701 001466
015314 012737 000022 001226
1306
1307
1308 015322 012737 001060 001446
1309 015330 013737 001334 001420
1310 015336 112737 000036 001420
1311 015344 012737 054420 001416
1312 015352 012737 177776 001414
1313 015360 012737 010000 001444
1314 015366 012737 000073 001412
1315
1316 015374 004737 022264
    015400 000403
    015402 104000
1317 015404 000137 016524
1318 015410
1319 015410 004737 023136
    015414 000403
    015416 104000
1320 015420 000137 016524
1321
1322
1323 015424
1324 015424 010037 001136
1325 015430 062737 000040 001136
1326 015436 012702 000004
1327 015442 012704 000021
1328
1329
1330 015446
1331 015446 012760 055401 000024
1332 015454 012760 051401 000024
1333 015462 016003 000024
1334 015
    CZVTMAO VT61 ACCPT TST
    ZVTN      AO
    ZVTO      AO
    CZVTNAO  VT105 ACCPT TST
    CZVTOAO  M7142 ACCPT TST
    
```

```

TST22:
    SCOPE
    NOP
    MOV #STACK,SP ;LOAD THE STACK POINTER
    MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
    MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
    MOV #22,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
;LOAD REGISTER OUTPUT BUFFER FOR READ HEADER COMMAND
    MOV #560.,RMDCO ;LAST CYLINDER
    MOV LSTRK,RMDAO ;SET LAST TRACK AND
    MOVB #30.,RMDAO ;LAST SECTOR
    MOV #BUFFER,RMBAO ;DATA ADDRESS
    MOV #-2,RMWCO ;2 WORDS (2'S COMP)
    MOV #FMT16,RMOFO ;16 BIT MODE
    MOV #RH!GO,RMC$10 ;READ HEADER AND DATA FUN
    JSR PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE
    ;SEARCH IS ENABLED USING SUBROUTINE.
    BR 10$ ;BRANCH TO 10$ IF NO ERROR
    EMT
    JMP 230$ ;SKIP REST OF TEST IF ERROR
10$:
    JSR PC,SCTCMP ;FORCE SECTOR COMPARE USING SUBROUTINE
    BR 20$ ;BRANCH TO 20$ IF NO ERROR
    EMT
    JMP 230$
;READ GATE SHOULD BE OFF FOR 3 PROM STROBES
20$:
    MOV R0,$BDADR ;SETUP ERROR MESSAGE
    ADD #RMR2,$BDADR
    MOV #4,R2 ;R2=NUMBER OF PROM STROBES
30$:
    MOV #17.,R4 ;MAX BIT CLOCKS
;CLOCK BIT CLOCK UNTIL PROM STROBE COMES ON
40$:
    MOV #MR1AAA!MCLK,RMR1(R0) ;LOAD RMR1
    MOV #MR1AAA,RMR1(R0) ;LOAD RMR1
    MOV RMR1(R0),R3 ;STORE RMR1 AT R3
    
```

** END OF REPORT **