

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

IDENTIFICATION

PRODUCT CODE: AC-F933B-MC
PRODUCT NAME: CZRMQBO RM05/3/2 DISKLESS TEST, PT 2
PRODUCT DATE: APRIL 1981
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980,1981 DIGITAL EQUIPMENT CORPORATION

CONTENTS

1
2
3
4
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

- 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 RM05/3/2 DISKLESS DIAGNOSTIC IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL AND DIAGNOSTIC MEANS TO VERIFY THE OPERABILITY OF THE 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 RM MASSBUS ADAPTER;

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

1.2 UNIT UNDER TEST

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

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

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

- PDP-11 PROCESSOR
- 12K MEMORY
- KW11-L OR LW11-P CLOCK
- PROGRAM LOADING DEVICE
- TERMINAL
- RH11 OR RH70 CONTROLLER
- 1 TO 8 DISK DRIVES (ANY COMBINATION OF RM05'S, RM03'S OR RM02'S)

2.2 MEDIA REQUIREMENTS

NONE

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

RM05/3/2 DISKLESS TEST, PT 1

3.0 OPERATING PROCEDURE

3.1 LOADING

58 THE PROGRAM MAY BE LOADED BY EITHER PAPER TAPE, USING THE
59 STANDARD PAPER TAPE LOADING PROCEDURE, OR XXDP MEDIA, USING THE
60 APPROPRIATE LOADING DEVICE.
61
62

63 3.2 SWITCH OPTIONS

64 THE FOLLOWING SWITCH OPTIONS ARE INVOKED WHEN THE APPROPRIATE
65 SWITCH IS ON.
66

67 SW15 HALT ON ERROR
68 SW14 LOOP ON TEST (CURRENTLY BEING EXECUTED)
69 SW13 INHIBIT ERROR TYPEOUTS
70 SW12 UNUSED
71 SW11 INHIBIT TEST ITERATIONS
72 SW10 BELL ON ERROR
73 SW09 LOOP ON ERROR
74 SW08 LOOP ON TEST IN SW07-00
75
76
77

78 THE LOW ORDER 8 SWITCHES ARE USED IN CONJUNCTION WITH SW08 TO SPECIFY
79 THE OCTAL NUMBER OF THE TEST WHICH THE PROGRAM WILL LOOP ON.
80
81

82 3.3 STARTING

83 THE PROGRAM MAY BE STARTED AT LOCATION 200 OR 204. STARTING AT
84 200 WILL BE THE NORMAL STARTING ADDRESS. STARTING AT 204 WILL ENABLE
85 THE RH/RM BASE ADDRESS TO BE CHANGED. IF RUNNING IN A STAND-ALONE
86 ENVIRONMENT, THE PROGRAM USES CONSOLE DIALOGUE TO ALLOW THE OPERATOR
87 TO CONTROL TEST CONDITIONS.
88
89
90

91 3.4 HALTING

92 THE PROGRAM CAN BE HALTED BY TYPING CONTROL C FROM THE CONSOLE OR
93 BY PRESSING THE HALT SWITCH ON THE PROCESSOR FRONT PANEL.
94
95

96 3.5 RESTARTING

97 THE PROGRAM CAN BE RESTARTED AT ADDRESS 200 OR 204. (SEE SECTION 3.3)
98
99

100 4.0 OPERATOR INTERFACE

101 4.1 PROGRAM ID

102 THE PROGRAM TYPES ITS NAME AND MAINDEC NUMBER THE FIRST TIME IT
103 IS STARTED AFTER BEING LOADED.
104
105
106
107
108
109
110
111
112
113
114

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  RM03
1  LOAD DEVICE
2  OFFLINE RM05
3  NOT PRESENT
4  NOT PRESENT
5  NOT AN RM05/3/2
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:

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

'DRIVE(S) TO BE TESTED, 0, 2'

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 QUE. 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 RM05/3/2 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 - RM03, 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 RM05/3/2 DISKLESS DIAGNOSTIC IS EXECUTABLE ON ANY PDP-11 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET.

5.2 DUAL PORT CONFIGURATIONS

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

5.3 MEMORY PARITY HARDWARE

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

286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342

5.4 MEMORY MANAGEMENT HARDWARE

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

5.5 ACT11, APT11 COMPATIBILITY

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

5.6 XXDP COMPATIBILITY

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

5.7 OPERATING SYSTEM COMPATIBILITY

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

6.0 TEST DESCRIPTION

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

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

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

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

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

IF
CS
DS
MASSBUS MODULE

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

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

PROCEDURE:

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

PROBABLE FAULT:

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

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

TEST 2 CTOD TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

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

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.

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

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

REGISTER SELECT LINES 1, 2, 4 AND 8 ARE TESTED IN THIS 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 RM05, RM03 OR RM02 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 'DPI', EXCEPT WHEN 'SEARCH TO DISABLE' IS ACTIVE.

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

PROCEDURE:

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 MODULF

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 'CS5 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 'CS5 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:

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

TO VERIFY THAT WORD CLOCK AND PROM STROBE CAN BE MANIPULATED 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.

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

USING BIT 10 OF THE MAINTENANCE REGISTER (RMMR1) TO SIMULATE 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:

TO 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.

AN RM02/3 USES CYLINDER 822., TRACK 4., SECTOR 31. AND 16 BIT FORMAT FOR THE TEST, WHICH CORRESPONSE TO THE FOLLOWING:

HEADER:

WORD 1 - 1101001100110110

WORD 2 - 0000010000011111

AN RM05 USES CYLINDER 822., TRACK 18., SECTOR 31. AND 16 BIT FORMAT FOR THIS TEST, WHICH CORRESPONSE TO THE FOLLOWING:

HEADER:

WORD 1 - 1101001100110110

WORD 2 - 0001001000011111

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 (RMMRT) TO SIMULATE THE DATA BITS FOR THE FIRST AND SECOND

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

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;
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
677
678

```

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

```

679

```

.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

```

680

```

.SBTTL BASIC DEFINITIONS

```

681
682

```

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK = 1100
104000 ERROR = EMT          ;;BASIC DEFINITION OF ERROR CALL
000004 SCOPE = IOT         ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS
000011 HT = 11             ;;CODE FOR HORIZONTAL TAB
000012 LF = 12             ;;CODE FOR LINE FEED
000015 CR = 15             ;;CODE FOR CARRIAGE RETURN
000200 CRLF = 200          ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776 PS = 177776        ;;PROCESSOR STATUS WORD
177776 PSW=PS
177774 STKLMT = 177774     ;;STACK LIMIT REGISTER
177772 PIRQ = 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR = 177570      ;;HARDWARE SWITCH REGISTER
177570 DDISP = 177570     ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS
000000 R0 = %0             ;;GENERAL REGISTER
000001 R1 = %1             ;;GENERAL REGISTER
000002 R2 = %2             ;;GENERAL REGISTER
000003 R3 = %3             ;;GENERAL REGISTER
000004 R4 = %4             ;;GENERAL REGISTER

```

```

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

```

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

```

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

```

                  ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000            BIT15     = 100000
040000            BIT14     = 40000
020000            BIT13     = 20000
010000            BIT12     = 10000
004000            BIT11     = 4000
002000            BIT10     = 2000
001000            BIT09     = 1000
000400            BIT08     = 400
000200            BIT07     = 200
000100            BIT06     = 100
000040            BIT05     = 40
000020            BIT04     = 20
  
```

```

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

.*BASIC 'CPU' TRAP VECTOR ADDRESSES

```

000004 ERRVEC = 4 ;:TIME OUT AND OTHER ERRORS
000010 RESVEC = 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC = 14 ;:T BIT
000014 TRTVEC = 14 ;:TRACE TRAP
000014 BPTVEC = 14 ;:BREAKPOINT TRAP (BPT)
000020 IOTVEC = 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC = 24 ;:POWER FAIL
000030 EMTVEC = 30 ;:EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC = 34 ;:'TRAP' TRAP
000060 TKVEC = 60 ;:TTY KEYBOARD VECTOR
000064 TPVEC = 64 ;:TTY PRINTER VECTOR
000240 PIRQVEC = 240 ;:PROGRAM INTERRUPT REQUEST VECTOR
    
```

.SBTTL RM REGISTER BIT DEFINITIONS

.*RMCS1 CONTROL STATUS REGISTER

```

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

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

```

000000 NOP = 000000 ;:NOP COMMAND
000002 ILF02 = 000002 ;:ILLEGAL COMMAND
000004 SEEK = 000004 ;:SEEK COMMAND
000006 RECAL = 000006 ;:RECALIBRATE COMMAND
000010 DRVCLR = 000010 ;:DRIVE CLEAR COMMAND
000012 RELEASE = 000012 ;:RELEASE COMMAND
000014 OFFSET = 000014 ;:OFFSET COMMAND
000016 RTC = 000016 ;:RETURN TO CENTERLINE COMMAND
000020 RIP = 000020 ;:READ IN PRESET COMMAND
000022 PAKACK = 000022 ;:PACK ACKNOWLEDGE COMMAND
000022 PACACK = PAKACK
000024 ILF24 = 000024 ;:ILLEGAL COMMAND
000026 ILF26 = 000026 ;:ILLEGAL COMMAND
000030 SEARCH = 000030 ;:SEARCH COMMAND
    
```

683
 684
 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

```

714      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
715      000050      WCD   = 000050      ;WRITE CHECK DATA COMMAND
716      000052      WCH   = 000052      ;WRITE CHECK HEADER AND DATA
717      000054      ILF54 = 000054      ;ILLEGAL COMMAND
718      000056      ILF56 = 000056      ;ILLEGAL COMMAND
719      000060      WD    = 000060      ;WRITE DATA COMMAND
720      000062      WH    = 000062      ;WRITE HEADER AND DATA COMMAND
721      000064      ILF64 = 000064      ;ILLEGAL COMMAND
722      000066      ILF66 = 000066      ;ILLEGAL COMMAND
723      000070      RD    = 000070      ;READ DATA COMMAND
724      000072      RH    = 000072      ;READ HEADER AND DATA COMMAND
725      000074      ILF74 = 000074      ;ILLEGAL COMMAND
726      000076      ILF76 = 000076      ;ILLEGAL COMMAND
727
728      ;*RMDA DISK ADDRESS REGISTER
729
730      ;TRACK ADDRESS DEFINITIONS
731      010000      TA16  = BIT12      ;TRACK ADDRESS 16.
732      004000      TA8   = BIT11      ;TRACK ADDRESS 8.
733      002000      TA4   = BIT10      ;TRACK ADDRESS 4
734      001000      TA2   = BIT09      ;TRACK ADDRESS 2
735      000400      TA1   = BIT08      ;TRACK ADDRESS 1
736
737      ;SECTOR ADDRESS DEFINITIONS
738      000020      SA16  = BIT04      ;SECTOR ADDRESS 16.
739      000010      SA8   = BIT03      ;SECTOR ADDRESS 8.
740      000004      SA4   = BIT02      ;SECTOR ADDRESS 4
741      000002      SA2   = BIT01      ;SECTOR ADDRESS 2
742      000001      SA1   = BIT00      ;SECTOR ADDRESS 1
743
744      ;TRACK & SECTOR MASKS
745      177400      TADMSK = 177400    ;TRACK ADDRESS MASK
746      000377      SADMSK = 000377    ;SECTOR ADDRESS MASK
747
748      ;*RMDS DRIVE STATUS REGISTER
749
750      100000      ATA   = BIT15      ;ATTENTION ACTIVE
751      040000      ERR   = BIT14      ;COMPOSITE ERROR
752      020000      PIP   = BIT13      ;POSITIONING IN PROGRESS
753      010000      MOL   = BIT12      ;MEDIUM ON LINE
754      004000      WRL   = BIT11      ;WRITE LOCK
755      002000      LBT   = BIT10      ;LAST BLOCK TRANSFERRED
756      001000      PGM   = BIT09      ;PROGRAMMABLE
757      000400      DPR   = BIT08      ;DRIVE PRESENT
758      000200      DRY   = BIT07      ;DRIVE READY
759      000100      VV    = BIT06      ;VOLUME VALID
760      000001      OM    = BIT00      ;OFFSET MODE ACTIVE
761
762      ;*RMER1 ERROR REGISTER #1
763

```

764	100000	DCK	= BIT15	;DATA CHECK ERROR
765	040000	UNS	= BIT14	;DRIVE UNSAFE
766	020000	OPI	= BIT13	;OPERATION INCOMPLETE
767	010000	DTE	= BIT12	;DRIVE TIMING ERROR
768	004000	WLE	= BIT11	;WRITE LOCK ERROR
769	002000	IAE	= BIT10	;INVALID ADDRESS ERROR
770	001000	AOE	= BIT09	;ADDRESS OVERFLOW ERROR
771	000400	HCRC	= BIT08	;HEADER CRC ERROR
772	000200	HCE	= BIT07	;HEADER COMPARE ERROR
773	000100	ECH	= BIT06	;ECC 'HARD' ERROR
774	000040	WCF	= BIT05	;WRITE CLOCK FAILURE
775	000020	FER	= BIT04	;FORMAT ERROR
776	000010	PAR	= BIT03	;PARITY ERROR
777	000004	RMR	= BIT02	;REGISTER MODIFICATION REFUSED
778	000002	ILR	= BIT01	;ILLEGAL REGISTER
779	000001	ILF	= BIT00	;ILLEGAL FUNCTION
780				
781	115760	NDTMSK	= DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER	
782				; 'NDTMSK' IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
783				; COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
784				
785				; *RMAS ATTENTION SUMMARY REGISTER
786				
787	000377	ATNMSK	= 377	; MASK FOR ATTENTION BITS
788				
789				; *RMLA LOOK AHEAD REGISTER
790				
791	002000	SC4	= BIT10	; SECTOR COUNT = 16
792	001000	SC3	= BIT09	; SECTOR COUNT = 8
793	000400	SC2	= BIT08	; SECTOR COUNT = 4
794	000200	SC1	= BIT07	; SECTOR COUNT = 2
795	000100	SC0	= BIT06	; SECTOR COUNT = 1
796				
797	003700	SCTMSK	= 003700	; SECTOR COUNT MASK
798				
799				; *RMR1 MAINTENANCE REGISTER #1
800				
801				; WRITE ONLY BITS
802	100000	DBCK	= BIT15	; DEBUG CLOCK
803	040000	DBEN	= BIT14	; DEBUG CLOCK ENABLE
804	020000	DEBL	= BIT13	; DIAGNOSTIC END OF BLOCK
805	010000	MSEN	= BIT12	; SEARCH TIMEOUT ENABLE
806	004000	MCLK	= BIT11	; MAINTENANCE CLOCK
807	002000	MRD	= BIT10	; READ DATA
808	001000	MUR	= BIT09	; UNIT READY
809	000400	MOC	= BIT08	; ON CYLINDER
810	000200	MSER	= BIT07	; SEEK ERROR
811	000100	MDF	= BIT06	; DRIVE FAULT
812	000040	MS	= BIT05	; SECTOR PULSE
813	000010	MWP	= BIT03	; WRITE PROTECT
814	000004	MI	= BIT02	; INDEX PULSE
815	000002	MSC	= BIT01	; SECTOR COMPARE
816	000001	DMD	= BIT00	; DIAGNOSTIC MODE
817				
818				; READ ONLY BITS
819	100000	OCC	= BIT15	; OCCUPIED
820	040000	RG	= BIT14	; RUN AND GO

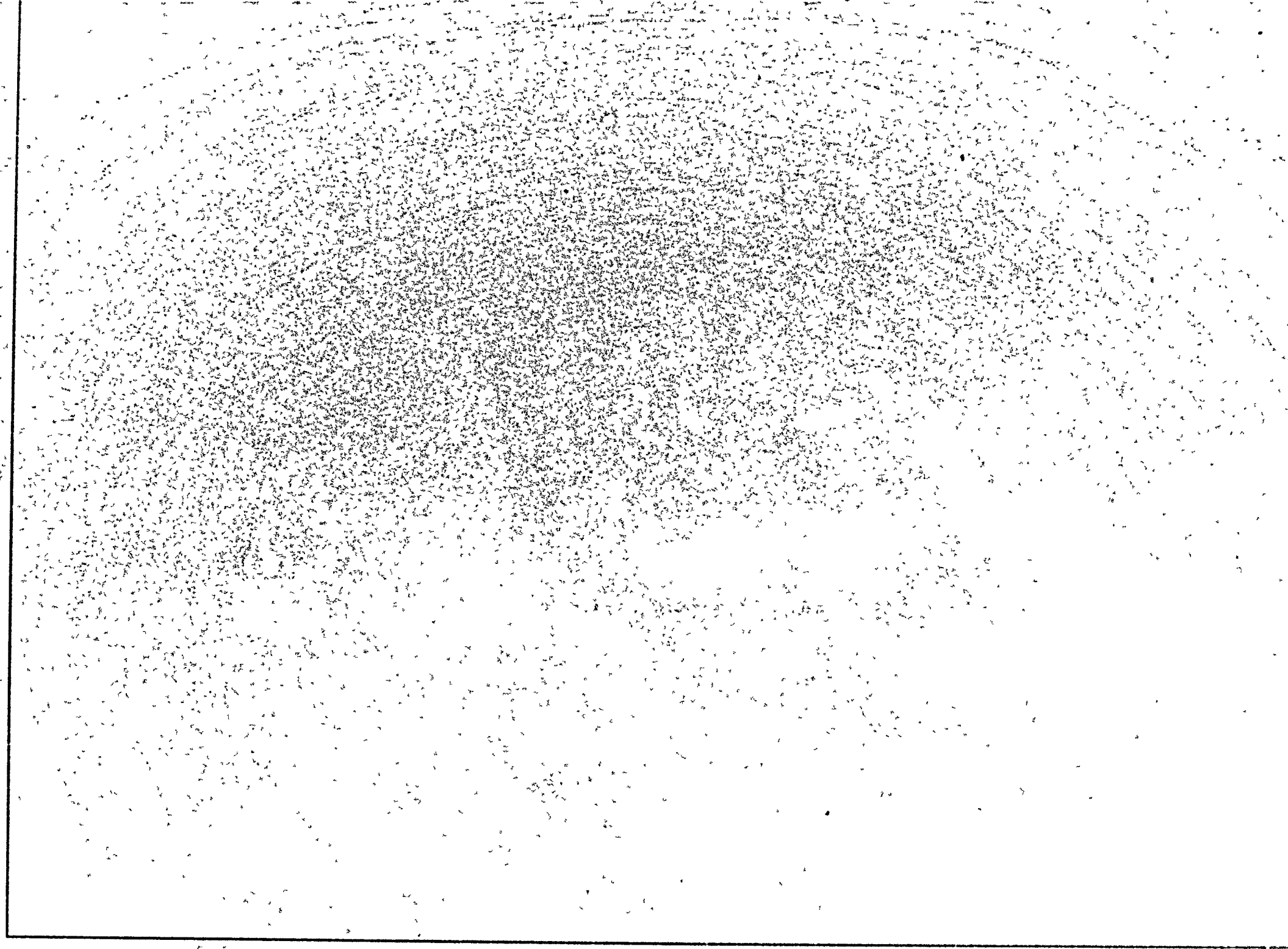
821	020000	EBL	= BIT13	:END OF BLOCK
822	010000	REX	= BIT12	:EXCEPTION
823	004000	ESRC	= BIT11	:ENABLE SEARCH
824	002000	PLFS	= BIT10	:LOOKING FOR SYNC
825	001000	ECRC	= BIT09	:ENABLE CRC OUT
826	000400	PDA	= BIT08	:DATA AREA
827	000200	PHA	= BIT07	:HEADER AREA
828	000100	CONT	= BIT06	:CONTINUE
829	000040	WC	= BIT05	:WORD CLOCK
830	000020	EECC	= BIT04	:ENABLE ECC OUT
831	000010	MWD	= BIT03	:WRITE DATA BIT
832	000004	LS	= BIT02	:LAST SECTOR
833	000002	LST	= BIT01	:LAST SECTOR AND TRACK
834	000001	DMD	= BIT00	:DIAGNOSTIC MODE
835	051401	MR1AAA	= DMD!MUR!DBEN!MOC!MSEN	
836				
837		:*RMDT	DRIVE TYPE REGISTER	
838				
839	100000	NSA	= BIT15	:NOT SECTOR ADDRESSED = 0
840	040000	TAP	= BIT14	:TAPE DRIVE = 0
841	020000	MOH	= BIT13	:MOVING HEAD = 1
842	004000	DRQ	= BIT11	:DRIVE REQUEST REQUIRED
843				
844	020024	SNGPRT	= 020024	:SINGLE PORT DRIVE TYPE
845	024024	DULPRT	= 024024	:DUAL PORT DRIVE TYPE
846				
847		:*RMOF	OFFSET REGISTER	
848				
849	010000	FMT16	= BIT12	:16 BIT WORD FORMAT
850	004000	ECI	= BIT11	:ECC INHIBIT
851	002000	HCI	= BIT10	:HEADER COMPARE INHIBIT
852	000200	OFD	= BIT07	:OFFSET FORWARD
853	161577	XNUOF	= 161577	:UNUSED BITS OF RMOF
854				
855		:*RMDC	DESIRED CYLINDER ADDRESS REGISTER	
856				
857	001777	CYLSK	= 001777	:MASK FOR CYLINDER ADDRESS
858	176000	XNUDC	= 176000	:UNUSED BITS OF RMDC
859				
860		:*RMR2	MAINTENANCE REGISTER #2	
861				
862		:READ ONLY BITS		
863	100000	RQA	= BIT15	:PORT A REQUEST
864	040000	ROB	= BIT14	:PORT B REQUEST
865	020000	TAG	= BIT13	:TAG CONTROL
866	010000	TST	= BIT12	:COMMAND SEQUENCE TEST BIT
867	004000	CC	= BIT11	:CONTROL OR CYLINDER TAG
868	002000	CH	= BIT10	:CONTROL OR HEAD TAG
871	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


```

872      000001      BB00      = BIT00      ;TAG BUS
873      ;*RMER2 ERROR REGISTER 2
874
875      100000      BSE      = BIT15      ;BAD SECTOR ERROR
876      040000      SKI      = BIT14      ;SEEK INCOMPLETE
877      020000      OPE      = BIT13      ;OPERATOR PLUG ERROR
878      010000      IVC      = BIT12      ;INVALID COMMAND ERROR
879      004000      LSC      = BIT11      ;LOSS OF SYSTEM CLOCK
880      002000      LBC      = BIT10      ;LOSS OF BIT CLOCK
881      000200      DVC      = BIT07      ;DEVICE CHECK
882      000010      DPF      = BIT03      ;DATA PARITY ERROR
883      001567      XNUER2   = 001567      ;UNSED BITS OF RMER2
884
885      .SBTTL PROGRAM MNEMONICS
886
887      100000      MSE      = BIT15      ;MANUFACTURING DETECTED SECTOR ERROR
888      040000      JSE      = BIT14      ;USER DETECTED SECTOR ERROR
889
890      .SBTTL RM REGISTER INDEX VALUES
891
892      000000      RMCS1    = 00      ;CONTROL STATUS REGISTER #1
893      000006      RMDA     = 06      ;DISK ADDRESS REGISTER
894      000012      RMDS     = 12      ;DRIVE STATUS REGISTER
895      000014      RMER1    = 14      ;ERROR REGISTER #1
896      000016      RMAS     = 16      ;ATTENTION SUMMARY REGISTER
897      000020      RMLA     = 20      ;LOOK AHEAD REGISTER
898      000024      RMMR1    = 24      ;MAINTENANCE REGISTER
899      000026      RMDT     = 26      ;DRIVE TYPE REGISTER
900      000030      RMSN     = 30      ;SERIAL NUMBER REGISTER
901      000032      RMOF     = 32      ;OFFSET REGISTER
902      000034      RMDC     = 34      ;DESIRED CYLINDER REGISTER
903      000036      RMHR     = 36      ;HOLDING REGISTER
904      000040      RMMR2    = 40      ;MAINTENANCE REGISTER #2
905      000042      RMER2    = 42      ;ERROR REGISTER #2
906      000044      RMEC1    = 44      ;ECC POSITION REGISTER
907      000046      RMEC2    = 46      ;ECC PATTERN REGISTER
910      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
911
912
913      000077      IDXMSK   = 77      ;MASK FOR REGISTER INDEX NUMBER
914
915      .SBTTL RH CONTROLLER REGISTER BIT DEFINITIONS
916
917      ;*RMCS1 CONTROL STATUS REGISTER #1
918

```

919	100000	SC	= BIT15	;SPECIAL CONDITION-READ ONLY
920	040000	TRE	= BIT14	;TRANSFER ERROR
921	020000	MCPE	= BIT13	;MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
922	002000	PSEL	= BIT10	;PORT B SELECT
923	001000	A17	= BIT09	;ADDRESS EXTENSION
924	000400	A16	= BIT08	;ADDRESS EXTENSION
925	000200	RDY	= BIT07	;READY-READ ONLY
926	000100	IE	= BIT06	;INTERRUPT ENABLE
927				
928		;*RMCS2 RH CONTROL STATUS REGISTER #2		
929				
930	100000	DLT	= BIT15	;DATA LATE-READ ONLY
931	040000	WCE	= BIT14	;WRITE CHECK ERROR-READ ONLY
932	020000	UPE	= BIT13	;UNIBUS PARITY ERROR
933	010000	NED	= BIT12	;NONEXISTANT DRIVE-READ ONLY
934	004000	NEM	= BIT11	;NONEXISTANT MEMORY-READ ONLY
935	002000	PGE	= BIT10	;PROGRAM ERROR-READ ONLY
936	001000	MXF	= BIT09	;MISSED TRANSFER
937	000400	MDPE	= BIT08	;MASSBUS DATA BUS PARITY ERROR-READ ONLY
938	000200	OR	= BIT07	;OUTPUT READY-READ ONLY
939	000100	IR	= BIT06	;INPUT READY-READ ONLY
940	000040	CLR	= BIT05	;CONTROLLER CLEAR
941	000020	PAT	= BIT04	;PARITY TEST
942	000010	BAI	= BIT03	;UNIBUS ADDRESS INCREMENT INHIBIT
945	000004	U2	= BIT02	;UNIT SELECT
	000002	U1	= BIT01	;UNIT SELECT
	000001	U0	= BIT00	;UNIT SELECT
946				
947		;UNIT SELECT MASK		
948				
949	000007	UNMSK	= 7	;UNIT SELECT MASK
950				
951		;*RMCS3 RH70 CONTROL STATUS REGISTER #3		
952				
953	100000	APE	= BIT15	;ADDRESS PARITY ERROR
954	040000	DPEHI	= BIT14	;DATA PARITY ERROR HIGH WORD
955	020000	DPELO	= BIT13	;DATA PARITY ERROR LOW WORD
956	010000	WCEHI	= BIT12	;WRITE CHECK ERROR HIGH WORD
957	004000	WCELO	= BIT11	;WRITE CHECK ERROR LOW WORD
958	002000	DBL	= BIT10	;DOUBLE WORD TRANSFER
959	000100	IE	= BIT06	;INTERRUPT ENABLE
960	000010	IPCK3	= BIT03	;INVERT PARITY CHECK
961	000004	IPCK2	= BIT02	;INVERT PARITY CHECK
962	000002	IPCK1	= BIT01	;INVERT PARITY CHECK
963	000001	IPCK0	= BIT00	;INVERT PARITY CHECK
964				
965		.SBTTL RH CONTROLLER REGISTER INDEX VALUES		
966				
967	000000	RMCS1	= 00	;CONTROL, STATUS REGISTER #1
968	000002	RMWC	= 02	;WORD COUNT REGISTER
969	000004	RMBA	= 04	;BUS ADDRESS REGISTER
970	000010	RMCS2	= 10	;CONTROL, STATUS REGISTER #2
971	000022	RMDB	= 22	;DATA BUFFER
972	000050	RMBAE	= 50	;BUS ADDRESS EXTENSION
973	000052	RMCS3	= 52	;CONTROL, STATUS REGISTER #3
974				
975	176700	ABASE	= 176700	;UNIBUS ADDRESS



1
2
3
4
5
6
7
8
9

```

.SBTTL TRAP CATCHER

      000000      .=0
      000174      : *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,HALT'
000174 000000      : *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
000176 000000      : *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      000174      .=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

.SBTTL ACT11 HOOKS

:*****
:HOOKS REQUIRED BY ACT11
000210      $SVPC=.      ;SAVE PC
000046 000046      .=46
000046 021530      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
000052 000052      .=52
000052 000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
000210 000210      .=$SVPC      ;; RESTORE PC

001100      .=1100
.SBTTL APT PARAMETER BLOCK

:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
001100      .SX=.      ;;SAVE CURRENT LOCATION
000024 000024      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024 000200      200      ;;FOR APT START UP
000044 000044      .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044 001100      $APTHDR ;;POINT TO APT HEADER BLOCK
001100 001100      .=.SX      ;;RESET LOCATION COUNTER

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

001100      $APTHD:
001100 000000      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102 001222      $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104 000602      $STSM: .WORD 2      ;;RUN TIM OF LONGEST TEST
001106 000002      $PASTM: .WORD 2      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110 000002      $UNITM: .WORD 2      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112 000042      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
001114      TAGADR=.

```

0

.SBTTL COMMON TAGS

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

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

.SBTTL APT MAILBOX-ETABLE

::*****

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

001230	000000	\$PASS:	.WORD	APASS	::PASS COUNT
001232	000000	\$DEVCT:	.WORD	ADEVCT	::DEVICE COUNT
001234	000000	\$UNIT:	.WORD	AUNIT	::I/O UNIT NUMBER
001236	000000	\$MSGAD:	.WORD	AMSGAD	::MESSAGE ADDRESS
001240	000000	\$MSGLG:	.WORD	AMSGLG	::MESSAGE LENGTH
001242		\$ETABLE:			::APT ENVIRONMENT TABLE
001242	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
001243	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
001244	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
001246	000000	\$USWR:	.WORD	AJSWR	::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	\$DEVW:	.WORD	ADEVW	::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	AUTSIZ: .WORD	0	:ALLOW AUTO DRIVE SIZING = 0, USE MANUALLY INPUT DRIVES = 1
001330	000000	CHGADR: .WORD	0	:CHANGE RH/RM BUS ADDRESS = -1, NO CHANGE = 0
001332	000000	XXDP: .WORD	0	:THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH :THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE :'XXDP' DEVICE CODE FOR THE RM05/3/2.
001334	000	LSTRK: .BYTE	0	:LO BYTE = 0
001335	000	.BYTE	0	:HI BYTE, CONTAINS LAST TRACK ADDRESS OF UNIT :UNDER TEST. RM02/3 = 4., RM05 = 18.

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

001336

GETBUF:

		:REGISTER INPUT BUFFER		
001336	000000	RMCS1I: .WORD	0	:CONTROL, STATUS REGISTER #1
001340	000000	RMWCI: .WORD	0	:WORD COUNT REGISTER
001342	000000	RMBAI: .WORD	0	:BUS ADDRESS REGISTER
001344	000000	RMDAI: .WORD	0	:DISK ADDRESS REGISTER
001346	000000	RMCS2I: .WORD	0	:CONTROL, STATUS REGISTER #2
001350	000000	RMDSI: .WORD	0	:DRIVE STATUS REGISTER
001352	000000	RMER1I: .WORD	0	:ERROR REGISTER #1
001354	000000	RMA5I: .WORD	0	:ATTENTION SUMMARY REGISTER
001356	000000	RMLAI: .WORD	0	:LOOK AHEAD REGISTER
001360	000000	RMDBI: .WORD	0	:DATA BUFFER
001362	000000	RMMR1I: .WORD	0	:MAINTENANCE REGISTER #1
001364	000000	RMDTI: .WORD	0	:DRIVE TYPE REGISTER
001366	000000	RMSNI: .WORD	0	:SERIAL NUMBER REGISTER
001370	000000	RMOFI: .WORD	0	:OFFSET REGISTER
001372	000000	RMDCI: .WORD	0	:DESIRED CYLINDER REGISTER
001374	000000	RMHRI: .WORD	0	:HOLDING REGISTER
001376	000000	RMMR2I: .WORD	0	:MAINTENANCE REGISTER #2
001400	000000	RMER2I: .WORD	0	:ERROR REGISTER #2
001402	000000	RMEC1I: .WORD	0	:ECC POSITION REGISTER
001404	000000	RMEC2I: .WORD	0	:ECC PATTERN REGISTER
001406	000000	RMBAEI: .WORD	0	:BUS ADDRESS EXTENSION REGISTER
001410	000000	RMCS3I: .WORD	0	:CONTROL, STATUS REGISTER #3

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

001412

PUTBUF:

		:REGISTER OUTPUT BUFFER		
001412	000000	RMCS1O: .WORD	0	:CONTROL, STATUS REGISTER #1
001414	000000	RMWCO: .WORD	0	:WORD COUNT REGISTER
001416	000000	RMBAO: .WORD	0	:BUS ADDRESS REGISTER
001420	000000	RMDAO: .WORD	0	:DISK ADDRESS REGISTER
001422	000000	RMCS2O: .WORD	0	:CONTROL, STATUS REGISTER #2
001424	000000	RMDSO: .WORD	0	:DRIVE STATUS REGISTER
001426	000000	RMER1O: .WORD	0	:ERROR REGISTER #1
001430	000000	RMA5O: .WORD	0	:ATTENTION SUMMARY REGISTER
001432	000000	RMLAO: .WORD	0	:LOOK AHEAD REGISTER
001434	000000	RMDBO: .WORD	0	:DATA BUFFER
001436	000000	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	RMMR20:	.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	\$LPCSR:	.WORD	172540	;KW11-P CONTROL + STATUS REGISTER
001514	172542	\$LPCSE:	.WORD	172542	;KW11-P COUNT SET BUFFER
001516	000104	\$LPVEC:	.WORD	104	;KW11-P INTERRUPT VECTOR
001520	000106		.WORD	106	
001522	177546	\$LLCSR:	.WORD	177546	;KW11-L CONTROL + STATUS REGISTER
001524	000100	\$LLVEC:	.WORD	100	;KW11-L INTERRUPT VECTOR
001526	000102		.WORD	102	
001530	000000	\$PSW:	.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 \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 .*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRP).
 .*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

\$ERRTB:

1
2
3

;ERROR 1 CANNOT CLEAR NED STATUS

001542 032460
 001544 040364
 001546 040500
 001550 040534

EMT1
 EHT1
 EDT1
 EFT1

4
5
6

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

001552 032466
 001554 040370
 001556 040502
 001560 040536

EMT2
 EHT2
 EDT2
 EFT2

7
8
9

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

001562 032514
 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 032534
 001574 000000
 001576 000000
 001600 000000

EMT4
 0
 0
 0

13
14
15

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

001602 032556
 001604 040374
 001606 040504
 001610 040540

EMT5
 EHT5
 EDT5
 EFT5

16
17

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

18

001612	032602	EMT6
001614	040374	EHT5
001616	040504	EDT5
001620	040540	EFT5

19

20

21

22

:ERROR 7 CANNOT WRITE/READ SHIFTING ONE BIT TO ALL BIT POSITIONS
 OF DEVICE REGISTERS

001622	032624	EMT7
001624	040400	EHT7
001626	040504	EDT5
001630	040540	EFT5

23

24

25

:ERROR 10 REGISTER SELECT 1 APPEARS S-A-0

001632	032646	EMT10
001634	000000	0
001636	000000	0
001640	000000	0

26

27

28

:ERROR 11 REGISTER SELECT 1 APPEARS S-A-1

001642	032664	EMT11
001644	000000	0
001646	000000	0
001650	000000	0

29

30

31

:ERROR 12 REGISTER SELECT 2 APPEARS S-A-0

001652	032702	EMT12
001654	000000	0
001656	000000	0
001660	000000	0

32

33

34

:ERROR 13 REGISTER SELECT 2 APPEARS S-A-1

001662	032720	EMT13
001664	000000	0
001666	000000	0
001670	000000	0

35

36

37

:ERROR 14 REGISTER SELECT 4 APPEARS S-A-0

001672	032736	EMT14
001674	000000	0
001676	000000	0
001700	000000	0

38

39

:ERROR 15 REGISTER SELECT 4 APPEARS S-A-1

40	001702 032754	EMT15	
	001704 000000	0	
	001706 000000	0	
	001710 000000	0	
41			
42		:ERROR 16	REGISTER SELECT 8 APPEARS S-A-0
43			
	001712 032772	EMT16	
	001714 000000	0	
	001716 000000	0	
	001720 000000	0	
44			
45		:ERROR 17	REGISTER SELECT 8 APPEARS S-A-1
46			
	001722 033010	EMT17	
	001724 000000	0	
	001726 000000	0	
	001730 000000	0	
47			
48		:ERROR 20	OPI SET WITH SFACH TIMEOUT DISABLED
49			
	001732 033026	EMT20	
	001734 040364	EHT1	
	001736 040500	EDT1	
	001740 040534	EFT1	
50			
51		:ERROR 21	OPI SET WITH SECTOR PULSE,SECTOR COMPARE WAS RESET
52			
	001742 033050	EMT21	
	001744 040364	EHT1	
	001746 040500	EDT1	
	001750 040534	EFT1	
53			
54		:ERROR 22	DTE SET WITH INDEX PULSE,SEARCH WAS NOT ENABLED
55			
	001752 033072	EMT22	
	001754 040364	EHT1	
	001756 040500	EDT1	
	001760 040534	EFT1	
56			
57		:ERROR 23	DTE SET WITH SFCTOR PULSE,SECTOR COMPARE WAS RESET
58			
	001762 033116	EMT23	
	001764 040364	EHT1	
	001766 040500	EDT1	
	001770 040534	EFT1	
59			
60		:ERROR 24	DTE DID NOT SET WITH SECTOR PULSE,SECTOR COMPARE WAS SET
61			

001772	033142	EMT24
001774	040364	EHT1
001776	040500	EDT1
002000	040534	EFT1
62		
63		:ERROR 25 DTE SET WITH SECTOR PULSE DURING FORMAT CHANGE
64		
002002	033166	EMT25
002004	040364	EHT1
002006	040500	EDT1
002010	040534	EFT1
65		
66		:ERROR 26 MBA CLR L IS STUCK ACTIVE
67		
002012	033206	EMT26
002014	000000	0
002016	000000	0
002020	000000	0
68		
69		:ERROR 27 COULD NOT SET DTE AFTER FORMAT CHANGE
70		
002022	033240	EMT27
002024	040364	EHT1
002026	040500	EDT1
002030	040534	EFT1
71		
72		:ERROR 30 CANNOT SET PROM STROBE WITH BIT CLOCK
73		
002032	033260	EMT30
002034	040364	EHT1
002036	040500	EDT1
002040	040534	EFT1
74		
75		:ERROR 31 CANNOT CLEAR RMER1,DTE
76		
002042	033300	EMT31
002044	040364	EHT1
002046	040500	EDT1
002050	040534	EFT1
77		
78		:ERROR 32 PROM STROBE RESET EARLY
79		
002052	033314	EMT32
002054	040364	EHT1
002056	040500	EDT1
002060	040534	EFT1
80		
81		:ERROR 33 PROM STROBE SET EARLY
82		
002062	033326	EMT33

	002064	040364		EHT1	
	002066	040500		EDT1	
	002070	040534		EFT1	
83					
84			:ERROR 34		LOOKING FOR SYNC SET EARLY
85					
	002072	033340		EMT34	
	002074	040364		EHT1	
	002076	040500		EDT1	
	002100	040534		EFT1	
86					
87			:ERROR 35		LOOKING FOR SYNC DID NOT SET
88					
	002102	033352		EMT35	
	002104	040364		EHT1	
	002106	040500		EDT1	
	002110	040534		EFT1	
89					
90			:ERROR 36		PROM STROBE SET WHILE LOOKING FOR SYNC
91					
	002112	033364		EMT36	
	002114	040364		EHT1	
	002116	040500		EDT1	
	002120	040534		EFT1	
92					
93			:ERROR 37		SYNC DETECTED WITH WRONG PATTERN
94					
	002122	033404		EMT37	
	002124	040440		EHT115	
	002126	040524		EDT115	
	002130	040552		EFT115	
95					
96			:ERROR 40		SYNC NOT DETECTED
97					
	002132	033434		EMT40	
	002134	040364		EHT1	
	002136	040500		EDT1	
	002140	040534		EFT1	
98					
99			:ERROR 41		DRIVE TIMING ERROR DID NOT CLEAR LOOKING FOR SYNC
100					
	002142	033456		EMT41	
	002144	040364		EHT1	
	002146	040500		EDT1	
	002150	040534		EFT1	
101					
102			:ERROR 42		WRITE GATE DID NOT COME ON OR RESET EARLY
103					
	002152	033504		EMT42	
	002154	040364		EHT1	

	002156	040500		EDT1	
	002160	040534		EFT1	
104					
105			:ERROR	43	INCORRECT SYNC PATTERN DURING HEADER
106					
	002162	033522		EMT43	
	002164	000000		0	
	002166	000000		0	
	002170	000000		0	
107					
108			:ERROR	44	INCORRECT SYNC PATTERN DURING HEADER
109					
	002172	033522		EMT43	
	002174	040400		EHT7	
	002176	040504		EDT5	
	002200	040540		EFT5	
110					
111			:ERROR	45	HEADER AREA DID NOT COME ON OR RESET EARLY
112					
	002202	033554		EMT45	
	002204	040364		EHT1	
	002206	040500		EDT1	
	002210	040534		EFT1	
113					
114			:ERROR	46	CRC ENABLE DID NOT SET
115					
	002212	033570		EMT46	
	002214	040364		EHT1	
	002216	040500		EDT1	
	002220	040534		EFT1	
116					
117			:ERROR	47	INCORRECT HEADER GENERATED DURING WRITE
118					
	002222	033604		EMT47	
	002224	040404		EHT47	
	002226	040506		EDT47	
	002230	040534		EFT1	
119					
120			:ERROR	50	READ GATE INCORRECT DURING HEADER AREA
121					
	002232	033622		EMT50	
	002234	040364		EHT1	
	002236	040500		EDT1	
	002240	040534		EFT1	
122					
123			:ERROR	51	UNEXPECTED HEADER ERROR DURING DIAGNOSTIC MODE
124					
	002242	033640		EMT51	
	002244	040364		EHT1	
	002246	040500		EDT1	

002250	040534	EFT1	
125			
126		:ERROR 52	INCORRECT HEADER READ IN DIAGNOSTIC MODE
127			
002252	033654	EMT52	
002254	040410	EHT52	
002256	040510	EDT52	
002260	040542	EFT57	
128			
129		:ERROR 53	INCORRECT TAG BUS DURING DATA COMMAND
130			
002262	040242	EMT276	
002264	040364	EHT1	
002266	040500	EDT1	
002270	040534	EFT1	
131			
132		:ERROR 54	DATA TIMING SEQUENCER CONTROLS INCORRECT DURING DATA COMMAND
133			
002272	033706	EMT54	
002274	040364	EHT1	
002276	040500	EDT1	
002300	040534	EFT1	
134			
135		:ERROR 55	DATA AREA DID NOT COME ON OR RESET EARLY
136			
002302	033724	EMT55	
002304	040364	EHT1	
002306	040500	EDT1	
002310	040534	EFT1	
137			
138		:ERROR 56	ECC ENABLE DID NOT SET
139			
002312	033742	EMT56	
002314	040364	EHT1	
002316	040500	EDT1	
002320	040534	EFT1	
140			
141		:ERROR 57	DEVICE IS NOT AN RM05/3/2
142			
002322	033756	EMT57	
002324	040414	EHT57	
002326	040512	EDT57	
002330	040542	EFT57	
143			
144		:ERROR 60	DEVICE AVAILABLE IS NOT SET
145			
002332	033772	EMT60	
002334	040364	EHT1	
002336	040500	EDT1	
002340	040534	EFT1	

147		:ERROR 61	INCORRECT ECC PATTERN GENERATED DURING WRITE
148	002342 034006	EMT61	
	002344 040420	EHT61	
	002345 040514	EDT61	
	002350 040542	EFT57	
149		:ERROR 62	CANNOT CLEAR PROM STROBE WITH BIT CLOCK
150		EMT62	
151	002352 034024	EHT1	
	002354 040364	EDT1	
	002356 040500	EFT1	
	002360 040534		
152		:ERROR 63	DATA AREA DID NOT RESET
153		EMT63	
154	002362 034042	EHT1	
	002364 040364	EDT1	
	002366 040500	EFT1	
	002370 040534		
155		:ERROR 64	UNEXPECTED ECC ERROR IN DIAGNOSTIC MODE
156		EMT64	
157	002372 034054	EHT1	
	002374 040364	EDT1	
	002376 040500	EFT1	
	002400 040534		
158		:ERROR 65	INCORRECT DATA TRANSFERRED TO MEMORY
159		EMT65	
160	002402 034070	EHT1	
	002404 040364	EDT1	
	002406 040500	EFT1	
	002410 040534		
161		:ERROR 66	
162		EMT66	
163	002412 034102	0	
	002414 000000	0	
	002416 000000	0	
	002420 000000	0	
164		:ERROR 67	
165		EMT67	
166	002422 034116	0	
	002424 000000	0	
	002426 000000	0	
	002430 000000	0	

167
168 :ERROR 70
169
002432 034134 EMT70
002434 000000 0
002436 000000 0
002440 000000 0

170
171 :ERROR 71
172
002442 034154 EMT71
002444 000000 0
002446 000000 0
002450 000000 0

173
174 :ERROR 72
175
002452 034200 EMT72
002454 000000 0
002456 000000 0
002460 000000 0

176
177 :ERROR 73
178
002462 034224 EMT73
002464 000000 0
002466 000000 0
002470 000000 0

179
180 :ERROR 74
181
002472 034244 EMT74
002474 000000 0
002476 000000 0
002500 000000 0

182
183 :ERROR 75
184
002502 034254 EMT75
002504 000000 0
002506 000000 0
002510 000000 0

185
186 :ERROR 76
187
002512 034266 EMT76
002514 000000 0
002516 000000 0
002520 000000 0

188

189		:ERROR 77	
190	002522 034302	EMT77	
	002524 000000	0	
	002526 000000	0	
	002530 000000	0	
191			
192		:ERROR 100	CANT SET VOLUME VALID
193	002532 036110	EMT170	
	002534 040464	EMT150	
	002536 040524	EDT115	
	002540 040552	EFT115	
194			
195		:ERROR 101	RUN AND GO WONT SET
196	002542 040222	EMT275	
	002544 040364	EMT1	
	002546 040500	EDT1	
	002550 040534	EFT1	
197			
198		:ERROR 102	SEARCH ENABLE WONT SET
199	002552 034356	EMT102	
	002554 040364	EMT1	
	002556 040500	EDT1	
	002560 040534	EFT1	
200			
201		:ERROR 103	OCCUPIED IS INCORRECT FOR FUNCTION CODE
202	002562 036154	EMT173	
	002564 040464	EMT150	
	002566 040524	EDT115	
	002570 040552	EFT115	
203			
204		:ERROR 104	READ IN PRESET DIDNT CLEAR RMDA, RMDC OR RMOF
205	002572 036176	EMT174	
	002574 000000	0	
	002576 000000	0	
	002600 000000	0	
206			
207		:ERROR 105	READ IN PRESET DIDNT CLEAR RMOF
208	002602 036224	EMT175	
	002604 040364	EMT1	
	002606 040500	EDT1	
	002610 040534	EFT1	
209			
210		:ERROR 106	READ IN PRESET DIDNT CLEAR RMDA

211
002612 036242 EMT176
002614 040364 EMT1
002616 040500 EDT1
002620 040534 EFT1

212
213 :ERROR 107 READ IN PRESET DIDNT CLEAR RMDC
214
002622 040346 EMT302
002624 040364 EMT1
002626 040500 EDT1
002630 040534 EFT1

215
216 :ERROR 110 CANT SET OFFSET MODE BY OFFSET COMMAND
217
002632 036260 EMT200
002634 040364 EMT1
002636 040500 EDT1
002640 040534 EFT1

218
219 :PUT ERROR TABLE HERE

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
002642 011600
002644 005740
002646 022626
002650 104401 002656
002654 000417
002714
002714 010046
002716 104402
002720 000240
002722 012737 177777 001330
002730 000402
002732 005037 001330
002736 000240
002740 005227 000000
002744 001375
002746 000005
002750 012706 001114
002754 005026
002756 022706 001154
002762 001374
002764 012706 001100
002770 012737 025222 000020
002776 012737 000340 000022
003004 012737 025752 000030
003012 012737 000340 000032
003020 012737 031006 000034
003026 012737 000340 000036
003034 012737 031114 000024
003042 012737 000340 000026
003050 013737 021366 021360
003056 005037 001206
003062 005037 001210
003066 112737 000001 001131
003074 012737 003074 001122
003102 012737 003102 001124
003110 013746 000004
003114 012737 003150 000004
003122 012737 177570 001154
003130 012737 177570 001156
003136 022777 177777 176010
003144 001012
003146 000403

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

.SBTTL START OF PROGRAM
START1: MOV #-1,CHGADR ;CHANGE RH/RM BUS ADDRESS
BR START2
START: CLR CHGADR ;NO CHANGE IN ADDRESS
START2: NOP
INC #0 ;TTY LOOP, WAIT FOR INCREMENT
BNE -4 ;OF WORD
RESET ;RESET THE WORLD

.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@IOTVEC+2 ;;LEVEL 7
MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2;LEVEL 7
MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@PWRVEC+2 ;;LEVEL 7
MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOV #1,$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 0127'6 003156      64$:  MOV    #65$, (SP)      ;;SET UP FOR TRAP RETURN
003154 000002
003156 012737 000176 001154 65$:  MOV    #SWREG,SWR      ;;POINT TO SOFTWARE SWR
003164 012737 000174 001156 66$:  MOV    #DISPREG,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 67$:  MOV    #$$SWREG,SWR   ;;NO,USE APT SWITCH REGISTER
003220

24 003220 012737 002642 000004 67$:  ;SETUP 'TIMEOUT' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 003220 012737 002642 000004 MOV    #BADTMO,ERRVEC  ;;SETUP FOR UNEXPECTED TIMEOUT
26 003226 012737 000300 000006 MOV    #PR6,ERRVEC+2  ;;LEVEL 6
27
28 .SBTTL TYPE PROGRAM NAME
   ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
003234 005227 177777      INC    #-1           ;;FIRST TIME?
003240 001034      BNE    68$          ;;BRANCH IF NO
003242 022737 021530 000042 CMP    #SENDAD,@#42  ;;ACT-11?
003250 001430      BEQ    68$          ;;BRANCH IF YES
003252 104401 003260      TYPE   ,69$         ;;TYPE ASCIZ STRING
003256 000425      BR     68$         ;;GET OVER THE ASCIZ
   ;;69$: .ASCIZ <CRLF>@CZRMQBO - RM05/3/2 DISKLESS TEST, PT 2@<CRLF>
   68$:
   .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
003332 005737 000042      TST    @#42         ;;ARE WE RUNNING UNDER XXDP/ACT?
003336 001012      BNE    70$         ;;BRANCH IF YES
003340 123727 001242 000001 CMPB   $ENV,#1       ;;ARE WE RUNNING UNDER APT?
003346 001406      BEQ    70$         ;;BRANCH IF YES
003350 023727 001154 000176 CMP    SWR,#SWREG   ;;SOFTWARE SWITCH REG SELECTED?
003356 001005      BNE    71$         ;;BRANCH IF NO
003360 104407      GTSWR           ;;GET SOFT-SWR SETTINGS
003362 000403      BR     71$
003364 112737 000091 001150 70$:  MOVB   #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
003372 71$:

29
30 ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
31 ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
32
33 003372 005037 001332      CLR    XXDP         ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
34 003376 122737 000016 000041 CMPB   #16,@#41     ;;LOADED FROM AN RM05/3/2 ?
35 003404 001160      BNE    5$          ;;BRANCH IF NOT
36 003406 013737 000040 001332 MOV    @#40,XXDP    ;;GET DEVICE INDICATOR AND NUMBER
37 003414 122737 000007 001332 CMPB   #7,XXDP     ;;IS IT A VALID NUMBER ?
38 003422 103002      BUIS   1$         ;;YES
39 003424 105037 001332      CLRB  XXDP        ;;NO, DEFAULT TO DRIVE 0
40 003430 005737 000042 1$:  TST    @#42         ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
41 003434 001425      BEQ    3$         ;;BR IF NEITHER
42 003436 104401 003444      TYPE   ,73$       ;;TYPE ASCIZ STRING
003442 000412      BR     72$       ;;GET OVER THE ASCIZ
   ;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
   72$:
43 003470 005046      CLR    -(SP)       ;;CLEAR WORD ON STACK
44 003472 113716 001332 MOVB   XXDP,(SP)    ;;GET DRIVE ADDRESS
45 003476 104403      TYPOS           ;;TYPE THE ADDRESS
46 003500 001      .BYTE 1         ;;ONLY 1 CHARACTER

```

```

47 003501      000      .BYTE 0      ;SUPRESS LEADING ZEROS
48 003502 104401 001217  TYPE $CRLF ;CR-LF
49 003506 000517      BR 5$      ;GET NUMBER OF DRIVES
50
51 003510 005227 177777 3$: INC #=1 ;FIRST TIME THRU HERE ?
52 003514 001114      BNE 5$      ;NO
53 003516 104401 003524  TYPE ,75$ ;:TYPE ASCIZ STRING
   003522 000410      BR 74$      ;:GET OVER THE ASCIZ
   ;:75$: .ASCIZ <CRLF>/TO TEST DRIVE /
   74$:
54 003544 005046      CLR -(SP) ;CLEAR WORD ON STACK
55 003546 113716 001332  MOVB XXDP,(SP) ;GET DRIVE ADDRESS
56 003552 104403      TYPOS ;TYPE DRIVE ADDRESS
57 003554      001      .BYTE 1 ;ONLY 1 CHARACTER
58 003555      000      .BYTE 0 ;SUPRESS LEADING ZEROS
59 003556 104401 003564  TYPE ,77$ ;:TYPE ASCIZ STRING
   003562 000431      BR 76$      ;:GET OVER THE ASCIZ
   ;:77$: .ASCIZ /, HALT PROGRAM, REMOVE RRD PCK AND REPLACE IT/<CRLF>
   76$:
60 003646 104401 003654  TYPE ,78$ ;:TYPE ASCIZ STRING
   003652 000435      BR 5$      ;:GET OVER THE ASCIZ
   ;:78$: .ASCIZ /WITH A WORK PCK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
   5$:
64 ;CHECK FOR AUTO MODE OR STANDALONE
65 003746 005037 001326  CLR AUTSIZ ;LET AUTO DRIVE SIZING OCCUR
66 003752 105737 001150  TSTB $AUTOB ;RUNNING IN AUTO MODE ?
67 003756 001561      BEQ STANDALONE ;BR IF NO
68 003760 012737 000377 001300  MOV #377,$DEV ;SET DEVICE MAP FOR ALL DRIVES
69
70 ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
71 003766      XSIZ:
72 003766 132737 000200 001243  BITB #BIT7,$ENVM ;SIZING ALLOWED ?
73 003774 001146      BNE 12$ ;NO
74
75 003776 005001      CLR R1 ;START FROM DRIVE 0
76 004000 013700 001276  MOV $BASE,R0 ;LOAD THE BASE ADDRESS
77 004004 104401 032166  TYPE ,SYSTAT ;TYPE 'UNIT STATUS:'
78
79 004010 136137 032450 001300 1$: BITB ATNTBL(R1),$DEV ;IS DEVICE PRESENT IN MAP ?
80 004016 001531      BEQ 11$ ;BR IF NO
81 004020 104401 001217  TYPE ,CRLF ;CR-LF
82 004024 010146      MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
   004026 104403      TYPOS ;GO TYPE--OCTAL ASCII
   004030      002      .BYTE 2 ;:TYPE 2 DIGIT(S)
   004031      000      .BYTE 0 ;:SUPPRESS LEADING ZEROS
83 004032 104401 032342  TYPE ,BLNKS4 ;TYPE 4 BLANKS
84
85 004036 012760 000040 000010  MOV #CLR, RMCS2(R0) ;CLEAR MASS BUS
86 004044 010160 000010  MOV R1, RMCS2(R0) ;LOAD THE DRIVE ADDRESS
87 004050 005760 000012  TST RMD5(R0) ;ACCESS DRIVE REGISTER
88 004054 032760 010000 000010  BIT #NED, RMCS2(R0) ;IS DRIVE PRESENT ?
89 004062 001051      BNE 3$ ;BR IF NO
90 004064 032760 004000 000000  BIT #DVA, RMCS1(R0) ;IS DRIVE AVAILABLE ?
91 004072 001450      BEQ 4$ ;BR IF NO
92 004074 012737 032204 004300  MOV #SRM02,10$ ;ASSUME RM02 DEVICE
93 004102 016002 000026  MOV RMDT(R0),R2 ;SAVE DRIVE TYPE REGISTER IN R2
94 004106 022702 020025  CMP #20025,R2 ;SINGLE PORT RM02 ?
    
```

95	004112	001430			BEQ	2\$;BR IF YES
96	004114	022702	024025		CMP	#24025,R2			;DUAL PORT RM02 ?
97	004120	001425			BEQ	2\$;BR IF YES
98	004122	012737	032211	004300	MOV	#\$RM03,10\$;ASSUME RM03 DEVICE
99	004130	022702	020024		CMP	#20024,R2			;SINGLE PORT RM03 ?
100	004134	001417			BEQ	2\$;BR IF YES
101	004136	022702	024024		CMP	#24024,R2			;DUAL PORT RM03 ?
102	004142	001414			BEQ	2\$;BR IF YES
103	004144	012737	032216	004300	MOV	#\$RM05,10\$;ASSUME RM05 DEVICE
104	004152	022702	020027		CMP	#20027,R2			;SINGLE PORT RM05 ?
105	004156	001406			BEQ	2\$;BR IF YES
106	004160	022702	024027		CMP	#24027,R2			;DUAL PORT RM05 ?
107	004164	001403			BEQ	2\$;BR IF YES
108	004166	104401	032223		TYPE	,NOTRM			;DRIVE NOT AN RM05/3/2
109	004172	000443			BR	11\$;CHECK NEXT DRIVE
110	004174	032760	010000	000012	2\$: BIT	,\$MOL,RMDS(R0)			;IS MEDIUM ON LINE ?
111	004202	001415			BEQ	6\$;BR IF NO
112	004204	000417			BR	7\$			
113									
114	004206	104401	032261		3\$: TYPE	,NOTPRS			;DRIVE NOT PRESENT
115	004212	000402			BR	5\$;CHECK NEXT DRIVE
116									
117	004214	104401	032276		4\$: TYPE	,NOTAVL			;DRIVE NOT AVAILABLE
118	004220	005737	001326		5\$: TST	AUTSIZ			;AUTO SIZING ON ?
119	004224	001026			BNE	11\$;BR IF NO
120	004226	146137	032450	001300	BICB	ATNTBL(R1),\$DEV			;CLEAR DEVICE FROM BIT MAP
121	004234	000422			BR	11\$;CHECK NEXT DRIVE
122									
123	004236	104401	032315		6\$: TYPE	,UNTOFF			;DRIVE OFFLINE
124	004242	000413			BR	9\$;PRINT DRIVE TYPE
125									
126	004244	005737	001332		7\$: TST	XXDP			;LOADED FROM RM80 ?
127	004250	001406			BEQ	8\$;NO
128	004252	123701	001332		CMPB	XXDP,R1			;IS THIS THE DRIVE ?
129	004256	001360			BNE	5\$;BR IF NO
130	004260	104401	032244		TYPE	,LODEV			;DRIVE IS LOAD DEVICE
131	004264	000755			BR	5\$			
132									
133	004266	104401	032326		8\$: TYPE	,UNTON			;DRIVE ONLINE
134	004272	104401	032344		9\$: TYPE	,BLNKS2			;TYPE 2 BLANKS
135	004276	104401			TYPE				;PRINT DRIVE TYPE
136	004300	000000			10\$: .WORD	0			;MESSAGE ADDRESS HERE
137									
138	004302	005201			11\$: INC	R1			;INCREMENT THE DRIVE ADDRESS
139	004304	020127	000007		CMP	R1,#7			;ALL DRIVES ARE CHECKED ?
140	004310	003637			BLE	1\$;BRANCH IF NOT
141									
142	004312	104401	001217		12\$: TYPE	,\$CRLF			;CR-LF
143	004316	000137	005004		JMP	,\$MNSTART			;JUMP TO COMMON START

```

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

```

```

.SBTTL STANDALONE INPUT ROUTINES

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 ,CNSL01 ;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 ,CNSL02 ;TYPE WARNING MESSAGE
BR 3$ ;TRY AGAIN
4$: MOV $TMP1,$BASE ;STORE NEW BUS ADDRESS

5$: TYPE ,CNSL03
CLR -(SP)
MOV $VECT1,(SP) ;GET CURRENT VECTOR ADDRESS
TYPOC
TYPE ,BLNKS2 ;TYPE 2 BLANKS
RDOCT ;GET NEW VECTOR ADDRESS
MOV (SP)+,$TMP1 ;CARRIAGE RETURN?
BEQ 7$ ;YES-SKIP TO NEXT ENTRY
CMP #1000,$TMP1 ;VECTOR ADDRESS < 1000 ?
BHI 6$ ;YES!!
TYPE ,CNSL04 ;TYPE WARNING MESSAGE
BR 5$ ;RETRY
6$: MOV $TMP1,$VECT1 ;STORE NEW VECTOR ADDRESS

```



```

57
58
59 004546 005227 177777
60 004552 001002
61 004554 104401 031724
62 004560 104401 001217
63 004564 005037 001300
64 004570 104401 032144
65 004574 104411
66 004576 012637 001176
67 004602 023727 001176 000101
68 004610 001007
69 004612 104401 031542
70 004616 012737 000377 001300
71 004624 000137 003766
72
73 004630 023727 001176 000015 10$:
74 004636 001436
75 004640 104401 001176
76 004644 023727 001176 000060
77 004652 002430
78 004654 023727 001176 000067
79 004662 003427
80 004664 000423
81
82 004666 104411
83 004670 012637 001176
84 004674 023727 001176 000015
85 004702 001432
86 004704 104401 031553
87 004710 104401 001176
88 004714 023727 001176 000060
89 004722 002404
90 004724 023727 001176 000067
91 004732 003403
92 004734 104401 032066 12$:
93 004740 000711
94
95 004742 013701 001176 13$:
96 004746 042701 177770
97 004752 156137 032450 001300
98 004760 122737 000377 001300
99 004766 101337
100 004770 005237 001326 14$:
101 004774 104401 001217
102 005000 000137 003766

```

:DIALOGUE TO INPUT DEVICE NUMBERS

```

7$: INC #=1 ;FIRST TIME THRU ?
    BNE R8 ;BR IF NO
    TYPE ,CNSLO7 ;TYPE INPUT INSTRUCTIONS
8$: TYPE ,%CRLF ;CR-LF
9$: CLR $DEV ;CLEAR DEVICE MAP
    TYPE ,MSDRVS ;TYPE 'DRIVE(S) : '
    RDCHR
    MOV (SP)+,$TMP1 ;GET RESPONSE
    CMP $TMP1,#'A ;IS INPUT 'A' ?
    BNE 10$ ;NO
    TYPE ,ALL ;YES, TYPE 'ALL' AND GO
    MOV #377,$DEV ;SET DEVICE MAP FOR ALL DRIVES
    JMP XSIZ ;AUTO SIZE.
10$: CMP $TMP1,#CR ;CARRIAGE RETURN ?
    BEQ 12$ ;YES
    TYPE , $TMP1 ;ECHO RESPONSE
    CMP $TMP1,#'0 ;NUMBER < 0 ?
    BLT 12$ ;YES
    CMP $TMP1,#'7 ;NUMBER > 7 ?
    BLE 13$ ;NO
    BR 12$ ;ILLEGAL INPUT
11$: RDCHR
    MOV (SP)+,$TMP1 ;GET RESPONSE
    CMP $TMP1,#CR ;CARRIAGE RETURN ?
    BEQ 14$ ;YES
    TYPE ,COMMA ;TYPE ' , '
    TYPE , $TMP1 ;ECHO RESPONSE
    CMP $TMP1,#'0 ;NUMBER < 0 ?
    BLT 12$ ;YES
    CMP $TMP1,#'7 ;NUMBER > 7 ?
    BLE 13$ ;NO
    TYPE ,CNSLO8 ;TYPE '' ?ILLEGAL INPUT''
    BR 9$ ;RETRY
12$:
13$: MOV $TMP1,R1 ;R1 = DRIVE NUMBER
    BIC #'C7,R1
    BISB ATNTBL(R1),$DEV ;SET DEVICE IN MAP
    CMPB #377,$DEV ;DONE ?
    BHI 11$ ;NO
14$: INC AUTSIZ ;DO NOT AUTO SIZE WHEN TYPING DRIVE STATUS
    TYPE ,%CRLF ;CR-LF
    JMP XSIZ ;GO SIZE DEVICES

```

```

1
2 005004
3 005004 104401 032110
4 005010 013700 001300
5 005014 001004
6 005016 104401 031553
7 005022 104401 032137
8 005026 012701 001470
9 005032 010137 001466
10 005036 012702 000001
11 005042 005003
12 005044 030200
13 005046 001413
14 005050 104401 031553
15 005054 010311
16 005056 010346
    005060 104403
    005062 001
    005063 000
17 005064 116361 032450 000001
18 005072 062701 000002
19 005076 006302
20 005100 105702
21 005102 001402
22 005104 005203
23 005106 000756
24 005110 005011
25 005112 104401 001217
26
27
28 005116 004737 021550
29 005122 000425
30 005124 104401 005132
    005130 000413
    005160
31 005160 005737 000042
32 005164 001002
33 005166 000137 002732
34 005172 000137 021520
35 005176 000413
36
37 005200 000240
38 005202 105737 001300
39 005206 001007
40 005210 005737 000042
41 005214 001002
42 005216 000137 002732
43 005222 000137 021520
44 005226
45
46 005226 105037 001116
47 005232 005037 001206
48 005236 004737 027422
49 005242 012746 000000
    005246 012746 005254
    005252 000002

```

```

:ASSEMBLE TEST QUE FROM DEVICE MAP
MNSTART:
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
MOV# 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$: CLR (R1) ;TERMINATE TEST QUE
TYPE ,$CRLF ;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$: .ASCIIZ <CRLF>/NO 'L' OR 'P' CLOCK/
64$:
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 $STNM ;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

```

```

005254
50 005254 117737 174206 001234 64$:      MOV      @TSTQUE,$UNIT ;LOAD DRIVE NUMBER
51
52      ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND DETERMINE THE LAST TRACK
53      ;OF THE DIFFERENT DRIVE TYPES
54
55 005262 012737 002000 001334      MOV      #TA4,LSTRK ;ASSUME LAST TRACK FOR RM02/3 = 4.
56 005270 013700 001276      MOV      $BASE,R0 ;RO = UNIBUS ADDRESS
57 005274 012760 000040 000010      MOV      #CLR,RMCS2(R0) ;CLEAR MASSBUS
58 005302 117760 174160 000010      MOV      @TSTQUE,RMCS2(R0) ;SELECT DEVICE UNDER TEST
59 005310 016002 000026      MOV      RMDI(R0),R2 ;GET RMDI AND
60 005314 042702 177770      BIC      #177770,R2 ;SAVE DRIVE TYPE BITS
61 005320 022702 000007      CMP      #7,R2 ;IS IT AN RM05 ?
62 005324 001003      BNE      3$ ;NO, MUST BE AN RM02 OR RM03
63 005326 012737 011000 001334      MOV      #TA16!TA2,LSTRK ;YES--SET LAST TRACK = 18.
64
65      ;TYPE DRIVE NUMBER TO BE TESTED($UNIT)
66
67 005334 104401 001217 3$:      TYPE      ,SCLF ;CR-LF
68 005340 104401 032160      TYPE      ,MSGDRV ;TYPE 'DRIVE'
69 005344 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
005350 104403      TYPOS
005352 002      .BYTE 2
005353 000      .BYTE 0
70 005354 005004      CLR      R4
71 005356 005304      DEC      R4
72 005360 001376      BNE      -2
73 005362 005304      DEC      R4
74 005364 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 RM CONTROLLER
:R1 = ADDRESS OF ENTRY IN TEST QUE CORRESPONDING TO THE
UNIT UNDER TEST
:R2,R3 = WORKING REGISTERS FOR TEST IN PROGRESS, MUST BE
SAVED BY SUBROUTINES
:R4,R5 = GENERAL WORKING REGISTERS, ARE NOT SAVED BY
SUBROUTINES
:R6 = STACK POINTER
:R7 = LINKAGE REGISTER TO SUBROUTINES

:STORAGE ASSIGNMENTS

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

005366
005366 000004
005370 000240
005372 012706 001100
005376 013700 001276
005402 013701 001466
005406 012737 000001 001226
2
3 005414 012702 000000
4
5
6 005420
7 005420 004737 022404
8 005424 016037 000010 001142
9 005432 032737 010000 001142
10 005440 001417
11 005442 111137 001140
12 005446 042737 177770 001140
13 005454 052737 000100 001140
14 005462 010037 001136
15 005466 062737 000010 001136
16 005474 104001
17 005476 000475
18
19
20
21 005500
22 005500 010003
23 005502 060203
24 005504 011304
25 005506 032760 010000 000010
26 005514 001470
27
28 005516 004737 022404
29 005522 016037 000010 001142
30 005530 032737 010000 001142
31 005536 001417
32 005540 111137 001140
33 005544 042737 177770 001140
34 005552 052737 000100 001140
35 005560 010037 001136
36 005564 062737 000010 001136
37 005572 104001
38 005574 000436
39
40
41
42 005576
43 005576 012713 000000
44 005602 032760 010000 000010
45 005610 001432
46
47

```

TST1:
SCOPE                                ;SCOPE CALL
NOP
MOV #STACK,SP                        ;LOAD THE STACK POINTER
MOV $BASE,R0                          ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1                         ;R1 = POINTER TO DEVICE
MOV #1,$TESTN                         ;;SET TEST NUMBER IN APT MAIL BOX

MOV #0,R2                              ;R2 = REGISTER INDEX.

;CLEAR THE MASSBUS AND VERIFY THAT NONEXISTANT DEVICE ERROR IS RESET
10$:
JSR PC,CNTCLR                          ;GO CLEAR CONTROLLER
MOV RMCS2(R0),$BDDAT                    ;STORE RMCS2 AT $BDDAT
BIT #NED,$BDDAT
BEQ 20$
MOVB (R1),$GDDAT
BIC #^CUNTMASK,$GDDAT
BIS #IR,$GDDAT
MOV R0,$BDADR
ADD #RMCS2,$BDADR
EMT 1
BR 60$

;READ THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE READ
;DOES NOT SET 'NED' ERROR
20$:
MOV R0,R3                              ;R3 = REGISTER ADDRESS
ADD R2,R3
MOV (R3),R4                             ;READ REGISTER
BIT #NED,RMCS2(R0)                       ;IS 'NED' SET??
BEQ 70$                                  ;NO!!

JSR PC,CNTCLR                          ;GO CLEAR CONTROLLER
MOV RMCS2(R0),$BDDAT                    ;STORE RMCS2 AT $BDDAT
BIT #NED,$BDDAT
BEQ 30$
MOVB (R1),$GDDAT
BIC #^CUNTMASK,$GDDAT
BIS #IR,$GDDAT
MOV R0,$BDADR
ADD #RMCS2,$BDADR
EMT 1
BR 60$

;WRITE THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE WRITE
;DOES NOT SET 'NED' ERROR
30$:
MOV #0,(R3)                             ;WRITE REGISTER
BIT #NED,RMCS2(R0)                       ;IS 'NED' SET??
BEQ 70$                                  ;NO!!

;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$:
51 ADD #2,R2 ;ADVANCE TO NEXT REGISTER
52 CMP #RMWC,R2 ;IS THIS RMWC??
53 BEQ 40$ ;YES - TRY NEXT REGISTER
54 CMP #RMBA,R2 ;IS THIS RMBA??
55 BEQ 40$ ;YES - TRY NEXT REGISTER
56 CMP #RMCS2,R2 ;IS THIS RMCS2??
57 BEQ 40$ ;YES - TRY ANOTHER REGISTER
58 CMP #RMAS,R2 ;IS THIS RMAS ??
59 BEQ 40$ ;YES - TRY ANOTHER REGISTER
60 CMP #RMDB,R2 ;IS THIS RMDB??
61 BEQ 40$ ;YES - TRY ANOTHER REGISTER
62 CMP #RMEC2,R2 ;IS THIS A LEGAL REGISTER
63 BHIS 10$ ;YES - TRY THIS REGISTER
64
65 ;GOT 'NONEXISTENT DEVICE' ERROR FOR EVERY REMOTE REGISTER ADDRESS
66 50$:
67 MOV $BASE,$BDADR ;STORE BASE ADDRESS
68 EMT 2
69 60$: JMP $EOSP ;GO SELECT NEXT DEVICE
70
71 70$:
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
```

*TEST 2 CTOD TEST

```
TEST2:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
74 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
75
76 ;WRITE ONES IN REMOTE REGISTERS
77 MOV #ILF76,RMCS1(R0) ;LOAD RMCS1
78 MOV #-1,RMDA(R0) ;LOAD RMDA
79 MOV #CYLMSK,RMDC(R0) ;LOAD RMDC
80 MOV #^CXAUOF,RMOF(R0) ;LOAD RMOF
81
82 ;READ REMOTE REGISTERS TWICE
83 MOV #1,R2
84 10$:
85 MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
86 MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
87 MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
88 MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
89 DEC R2
90 BPL 10$
91
92 ;SEE IF ANY ONE BITS CAME BACK
93 BIC #^CILF76,RMCS1I ;IS RMCS1 0??
```

```

94 006026 001014      BNE      20$          ;NO!!
95 006030 005737 001344  TST      RMDA1       ;IS RMDA 0??
96 006034 001011      BNE      20$          ;NO!!
97 006036 042737 176000 001372  BIC      #XNUDC,RMDC1 ;IS RMDC 0??
98 006044 001005      BNE      20$          ;NO!!
99 006046 042737 161577 001370  BIC      #XNUOF,RMOF1 ;IS RMOF 0 ??
100 006054 001001     BNE      20$          ;NO!!
101
102                ;CANNOT READ/WRITE A ONE FROM ANY REMOTE REGISTER
103 006056 104003     EMT      3
104 006060          20$:
105
106                ;*****
                ;*TEST 3          MASSBUS INITIALIZE TEST
                ;*****
006060          TST3:
006060 000004          SCOPE          ;SCOPE CALL
006062 000240          NOP
006064 012706 001100  MOV      #STACK,SP    ;LOAD THE STACK POINTER
006070 013700 001276  MOV      $BASE,R0     ;R0 = UNIBUS ADDRESS
006074 013701 001466  MOV      TSTQUE,R1    ;R1 = POINTER TO DEVICE
006100 012737 000003 001226  MOV      #3,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
107
108 006106 004737 022404  JSR      PC,CNTCLR    ;GO CLEAR CONTROLLER
109
110                ;WRITE ONES IN SELECTED REGISTERS
111 006112 012760 000076 000000  MOV      #ILF76,RMCS1(R0) ;LOAD RMCS1
112 006120 012760 177777 000014  MOV      #-1,RMER1(R0) ;LOAD RMER1
113 006126 012760 177777 000042  MOV      #-1,RMER2(R0) ;LOAD RMER2
114
115                ;INITIALIZE MASSBUS WITH A CLEAR
116 006134 004737 022404  JSR      PC,CNTCLR    ;GO CLEAR CONTROLLER
117
118                ;READ THE REGISTERS THAT WERE WRITTEN
119 006140 016037 000000 001336  MOV      RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
120 006146 016037 000014 001352  MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
121 006154 016037 000042 001400  MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
122
123                ;SEE IF ANY REGISTER BITS WERE CLEARED
124 006162 052737 177701 001336  BIS      #^C1LF76,RMCS1I ;SET ANY BIT NOT WRITTEN
125 006170 052737 001567 001400  BIS      #XNUER2,RMER2I
126 006176 022737 177777 001336  CMP      #-1,RMCS1I    ;ANY ZEROS IN RMCS1??
127 006204 001011      BNE      10$          ;YES!!
128 006206 022737 177777 001352  CMP      #-1,RMER1I    ;ANY ZEROS IN RMER1??
129 006214 001005      BNE      10$          ;YES!!
130 006216 022737 177777 001400  CMP      #-1,RMER2I    ;ANY ZEROS IN RMER2??
131 006224 001001     BNE      10$
132
133                ;NONE OF THE BITS WERE CLEARED
134 006226 104004     EMT      4
135 006230          10$:
136
137                ;*****
                ;*TEST 4          CLEAR STUCK ACTIVE TEST
                ;*****

```

```

006230
006230 000004
006232 000240
006234 012706 001100
006240 013700 001276
006244 013701 001466
006250 012737 000004 001226
*38
139 006256 004737 022404
140
141
142 006262 012760 177777 000014
143 006270 012760 177777 000042
144 006276 012760 000001 000024
145
146
147 006304 016037 000014 001352
148 006312 016037 000042 001400
149 006320 016037 000024 001362
150 006326 042737 040000 001352
151 006334 001011
152 006336 042737 040200 001400
153 006344 001005
154 006346 032737 000001 001362
155 006354 001001
156 006356 104026
157 006360
*58
159

```

```

TST4:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #4,$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

```

*TEST 5 TRISTATE TRANSFER TEST

```

006360
006360 000004
006362 000240
006364 012706 001100
006370 013700 001276
006374 013701 001466
006400 012737 000005 001226
160
161 006406 005002
162 006410 004737 022404
163
164
165 006414 012760 000076 000000
166 006422 012760 177777 000006
167 006430 012760 177777 000014
168 006436 012760 177777 000032
169 006444 012760 177777 000042
170
171
172 006452 012760 000000 000000
173 006460 012760 000000 000006
174 006466 012760 000000 000014
175 006474 012760 000000 000032
176 006502 012760 000000 000034
177 006510 012760 000000 000042

```

```

TST5:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #5,$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,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
MOV #0,RMDC(R0) ;LOAD RMDC
MOV #0,RMER2(R0) ;LOAD RMER2

```



```

178
179
180 006516 016037 000000 001336 ;READ BACK ALL REGISTERS
181 006524 016037 000006 001344 MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
182 006532 016037 000014 001352 MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
183 006540 016037 000032 001370 MOV RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
184 006546 016037 000034 001372 MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
185 006554 016037 000042 001400 MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
186
187
188 006562 012702 177777 001336 ;CHECK EACH REGISTER CONTENT FOR ZERO BITS WRITTEN & READ
189 006566 052737 177701 001336 MOV #-1,R2 ;ACCUMULATE ZEROS IN R2
190 006574 052737 161577 001370 BIS #^C1LF76,RMCS1I ;SET ALL BITS NOT WRITTEN
191 006602 052737 176000 001372 BIS #XNUOF,RMOFI
192 006610 052737 001567 001400 BIS #XNUDC,RMDCI
193 006616 005137 001336 COM RMCS1I ;COMPLEMENT REGISTER CONTENTS
194 006622 005137 001344 COM RMDAI
195 006626 005137 001352 COM RMER1I
196 006632 005137 001370 COM RMOFI
197 006636 005137 001372 COM RMDCI
198 006642 005137 001400 COM RMER2I
199 006646 043702 001336 BIC RMCS1I,R2 ;ACCUMULATE ALL ZERO BITS
200 006652 043702 001344 BIC RMDAI,R2
201 006656 043702 001352 BIC RMER1I,R2
202 006662 043702 001370 BIC RMOFI,R2
203 006666 043702 001372 BIC RMDCI,R2
204 006672 043702 001400 BIC RMER2I,R2
205 006676 001407 BEQ 10$ ;BRANCH IF EACH BIT IS ZERO
206
207
208 006700 010237 001142 ;ONE OR MORE BIT POSITIONS ARE NOT ZERO
209 006704 005037 001140 MOV R2,$BDDAT ;SAVE RESULT FOR TYPE
210 006710 104005 CLR $GDDAT ;LOAD EXPECTED RESULT
211 006712 052702 000001 EMT 5
212 BIS #BIT0,R2 ;SET ERROR FLAG
213 006716 004737 022404 10$: JSR PC,CNTCLR ;GO CLEAR CONTROLLER
214
215
216 ;PRESET SELECTED REGISTERS TO ZEROS
217 006722 012760 000600 000006 ;(ASSUME RMCS1, RMER1, RMER2 WERE CLEARED BY INIT)
218 006730 012760 000000 000032 MOV #0,RMDA(R0) ;LOAD RMDA
219 006736 012760 000000 000034 MOV #0,RMOF(R0) ;LOAD RMOF
220 MOV #0,RMDC(R0) ;LOAD RMDC
221
222 ;WRITE ONES IN SELECTED REGISTERS
223 006744 012760 000076 000000 MOV #1LF76,RMCS1(R0) ;LOAD RMCS1
224 006752 012760 177777 000006 MOV #-1,RMDA(R0) ;LOAD RMDA
225 006760 012760 016200 000032 MOV #^CXNUOF,RMOF(R0) ;LOAD RMOF
226 006766 012760 001777 000034 MOV #^CXNUDC,RMDC(R0) ;LOAD RMDC
227 006774 012760 177777 000014 MOV #-1,RMER1(R0) ;LOAD RMER1
228 007002 012760 176210 000042 MOV #^CXNUER2,RMER2(R0) ;LOAD RMER2
229
230 ;READ ALL REGISTERS
231 007010 016037 000000 001336 MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
232 007016 016037 000006 001344 MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
233 007024 016037 000032 001370 MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
234 007032 016037 000034 001372 MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER

```

```

234 007040 016037 000014 001352      MOV      RMER1(RO),RMER1I      ;STORE RMER1 IN INPUT BUFFER
235 007046 016037 000042 001400      MOV      RMER2(RO),RMER2I      ;STORE RMER2 IN INPUT BUFFER
236
237
238 007054 042737 177701 001336      ;CHECK EACH REGISTER CONTENT FOR ONE BITS WRITTEN & READ
      BIC      #^CILF76,RMCS1I      ;CLEAR ALL BITS NOT WRITTEN
239 007062 042737 161577 001370      BIC      #XNUOF,RMOFI
240 007070 042737 176000 001372      BIC      #XNUDC,RMDCI
241 007076 042737 001567 001400      BIC      #XNUER2,RMER2I
242 007104 005002                CLR      R2                      ;ACCUMULATE ONES IN R2
243 007106 053702 001336      BIS      RMCS1I,R2              ;ACCUMULATE ALL ONE BITS
244 007112 053702 001344      BIS      RMDAI,R2
245 007116 053702 001370      BIS      RMOFI,R2
246 007122 053702 001372      BIS      RMDCI,R2
247 007126 053702 001352      BIS      RMER1I,R2
248 007132 053702 001400      BIS      RMER2I,R2
249 007136 022702 177777      CMP      #-1,R2                ;SEE IF EACH BIT POSITION WAS ONE
250 007142 001410                BEQ      20$                    ;BRANCH IF NONE STUCK
251
252
253 007144 010237 001142                ;ONE OR MORE BIT POSITIONS ARE NOT ONE
      MOV      R2,$BDDAT          ;SAVE RESULT FOR TYPE
254 007150 012737 177777 001140      MOV      #-1,$GDDAT          ;EXPECTED RESULT
255 007156 104006                EMT      6
256 007160 052702 000002                BIS      #BIT1,R2            ;SET ERROR FLAG
257 007164                20$:
258 007164 005702                IST      R2                    ;ANY ERRORS DETECTED ??
259 007166 001126                BNE      30$                    ;YES - DONT DO BIT TEST
260 007170 012702 000001                MOV      #1,R2                ;R2=BIT POSITION
261
262 007174                25$:
      JSR      PC,CNTCLR          ;GO CLEAR CONTROLLER
263 007174 004737 022404
264
265 007200 010260 000006                ;WRITE THE BIT PATTERN IN SELECTED DEVICE REGISTERS
      MOV      R2,RMDA(RO)        ;LOAD RMDA
266 007204 010260 000032                MOV      R2,RMOF(RO)         ;LOAD RMOF
267 007210 010260 000034                MOV      R2,RMDC(RO)         ;LOAD RMDC
268 007214 010260 000014                MOV      R2,RMER1(RO)        ;LOAD RMER1
269 007220 010260 000042                MOV      R2,RMER2(RO)        ;LOAD RMER2
270
271
272 007224 016037 000006 001344      ;READ BACK THE REGISTERS
      MOV      RMDA(RO),RMDAI    ;STORE RMDA IN INPUT BUFFER
273 007232 016037 000032 001370      MOV      RMOF(RO),RMOFI      ;STORE RMOF IN INPUT BUFFER
274 007240 016037 000034 001372      MOV      RMDC(RO),RMDCI      ;STORE RMDC IN INPUT BUFFER
275 007246 016037 000014 001352      MOV      RMER1(RO),RMER1I    ;STORE RMER1 IN INPUT BUFFER
276 007254 016037 000042 001400      MOV      RMER2(RO),RMER2I    ;STORE RMER2 IN INPUT BUFFER
277
278
279 007262 005003                ;CHECK REGISTER CONTENTS FOR CORRECT PATTERN
      CLR      R3                      ;R3=ACCUMULATED ONE BIT
280 007264 012704 177777                MOV      #-1,R4                ;R4=ACCUMULATED ZERO BITS
281 007270 013705 001344                MOV      RMDAI,R5              ;GET ANY GOOD BITS FROM RMDA
282 007274 050503                BIS      R5,R3
283 007276 005105                COM      R5
284 007300 040504                BIC      R5,R4
285 007302 013705 001370                MOV      RMOFI,R5              ;GET GOOD BITS FROM RMOF
286 007306 042705 161577                BIC      #XNUOF,R5
287 007312 050503                BIS      R5,R3
288 007314 005105                COM      R5
289 007316 042705 161577                BIC      #XNUOF,R5

```

```

290 007322 040504          BIC      R5,R4
291 007324 013705 001372  MOV      RMDC1,R5          ;GET GOOD BITS FROM RMDC
292 007330 042705 176000  BIC      #XNUDC,R5
293 007334 050503          BIS      R5,R3
294 007336 005105          COM      R5
295 007340 042705 176000  BIC      #XNUDC,R5
296 007344 040504          BIC      R5,R4
297 007346 013705 001352  MOV      RMER11,R5        ;GET GOOD BITS FROM RMER1
298 007352 050503          BIS      R5,R3
299 007354 005105          COM      R5
300 007356 040504          BIC      R5,R4
301 007360 013705 001400  MOV      RMER21,R5        ;GET GOOD BITS FROM RMER2
302 007364 042705 001567  BIC      #XNUER2,R5
303 007370 050503          BIS      R5,R3
304 007372 005105          COM      R5
305 007374 042705 001567  BIC      #XNUER2,R5
306 007400 040504          BIC      R5,R4
307 007402 010205          MOV      R2,R5          ;RESET ALL ONES IN R3 EXCEPT
308 007404 005105          COM      R5          ;FOR THE TEST BIT
309 007406 040503          BIC      R5,R3
310 007410 040204          BIC      R2,R4          ;RESET TEST BIT IN R4
311 007412 050403          BIS      R4,R3          ;COMBINE ACCUMULATED 1'S + 0'S
312 007414 020302          CMP      R3,R2          ;IS PATTERN OK??
313 007416 001406          BEQ     26$            ;YES!!
314 007420 010237 001140  MOV      R2,$GDDAT       ;SAVE TEST PATTERN
315 007424 010337 001142  MOV      R3,$BDDAT       ;SAVE RESULT
316 007430 104007          EMT
317 007432 000404          BR      30$            ;SKIP TO NEXT
318
319          ;ADVANCE R2 TO THE NEXT PATTERN AND REPEAT TEST
320 007434          26$:
321 007434 006302          ASL     R2              ;SHIFT THE BIT
322 007436 001402          BEQ     30$            ;EXIT IF DONE
323 007440 000137 007174  JMP     25$
324 007444          30$:
325
326

```

 ;*TEST 6 REGISTER SELECT TEST

```

007444          TST6:
007444 000004          SCOPE          ;SCOPE CALL
007446 000240          NOP
007450 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
007454 013700 001276  MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
007460 013701 001466  MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
007464 012737 000006  MOV      #6,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

```

327
 328 ;THE FOLLOWING TABLE GIVES MASSBUS REGISTER SELECT VALUES FOR
 329 ;EACH DEVICE REGISTER

REGISTER NAME	REG SFL (16,8,4,2,1)
RMCS1	00000
RMDS	00001
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 : RMR2 01011
346 : RMMR2 01100
347 : RMR2 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, RMR1 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 RMR1, AND RMR1
;WILL NOT BE 0 WHEN READ BACK.

```

```

360 007472 005002 CLR R2 ;R2= ZEROS SOURCE
361 007474 012703 177777 MOV #1,R3 ;R3= ONES SOURCE

```

;TEST REG SEL 1 FOR S-A-0

```

365 007500 004737 022404 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
366 007504 010260 000014 MOV R2,RMR1(RO) ;LOAD RMR1
367 007510 010260 000034 MOV R2,RMDC(RO) ;LOAD RMDC
368 007514 010360 000024 MOV R3,RMMR1(RO) ;LOAD RMMR1
369 007520 010360 000036 MOV R3,RMR2(RO) ;LOAD RMR2
370 007524 016037 000014 001352 MOV RMR1(RO),RMR1I ;STORE RMR1 IN INPUT BUFFER
371 007532 016037 000034 001372 MOV RMDC(RO),RMDCI ;STORE RMDC IN INPUT BUFFER
372 007540 020337 001352 CMP R3,RMR1I
373 007544 001007 BNE 10$
374 007546 052737 176000 001372 BIS #XNDC,RMDCI
375 007554 020337 001372 CMP R3,RMDCI
376 007560 001001 BNE 10$
377 007562 104010 EMT 10

```

;TEST REG SEL 1 FOR S-A-1

```

380 007564 10$
381 007564 004737 022404 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
382 007570 010260 000006 MOV R2,RMDA(RO) ;LOAD RMDA
383 007574 010260 000032 MOV R2,RMOF(RO) ;LOAD RMOF
384 007600 010260 000042 MOV R2,RMR2(RO) ;LOAD RMR2
385 007604 010360 000016 MOV R3,RMAS(RO) ;LOAD RMAS
386 007610 010360 000030 MOV R3,RMSN(RO) ;LOAD RMSN
387 007614 010360 000040 MOV R3,RMMR2(RO) ;LOAD RMMR2
388 007620 016037 000006 001344 MOV RMDA(RO),RMDAI ;STORE RMDA IN INPUT BUFFER
389 007626 016037 000032 001370 MOV RMOF(RO),RMOFI ;STORE RMOF IN INPUT BUFFER
390 007634 016037 000042 001400 MOV RMR2(RO),RMR2I ;STORE RMR2 IN INPUT BUFFER
391 007642 020337 001344 CMP R3,RMDAI
392 007646 001015 BNE 20$
393 007650 052737 161577 001370 BIS #XNUOF,RMOFI

```

```

394 007656 020337 001370      CMP      R3,RMOF I
395 007662 001007      BNE      20$
396 007664 052737 001567 001400  BIS      #XNUER2,RMER2I
397 007672 020337 001400      CMP      R3,RMER2I
398 007676 001001      BNE      20$
399 007700 104011      EMT      11
400
401      ;TEST REG SEL 2 FOR S-A-0
402 007702 20$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
403 007702 004737 022404      MOV      R2,RMDA(RO)    ;LOAD RMDA
404 007706 010260 000006      MOV      R2,RMOF(RO)    ;LOAD RMOF
405 007712 010260 000032      MOV      R2,RMER2(RO)   ;LOAD RMER2
406 007716 010260 000042      MOV      R3,RMLA(RO)    ;LOAD RMLA
407 007722 010360 000020      MOV      R3,RMHR(RO)    ;LOAD RMHR
408 007726 010360 000036      MOV      R3,RMEC2(RO)   ;LOAD RMEC2
409 007732 010360 000046      MOV      RMDA(RO),RMDAI ;STORE RMDA IN INPUT BUFFER
410 007736 016037 000006 001344  MOV      RMOF(RO),RMOFI ;STORE RMOF IN INPUT BUFFER
411 007744 016037 000032 001370  MOV      RMER2(RO),RMER2I ;STORE RMER2 IN INPUT BUFFER
412 007752 016037 000042 001400  CMP      R3,RMDAI
413 007760 020337 001344      BNE      30$
414 007764 001015      BIS      #XNUOF,RMOF I
415 007766 052737 161577 001370  CMP      R3,RMOF I
416 007774 020337 001370      BNE      30$
417 010000 001007      BIS      #XNUER2,RMER2I
418 010002 052737 001567 001400  CMP      R3,RMER2I
419 010010 020337 001400      BNE      30$
420 010014 001001      EMT      12
421 010016 104012
422
423      ;TEST REG SEL 2 FOR S-A-1
424 010020 30$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
425 010020 004737 022404      MOV      R2,RMER1(RO)   ;LOAD RMER1
426 010024 010260 000014      MOV      R2,RMDC(RO)    ;LOAD RMDC
427 010030 010260 000034      MOV      #ILF76,RMCS1(RO) ;LOAD RMCS1
428 010034 012760 000076 000000  MOV      R3,RMSN(RO)    ;LOAD RMSN
429 010042 010360 000030      MOV      RMER1(RO),RMER1I ;STORE RMER1 IN INPUT BUFFER
430 010046 016037 000014 001352  MOV      RMDC(RO),RMDCI ;STORE RMDC IN INPUT BUFFER
431 010054 016037 000034 001372  BIS      #^CILF76,RMER1I
432 010062 052737 177701 001352  CMP      R3,RMER1I
433 010070 020337 001352      BNE      40$
434 010074 001007      BIS      #XNUDC,RMDC I
435 010076 052737 176000 001372  CMP      R3,RMDC I
436 010104 020337 001372      BNE      40$
437 010110 001001      EMT      13
438 010112 104013
439
440      ;TEST REG SEL 4 FOR S-A-0
441 010114 40$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
442 010114 004737 022404      MOV      R2,RMER1(RO)   ;LOAD RMER1
443 010120 010260 000014      MOV      R2,RMOF(RO)    ;LOAD RMOF
444 010124 010260 000032      MOV      R2,RMDC(RO)    ;LOAD RMDC
445 010130 010260 000034      MOV      R3,RMDT(RO)    ;LOAD RMDT
446 010134 010360 000026      MOV      R3,RMER2(RO)   ;LOAD RMER2
447 010140 010360 000042      MOV      R3,RMEC1(RO)   ;LOAD RMEC1
448 010144 010360 000044      MOV      RMER1(RO),RMER1I ;STORE RMER1 IN INPUT BUFFER
449 010150 016037 000014 001352  MOV      RMOF(RO),RMOFI ;STORE RMOF IN INPUT BUFFER
450 010156 016037 000032 001370

```

```

451 010164 016037 000034 001372      MOV      RMDC(RO),RMDCI ;STORE RMDC IN INPUT BUFFER
452 010172 020337 001352      CMP      R3,RMER1I
453 010176 001015          BNE     50$
454 010200 052737 161577 001370      BIS     #XNUOF,RMOFI
455 010206 020337 001370      CMP      R3,RMOFI
456 010212 001007          BNE     50$
457 010214 052737 176000 001372      BIS     #XNUDC,RMDCI
458 010222 020337 001372      CMP      R3,RMDCI
459 010226 001001          BNE     50$
460 010230 104014          EMT     14
461
462
463 010232          :TEST REG SEL 4 FOR S-A-1
464 010232 004737 022404      50$:    JSR     PC,CNTCLR ;GO CLEAR CONTROLLER
465 010236 010260 000006      MOV     R2,RMDA(RO) ;LOAD RMDA
466 010242 010260 000042      MOV     R2,RMER2(RO) ;LOAD RMER2
467 010246 010360 000012      MOV     R3,RMDS(RO) ;LOAD RMDS
468 010252 010360 000032      MOV     R3,RMOF(RO) ;LOAD RMOF
469 010256 016037 000006 001344      MOV     RMDA(RO),RMDAI ;STORE RMDA IN INPUT BUFFER
470 010264 016037 000042 001400      MOV     RMER2(RO),RMER2I ;STORE RMER2 IN INPUT BUFFER
471 010272 020337 001344      CMP     R3,RMDAI
472 010276 001007          BNE     60$
473 010300 052737 001567 001400      BIS     #XNUER2,RMER2I
474 010306 020337 001400      CMP     R3,RMER2I
475 010312 001001          BNE     60$
476 010314 104015          EMT     15
477
478
479 010316          :TEST REG SEL 8 FOR S-A-0
480 010316 004737 022404      60$:    JSR     PC,CNTCLR ;GO CLEAR CONTROLLER
481 010322 010260 000014      MOV     R2,RMER1(RO) ;LOAD RMER1
482 010326 010260 000006      MOV     R2,RMDA(RO) ;LOAD RMDA
483 010332 010360 000034      MOV     R3,RMDC(RO) ;LOAD RMDC
484 010336 010360 000042      MOV     R3,RMER2(RO) ;LOAD RMER2
485 010342 016037 000014 001352      MOV     RMER1(RO),RMER1I ;STORE RMER1 IN INPUT BUFFER
486 010350 016037 000006 001344      MOV     RMDA(RO),RMDAI ;STORE RMDA IN INPUT BUFFER
487 010356 020337 001352      CMP     R3,RMER1I
488 010362 001004          BNE     70$
489 010364 020337 001344      CMP     R3,RMDAI
490 010370 001001          BNE     70$
491 010372 104016          EMT     16
492
493
494 010374          :TEST REG SEL 8 FOR S-A-1
495 010374 004737 022404      70$:    JSR     PC,CNTCLR ;GO CLEAR CONTROLLER
496 010400 010260 000032      MOV     R2,RMOF(RO) ;LOAD RMOF
497 010404 010260 000034      MOV     R2,RMDC(RO) ;LOAD RMDC
498 010410 010260 000042      MOV     R2,RMER2(RO) ;LOAD RMER2
499 010414 010360 000012      MOV     R3,RMDS(RO) ;LOAD RMDS
500 010420 010360 000014      MOV     R3,RMER1(RO) ;LOAD RMER1
501 010424 010360 000006      MOV     R3,RMDA(RO) ;LOAD RMDA
502 010430 016037 000032 001370      MOV     RMOF(RO),RMOFI ;STORE RMOF IN INPUT BUFFER
503 010436 016037 000034 001372      MOV     RMDC(RO),RMDCI ;STORE RMDC IN INPUT BUFFER
504 010444 016037 000042 001400      MOV     RMER2(RO),RMER2I ;STORE RMER2 IN INPUT BUFFER
505 010452 052737 161577 001370      BIS     #XNUOF,RMOFI
506 010460 001015          BNE     80$
507 010462 022737 176000 001372      CMP     #XNUDC,RMDCI
    
```

508 010470 020337 001372
 509 010474 001007
 510 010476 052737 001567 001400
 511 010504 020337 001400
 512 010510 001001
 513 010512 104017
 514 010514
 515
 516
 517
 518

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

80\$:

;REGISTER SELECT 16 IS TESTED BY THE ILR TEST

 ;*TEST 7 DRIVE TYPE TEST

010514
 010514 000004
 010516 000240
 010520 012706 001100
 010524 013700 001276
 010530 013701 001466
 010534 012737 000007 001226
 519
 520 010542 016002 000026
 521 010546 022702 020024
 522 010552 001431
 523 010554 022702 024024
 524 010560 001426
 525
 526 010562 022702 020025
 527 010566 001423
 528 010570 022702 024025
 529 010574 001420
 530
 531 010576 022702 020027
 532 010602 001415
 533 010604 022702 024027
 534 010610 001412
 535
 536 010612 010237 001142
 537 010616 010037 001136
 538 010622 062737 000026 001136
 539 010630 104057
 540 010632 000137 021274
 541 010636
 542
 543

TST7:

SCOPE ;SCOPE CALL
 NOP
 MOV #STACK,SP ;LOAD THE STACK POINTER
 MOV \$BASE,R0 ;RO = UNIBUS ADDRESS
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
 MOV #7,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
 MOV RMDT(R0),R2 ;STORE RMDT AT R2
 CMP #SNGPRT,R2 ;SINGLE PORT RM03 ?
 BEQ 10\$;YES !!
 CMP #DULPRT,R2 ;DUAL PORT RM03 ?
 BEQ 10\$;YES !!
 CMP #SNGPRT!BIT0,R2 ;SINGLE PORT RM02 ?
 BEQ 10\$;YES !!
 CMP #DULPRT!BIT0,R2 ;DUAL PORT RM02 ?
 BEQ 10\$;YES !!
 CMP #SNGPRT!BIT1!BIT0,R2 ;SINGLE PORT RM05 ?
 BEQ 10\$;YES !!
 CMP #DULPRT!BIT1!BIT0,R2 ;DUAL PORT RM05 ?
 BEQ 10\$
 MOV R2,\$BDDAT ;GET RECIEVED DRIVE TYPE
 MOV R0,\$BDADR ;LOAD BAD ADDRESS
 ADD #RMDT,\$BDADR
 EMT 57
 JMP \$EOSP ;GO TO NEXT DEVICE

10\$:

 ;*TEST 10 DEVICE AVAILABLE TEST

010636
 010636 000004
 010640 000240
 010642 012706 001100
 010646 013700 001276
 010652 013701 001466
 010656 012737 000010 001226
 544

TST10:

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

```

545 010664 004737 022404 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
546 010670 016037 000000 001142 MOV RMCS1(R0),%BDDAT ;STORE RMCS1 AT %BDDAT
547 010676 042737 173777 001142 BIC #^CDVA,%BDDAT ;CLEAR ALL BUT DVA
548 010704 001006 BNE 10$ ;BRANCH IF DVA SET
549 010706 012737 004000 001140 MOV #DVA,%GDDAT ;SETUP EXPECTED
550 010714 010037 001136 MOV R0,%BDADR ;SETUP REG ADDRESS
551 010720 104060 EMT 60

```

10\$:

::*****
:*TEST 11 SEARCH TIMEOUT TEST

::*****
TST11:

```

010722 SCOPE ;SCOPE CALL
010722 000004 NOP
010724 000240
010726 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
010732 013700 001276 MOV %BASE,R0 ;R0 = UNIBUS ADDRESS
010736 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
010742 012737 000011 001226 MOV #11,%TESTN ;;SET TEST NUMBER IN APT MAIL BOX

```

555
556 ;LOAD REGISTER OUTPUT BUFFER WITH COMMAND PARAMETERS

```

557 010750 012737 000000 001420 MOV #0,RMDAO ;SECTOR=0=TRACK
558 010756 012737 000000 001446 MOV #0,RMDCO ;CYLINDER=0
559 010764 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
560 010772 012737 054636 001416 MOV #BUFFER,RMBAO ;STARTING BUFFER ADDRESS
561 011000 012737 177777 001414 MOV #-1,RMWCO ;WORD COUNT
562 011006 012737 000061 001412 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND

```

```

563  
564 011014 004737 022432 JSR PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE  
;SEARCH IS ENABLED USING SUBROUTINE.
011020 000402 BR 10$ ;BRANCH TO 10$ IF NO ERROR
011022 104000 EMT

```

```

565 011024 000462 BR 50$ ;SLOE REST PF TEST

```

;START THE CLOCK AND WAIT FOR 100 MS

```

10$:
568 011026
569 011026 012737 000144 001534 MOV #100,%WATCH ;SET WATCHDOG TIMER VALUE
011034 004777 170476 JSR PC,@CLOCK ;START THE CLOCK
570 011040 005737 001534 20$: TST WATCH
571 011044 001375 BNE 20$
572 011046 004777 170466 JSR PC,@STOPCL ;STOP THE CLOCK

```

;VERIFY THAT OPI IS NOT SET (SEARCH TIMEOUT IS DISABLED)

```

574  
575 011052 016037 000014 001142 MOV RMER1(R0),%BDDAT ;STORE RMER1 AT %BDDAT
576 011060 010037 001136 MOV R0,%BDADR ;SET UP FOR ERROR MSG
577 011064 062737 000014 001136 ADD #RMER1,%BDADR
578 011072 042737 157777 001142 BIC #^COPI,%BDDAT
579 011100 001404 BEQ 30$ ;BRANCH IF NO ERROR
580 011102 005037 001140 CLR %GDDAT
581 011106 104020 EMT 20

```

;DISABLED
;SKIP

```

582  
583 011110 000430 BR 50$

```

;ENABLE SEARCH TIMEOUT, THEN WAIT 100 MS

```

584  
585  
586 011112 30$:
587 011112 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMMR1(R0) ;LOAD RMMR1

```



```

588 011120 012737 000144 001534      MOV      #100,,WATCH      ;SET WATCHDOG TIMER VALUE
      011126 004777 170404              JSR      PC,@CLOCK        ;START THE CLOCK
589 011132 005737 001534      40$:    TST      WATCH
590 011136 001375              BNE     40$
591 011140 004777 170374      JSR      PC,@STOPCL      ;STOP THE CLOCK
592
593      ;OPI SHOULD NOW BE SET (SEARCH TIMEOUT IS ENABLED)
594 011144 016037 000014 001142      MOV      RMER1(RO),$BDDAT ;STORE RMER1 AT $BDDAT
595 011152 042737 157777 001142      BIC     #^COPI,$BDDAT
596 011160 001004              BNE     50$
597 011162 012737 020000 001140      MOV      #OPI,$GDDAT
598 011170 104021              EMT     21
599 011172              50$:
600
601      ;*****
      ;*TEST 12      SET DTE TEST
      ;*****
      TST12:
      SCOPE              ;SCOPE CALL
      NOP
      MOV      #STACK,SP    ;LOAD THE STACK POINTER
      MOV      $BASE,RO     ;RO = UNIBUS ADDRESS
      MOV      TSTQUE,R1    ;R1 = POINTER TO DEVICE
      MOV      #12,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
602
603 011220 010037 001136              MOV      RO,$BDADR      ;SETUP ERROR MSG
604 011224 062737 000014 001136      ADD     #RMER1,$BDADR
605 011232 005037 001140              CLR     $GDDAT
606
607      ;INITILAIZE AND VERIFY THAT DRIVE TIMING ERROR IS RESET
608 011236 004737 022404      JSR     PC,CNTCLR      ;GO CLEAR CONTROLLER
609
610 011242 016037 000014 001142      MOV      RMER1(RO),$BDDAT ;STORE RMER1 AT $BDDAT
611 011250 042737 167777 001142      BIC     #^CDTE,$BDDAT
612 011256 001402              BEQ     10$             ;BRANCH IF DTE=0
613 011260 104031              EMT     31
614 011262 000517              BR      50$             ;SKIP REST OF TEST
615
616      ;SET MAINTENANCE INDEX PULSE AND VERIFY DTE REMAINS RESET
617 011264              10$:
618 011264 012760 000001 000024      MOV      #DMD,RMMR1(RO) ;LOAD RMMR1
619 011272 012760 000005 000024      MOV      #DMD!MI,RMMR1(RO) ;LOAD RMMR1
620 011300 016037 000014 001142      MOV      RMER1(RO),$BDDAT ;STORE RMER1 AT $BDDAT
621 011306 042737 167777 001142      BIC     #^CDTE,$BDDAT
622 011314 001402              BEQ     20$
623 011316 104000              EMT     20
624 011320 000500              BR      50$             ;COMPARE SHOULD BE RESET
625
626      ;EXECUTE DUMMY DATA COMMAND TO ENABLE SEARCH
627 011322              20$:
628 011322 012737 000000 001420      MOV      #0,RMDAO
629 011330 012737 000005 001446      MOV      #5,RMDCO
630 011336 012737 010000 001444      MOV      #FMT16,RMOFO
631 011344 012737 054636 001416      MOV      #BUFFER,RMBAO
632 011352 012737 177777 001414      MOV      #-1,RMWCO
633 011360 012737 000061 001412      MOV      #WD.GO,RMCS10
  
```

```

635 011366 004737 022432      JSR      PC,ENBSCH      ;EXECUTE DATA COMMAND TO POINT WHERE
                                ;SEARCH IS ENABLED USING SUBROUTINE.
                                ;BRANCH TO 30$ IF NO ERROR
                                011372 000402      BR       30$
                                011374 104000      EMT
636 011376 000451      BR       50$
637
638 ;WITH SEARCH ENABLED, SET AND RESET SECTOR PULSE TO SET ENABLE
639 ;SEARCH FLOP.
640 011400      30$:
641 011400 012760 051441 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN!MS,RMMR1(RO) ;LOAD RMMR1
642 011406 012760 051401 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
643
644 ;SET SECTOR PULSE AND VERIFY DTE DOES NOT SET
645 ;
646 ;
647 ;
648 011414 016037 000014 001142      MOV      RMR1(RO),SBDDAT ;STORE RMR1 AT SBDDAT
649 011422 042737 167 , ' 001142      BIC      #^CDTE,SBDDAT
650 011430 001402      BEQ     40$
651 011432 104023      EMT     23
652 011434 001432      BEQ     50$ ;COMPARE SHOULD BE RESET
653
654 ;FORCE SECTOR COMPARE
655 011436      40$:
656 011436 012760 051403 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO) ;LOAD RMMR1
657 011444 012760 051443 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC!MS,RMMR1(RO) ;LOAD RMMR1
658 011452 012760 051403 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO) ;LOAD RMMR1
659
660 ;SET SECTOR PULSE AND VERIFY DTE SETS
661 011460 012760 051441 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN!MS,RMMR1(RO) ;LOAD RMMR1
662 011466 012760 051401 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
663 011474 016037 000014 001142      MOV      RMR1(RO),SBDDAT ;STORE RMR1 AT SBDDAT
664 011502 042737 167777 001142      BIC      #^CDTE,SBDDAT
665 011510 001004      BNE     50$
666 011512 012737 010000 001142      MOV      #DTE,SBDDAT
667 011520 104024      EMT     24 ;SECTOR COMPARE SET
668
669 011522      50$:
670
671 ;*****
; *TEST 13          FORMAT CHANGE TEST
;*****
;TST13:
011522
011522 000004      SCOPE ;SCOPE CALL
011524 000240      NOP
011526 012706 001100      MOV      #STACK,SP ;LOAD THE STACK POINTER
011532 013700 001276      MOV      $BASE,R0 ;R0 = UNIBUS ADDRESS
011536 013701 001466      MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
011542 012737 000013 001226      MOV      #13,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
672
673 011550 012702 012100      MOV      #50$,R2 ;R2=TABLE POINTER
674
675 ;INITILAIZE AND SET THE FORMAT BIT, USE INDEX PULSE TO CLEAR FORMAT CHANGE
676 011554      10$:
677 011554 004737 022404      JSR      PC,CNTCLR ;GO CLEAR CONTROLLER

```

```

678 011560 011260 000032      MOV      (R2),RMOF(RO)      ;LOAD FORMAT MODE IN RMOF
679 011564 012760 000001 000024  MOV      #DMD,RMMR1(RO)    ;LOAD RMMR1
680 011572 012760 000005 000024  MOV      #DMD!MI,RMMR1(RO)  ;LOAD RMMR1
681
682                               ;SETUP AND EXECUTE DUMMY DATA COMMAND USING OPPOSITE FORMAT
683 011600 012737 000000 001420  MOV      #0,RMDAO
684 011606 012737 000005 001446  MOV      #5,RMDCO
685 011614 016237 000002 001444  MOV      2(R2),RMOFO
686 011622 012737 054636 001416  MOV      #BUFFER,RMBAO
687 011630 012737 177777 001414  MOV      #-1,RMWCO
688 011636 012737 000061 001412  MOV      #WD!GO,RMCS10
689
690 011644 004737 022432      JSR      PC,ENBSCH          ;EXECUTE DATA COMMAND TO POINT WHERE
                               ;SEARCH IS ENABLED USING SUBROUTINE.
                               ;BRANCH TO 20$ IF NO ERROR
        011650 000402      BR      20$
        011652 104000      EMT
691 011654 000514      BR      60$
692
693                               ;FORMAT CHANGE FLOP SHOULD BE SET - VERIFY BY TRYING TO FORCE A
694                               ;DRIVE TIMING ERROR WHICH SHOULD NOT SET.
695 011656
696 011656 012760 051403 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO) ;LOAD RMMR1
697 011664 012760 051443 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC!MS,RMMR1(RO) ;LOAD RMMR1
698 011672 012760 051403 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO) ;LOAD RMMR1
699 011700 012760 051401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
700
701                               ;VERIFY THAT DRIVE TIMING ERROR DIDNT SET
702 011706 016037 000014 001142  MOV      RMER1(RO),%BDDAT    ;STORE RMER1 AT %BDDAT
703 011714 042737 167777 001142  BIC      #^CDTE,%BDDAT
704 011722 001416      BEQ     30$
705 011724 010037 001136      MOV     RO,%BDADR           ;SETUP ERROR MESSAGE
706 011730 062737 000014 001136  ADD     #RMER1,%BDADR
707 011736 005037 001140      CLR     %GDDAT
708 011742 011237 001174      MOV     (R2),%TMP0
709 011746 016237 000002 001176  MOV     2(R2),%TMP1
710 011754 104025      EMT
711 011756 000453      BR      60$                ;A FORMAT CHANGE
712
713                               ;CLEAR THE FORMAT CHANGE FLOP W/INDEX PULSE
714 011760
715 011760 012760 051405 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MI,RMMR1(RO) ;LOAD RMMR1
716
717                               ;ENABLE SEARCH AND FORCE SECTOR COMPARE
718 011766 012760 051403 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO) ;LOAD RMMR1
719 011774 012760 051443 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC!MS,RMMR1(RO) ;LOAD RMMR1
720 012002 012760 051403 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO) ;LOAD RMMR1
721
722                               ;SET DTE W/ANOTHER SECTOR PULSE - VERIFY DTE IS SET
723 012010 012760 051441 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN!MS,RMMR1(RO) ;LOAD RMMR1
724 012016 012760 051401 000024  MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
725 012024 016037 000014 001142  MOV      RMER1(RO),%BDDAT    ;STORE RMER1 AT %BDDAT
726 012032 042737 167777 001142  BIC      #^CDTE,%BDDAT
727 012040 001012      BNE     40$
728 012042 010037 001136      MOV     RO,%BDADR           ;SETUP ERROR MESSAGE
729 012046 062737 000014 001136  ADD     #RMER1,%BDADR
730 012054 012737 010000 001140  MOV     #DTE,%GDDAT
731 C*2062 104027      EMT
        27
  
```

732 012064 000410
 733
 734
 735 012066
 736 012066 022702 012100
 737 012072 001005
 738 012074 005722
 739 012076 000626
 740
 741 012100 010000
 742 012102 000000
 743 012104 010000
 744 012106
 745
 746

BR 60\$
 ;DO TEST W/18 TO 16 FORMAT CHANGE DURING SECOND EXECUTION
 40\$:
 CMP #50\$,R2
 BNE 60\$;BRANCH IF DONE TEST
 TST (R2)+ ;MOVE POINTER
 BR 10\$;REPEAT TEST
 50\$:
 .WORD FMT16 ;16-18 FORMAT CHANGE
 .WORD 0 ;18-16 FORMAT CHANGE
 .WORD FMT16
 60\$:
 ;END OF TEST
 ;*****
 ;*TEST 14 PROM STROBE TEST
 ;*****

012106
 012106 000004
 012110 000240
 012112 012706 001100
 012116 013700 001276
 012122 013701 001466
 012126 012737 000014 001226
 747
 748 012134 004737 022404
 749 012140 010037 001136
 750 012144 062737 000024 001136
 751
 752
 753 012152 012760 000001 000024
 754 012160 012702 000021
 755 012164
 012164 012760 004001 000024
 756 012172 012760 000001 000024
 757 012200 016037 000024 001142
 758 012206 042737 177737 001142
 759 012214 001406
 760 012216 005302
 761 012220 001361
 762 012222 005037 001140
 763 012226 104000
 764 012230 000501
 765 012232 012702 000021
 766 012236
 012236 012760 004001 000024
 767 012244 012760 000001 000024
 768 012252 016037 000024 001142
 769 012260 042737 177737 001142
 770 012266 001007
 771 012270 005302
 772 012272 001361
 773 012274 012737 000040 001140
 774 012302 104030
 775 012304 000453
 776

TST14:
 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 #14,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
 MOV R0,\$BDADR ;SETUP ERROR MESSAGE
 ADD #RMMR1,\$BDADR
 ;SET DIAGNOSTIC MODE AND TRY TO RESET PROM STROBE
 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
 MOV #17.,R2 ;R2=MAXIMUM # CLOCK PULSES
 5\$:
 MOV #DMD!MCLK,RMMR1(R0) ;LOAD RMMR1
 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
 MOV RMMR1(R0),\$BDDAT ;STORE RMMR1 AT \$BDDAT
 BIC #^CWC,\$BDDAT
 BEQ 6\$
 DEC R2
 BNE 5\$
 CLR \$GDDAT
 EMT
 BR 60\$
 6\$:
 MOV #17.,R2
 10\$:
 MOV #DMD!MCLK,RMMR1(R0) ;LOAD RMMR1
 MOV #DMD,RMMR1(R0) ;LOAD RMMR1
 MOV RMMR1(R0),\$BDDAT ;STORE RMMR1 AT \$BDDAT
 BIC #^CWC,\$BDDAT
 BNE 20\$;EXIT LOOP WHEN PROM STROBE ON
 DEC R2
 BNE 10\$
 MOV #WC,\$GDDAT
 EMT 30
 BR 60\$

```

777 ;VERIFY PROM STROBE IS ON FOR 3 BIT CLOCKS
778 012306 20$:
779 012306 012702 000003 MOV #3.,R2
780 012312 25$:
    012312 012760 004001 000024 MOV #DMD!MCLK,RMMR1(RO) ;LOAD RMMR1
781 012320 012760 000001 000024 MOV #DMD,RMMR1(RO) ;LOAD RMMR1
782 012326 016037 000024 001142 MOV RMMR1(RO),SBDDAT ;STORE RMMR1 AT SBDDAT
783 012334 042737 177737 001142 BIC #^CWC,SBDDAT
784 012342 001005 BNE 30$ ;BRANCH IF PROM STORBE DID NOT
785 ;DROP EARLY
786 012344 012737 000040 001140 MOV #WC,$GDDAT
787 012352 104032 EMT 32
788 012354 000427 BR 60$
789 012356 005302 30$: DEC R2
790 012360 001354 BNE 25$

791 ;VERIFY PROM STROBE IS OFF FOR 8 BIT CLOCKS
792
793 012362 012702 000010 40$:
794 012366 MOV #8.,R2
    C12366 012760 004001 000024 MOV #DMD!MCLK,RMMR1(RO) ;LOAD RMMR1
795 012374 012760 000001 000024 MOV #DMD,RMMR1(RO) ;LOAD RMMR1
796 012402 016037 000024 001142 MOV RMMR1(RO),SBDDAT ;STORE RMMR1 AT SBDDAT
797 012410 042737 177737 001142 BIC #^CWC,SBDDAT
798 012416 001404 BEQ 50$ ;BRANCH IF PROM STROBE
799 012420 005037 001140 CLR $GDDAT ;DID NOT SET EARLY
800 012424 104033 EMT 33
801 012426 000402 BR 60$
802 012430 005302 50$: DEC R2
803 012432 001355 BNE 40$
804 012434 60$:
805
806 ;*****
; *TEST 15 SYNC WORD COUNT INHIBIT TEST
;*****

012434 TST15:
012434 000004 SCOPE ;SCOPE CALL
012436 000240 NOP
012440 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
012444 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
012450 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
807 C12454 012737 000015 001226 MOV #15,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

808 ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES
809 012462 012737 000005 001446 MOV #5,RMDCO ;CYLINDER=5
810 012470 012737 000000 001420 MOV #0,RMDAO ;SECTOR=0, TRACK=0
811 012476 012737 010000 001444 MOV #FMT16,RMOFO ;16 BIT FORMAT
812 012504 012737 054636 001416 MOV #BUFFER,RMBAO ;STARTING BUFFER ADDRESS
813 012512 012737 177777 001414 MOV #-1,RMWCO ;WORD COUNT
814 012520 012737 000061 001412 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
815
816 012526 004737 022432 JSR PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE
;SEARCH IS ENABLED USING SUBROUTINE.
012532 000402 BR 10$ ;BRANCH TO 10$ IF NO ERROR
012534 104000 EMT
817 012536 000562 BR 120$ ;SKIP IF FAILURE
818 012540 10$:

```

```

819 012540 004737 023304      JSR      PC,SCTCMP      ;FORCE SECTOR COMPARE USING SUBROUTINE
      012544 000402      BR       20$           ;BRANCH TO 20$ IF NO ERROR
      012546 104000      EMT
820 012550 000555      BR       120$
821
822      ;CAN NOW STEP DATA TIMING SEQUENCER WITH SECTOR COMPARE SET
823
824      ;STEP THE SEQUENCER TO LOCATION 10(10) WHILE CHECKING 'PLFS'
825 012552 012702 000000      20$:    MOV      #0,R2           ;R2=SEQUENCER ADDRESS
826
827      ;PULSE MAINTENANCE CLOCK UNTIL PROM STROBE SETS
828 012556      30$:
829 012556 012760 055401 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(R0) ;LOAD RMMR1
830 012564 012760 051401 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
831
832      ;SEE IF PROM STROBE IS SET
833 012572 015003 000024      MOV      RMMR1(R0),R3      ;STORE RMMR1 AT R3
834 012576 032703 000040      BIT      #WC,R3
835 012602 001765      BEQ      30$           ;ISSUE NEXT BIT CLOCK
836 012604 005202      INC      R2           ;INCREMENT SEQUENCER ADDRESS
837
838      ;CHECK 'LOOKING FOR SYNC' - SHOULD BE ZERO UNTIL LOCATION 11
839 012606 010337 001142      MOV      R3,$BDDAT
840 012612 042737 175777 001142      BIC      #^CPLFS,$BDDAT
841 012620 001413      BEQ      50$           ;BRANCH IF PLFS IS ZERO
842 012622 005037 001140      CLR      $GDDAT
843 012626 010037 001136      MOV      R0,$BDADR
844 012632 062737 000024 001136      ADD      #RMMR1,$BDADR
845 012640 010237 001174      MOV      R2,$TMP0
846 012644 104034      EMT
847 012646 000516      BR       120$           ;SKIP REST OF TEST
848
849      ;EXIT LOOP IF SEQUENCER NOW AT LOCATION 10
850 012650      50$:
851 012650 022702 000012      CMP      #10.,R2
852 012654 001414      BEQ      70$
853
854      ;PULSE MAINTENANCE CLOCK UNTIL PROM STROBE RESETS
855 012656      60$:
856 012656 012760 055401 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(R0) ;LOAD RMMR1
857 012664 012760 051401 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
858
859      ;SEE IF PROM STROBE IS RESET
860 012672 016003 000024      MOV      RMMR1(R0),R3      ;STORE RMMR1 AT R3
861 012676 032703 000040      BIT      #WC,R3
862 012702 001365      BNE      60$
863 012704 000724      BR       30$           ;GO STEP SEQUENCER TO NEXT LOC
864
865      ;THE NEXT PROM STROBE SHOULD SET 'PLFS' AT LOCATION 11(10) OF THE
866      ;DATA TIMING SEQUENCER.
867 012706      70$:
868 012706 012702 000020      MOV      #16.,R2           ;R2=NUMBER OF BIT CLOCKS
869
870      ;ISSUE 16 BIT CLOCKS TO GET NEXT PROM STROBE
871 012712      80$:
872 012712 012760 055401 000024      MOV      #DMD.MUR.DBEN!MOC!MSEN.MCLK,RMMR1(R0) ;LOAD RMMR1
873 012720 012760 051401 000024      MOV      #DMD.MUR.DBEN.MOC!MSEN,RMMR1(R0) ;LOAD RMMR1
  
```

```

874 012726 005302          DEC      R2
875 012730 001370          BNE     80$
876
877                          ;VERIFY THAT 'PLFS' IS NOW SET
878 012732 016037 000024 001142  MOV     RMMR1(R0), $BDDAT      ;STORE RMMR1 AT $BDDAT
879 012740 042737 175777 001142  BIC     #^CPLFS, $BDDAT
880 012746 001012          BNE     90$                    ;BRANCH IF PLFS IS SET
881 012750 012737 002000 001140  MOV     #PLFS, $GDDAT        ;SETUP ERROR MESSAGE
882 012756 010037 001136          MOV     R0, $BDADR
883 012762 062737 000024 001136  ADD     #RMMR1, $BDADR
884 012770 104035          EMT     35
885 012772 000444          BR      120$                    ;SKIP REST OF TEST
886
887                          ;ISSUE 3 MORE BIT CLOCKS TO RESET PROM STROBE
888 012774          90$:
889 012774 012702 000003          MOV     #3, R2
890 013000          95$:
891 013006 012760 055401 000024  MOV     #DMD!MUR!DBEN!MOC!MSEN!MCLK, RMMR1(R0) ;LOAD RMMR1
892 013014 005302          MOV     #DMD.MUR.DBEN!MOC!MSEN, RMMR1(R0) ;LOAD RMMR1
893 013016 001370          DEC     R2
894                          BNE     95$
895                          ;WITH 'LOOKING FOR SYNC SET, FURTHER BIT CLOCKS SHOULD NOT SET
896                          ;PROM STROBE
897 013020 012702 000040          MOV     #32, R2                ;R2=NUMBER OF BIT CLOCKS
898
899                          ;PULSE BIT CLOCK AND VERIFY PROM STROBE DOES NOT SET
900 013024          100$:
901 013024 012760 055401 000024  MOV     #DMD!MUR!DBEN!MOC!MSEN!MCLK, RMMR1(R0) ;LOAD RMMR1
902 013032 012760 051401 000024  MOV     #DMD!MUR!DBEN!MOC!MSEN, RMMR1(R0) ;LOAD RMMR1
903 013040 016003 000024          MOV     RMMR1(R0), R3        ;STORE RMMR1 AT R3
904 013044 042703 177737          BIC     #^CWC, R3
905 013050 001413          BEQ     110$                    ;BRANCH IF PROM STROBE IS 0
906 013052 005037 001140          CLR     $GDDAT                ;SETUP ERROR MESSAGE
907 013056 010337 001142          MOV     R3, $BDDAT
908 013062 010037 001136          MOV     R0, $BDADR
909 013066 062737 000024 001136  ADD     #RMMR1, $BDADR
910 013074 104036          EMT     36
911 013076 000402          BR      120$
912 013100 005302          110$: DEC     R2                ;CONTINUE FOR 32 BIT CLOCKS
913 013102 001350          BNE     100$
914
915 013104          120$:                    ;END OF TEST
916
917

```

; *TEST 16 SYNC DETECTION TEST

```

013104          TST16:
013104 000004          SCOPE                    ;SCOPE CALL
013106 000240          NOP
013110 012706 001100          MOV     #STACK, SP        ;LOAD THE STACK POINTER
013114 013700 001276          MOV     $BASE, R0         ;R0 = UNIBUS ADDRESS
013120 013701 001466          MOV     TSTQUE, R1        ;R1 = POINTER TO DEVICE
013124 012737 000016 001226  MOV     #16, $TESTN       ;:SET TEST NUMBER IN APT MAIL BOX
918
919                          ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES

```

```

920 013132 012737 000000 001420      MOV      #0,RMDAO      ;SECTOR 0,TRACK 0
921 013140 012737 000005 001446      MOV      #5,RMDCO      ;CYLINDER 5
922 013146 012737 010000 001444      MOV      #FMT16,RMOFO   ;16 BIT MODE
923 013154 012737 054636 001416      MOV      #BUFFER,RMBAO  ;BUFFER ADDRESS
924 013162 012737 177777 001414      MOV      #-1,RMWCO     ;WORD COUNT
925 013170 012737 000071 001412      MOV      #RD.GO,RMCS10 ;READ DATA COMMAND
926
927 013176 004737 022432      JSR      PC,ENBSCH     ;EXECUTE DATA COMMAND TO POINT WHERE
                                ;SEARCH IS ENABLED USING SUBROUTINE.
                                ;BRANCH TO 10$ IF NO ERROR
                                10$:
                                BR      10$
                                EMT
                                BR      100$
928 013206 000550
929 013210
930 013210 004737 023304      JSR      PC,SCTCMP     ;FORCE SECTOR COMPARE USING SUBROUTINE
                                ;BRANCH TO 20$ IF NO ERROR
                                20$:
                                BR      20$
                                EMT
                                BR      100$
931 013220 000543
932 013222
933 013222 004737 023412      JSR      PC,SETLFS     ;SET 'LOOKING FOR SYNC' USING SUBROUTINE
                                ;BRANCH TO 30$ IF NO ERROR
                                30$:
                                BR      30$
                                EMT
                                BR      100$
934 013232 000536
935
936
937
938 013234
939 013234 012702 000020      :CLOCK ALL ONES SYNC PATTERN AND VERIFY SYNC IS NOT DETECTED
                                ;(USING PROM STORAGE AS INDICATION)
                                30$:
                                MOV      #16.,R2      ;NUMBER OF ONE BITS
                                MOV      RO,$BDADR     ;SETUP ERROR MESSAGE
940 013240 010037 001136      ADD      #RMMR1,$BDADR
941 013244 062737 000024 001136      CLR      $GDDAT
942 013252 005037 001140      MOV      #MR1AAA!MRD,RMMR1(RO) ;LOAD RMMR1
                                40$:
                                MOV      #MR1AAA!MRD!MCLK,RMMR1(RO) ;LOAD RMMR1
                                MOV      #MR1AAA!MRD,RMMR1(RO) ;LOAD RMMR1
943 013256 012760 053401 000024      MOV      RMMR1(RO),$BDDAT ;STORE RMMR1 AT $BDDAT
944 013264
945 013264 012760 057401 000024      BIC     #^CWC,$BDDAT
946 013272 012760 053401 000024      BEQ     50$
947 013300 016037 000024 001142      MOV     #177777,$TMP0
948 013306 042737 177737 001142      EMT     37
949 013314 001405
950 013316 012737 177777 001174      BR      100$
951 013324 104037
952 013326 000500
953 013330
954 013330 005302      50$:
                                DEC      R2          ;REPEAT FOR 16 BITS
955 013332 001354      BNE     40$
956 013334 012702 000020      MOV     #16.,R2      ;NUMBER OF ZERO BITS
957
958
959 013340      :CLOCK ALL ZERO SYNC PATTERN AND VERIFY SYNC IS NOT DETECTED
                                60$:
                                MOV      #MR1AAA!MCLK,RMMR1(RO) ;LOAD RMMR1
960 013340 012760 055401 000024      MOV      #MR1AAA,RMMR1(RO) ;LOAD RMMR1
961 013346 012760 051401 000024      MOV      RMMR1(RO),$BDDAT ;STORE RMMR1 AT $BDDAT
962 013354 016037 000024 001142      BIC     #^CWC,$BDDAT ;MAKE SURE SYNC NOT DETECTED
963 013362 042737 177737 001142      BEQ     70$
964 013370 001404
965 013372 005037 001174      CLR     $TMP0
966 013376 104037      EMT     37
967 013400 000453      BR      100$ ;SKIP IF FAILURE
968 013402
969 013402 005302      70$:
                                DEC      R2          ;REPEAT FOR 16 BITS

```



```

970 013404 001355          BNE      60$
971 013406 012702  C00040  MOV      #32,R2          ;R2=NUMBER OF BITS
972 013412 012703  000031  MOV      #000031,R3     ;R3=SYNC PATTERN FOR LEFT SHIFT
973
974          ;CLOCK THE BINARY SYNC PATTERN (10011000) AND VERIFY THAT SYNC IS
975          ;DETECTED
976 013416 012737  051401  001436  80$:    MOV      #MR1AAA,RMMR10 ;GENERATE VALUE OF RMMR1
977 013424 000241          CLC          ;CLEAR THE CARR
978 013426 006003          ROR      R3          ;SHIFT RIGHT
979 013430 103003          BCC     90$          ;BRANCH IF C BIT CLEAR
980 013432 052737  002000  001436  BIS      #MRD,RMMR10    ;SET MRD IF PATTERN BIT SETS
981 013440
982 013440 013760  001436  000024  90$:    MOV      RMMR10,RMMR1(RO) ;LOAD RMMR1
983 013446 052737  004000  001436  BIS      #MCLK,RMMR10    ;SET THE MAINTENANCE DATA CLK
984 013454 013760  001436  000024  MOV      RMMR10,RMMR1(RO) ;LOAD RMMR1
985 013462 042737  004000  001436  BIC      #MCLK,RMMR10    ;RESET BIT CLOCK
986 013470 013760  001436  000024  MOV      RMMR10,RMMR1(PO) ;LOAD RMMR1
987 013476 016037  000024  001142  MOV      RMMR1(RO), $BDDAT ;STORE RMMR1 AT $BDDAT
988 013504 042737  177737  001142  BIC      #^CWC,$BDDAT
989 013512 001006          BNE     100$         ;BRANCH IF PROM STROBE IS SET
990 013514 005302          DEC     R2          ;CONTINUE FOR 16 BIT CLOCKS
991 013516 001337          BNE     80$
992 013520 012737  C00040  001140  MOV      #WC,$GDDAT
993 013526 104040          EMT     40
994          100$:          ;END OF TEST
995

```

 *TEST 17 ABORT SYNC DETECTION TEST

```

013530
013530 000004          TST17:  SCOPE          ;SCOPE CALL
013532 000240          NOP
013534 012706  001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
013540 013700  001276  MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
013544 013701  001466  MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
013550 012737  000017  001226  MOV      #17,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX
996
997          ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES
998 013556 012737  000000  001420  MOV      #0,RMDAO      ;SECTOR 0, TRACK 0
999 013564 012737  000005  001446  MOV      #5,RMDCO      ;CYLINDER 5
1000 013572 012737  010000  001444  MOV      #FMT16,RMOFO   ;16 BIT FORMAT
1001 013600 012737  054636  001416  MOV      #BUFFER,RMBAD  ;STARTING BUFFER ADDRESS
1002 013606 012737  177777  001414  MOV      #-1,RMWCO     ;WORD COUNT
1003 013614 012737  000071  001412  MOV      #RD!GO,RMCS10 ;READ DATA COMMAND
1004
1005 013622 004737  022432          JSR     PC,ENBSCH      ;EXECUTE DATA COMMAND TO POINT WHERE
                          ;SEARCH IS ENABLED USING SUBROUTINE.
013626 000402          BR     10$           ;BRANCH TO 10$ IF NO ERROR
013630 104000          EMT
013632 000447          BR     50$
1006          10$:
1007 013634          JSR     PC,SCTCMP     ;FORCE SECTOR COMPARE USING SUBROUTINE
1008 013640 000402  023304  BR     20$           ;BRANCH TO 20$ IF NO ERROR
013642 104000          EMT
1009 013644 000442          BR     50$
1010 013646

```

```

1011 013646 004737 023412      JSR    PC,SETLFS      ;SET 'LOOKING FOR SYNC' USING SUBROUTINE
      013652 000402              BR     30$           ;BRANCH TO 30$ IF NO ERROR
      013654 104000              EMT
1012 013656 000435              BR     50$
1013
1014                               ;LOOKING FOR SYNC INHIBITS WORD CLOCK, HOWEVER, 'DRIVE TIMING ERROR'
1015                               ;SHOULD RESET 'LFS WORD COUNT INHIBIT' FLOP
1016                               ;FORCE DRIVE TIMING ERROR
1017 013660                               30$:
1018 013660 012760 051441 000024      MOV    #MR1AAA!MS,RMMR1(R0) ;LOAD RMMR1
1019
1020                               ;PULSE BIT CLOCK AND VERIFY PROM STROBE SETS
1021 013666 012702 000020      MOV    #16.,R2        ;R2, NUMBER OF BIT CLOCKS
1022 013672                               40$:
1023 013672 012760 055441 000024      MOV    #MR1AAA!MS!MCLK,RMMR1(R0) ;LOAD RMMR1
1024 013700 012760 051441 000024      MOV    #MR1AAA!MS,RMMR1(R0) ;LOAD RMMR1
1025 013706 016037 000024 001142      MOV    RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
1026 013714 042737 177737 001142      BIC    #^CWC,SBDDAT
1027 013722 001013              BNE    50$
1028 013724 005302              DEC    R2
1029 013726 001361              BNE    40$
1030 013730 012737 000040 001140      MOV    #WC,SGDDAT
1031 013736 010037 001136              MOV    R0,$BDADR
1032 013742 062737 000024 001136      ADD    #RMMR1,$BDADR
1033 013750 104041              EMT    41
1034 013752                               50$:
1035
1036
1037

```

 :*TEST 20 SYNC GENERATION TEST

```

013752                               TST20:
013752 000004              SCOPE                ;SCOPE CALL
013754 000240              NOP
013756 012706 001100      MOV    #STACK,SP    ;LOAD THE STACK POINTER
013762 013700 001276      MOV    $BASE,R0     ;R0 = UNIBUS ADDRESS
013766 013701 001466      MOV    TSTQUE,R1    ;R1 = POINTER TO DEVICE
013772 012737 000020 001226      MOV    #20,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
1038
1039                               ;SETUP REGISTER OUTPUT BUFFER FOR SUBROUTINES
1040 014000 012737 000000 001420      MOV    #0,RMDAO     ;SECTOR 0, TRACK 0
1041 014006 012737 000005 001446      MOV    #5,RMDCO     ;CYL 5
1042 014014 012737 010000 001444      MOV    #FMT16,RMOFO ;16 BIT MODE
1043 014022 012737 054636 001416      MOV    #BUFFER,RMBAO ;BUFFER ADDRESS
1044 014030 012737 177776 001414      MOV    #-2,RMWCO    ;WORD COUNT
1045 014036 012737 000063 001412      MOV    #WH!GO,RMCS10 ;WHITE HEAD AND DATA COM
1046
1047 014044 004737 022432      JSR    PC,ENBSCH    ;EXECUTE DATA COMMAND TO POINT WHERE
                               ;SEARCH IS ENABLED USING SUBROUTINE.
      014050 000403              BR     10$         ;BRANCH TO 10$ IF NO ERROR
      014052 104000              EMT
1048 014054 000137 014444      JMP    160$
1049 014060                               10$:
1050 014060 004737 023304      JSR    PC,SCTCMP    ;FORCE SECTOR COMPARE USING SUBROUTINE
      014064 000402              BR     20$         ;BRANCH TO 20$ IF NO ERROR
      014066 104000              EMT

```

```

1051 014070 000565 BR 160$
1052
1053 ;WRITE PROM OF DATA TIMING SEQUENCER SHOULD NOW BE ENABLED
1054 ;FIRST, VERIFY THAT WRITE GATE IS ON, INDICATING THAT
1055 ;SECTOR COMPARE SET
1056 014072 20$:
1057 014072 016037 000040 001142 MOV RMMR2(R0), $BDDAT ;STORE RMMR2 AT $BDDAT
1058 014100 042737 177776 001142 BIC #^CB00, $BDDAT ;BUS BIT 0 IS WRITE GATE
1059 014106 001011 BNE 30$ ;
1060 014110 005037 001140 CLR $GDDAT
1061 014114 010037 001136 MOV R0, $BDADR
1062 014120 062737 000040 001136 ADD #RMMR2, $BDADR
1063 014126 104042 EMT 42
1064 014130 000545 BR 160$ ;FORMAT-CS FAILURE
1065
1066 ;STEP THE DATA TIMING SEQUENCER TO LOCATION 16 AND VERIFY WRITE
1067 ;GATE STAYS ON
1068 014132 30$:
1069 014132 012702 000016 MOV #14., R2 ;MAXMUM NUMBER OF CLOCK
1070 014136 40$:
1071 014136 016037 000040 001142 MOV RMMR2(R0), $BDDAT ;STORE RMMR2 AT $BDDAT
1072 014144 032737 000001 001142 BIT #BB00, $BDDAT ;
1073 014152 001014 BNE 50$ ;BRANCH IF WRITE GATE ON
1074 014154 005302 DEC R2
1075 014156 001367 BNE 40$
1076 014160 010037 001136 MOV R0, $BDADR
1077 014164 062737 000040 001136 ADD #RMMR2, $BDADR
1078 014172 012737 000001 001140 MOV #BB00, $GDDAT
1079 014200 104042 EMT 42
1080 014202 000520 BR 160$ ;FORMAT - CS FAILURE
1081 014204 50$:
1082
1083 014204 60$:
1084 014204 012760 055401 000024 MOV #MR1AAA!MCLK, RMMR1(R0) ;LOAD RMMR1
1085 014212 012760 051401 000024 MOV #MR1AAA, RMMR1(R0) ;LOAD RMMR1
1086 014220 016003 000024 MOV RMMR1(R0), R3 ;STORE RMMR1 AT R3
1087 014224 032703 000040 BIT #WC, R3
1088 014230 001765 BEQ 60$
1089
1090 ;PROM STROBE CAME ON-DECREMENT PROM STROBE COUNT
1091 014232 005302 DEC R2
1092 014234 001414 BEQ 80$
1093
1094 ;WAIT FOR PROM STROBE TO GO OFF, THEN REPEAT LOOP
1095 014236 70$:
1096 014236 012760 055401 000024 MOV #MR1AAA!MCLK, RMMR1(R0) ;LOAD RMMR1
1097 014244 012760 051401 000024 MOV #MR1AAA, RMMR1(R0) ;LOAD RMMR1
1098 014252 016003 000024 MOV RMMR1(R0), R3 ;STORE RMMR1 AT R3
1099 014256 032703 000040 BIT #WC, R3
1100 014262 001725 BEQ 40$
1101 014264 000764 BR 70$
1102
1103 ;VERIFY HEADER SYNC GENERATION
1104 ;WRITE DATA BIT IS INVERTED AT MAINTENANCE REGISTER
1105 ;FIRST, COUNT NUMBER OF ZERO BITS UNTIL FIRST ONE BIT
1106 014266 80$:
1107 014266 012702 000020 MOV #16., R2 ;MAX TIMES THRU LOOP

```

```

1108 014272 005003          CLR      R3
1109 014274          90$:      MOV      RMMR1(R0),R4      ;STORE RMMR1 AT R4
      014274 016004 000024    BIT      #MWD,R4
1110 014300 032704 000010    BEQ     110$              ;JUMP OUT OF LOOP WITH FIRST 1
1111 014304 001414          INC      R3                ;INCREMENT ZERO COUNT
1112 014306 005203          DEC     R2                ;DECREMENT MAX LOOP COUNT
1113 014310 005302          BNE    100$
1114 014312 001002          EMT    43
1115 014314 104043          BR     160$              ;HEADER SYNC ,CANT GET FIRST 1
1116 014316 000452
1117 014320          100$:    MOV      #MR1AAA.MCLK,RMMR1(R0) ;LOAD RMMR1
1118 014320 012760 055401 000024  MOV     #MR1AAA,RMMR1(R0) ;LOAD RMMR1
1119 014326 012760 051401 000024  BR     90$
1120 014334 000757
1121
1122          ;MAKE SURE THERE WERE AT LEAST 8 ZERO BITS IN HEADER SYNC
1123 014336          110$:    CMP      R3,#8.
1124 014336 020327 000010    BHIS   120$
1125 014342 103002          EMT    43
1126 014344 104043          BR     160$              ;HEADER SYNC
1127 014346 000436
1128
1129          ;SAMPLE AND STORE THE REST OF THE HEADER SYNC
1130 014350          120$:    MOV      #8.,R2              ;NUMBER OF SAMPLES
1131 014350 012702 000010    CLR    R3                ;HEADER SYNC
1132 014354 005003
1133 014356          130$:    MOV      RMMR1(R0),R4      ;STORE RMMR1 AT R4
1134 014356 016004 000024    ASL    R3
1135 014362 006303          BIT      #MWD,R4          ;SHIFT SAMPLE,IF SYNC BIT IS 1
1136 014364 032704 000010    BNE    140$              ;THEN SET SAMPLE INPUT
1137 014370 001002          BIS    #BIT0,R3
1138 014372 052703 000001    DEC     R2
1139 014376 005302          BEQ     150$
1140 014400 001407          MOV     #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1141 014402 012760 055401 000024  MOV     #MR1AAA,RMMR1(R0) ;LOAD RMMR1
1142 014410 012760 051401 000024  BR     130$              ;DO LOOP TIL R2=0
1143 014416 000757
1144
1145          ;VERIFY THE SYNC BIT STREAM IS 0000000010011000
1146 014420          150$:    MOV      #000230,$GDDAT
1147 014420 012737 000230 001140  MOV     R3,$BDDAT
1148 014426 010337 001142          CMP     $GDDAT,$BDDAT
1149 014432 023737 001140 001142  BEQ     160$
1150 014440 001401          EMT    44
1151 014442 104044
1152 014444
1153
1154

```

 ;*TEST 21 WRITE HEADER TEST

```

014444          TST21:    SCOPE
014444 000004          NOP
014446 000240          MOV     #STACK,SP      ;LOAD THE STACK POINTER
014450 012706 001100    MOV     $BASE,R0       ;R0 = UNIBUS ADDRESS
014454 013700 001276    MOV     TSTQUE,R1      ;R1 = POINTER TO DEVICE
014460 013701 001466

```

```

1155 014464 012737 000021 001226      MOV      #21,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1156                                     ;:SETUP THE REGISTER OUTPUT BUFFER FOR SUBROUTINES
1157 014472 012737 001466 001446      MOV      #822.,RMDCO     ;:LAST CYLINDER
1158 014500 013737 001334 001420      MOV      LSTRK,RMDAO     ;:SET LAST TRACK AND
1159 014506 112737 000036 001420      MOVVB    #30.,RMDAO     ;:LAST SECTOR
1160 014514 012737 010000 001444      MOV      #FMT16,RMOFO    ;:IN 16 BIT FORMAT
1161 014522 012702 054636          MOV      #BUFFER,R2     ;:BUFFER ADDRESS
1162 014526 010237 001416          MOV      R2,RMBAO       ;:BUFFER STARTS
1163                                     ;:STORE FIRST HEADER WORD
1164 014532 012712 150000          MOV      #150000,(R2)    ;:SET MF/UF BITS GOOD, 16 BIT FORMAT AND
1165 014536 052722 001466          BIS      #822.,(R2)+    ;:LAST CYLINDER
1166                                     ;:STORE SECOND HEADER WORD
1167 014542 013712 001334          MOV      LSTRK,(R2)     ;:SET LAST TRACK ADDRESS AND
1168 014546 112712 000036          MOVVB    #30.,(R2)     ;:SECTOR ADDRESS.
1169 014552 012737 177776 001414      MOV      #-2,RMWCO      ;:2 WORDS (2'S COMP)
1170 014560 012737 000063 001412      MOV      #WH!GO,RMCS10  ;:WRITE HEADER COMMAND
1171
1172 014566 004737 022432          JSR      PC,ENBSCH      ;:EXECUTE DATA COMMAND TO POINT WHERE
                                     ;:SEARCH IS ENABLED USING SUBROUTINE.
                                     ;:BRANCH TO 10$ IF NO ERROR
1173 014572 000403          BR      10$
1174 014574 104000          EMT
1175 014576 000137 015432          JMP      160$
1176 014602 004737 023304          JSR      PC,SCTCMP     ;:FORCE SECTOR COMPARE USING SUBROUTINE
1177 014606 000403          BR      20$           ;:BRANCH TO 20$ IF NO ERROR
1178 014610 104000          EMT
1179 014612 000137 015432          JMP      160$
1180                                     ;:STEP THE DATA TIMING SEQUENCER UNTIL 'HEADER AREA' COMES ON
1181 014616 012702 000017          20$:
1182 014622 012704 000041          30$:      MOV      #15.,R2      ;:ALLOW 15 MORE PROM STROBES
1183                                     ;:PULSE BIT CLOCK UNTIL PROM STROBE IS ON
1184 014626 012760 055401 000024      40$:      MOV      #MR1AAA!MCLK,RMMR1(R0) ;:LOAD RMMR1
1185 014634 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0) ;:LOAD RMMR1
1186 014642 016003 000024          MOV      RMMR1(R0),R3   ;:STORE RMMR1 AT R3
1187 014646 032703 000040          BIT      #WC,R3
1188 014652 001004          BNE     50$
1189 014654 005304          DEC     R4
1190 014656 001363          BNE     40$
1191 014660 000137 015432          JMP     160$           ;:COUNT EXHAUSTED
1192
1193                                     ;:SEE IF HEADER AREA CAME ON
1194 014664 032703 000200          50$:      BIT      #PHA,R3
1195 014670 001062          BNE     80$
1196 014672 005302          DEC     R2
1197 014674 001015          BNE     60$
1198
1199                                     ;:ERROR-HEADER AREA NEVER CAME ON
1200 014676 012737 000200 001140      MOV      #PHA,$GDDAT    ;:SETUP ERROR MESSAGE
1201 014704 005037 001142          CLR     $BDDAT
1202 014710 010037 001136          MOV     R0,$BDADR
1203 014714 062737 000024 001136      ADD     #RMMR1,$BDADR
    
```

```

1206 014722 104045          EMT      45
1207 014724 000137 015432    JMP      160$          ;SKIP REST OF TEST
1208
1209                          ;WAIT FOR PROM STROBE TO GO OFF
1210 014730          60$:
1211 014730 012760 055401 000024    MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1212 014736 012760 051401 000024    MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1213 014744 016003 000024          MOV      RMMR1(R0),R3          ;STORE RMMR1 AT R3
1214 014750 032703 000040          BIT      #WC,R3
1215 014754 001404          BEQ      70$
1216 014756 005304          DEC      R4
1217 014760 001363          BNE     60$
1218 014762 000137 015432    JMP      160$          ;COUNT EXHAUSTED
1219
1220                          ;VERIFY THE TAG BUS ONCE EVERY PROM STROBE
1221 014766          70$:
1222 014766 016037 000040 001142    MOV      RMMR2(R0),$BDDAT      ;STORE RMMR2 AT $BDDAT
1223 014774 042737 176000 001142    BIC      #^C1777,$BDDAT
1224 015002 022737 000001 001142    CMP      #BB00,$BDDAT          ;WRITE GATE SHOULD BE ON
1225 015010 001704          BEQ      30$              ;ALL ELSE OFF
1226 015012 010037 001136          MOV      R0,$BDADR            ;
1227 015016 062737 000040 001136    ADD      #RMMR2,$BDADR
1228 015024 012737 000001 001140    MOV      #BB00,$GDDAT
1229 015032 104042          EMT      42
1230 015034 000576          BR       160$
1231
1232                          ;HEADER AREA IS NOW ON CAN START SAMPLE
1233                          ;DATA AFTER 5 MORE BIT CLOCKS
1234 015036          80$:
1235 015036 012702 000005          MOV      #5,R2
1236 015042          81$:
1237 015042 012760 055401 000024    MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1238 015050 012760 051401 000024    MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1239 015060 005302          DEC      R2
1240 015060 001370          BNE     81$
1241
1242                          ;SAMPLE AND STORE THE TWO HEADER WORDS
1243 015062 005037 001174          CLR      $TMP0
1244 015066 005037 001176          CLR      $TMP1
1245 015072 012702 001174          MOV      #$TMP0,R2            ;R2 NUMBER OF WORDS/ADDRESS
1246 015076 012703 000020          90$:    MOV      #16.,R3            ;NUMBER OF BITS
1247 015102 005004          CLR      R4                  ;WORD STORAGE
1248 015104          100$:
1249 015104 016005 000024          MOV      RMMR1(R0),R5         ;STORE RMMR1 AT R5
1250 015110 006304          ASL      R4                  ;SHIFT WORD SAMPLE
1251 015112 032705 000010          BIT      #MWD,R5             ;SET BIT 0 IF WRITE BIT ON
1252 015116 001002          BNE     110$
1253 015120 052704 000001          BIS      #BIT0,R4
1254 015124          110$:
1255 015124 012760 055401 000024    MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1256 015132 012760 051401 000024    MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1257 015140 005303          DEC      R3
1258 015142 001360          BNE     100$
1259 015144 010422          MOV      R4,(R2)+            ;STORE WORD SAMPLE
1260 015146 022702 001200          CMP      #$TMP0+4,R2         ;DONE ?
1261 015152 101415          BLOS    120$              ;YES

```

```

1261
1262 015154 042705 177577 ;HEADER AREA SHOULD STILL BE ON FOR SECOND HEADER WORD
1263 015160 001346 BIC #^C<PHA>,R5
1264 015162 010037 BNE 90$ ;BRANCH IF ON
1265 015166 062737 001136 MOV RO,$BDADR
1266 015174 012737 000024 001136 ADD #RMMR1,$BDADR
1267 015202 104045 MOV #PHA,$GDDAT
1268 015204 000512 EMT 45
1269 BR 160$
1270
1271 015206 ;WAIT UNTIL ECRC SETS
1272 015206 005003 120$: CLR R3 ;CHECK NUMBER BITS DIFFER BET PHA AND ECRC
1273
1274 015210 125$: MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
015210 016005 000024 BIC #^C<ECRC>,R5 ;CHECK OUT THE REST BITS
1275 015214 042705 176777 BNE 127$ ;BRANCH IF ECRC ON
1276 015220 001026 MOV #MR1AAA:MCLK,RMMR1(RO) ;LOAD RMMR1
1277 015222 012760 055401 000024 MOV #MR1AAA,RMMR1(RO) ;LOAD RMMR1
1278 015230 012760 051401 000024 INC R3 ;TOTAL NUMBER OF BIT DIFF
1279 015236 005203 CMP #16.,R3 ;OVER ONE WORD ?
1280 015240 022703 000020 BHI 125$ ;BRANCH IF NOT
1281 015244 101361 MOV RO,$BDADR
1282 015246 010037 001136 ADD #RMMR1,$BDADR
1283 015252 062737 000024 001136 MOV #ECRC,$GDDAT
1284 015260 012737 001000 001140 CLR $BDAT ;CLEAR THE EXP
1285 015266 005037 001142 EMT 46
1286 015272 104046 BR 160$ ;EXIT
1287 015274 000456
1288
1289 ;SAMPLE AND STORE CRC WORD
1290 015276 127$: MOV #16.,R3
1291 015276 012703 000020 CLR R4
1292 015302 005004
1293 015304 130$: MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
015304 016005 000024 ASL R4 ;SHIFT WORD SAMPLE
1294 015310 006304 BIT #RWD,R5 ;SET BIT 0 IF WRITE BIT ON
1295 015312 032705 000010 BNE 140$
1296 015316 001002 BIS #BIT0,R4
1297 015320 052704 000001
1298 015324 140$: MOV #MR1AAA:MCLK,RMMR1(RO) ;LOAD RMMR1
015324 012760 055401 000024 MOV #MR1AAA,RMMR1(RO) ;LOAD RMMR1
1299 015332 012760 051401 000024 DEC R3
1300 015340 005303 BNE 130$
1301 015342 001360 MOV R4,(R2)+
1302 015344 010422
1303
1304 ;VERIFY 3 HEADER WORDS
1305 ;NOTE THAT DATA WAS STORED IN THE REVERSE
1306 ;ORDER FROM WHICH IT WAS WRITTEN
1307 015346 022737 066313 001174 CMP #066313,$TMP0 ;CORRECT FIRST HDR WRD ?
1308 015354 001025 BNE 150$ ;NO !!
1309
1310 ;VERIFY 2ND AND 3RD HEADER WORDS FOR AN RM05
1311 015356 032737 010000 001334 BIT #TA16,LSTRK ;IS DEVICE AN RM05 ?
1312 015364 001411 BEQ 145$ ;NO !!
1313 015366 022737 074110 001176 CMP #074110,$TMP1 ;CORRECT SECOND HDR WRD ?
1314 015374 001015 BNE 150$ ;NO !!

```

```

1315 015376 022737 167073 001200      CMP      #167073,$TMP2      ;CORRECT HEADER CRC ?
1316 015404 001011                      BNE      150$              ;NO ..
1317 015406 000411                      BR       160$
1318
1319      ;VERIFY 2ND AND 3RD HEADER WORDS FOR AN RM03
1320 015410      145$:
1321 015410 022737 074040 001176      CMP      #074040,$TMP1      ;CORRECT SECOND HDR WRD ?
1322 015416 001004                      BNE      150$              ;NO !!
1323 015420 022737 067510 001200      CMP      #067510,$TMP2      ;CORRECT HEADER CRC ?
1324 015426 001401                      BEQ      160$              ;YES !!
1325 015430      150$:
1326 015430 104047                      EMT      47
1327 015432      160$:
1328                      ;END OF SUB TEST
1329
:*****
:*TEST 22      HEADER COMPARE TEST
:*****

015432      TST22:
015432 000004                      SCOPE                      ;SCOPE CALL
015434 000240                      NOP
015436 012706 001100      MOV      #STACK,SP          ;LOAD THE STACK POINTER
015442 013700 001276      MOV      $BASE,R0           ;R0 = UNIBUS ADDRESS
015446 013701 001466      MOV      TSTQUE,R1          ;R1 = POINTER TO DEVICE
015452 012737 000022 001226      MOV      #22,$TSTN          ;:SET TEST NUMBER IN APT MAIL BOX

1330
1331      ;LOAD REGISTER OUTPUT BUFFER FOR READ HEADER COMMAND
1332 015460 012737 001466 001446      MOV      #822.,RMDCO        ;LAST CYLINDER
1333 015466 013737 001334 001420      MOV      LSTRK,RMDAO        ;SET LAST TRACK AND
1334 015474 112757 000036 001420      MOV      #30.,RMDAO        ;LAST SECTOR
1335 015502 012737 054636 001416      MOV      #BUFFER,RMBAD      ;DATA ADDRESS
1336 015510 012737 177700 001414      MOV      #-2,RMWCO         ;2 WORDS (2'S COMP)
1337 015516 012737 010000 001444      MOV      #FMT16,RMFO        ;16 BIT MODE
1338 015524 012737 000073 001412      MOV      #RH!GO,RMCS10      ;READ HEADER AND DATA FUN
1339
1340 015532 004737 022432                      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      230$
                                ;SKIP REST OF TEST IF ERROR
1341 015542 000137 016700
1342 015546      10$:
1343 015546 004737 023304                      JSR      PC,SCTCMP          ;FORCE SECTOR COMPARE USING SUBROUTINE
                                BR       20$
                                ;BRANCH TO 20$ IF NO ERROR
                                EMT
                                JMP      230$
1344 015556 000137 016700
1345
1346      ;READ GATE SHOULD BE OFF FOR 3 PROM STROBES
1347 015562      20$:
1348 015562 010037 001136                      MOV      R0,$BDADR          ;SETUP ERROR MESSAGE
1349 015566 062737 000040 001136      ADD      #RMMR2,$BDADR
1350 015574 012702 000004                      MOV      #4,R2              ;R2=NUMBER OF PROM STROBES
1351 015600 012704 000021      30$:      MOV      #17.,R4              ;MAX BIT CLOCKS
1352
1353      ;CLOCK BIT CLOCK UNTIL PROM STROBE COMES ON
1354 015604      40$:
1355 015604 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1356 015612 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0)    ;LOAD RMMR1

```


1357 015620 016003 000024
 1358 015624 032703 000040
 1359 015630 001004
 1360 015632 005304
 1361 015634 001363
 1362 015636 000137 016700
 1363
 1364
 1365 015642
 1366 015642 016003 000040
 1367 015646 042703 176000
 1368 015652 001407
 1369 015654 010337 001142
 1370 015660 005037 001140
 1371 015664 104050
 1372 015666 000137 016700
 1373
 1374
 1375 015672
 1376 015672 012760 055401 000024
 1377 015700 012760 051401 000024
 1378 015706 016003 000024
 1379 015712 032703 000040
 1380 015716 001404
 1381 015720 005304
 1382 015722 001363
 1383 015724 000137 016700
 1384
 1385
 1386 015730
 1387 015730 005302
 1388 015732 001322
 1389
 1390
 1391
 1392
 1393 015734 012702 000006
 1394 015740 012703 000021
 1395
 1396
 1397 015744
 1398 015744 012760 055401 000024
 1399 015752 012760 051401 000024
 1400 015760 016004 000024
 1401 015764 032704 000040
 1402 015770 001004
 1403 015772 005303
 1404 015774 001363
 1405 015776 000137 016700
 1406
 1407
 1408 016002
 1409 016002 016004 000040
 1410 016006 042704 176000
 1411 016012 022704 000002
 1412 016016 001410
 1413 016020 010437 001142

```

MOV RMMR1(R0),R3 ;STORE RMMR1 AT R3
BIT #WC,R3
BNE 50$ ;BRANCH WHEN PROM STROBE ON
DEC R4
BNE 40$
JMP 230$

:VERIFY READ GATE IS OFF AND TAG BUS IS ZERO
50$:
MOV RMMR2(R0),R3 ;STORE RMMR2 AT R3
BIC #^C1777,R3
BEQ 60$ ;BRANCH IF TAG BUS ZERO
MOV R3,$BDDAT
CLR $GDDAT
EMT 50
JMP 230$ ;READ HEADER FUNCTION

:CLOCK BIT CLOCK TIL PROM STROBE GOES OFF
60$:
MOV #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
MOV #MR1AAA,RMMR1(R0) ;LOAD RMMR1
MOV RMMR1(R0),R3 ;STORE RMMR1 AT R3
BIT #WC,R3
BEQ 70$ ;BRANCH IF PROM STROBE OFF
DEC R4
BNE 60$
JMP 230$

:CONTINUE FOR 3 BIT CYCLE TOTAL (LOC 0 NOT SAMPLED)
70$:
DEC R2
BNE 30$

:READ GATE SHOULD COME ON WITH NEXT PROM STROBE AND STAY
:ON FOR 7 CYCLES (AND MORE)
80$:
MOV #6,R2
MOV #17.,R3

:CLOCK BIT CLOCK TIL PROM STROBE SETS
90$:
MOV #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
MOV #MR1AAA,RMMR1(R0) ;LOAD RMMR1
MOV RMMR1(R0),R4 ;STORE RMMR1 AT R4
BIT #WC,R4
BNE 100$ ;BRANCH WHEN PROM STROBE ON
DEC R3
BNE 90$
JMP 230$

:VERIFY READ GATE ON AND REST OF TAG BUS OFF
100$:
MOV RMMR2(R0),R4 ;STORE RMMR2 AT R4
BIC #^C1777,R4
CMP #B01,R4
BEQ 110$ ;BRANCH IF READ GATE ON
MOV R4,$BDDAT
  
```

```

1414 016024 012737 000002 001140      MOV      #BB01,$GDDAT
1415 016032 104050                EMT      50
1416 016034 000137 016700      JMP      230$          ;READ HEADER FUNCTION
1417
1418                                ;CLOCK BIT CLOCK TIL PROM STROBE GOES OFF
1419 016040      110$:
1420 016040 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1421 016046 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1422 016054 016004 000024      MOV      RMMR1(R0),R4           ;STORE RMMR1 AT R4
1423 016060 032704 000040      BIT      #WC,R4
1424 016064 001404                BEQ      120$          ;BRANCH WHEN PROM STROBE OFF
1425 016066 005303                DEC      R3
1426 016070 001363                BNE     110$
1427 016072 000137 016700      JMP      230$
1428
1429                                ;CONTINUE FOR TOTAL OF 7 CYCLES
1430 016076      120$:
1431 016076 005302                DEC      R2
1432 016100 001317                BNE     80$
1433
1434                                ;LOOKING FOR SYNC SHOULD SET WITH NEXT PROM STORBE
1435 016102 012702 000021      MOV      #17.,R2             ;MAX BIT CLOCKS
1436
1437                                ;CLOCK BIT CLOCK TIL PROM STROBE SETS
1438 016106      130$:
1439 016106 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1440 016114 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1441 016122 016003 000024      MOV      RMMR1(R0),R3           ;STORE RMMR1 AT R3
1442 016126 032703 000040      BIT      #WC,R3
1443 016132 001004                BNE     140$          ;BRANCH WHEN PROM STROBE ON
1444 016134 005302                DEC      R2
1445 016136 001363                BNE     130$
1446 016140 000137 016700      JMP      230$
1447
1448                                ;VERIFY 'PLFS' IS NOW ON
1449 016144      140$:
1450 016144 042703 175777      BIC      #^CPLFS,R3
1451 016150 001015                BNE     150$
1452 016152 010337 001142      MOV      R3,$BDDAT
1453 016156 012737 002000 001140      MOV      #PLFS,$GDDAT
1454 016164 010037 001136      MOV      R0,$BDADR
1455 016170 062737 000024 001136      ADD      #RMMR1,$BDADR
1456 016176 104035                EMT      35
1457 016200 000137 016700      JMP      230$
1458 016204      150$:
1459 016204 012703 000005      MOV      #5,R3
1460
1461                                ;WITH PLFS ON, WORD COUNT INHIBIT WILL SET IN 5 BIT CLOCKS
1462 016210      155$:
1463 016210 012760 055401 000024      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1464 016216 012760 051401 000024      MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
1465 016224 005303                DEC      R3
1466 016226 001370                BNE     155$
1467
1468                                ;SIMULATE THE SYNC PATTERN BEING READ
1469 016230 012702 000031      MOV      #31,R2             ;SYNC PATTERN=00011001
1470 016234 012703 000010      MOV      #8.,R3             ;8 BITS IN PATTERN

```

```

1471 016240
1472 016240 012737 055401 001436 160$: MOV #MR1AAA.MCLK,RMMR10
1473 016246 000241 CLC
1474 016250 006002 ROR R2
1475 016252 103003 BCC 165$
1476 016254 052737 002000 001436 BIS #MRD,RMMR10
1477 016262
1478 016262 013760 001436 000024 165$: MOV RMMR10,RMMR1(RO) ;LOAD RMMR1
1479 016270 042737 004000 001436 BIC #MCLK,RMMR10
1480 016276 013760 001436 000024 MOV RMMR10,RMMR1(RO) ;LOAD RMMR1
1481 016304 005303 DEC R3 ;CONTINUE TIL SHIFT COUNT
1482 016306 001354 BNE 160$ ;IS ZERO
1483
1484 ;SIMULATE THE HEADER BEING READ
1485 016310 012702 001174 MOV #STMP0,R2
1486 016314 012737 150000 001174 MOV #150000,$TMP0 ;SET MF/UF BITS GOOD, 16 BIT FORMAT AND
1487 016322 052737 001466 001174 BIS #822,$TMP0 ;LAST CYLINDER
1488 016330 013737 001334 001176 MOV LSTRK,$TMP1 ;SET LAST TRACK AND
1489 016336 112737 000036 001176 MOVB #30,$TMP1 ;LAST SECTOR
1490 016344 012737 011366 001200 MOV #011366,$TMP2 ;GET CRC PATTERN
1491 016352 032737 010000 001334 BIT #TA16,LSTRK ;IS DEVICE AN RM05 ?
1492 016360 001403 BFO 170$ ;NO !!
1493 016362 012737 156167 001200 MOV #156167,$TMP2 ;YES, ADJUST CRC PATTERN
1494 016370
1495 016370 012703 000020 170$: MOV #16.,R3 ;NUMBER OF BITS EACH WORD
1496 016374 012204 MOV (R2)+,R4 ;HEADER WORD 1,2 OR 3
1497 016376
1498 016376 012737 055401 001436 175$: MOV #MR1AAA!MCLK,RMMR10
1499 016404 000241 CLC
1500 016406 006004 ROR R4
1501 016410 103003 BCC 180$
1502 016412 052737 002000 001436 BIS #MRD,RMMR10
1503 016420
1504 016420 013760 001436 000024 180$: MOV RMMR10,RMMR1(RO) ;LOAD RMMR1
1505 016426 042737 004000 001436 BIC #MCLK,RMMR10
1506 016434 013760 001436 000024 MOV RMMR10,RMMR1(RO) ;LOAD RMMR1
1507 016442 005303 DEC R3 ;SHIFT OUT 16 BITS
1508 016444 001354 BNE 175$
1509 016446 020227 001200 CMP R2,#$TMP0+4 ;ALL DONE ?
1510 016452 101746 BLOS 170$ ;BRANCH IF NOT
1511 ;LOOKING FOR SYNC SHOULD COME ON WITHIN 6 STROBES
1512 016454 012702 000006 MOV #6,R2
1513 016460 012703 000021 185$: MOV #17.,R3
1514
1515 ;CLOCK UNTIL PROM STROBE ON
1516 016464 190$:
1517 016464 012760 055401 000024 MOV #MR1AAA!MCLK,RMMR1(RO) ;LOAD RMMR1
1518 016472 012760 051401 000024 MOV #MR1AAA,RMMR1(RO) ;LOAD RMMR1
1519 016500 016004 000024 MOV RMMR1(RO),R4 ;STORE RMMR1 AT R4
1520 016504 032704 000040 BIT #WC,R4
1521 016510 001003 BNE 195$
1522 016512 005303 DEC R3
1523 016514 001363 BNE 190$
1524 016516 000470 BR 230$
1525
1526 ;SEE IF 'PLFS' IS ON
  
```

1527 016520
 1528 016520 042704 175777
 1529 016524 001034
 1530
 1531
 1532 016526 005302
 1533 016530 001014
 1534 016532 010437 001142
 1535 016536 012737 002000 001140
 1536 016544 010037 001136
 1537 016550 062737 000024 001136
 1538 016556 104035
 1539 016560 000447
 1540
 1541
 1542 016562
 1543 016562 012760 055401 000024
 1544 016570 012760 051401 000024
 1545 016576 016004 000024
 1546 016602 032704 0C0040
 1547 016606 001724
 1548 016610 005303
 1549 016612 001363
 1550 016614 000431
 1551
 1552
 1553 016616
 1554 016616 016037 000014 001142
 1555 016624 042737 177157 001142
 1556 016632 001411
 1557 016634 005037 001140
 1558 016640 010037 001136
 1559 016644 062737 000014 001136
 1560 016652 104051
 1561 016654 000411
 1562
 1563
 1564 016656
 1565 016656 023737 054636 001174
 1566 016664 001004
 1567 016666 023737 054640 001176
 1568 016674 001401
 1569
 1570
 1571 016676
 1572 016676 104052
 1573 016700
 1574
 1575

```

195$:
      BIC      #*CPLFS,R4
      BNE      210$

:CONTINUE IF LESS THAN 6 PROM STROBES
      DEC      R2
      BNE      200$
      MOV      R4,$BDDAT      ;SETUP ERROR
      MOV      #PLFS,$GDDAT
      MOV      R0,$BDADR
      ADD      #RMMR1,$BDADR
      EMT      35
      BR       230$      ;HEADER

:CLOCK UNTIL PROM STROBE OFF
200$:
      MOV      #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
      MOV      #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
      MOV      RMMR1(R0),R4      ;STORE RMMR1 AT R4
      BIT      #WC,R4
      BEQ      185$
      DEC      R3
      BNE      200$
      BR       230$

:VERIFY NO HEADER ERROR IS SET
210$:
      MOV      RMER1(R0),$BDDAT      ;STORE RMER1 AT $BDDAT
      BIC      #*C<HCRC!HCE!FER>,$BDDAT
      BEQ      220$
      CLR      $GDDAT
      MOV      R0,$BDADR
      ADD      #RMER1,$BDADR
      EMT      51
      BR       230$      ;HEADER ERROR SET

:VERIFY DATA IN MEMORY OK
220$:
      CMP      BUFFER,$TMP0      ;COMPARE 1ST HEADER WORD
      BNE      225$
      CMP      BUFFER+2,$TMP1      ;COMPARE 2ND HEADER WORD
      BEQ      230$

:REPORT ERROR IN ONE OR MORE HEADER WORDS
225$:
      EMT      52

230$:

:*****
:*TEST 23      ECC GENERATION TEST
:*****

TST23:
      SCOPE
      NOP      ;SCOPE CALL
      MOV      #STACK,SP      ;LOAD THE STACK POINTER
      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
  
```

```

016714 013701 001466      MOV    TSTQUE,R1      ;R1 = POINTER TO DEVICE
016720 012737 000023 001226  MOV    #23,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

1576
1577      ;SETUP REGISTER OUTPUT BUFFER FOR WRITE DATA CAMOMMAND
1578 016726 012737 001466 001446  MOV    #822.,RMDCO    ;LAST CYLINDER
1579 016734 013737 001334 001420  MOV    LSTRK,RMDAO    ;SET LAST TRACK AND
1580 016742 112737 000036 001420  MOV    #30.,RMDAO    ;LAST SECTOR
1581 016750 012737 012000 001444  MOV    #FMT16!HCI,RMOFO ;16 BIT FORMAT, IGNORE HEADER
1582 016756 012702 054636      MOV    #BUFFER,R2
1583 016762 010237 001416      MOV    R2,RMBAO      ;DATA ADDRESS
1584 016766 012703 177400      MOV    #-256.,R3     ;256. WORDS (2'S COMP)
1585 016772 010337 001414      MOV    R3,RMWCO
1586 016776 012737 000061 001412  MOV    #WD!GO,RMCS10 ;WRITE DATA COMMAND
1587
1588      ;FILL THE DATA BUFFER WITH ALL ONES
1589 017004 012704 177777      MOV    #177777,R4    ;DATA PATTERN ALL ONES
1590 017010 012703 000400      MOV    #256.,R3      ;ONE SECTOR DATA FIELD
1591 017014 010422 30$:     MOV    R4,(R2)+
1592 017016 005303      DEC    R3
1593 017020 001375      BNE    10$
1594
1595 017022 004737 022432      JSR    PC,ENBSCH     ;EXECUTE DATA COMMAND TO POINT WHERE
                        ;SEARCH IS ENABLED USING SUBROUTINE.
                        ;BRANCH TO 20$ IF NO ERROR
                        '
017026 000403      BR    20$
017030 104000      EMT
1596 017032 000137 020066      JMP    400$
1597 017036 20$:
1598 017036 004737 023304      JSR    PC,SCTCMP    ;FORCE SECTOR COMPARE USING SUBROUTINE
                        ;BRANCH TO 30$ IF NO ERROR
017042 000403      BR    30$
017044 104000      EMT
1599 017046 000137 020066      JMP    400$
1600 017052 30$:
1601 017052 004737 023412      JSR    PC,SETLFS    ;SET 'LOOKING FOR SYNC' USING SUBROUTINE
                        ;BRANCH TO 40$ IF NO ERROR
017056 000403      BR    40$
017060 104000      EMT
1602 017062 000137 020066      JMP    400$
1603 017066 40$:
1604 017066 004737 023574      JSR    PC,CLKSNC    ;CLOCK THE SYNC PATTERN USING SUBROUTINE
                        ;BR TO 50$ IF NO ERROR
017072 000403      BR    50$
017074 104000      EMT
1605 017076 000137 020066      JMP    400$
1606
1607      ;HEADER COMPARE IS INHIBITED FOR THIS TEST
1608      ;CLOCK THE DATA TIMING SEQUENCER AND VERIFY THE FOLLOWING WAVEFORMS
1609      ;FOR EACH LEADING EDGD OF PROM STROBE
1610      ;   HEADER ARES, READ GATE (DATA TIMING SEQUENCER=200 OCTAL)
1611      ;   ENABLE CRC OUT
1612      ;   WRITE GATE
1613
1614 017102 012702 017634 50$:     MOV    #230$,R2      ;POINTER TO WAVE FORM TABLE
1615
1616      ;CLOCK BIT CLOCK (MCLK) UNTIL PROM STROBE COMES ON
1617 017106 70$:
1618 017106 004737 017654      JSR    PC,300$
1619
1620      ;PROM STROBE IS ON - VERIFY WAVEFORM
1621 017112 80$:

```

```

1622 017112 016037 000024 001142      MOV      RMMR1(R0), $BDDAT      ;STORE RMMR1 AT $BDDAT
1623 017120 042737 176157 001142      BIC      #^C<ECRC.PDA.PHA.EECC>, $BDDAT
1624 017126 012237 001140      MOV      (R2)+, $GDDAT
1625 017132 023737 001140      CMP      $GDDAT, $BDDAT
1626 017140 001410      BEQ      90$      ;BRANCH IF RMMR1 OK
1627
1628      ;ERROR - THE DATA TIMING SEQUENCER OUTPUT IS ONCORRECT
1629 017142 010037 001136      MOV      RO, $BDADR
1630 017146 062737 000024 001136      ADD      #RMMR1, $BDADR
1631 017154 104054      EMT      54
1632 017156 000137 020066      JMP      400$
1633
1634      ;VERIFY THE TAB BUS
1635 017162 90$:
1636 017162 016037 000040 001142      MOV      RMMR2(R0), $BDDAT      ;STORE RMMR2 AT $BDDAT
1637 017170 042737 176000 001142      BIC      #^C1777, $BDDAT
1638 017176 012237 001140      MOV      (R2)+, $GDDAT
1639 017202 023737 001140 001142      CMP      $GDDAT, $BDDAT
1640 017210 001410      BEQ      100$      ;BRANCH IF TAG BUS OK
1641
1642      ;ERROR- TAG BUS IS WRONG
1643 017212 010037 001136      MOV      RO, $BDADR
1644 017216 062737 000040 001136      ADD      #RMMR2, $BDADR
1645 017224 104053      EMT      53
1646 017226 000137 020066      JMP      400$
1647
1648      ;CLOCK BIT CLOCK TIL PROM STROBE RESETS
1649 017232 100$:
1650 017232 004737 017762      JSR      PC, 350$
1651
1652      ;CONTINUE CHECKING THROUGH 4 PROM CYCLES
1653 017236 110$:
1654 017236 020227 017652      CMP      R2, #230$+16
1655 017242 103721      BLO      70$
1656
1657      ;VERIFY THAT DATA AREA COMES ON WITHIN 9 PROM STROBES
1658 017244 012702 000011      MOV      #9., R2      ;R2=9 STROBES
1659
1660      ;CLOCK PROM STROBE ON
1661 017250 130$:
1662 017250 004737 017654      JSR      PC, 300$
1663
1664      ;PROM STROBE ON CKECK IF DATA AREA ON
1665 017254 140$:
1666 017254 016004 000024      MOV      RMMR1(R0), R4      ;STORE RMMR1 AT R4
1667 017260 042704 177377      BIC      #^CPDA, R4
1668 017264 001020      BNE      160$
1669
1670      ;CLOCK PROM STROBE OFF
1671 017266 150$:
1672 017266 004737 017762      JSR      PC, 350$
1673 017272 005302      DEC      R2
1674 017274 001365      BNE      130$
1675
1676      ;ERROR-DATA AREA DIDN'T COME ON
1677 017276 012737 000400 001140      MOV      #PDA, $GDDAT      ;SETUP ERROR MESSAGE
1678 017304 005037 001142      CLR      $BDDAT
  
```

```

1679 017310 010037 001136          MOV      RO,$BDADR
1680 017314 062737 C00024 001136      ADD      #RMMR1,$BDADR
1681 017322 104055          EMT      55
1682 017324 000541          BR       220$
1683
1684          ;VERIFY THAT DATA AREA IS ON FOR 256 CYCLES
160$:
1685 017326          MOV      #256.,R2
1686 017326 012702 000400
1687
1688          ;VERIFY THAT DATA AREA IS ON AND ECC IS OFF
170$:
1689 017332          MOV      RMMR1(RO),R3 ;STORE RMMR1 AT R3
1690 017332 016003 000024          BIC      #^C<PDA!ECC>,R3
1691 017336 042703 177357          CMP      #PDA,R3
1692 017342 022703 000400          BEQ     180$ ;DATA AREA IS ON
1693 017346 001414          MOV     R3,$BDDAT
1694 017350 010337 001142          MOV     #PDA,$GDDAT
1695 017354 012737 000400 001140          MOV     RO,$BDADR
1696 017362 010037 001136          ADD     #RMMR1,$BDADR
1697 017366 062737 C00024 001136      EMT     55
1698 017374 104055          BR     220$
1699 017376 000514
1700
1701          ;CLOCK PROM STROBE OFF
180$:
1702 017400          JSR     PC,350$
1703 017400 004737 017762
1704
1705          ;CLOCK PROM STROBE ON
1706 017404 004737 017654          JSR     PC,300$
1707
1708          ;CONTINUE TIL COUNT ZERO
1709 017410 005302          DEC     R2
1710 017412 001347          BNE     170$
1711
1712          ;ECC SHOULD BE ENABLED WITHIN 4 CLOCK BITS
182$:
1713 017414 005002          CLR     R2
1714 017416          MOV     RMMR1(RO),R3 ;STORE RMMR1 AT R3
1715 017422 042703 177357          BIC     #^C<PDA!ECC>,R3
1716 017426 022703 000020          CMP     #ECC,R3
1717 017432 001426          BEQ     190$
1718 017434 005202          INC     R2
1719 017436 012760 055401 000024          MOV     #R1AAA!MCLK,RMMR1(RO) ;LOAD RMMR1
1720 017444 012760 051401 000024          MOV     #R1AAA,RMMR1(RO) ;LOAD RMMR1
1721 017452 022702 000007          CMP     #7,R2
1722 017456 101357          BHI     182$
1723 017460 010337 001142          MOV     R3,$BDDAT
1724 017464 012737 000020 001140          MOV     #ECC,$GDDAT
1725 017472 010037 001136          MOV     RO,$BDADR
1726 017476 062737 000024 001136      ADD     #RMMR1,$BDADR
1727 017504 104056          EMT     56
1728 017506 000450          BR     220$
1729
1730          ;SAMPLE ECC CHARACTER FOR 32 BITS
190$:
1731 017510          MOV     #$TMP0,R2
1732 017510 012702 001174          CLR     R3
1733 017514 005003          MOV     #16.,R4
1734 017516 012704 000020

```

```

1735 017522          200$: MOV RMMR1(R0),R5 ;STORE RMMR1 AT R5
      017522 016005 000024 ASL R3
1736 017526 006303      BIT #MWD,R5
1737 017530 032705 000010 BNE 205$ ;BRANCH IF ECC BIT OFF (MWD INVERTED)
1738 017534 001002      BIS #BIT0,R3
1739 017536 052703 000001
1740 017542          205$: MOV #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
1741 017542 012760 055401 000024 MOV #MR1AAA,RMMR1(R0) ;LOAD RMMR1
1742 017550 012760 051401 000024 DEC R4
1743 017556 005304      BNE 200$
1744 017560 001360      MOV R3,(R2)+
1745 017562 010322      CMP #STMP0+4,R2
1746 017564 022702 001200 BHI 195$
1747 017570 101351
1748
1749          :VERIFY ECC CHARACTER GENERATED
1750 017572 022737 131177 001174 CMP #131177,$TMP0
1751 017600 001004      BNE 210$
1752 017602 022737 175154 001176 CMP #175154,$TMP1
1753 017610 001407      BEQ 220$
1754 017612 012737 131177 001200 210$: MOV #131177,$TMP2
1755 017620 012737 175154 001202 MOV #175154,$TMP3
1756 017626 104061      EMT 61
1757
1758 017630 000137 020066 220$: JMP 400$ ;GREAT ESCAPE
1759
1760 017634 000200 230$: .WORD PHA
1761 017636 000002      .WORD BB01
1762 017640 000200      .WORD PHA
1763 017642 000002      .WORD BB01
1764 017644 000000      .WORD 0
1765 017646 000002      .WORD 2
1766 017650 000000      .WORD 0
1767 017652 000000      .WORD 0
1768
1769 017654 012737 000021 017756 300$: MOV #17.,320$ ;CLOCK PROM STROBE ON
1770 017662          305$: MOV #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
      017662 012760 055401 000024 MOV #MR1AAA,RMMR1(R0) ;LOAD RMMR1
1771 017670 012760 051401 000024 MOV RMMR1(R0),330$ ;STORE RMMR1 AT 330$
1772 017676 016037 000024 017760 BIC #^CWC,330$
1773 017704 042737 177737 017760 BNE 310$ ;EXIT IF PROM STROBE ON
1774 017712 001020      DEC 320$
1775 017714 005337 017756      BNE 305$
1776 017720 001360      MOV 320$,$BDDAT ;SETUP ERROR MESSAGE
1777 017722 013737 017756 001142 MOV #WC,$GDDAT
1778 017730 012737 000040 001140 MOV R0,$BDADR
1779 017736 010037 001136      ADD #RMMR1,$BDADR
1780 017742 062737 000024 001136 EMT 30
1781 017750 104030      BR 400$ ;
1782 017752 000445          310$: RTS PC
1783 017754 000207          320$: .WORD 0 ;MAX BIT COUNT
1784 017756 000000          330$: .WORD 0 ;RMMR1 CONTENTS
1785 017760 000000
1786
1787 017762 012737 000021 020062 350$: MOV #17.,370$ ;CLOCK PROM STROBE OFF
1788
1789 017770          355$:
  
```



```

1790 017770 012760 055401 000024 MOV #MR1AAA.MCLK,RMMR1(R0) ;LOAD RMMR1
1791 017776 012760 051401 000024 MOV #MR1AAA,RMMR1(R0) ;LOAD RMMR1
1792 020004 016037 000024 020064 MOV RMMR1(R0),380$ ;STORE RMMR1 AT 380$
1793 020012 042737 177737 020064 BIC #^CWC,380$
1794 020020 001417 BEQ 360$
1795 020022 005337 020062 DEC 370$
1796 020026 001360 BNE 355$
1797 020030 013737 020064 001142 MOV 380$, $BDDAT
1798 020036 005037 001140 CLR $GDDAT
1799 020042 010037 001136 MOV R0, $BDADR
1800 020046 062737 000024 001136 ADD #RMMR1,$BDADR
1801 020054 104062 EMT 62
1802 020056 000403 BR 400$
1803 020060 000207 360$: RTS PC
1804 020062 000000 370$: .WORD 0 ;MAX BIT COUNT
1805 020064 000000 380$: .WORD 0 ;CONTENTS OF RMMR1
1806
1807 020066 400$: ;SUB-TEST EXIT POINT
1808 020066 000240 NOP
1809
1810
:::*****
:*TEST 24 ECC DETECTION TEST
:::*****
TST24:
020070 000004 SCOPE ;SCOPE CALL
020072 000240 NOP
020074 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
020100 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
020104 013701 001456 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
020110 012737 000024 001226 MOV #24,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1811
1812 ;SETUP REGISTER OUTPUT BUFFER FOR READ DATA COMMAND
1813 020116 012737 001466 001446 MOV #822.,RMDCO ;LAST CYLINDER
1814 020124 013737 001334 001420 MOV LSTRK,RMDAO ;SET LAST TRACK AND
1815 020132 112737 000036 001420 MOV #30.,RMDAO ;LAST SECTOR
1816 020140 012737 012000 001444 MOV #FMT16!HCI,RMOFO ;INHIBIT HEADER COMPARE
1817 020146 012702 054636 MOV #BUFONE,R2
1818 020152 010237 001416 MOV R2,RMBAD
1819 020156 012737 177400 001414 MOV #-256.,RMWCO ;256 WORDS (2'S COMP)
1820 020164 012737 000071 001412 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
1821
1822 020172 20$: JSR PC,ENBSCH ;EXECUTE DATA COMMAND TO POINT WHERE
020172 004737 022432 ;SEARCH IS ENABLED USING SUBROUTINE.
020176 000403 BR 30$ ;BRANCH TO 30$ IF NO ERROR
020200 104000 EMT
1823 020202 000137 021054 JMP 190$
1824
1825 020206 30$: JSR PC,SCTCMP ;FORCE SECTOR COMPARE USING SUBROUTINE
020206 004737 023304 BR 40$ ;BRANCH TO 40$ IF NO ERROR
020212 000403 EMT
020214 104000 JMP 190$
1826 020216 000137 021054
1827
1828 020222 40$:

```

```

020222 004737 023412      JSR    PC,SETLFS      ;SET 'LOOKING FOR SYNC' USING SUBROUTINE
020226 000403              BR     50$           ;BRANCH TO 50$ IF NO ERROR
020230 104000              EMT
1829 020232 000137 021054      JMP    190$
1830
1831 020236              50$:
020236 004737 023574      JSR    PC,CLKSNC     ;CLOCK THE SYNC PATTERN USING SUBROUTINE
020242 000403              BR     55$           ;BR TO 55$ IF NO ERROR
020244 104000              EMT
1832 020246 000137 021054      JMP    190$
1833
1834
1835 ;HEADER COMPARE IS INHIBITED FOR THIS TEST
1836 020252 012702 000002      55$: MOV    #2,R2        ;ALLOW 2 PASSES THRU LOOP
1837 020256 000405              BR     70$
1838 020260 004737 021166      60$: JSR    PC,350$     ;RESET PROM STROBE
1839 020264 004737 021060      JSR    PC,300$     ;SET PROM STROBE
1840 020270 000240              NOP
1841 020272              70$:
020272 016003 0C0024      MOV    RMMR1(R0),R3 ;STORE RMMR1 AT R3
1842 020276 032703 002000      BIT    #PLFS,R3
1843 020302 001421              BEQ    80$           ;BRANCH IF LOOKING FOR SYNC OFF
1844 020304 005302              DEC    R2
1845 020306 001364              BNE    60$
1846
1847 ;ERROR-LOOKING FOR SYNC DID NOT RESET DURING HEADER
1848 020310 010337 001142      MOV    R3,$BDDAT   ;SETUP ERROR MESSAGE
1849 020314 042737 175777 001142      BIC    #^CPLFS,$BDDAT
1850 020322 005037 001140              CLR    $GDDAT
1851 020326 010037 001136      MOV    R0,$BDADR
1852 020332 062737 000024 001136      ADD    #RMMR1,$BDADR
1853 020340 104040              EMT    40
1854 020342 000137 021054      JMP    190$
1855
1856 ;LOOKING FOR SYNC SHOULD COME ON FOR DATA AREA WITHIN 9 PROM CYCLES
1857 020346      80$:
1858 020346 012702 000011      MOV    #9,R2
1859 020352 004737 021166      JSR    PC,350$     ;SET STROBE OFF
1860 020356 004737 021060      JSR    PC,300$     ;SET PROM STROBE
1861 020362 016003 000024      MOV    RMMR1(R0),R3 ;STORE RMMR1 AT R3
1862 020366 032703 002000      BIT    #PLFS,R3
1863 020372 001022              BNE    90$         ;BRANCH IF SYNC ENABLED
1864 020374 005302              DEC    R2
1865 020376 001365              BNE    85$
1866
1867 ;ERROR-CAN'T LOOKING FOR SYNC DURING DATA AREA
1868 020400 012737 002000 001140      MOV    #PLFS,$GDDAT
1869 020406 010337 001142      MOV    R3,$BDDAT
1870
1871 020412 042737 175777 001142      BIC    #^CPLFS,$BDDAT
1872 020420 010037 001136      MOV    R0,$BDADR
1873 020424 062737 000024 001136      ADD    #RMMR1,$BDADR
1874 020432 104035              EMT    35
1875 020434 000137 021054      JMP    190$
1876 020440      90$:
1877 020440 004737 021166      JSR    PC,350$     ;RESET PROM STROBE
1878 020444 004737 023574      JSR    PC,CLKSNC   ;CLOCK THE SYNC PATTERN USING SUBROUTINE
  
```

```

020450 000402 BR 100$ ;BR TO 100$ IF NO ERROR
020452 104000 EMT
1879 020454 000577 BR 190$
1880
1881 ;SIMULATE READ DATA
1882 020456 100$:
1883 020456 012702 000400 MOV #256.,R2 ;WORD COUNT
1884 020462 012703 000020 MOV #16.,R3 ;BIT COUNT
1885 020466 115$:
020466 012760 057401 000024 MOV #MR1AAA!MRD!MCLK,RMMR1(R0) ;LOAD RMMR1
1886 020474 012760 053401 000024 MOV #MR1AAA!MRD,RMMR1(R0) ;LOAD RMMR1
1887 020502 005303 DEC R3
1888 020504 001370 BNE 115$
1889 020506 005302 DEC R2
1890 020510 001364 BNE 110$
1891
1892 ;SIMULATE ECC PATTERN
1893 020512 012737 177446 001174 MOV #177446,$TMP0 ;FIRST ECC WORD
1894 020520 012737 015457 001176 MOV #015457,$TMP1 ;SECOND ECC WORD
1895 020526 012702 001174 MOV #TMP0,R2
1896 020532 012703 000020 120$: MOV #16.,R3
1897 020536 012704 051401 125$: MOV #MR1AAA,R4
1898 020542 000241 CLC
1899 020544 006012 ROR (R2)
1900 020546 103002 BCC 130$
1901 020550 052704 002000 BIS #MRD,R4
1902 020554 130$:
020554 010460 000024 MOV R4,RMMR1(R0) ;LOAD RMMR1
1903 020560 052704 004000 BIS #MCLK,R4
1904 020564 010460 000024 MOV R4,RMMR1(R0) ;LOAD RMMR1
1905 020570 042704 004000 BIC #MCLK,R4
1906 020574 010460 000024 MOV R4,RMMR1(R0) ;LOAD RMMR1
1907 020600 005303 DEC R3
1908 020602 001355 BNE 125$ ;CLOCK A WORD
1909 020604 062702 000002 ADD #2,R2
1910 020610 022702 001176 CMP #TMP1,R2
1911 020614 001746 BEQ 120$ ;CLOCK THE NEXT WORD
1912
1913 ;VERIFY DATA AREA AND READ GATE RESET
1914 020616 012702 000005 MOV #5,R2
1915
1916 020622 004737 021060 140$: JSR PC,300$ ;SET PROM STROBE
1917 020626 004737 021166 JSR PC,350$ ;CLEAR PROM STROBE
1918 020632 016003 000024 MOV RMMR1(R0),R3 ;STORE RMMR1 AT R3
1919 020636 032703 000400 BIT #PDA,R3
1920 020642 001417 BEQ 150$
1921 020644 005302 DEC R2
1922 020646 001365 BNE 140$
1923 020650 042703 177377 BIC #^CPDA,R3 ;SETUP ERROR MESSAGE
1924 020654 010337 001142 MOV R3,$BDDAT
1925 020660 005037 001140 CLR $GDDAT
1926 020664 010037 001136 MOV R0,$BDADR
1927 020670 062737 000024 001136 ADD #RMMR1,$BDADR
1928 020676 104063 EMT 63
1929 020700 000465 BR 190$
1930 020702 150$:
1931 020702 016003 000040 MOV RMMR2(R0),R3 ;STORE RMMR2 AT R3

```

1932	020706	042703	176000		BIC	#*C1777,R3	
1933	020712	001413			BEQ	160\$	
1934	020714	010337	001142		MOV	R3,\$BDDAT	
935	020720	005037	001140		CLR	\$GDDAT	
1936	020724	010037	001136		MOV	R0,\$BDADR	
1937	020730	062737	000040	001136	ADD	#RMMR2,\$BDADR	
1938	020736	104053			EMT	53	
1939	020740	000445			BR	190\$	
1940							
1941							
1942	020742						
1943	020742	016003	000014		MOV	RMER1(R0),R3	;STORE RMER1 AT R3
1944	020746	042703	077677		BIC	#*C<DCK!ECH>,R3	
1945	020752	012737	000000	001140	MOV	#000000,\$GDDAT	
1946	020760	023703	001140		CMP	\$GDDAT,R3	
1947	020764	001411			BEQ	170\$	
1948	020766	010337	001142		MOV	R3,\$BDDAT	
1949	020772	010037	001136		MOV	R0,\$BDADR	
1950	020776	062737	000014	001136	ADD	#RMER1,\$BDADR	
1951	021004	104064			EMT	64	
1952	021006	000422			BR	190\$	
1953							
1954							
1955	021010						
1956	021010	012702	054636		MOV	#BUFONE,R2	;DATA WAS STORED HERE
1957	021014	012703	177777		MOV	#177777,R3	;EACH WORD ALL ONES
1958	021020	012704	000400		MOV	#256.,R4	
1959	021024	022203			180\$: CMP	(R2)+,R3	
1960	021026	001003			BNE	185\$	
1961	021030	005304			DEC	R4	
1962	021032	001374			BNE	180\$	
1963	021034	000407			BR	190\$;EXIT IF ALL THE LOCATIONS CHECKED
1964	021036	014237	001142		185\$: MOV	-(R2),\$BDDAT	;SETUP ERROR MESSAGE
1965	021042	010337	001140		MOV	R3,\$GDDAT	
1966	021046	010237	001136		MOV	R2,\$BDADR	
1967	021052	104065			EMT	65	
1968	021054	000137	021272		190\$: JMP	400\$;GREAT ESCAPE
1969	021060	012737	000021	021162	300\$: MOV	#17.,300\$	
1970							
1971	021066						
1972	021066	012760	055401	000024	305\$: MOV	#RMR1AAA!MCLK,RMMR1(R0)	;LOAD RMMR1
1973	021074	012760	051401	000024	MOV	#RMR1AAA,RMMR1(R0)	;LOAD RMMR1
1974	021102	016037	000024	021164	MOV	RMMR1(R0),330\$;STORE RMMR1 AT 330\$
1975	021110	042737	177737	021164	BIC	#*CWC,330\$	
1976	021116	001020			BNE	310\$;EXIT IF PROM STROBE ON
1977	021120	005337	021162		DEC	320\$	
1978	021124	001360			BNE	305\$	
1979	021126	013737	021162	001142	MOV	320\$,\$BDDAT	;SETUP ERROR MESSAGE
1980	021134	012737	000040	001140	MOV	#WC,\$GDDAT	
1981	021142	010037	001136		MOV	R0,\$BDADR	
1982	021146	062737	000024	001136	ADD	#RMMR1,\$BDADR	
1983	021154	104030			EMT	30	
1984	021156	000445			BR	400\$	
1985	021160	000207			310\$: RTS	PC	
1986	021162	000000			320\$: .WORD	0	
1987	021164	000000			330\$: .WORD	0	

```

1988
1989 021166 012737 000021 021266 350$: MOV #17.,370$ ;CLOCK THE STROBE OFF
1990 021174 012737 000021 021266 355$:
1991 021174 012760 055401 000024 MOV #MMR1AAA!MCLK,RMMR1(RO) ;LOAD RMMR1
1992 021202 012760 051401 000024 MOV #MMR1AAA,RMMR1(RO) ;LOAD RMMR1
1993 021210 016037 000024 021270 MOV RMMR1(RO),380$ ;STORE RMMR1 AT 380$
1994 021216 042737 177737 021270 BIC #^CWC,380$
1995 021224 001417 BEQ 360$
1996 021226 005337 021266 DEC 370$
1997 021232 001360 BNE 355$
1998 021234 013737 021270 001142 MOV 380$, $BDDAT
1999 021242 005037 001140 CLR $GDDAT
2000 021246 010037 001136 MOV RO, $BDADR
2001 021252 062737 000024 001136 ADD #RMP1, $BDADR
2002 021260 104062 .EMT 62
2003 021262 000403 BR 400$
2004 021264 000207 RTS PC
2005 021266 000000 360$: .WORD 0
2006 021270 000000 370$: .WORD 0
000000 380$: .WORD 0
000240 400$: NOP ;SUB-TEST EXIT POINT
  
```

```

1          .SBTTL  END OF SUB-PASS ROUTINE
2
3          :THIS IS THE END OF SUB-PASS ROUTINE. THIS ROUTINE IS USED TO
4          :TERMINATE THE OPERATION OF THE CURRENT DEVICE UNDER TEST AND
5          :SELECT THE NEXT DEVICE FOR TEST. IF THERE ARE NO MORE DEVICES
6          :TO TEST, EXIT IS MADE TO '$EOP' ROUTINE. OTHERWISE, RETURN
7          :IS MADE TO 'READY' ROUTINE.
8
9 021274 000004          $EOSP:  SCOPE
10 021276 000240          NOP
11 021300 013700 001466  MOV      TSTQUE,RO          ;GET POINTER TO TSTQUE
12 021304 062700 000002  ADD      #2,RO          ;ADJUST POINTER TO NEXT DEVICE
13 021310 010037 001466  MOV      RO,TSTQUE      ;SAVE POINTER TO TSTQUE
14 021314 005710          TST      (RO)          ;ANY MORE DEVICES FOR TEST ?
15 021316 001402          BEQ      1$          ;BR IF NO
16 021320 000137 005200  JMP      READY          ;YES, JUMP TO 'READY' ROUTINE
17 021324 012737 001470 001466 1$:  MOV      #TSTQUE+2,TSTQUE ;INITIALIZE POINTER TO FIRST DEVICE IN
18                                     ;TEST QUE TABLE
19
20

```

.SBTTL END OF PASS ROUTINE

```

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

```

```

021332          $EOP:
021332 000240          NOP
021334 005037 001116  CLR      $TSTNM          ;;ZERO THE TEST NUMBER
021340 005037 001206  CLR      $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
021344 005237 001230  INC      $PASS          ;;INCREMENT THE PASS NUMBER
021350 042737 100000 001230 BIC      #100000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
021356 005327          DEC      (PC)+          ;;LOOP?
021360 000001          $EOPCT: .WORD 1
021362 003066          BGT      $DOAGN          ;;YES
021364 012737          MOV      (PC)+,@(PC)+  ;;RESTORE COUNTER
021366 000001          $ENDCT: .WORD 1
021370 021360          $EOPCT
021372 104401 021400  TYPE      ,65$          ;;TYPE ASCIZ STRING
021376 000407          BR      64$          ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <12><15>/END PASS #/
64$:
021416          MOV      $PASS,-(SP)  ;;SAVE $PASS FOR TYPEOUT
021416 013746 001230  ;;TYPE PASS NUMBER
021422 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
021424 005737 001126  TST      $ERTTL          ;;SEE IF ANY ERRORS THIS PASS
021430 001431          BEQ      $GT42P          ;;BR IF NO ERRORS TO REPORT
021432 104401 021440  TYPE      ,67$          ;;TYPE ASCIZ STRING
021436 000421          BR      66$          ;;GET OVER THE ASCIZ
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:
021502          MOV      $ERTTL,-(SP)  ;;SAVE $ERTTL FOR TYPEOUT
021502 013746 001126  ;;TOTAL NUMBER OF ERRORS
021506 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
021510 005037 001126  CLR      $ERTTL          ;;CLEAR ERROR TOTAL

```

END OF PASS ROUTINE

021514	104401	001217		\$GT42P:	TYPE	,SCLRF		::TYPE CARRIAGE RETURN	LINE FEED
021520	013700	000042		\$GET42:	MOV	@#42,R0		::GET MONITOR ADDRESS	
021524	001405				BEQ	\$DOAGN		::BRANCH IF NO MONITOR	
021526	000005				RESET			::CLEAR THE WORLD	
021530	004710			\$ENDAD:	JSR	PC,(R0)		::GO TO MONITOR	
021532	000240				NOP			::SAVE ROOM	
021534	000240				NOP			::FOR	
021536	000240				NOP			::ACT11	
021540				\$DOAGN:					
021540	000137				JMP	@(PC)+		::RETURN	
021542	005200			\$RTNAD:	.WORD	READY			
021544	377	377	000	\$ENULL:	.BYTE	-1,-1,0		::NULL CHARACTER STRING	
					.EVEN				

```

1          .SBTTL  CLOCK SUBROUTINES
2
3          ;ROUTINE TO SIZE FOR CLOCKS (KW11-L OR KW11-P)
4
5 021550  000240          SIZECLK: NOP
6 021552  013746  000004  MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
7 021556  013746  000006  MOV      ERRVEC+2,-(SP)    ;;PUSH ERRVEC+2 ON STACK
8 021562  012737  021646  000004  MOV      #10$,ERRVEC      ;LOAD 04 TRAP VECTORS
9 021570  012737  000300  000006  MOV      #PR6,ERRVEC+2
10
11         ;SEE IF A KW11-P CLOCK IS PRESENT - GO TO 10$ IF NOT PRESENT
12 021576  005777  157710          TST      @SLPCSR          ;TEST FOR P CLOCK
13 021602  012737  022010  001536  MOV      #PCLOCK,CLOCK    ;LOAD SUBROUTINE ADDRESS
14 021610  012737  022132  001540  MOV      #PSTOP,STOPCL    ;LOAD STOP ADDRESS
15 021616  012777  022076  157672  MOV      #PCOUNT,@SLPVEC  ;LOAD P CLOCK INTERRUPT VECTOR
16 021624  012777  000300  157666  MOV      #PR6,@SLPVEC+2
17 021632  013777  001526  157664  MOV      $LLVEC+2,$LLVEC ;CLEAR L CLOCK INTERRUPT VECTOR
18 021640  005077  157662          CLR      @LLVEC+2
19 021644  000454          BR       30$
20 021646  012716  021654  10$:    MOV      #15$, (SP)      ;DUMMY RTI ADDRESS
21 021652  000002          RTI                       ;RESTORE PRIORITY
22
23         ;NO P CLOCK-SEE IF L CLOCK IS PRESENT-GO TO 20$ IF NOT PRESET
24 021654          15$:
25 021654  012737  021732  000004  MOV      #20$,ERRVEC      ;CHANGE 04 TRAP VECTOR
26 021662  005777  157634          TST      @LLCSR          ;TEST FOR L CLOCK
27 021666  012737  022026  001536  MOV      #LCLOCK,CLOCK    ;LOAD SUBROUTINE ADDRESS
28 021674  012737  022140  001540  MOV      #LSTOP,STOPCL   ;LOAD STOP ADDRESS
29 021702  012777  022076  157614  MOV      #LCOUNT,@LLVEC  ;LOAD L CLOCK INTERRUPT VECTOR
30 021710  012777  000300  157610  MOV      #PR6,@LLVEC+2
31 021716  013777  001520  157572  MOV      $LPVEC+2,$LPVEC ;CLEAR P CLOCK INTERRUPT VECTOR
32 021724  005077  157570          CLR      @SLPVEC+2
33 021730  000422          BR       30$
34 021732  012716  021740  20$:    MOV      #25$, (SP)      ;DUMMY RTI ADDRESS
35 021736  000002          RTI                       ;RESTORE PRIORITY
36
37         ;NO CLOCK AVAILABLE - AUGMENT RETURN ADDRESS
38 021740          25$:
39 021740  005037  001536          CLR      CLOCK          ;CLEAR SUBROUTINE ADDRESS
40 021744  012737  001520  001516  MOV      #$LPVEC+2,$LPVEC ;CLEAR P CLOCK INTERRUPT VECTOR
41 021752  005037  001520          CLR      $LPVEC+2
42 021756  012737  001526  001524  MOV      #$LLVEC+2,$LLVEC ;CLEAR L CLOCK INTERRUPT VECTOR
43 021764  005037  001526          CLR      $LLVEC+2
44 021770  062766  000002  000004  ADD      #2,4(SP)         ;CHANGE RETURN ADDRESS
45
46 021776          30$:
47 021776  012637  000006  MOV      (SP)+,ERRVEC+2    ;;POP STACK INTO ERRVEC+2
48 022002  012637  000004  MOV      (SP)+,ERRVEC     ;;POP STACK INTO ERRVEC
49 022006  000207          RTS      PC
50
51         ;ROUTINES TO START THE CLOCK (KW11-L OR KW11-P)
52 022010  012777  177777  157476  PCLOCK: MOV      #-1,@SLPCSB ;LOAD COUNT SET BUFFER
53 022016  012777  000135  157466  MOV      #135,@SLPCSR    ;LOAD CONTROL REGISTER
54 022024  000403          BR       PLCLK          ;GO TO COMMON CODE
55
56 022026  012777  000100  157466  LCLOCK: MOV      #100,@LLCSR ;LOAD CONTROL REGISTER
    
```



```

57
58 022034 005037 001532      PLCLK: CLR      TIME      ;CLEAR TIMER COUNT
59 022040 104400              TRAP                    ;;PUSH OLD PSW AND PC ON STACK
    022042 012605              MOV      (SP)+,R5      ;;SAVE THE PSW IN R5
60 022044 010537 001530      MOV      R5,$PSW      ;SAVE PRIORITY
61 022050 042705 177437      BIC     #^CPR7,R5     ;MASK X
62 022054 022705 000300      CMP     #PR6,R5      ;IS PRIORITY TOO HIGH??
63 022060 101005              BHI     40$           ;NO!!
64 022062 012746 000240      MOV     #PR5,-(SP)    ;;PUT NEW PS ON STACK
    022066 012746 022074      MOV     #30$,-(SP)   ;;PUT NEW PC ON STACK
    022072 000002              RTI                    ;;POP NEW PC AND PS
    022074
65 022074 000207      30$:
    40$: RTS      PC
66
67
68
69 022076              PCOUNT:
70 022076              LCOUNT:
71 022076 062737 000021 001532      ADD     #17.,TIME     ;ADD 17MS TO ELAPSED TIME
72 022104 103003              BCC     10$           ;BRANCH IF NO OVERFLOW
73 022106 012737 177777 001532      MOV     #-1.,TIME     ;RESTORE MAXIMUM COUNT
74 022114 162737 000021 001534      10$:   SUB     #17.,WATCH ;DECREMENT REMAINING TIME
75 022122 100002              BPL     20$           ;BRANCH IF POSITIVE
76 022124 005037 001534      CLR     WATCH        ;CLEAR REMAINING TIME
77 022130 000002      20$:   RTI                    ;RETURN TO USER
78
79
80
81 022132 005077 157354      PSTOP: CLR     @SLPCSR ;STOP P CLOCK
82 022136 00040?              BR      PLSTP        ;GO TO COMMON STOP CODE
83
84 022140 005077 157356      LSTOP: CLR     @LLCSR ;STOP L CLOCK
85
86 022144              PLSTP:
87 022144 013746 001530      MOV     $PSW,-(SP)   ;;PUT NEW PS ON STACK
    022150 012746 022156      MOV     #10$,-(SP)  ;;PUT NEW PC ON STACK
    022154 000002              RTI                    ;;POP NEW PC AND PS
    022156
88 022156 000207      10$:   RTS      PC
    
```

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

```

.SBTTL SET VOLUME VALID SUBROUTINE
;THIS SUBROUTINE INITIALIZES THE SUBSYSTEM AND SETS VOLUME VALID,
;RETURNING WITH THE DRIVE STILL IN DIAGNOSTIC MODE. THE SUBROUTINE
;RETURNS TO THE WORD FOLLOWING THE CALL, EXCEPT WHEN AN ERROR IS
;DETECTED, IN WHICH CASE IT RETURNS TO THE SECOND WORD FOLLOWING THE
;CALL.
:CALL: JSR PC,SETVV JUMP TO SUBROUTINE
: BR ?? RETURN HERE IF NO ERROR
: ERROR RETURN HERE IF ERROR

SETVV: JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD,RMMR1(RO) ;LOAD RMMR1
MOV #DMD!MUR,RMMR1(RO) ;LOAD RMMR1
MOV #0,RMER1(RO) ;LOAD RMER1
MOV #0,RMER2(RO) ;LOAD RMER2
MOV #PACACK!GO,RMCS1(RO) ;LOAD RMCS1
MOV RMDS(RO),$BDDAT ;STORE RMDS AT $BDDAT
BIC #^CVV,$BDDAT
BNE 10$ ;BRANCH IF VOLUME VALID SET
MOV RO,$BDADR ;SETUP FOR ERROR MSG
ADD #RMDS,$BDADR
MOV #VV,$GDDAT
ADD #2,(SP) ;MOVE RETURN ADDRESS TO ERROR
MOV #100,@(SP) ;WRITE ERROR NUMBER
MOV #PACACK,$TMPO
10$: RTS PC ;RETURN
  
```

13	022160			
14	022160	004737	022404	
15	022164	012760	000001	000024
16	022172	012760	001001	000024
17	022200	012760	000000	000014
18	022206	012760	000000	000042
19	022214	012760	000023	000000
20	022222	016037	0C0012	001142
21	022230	042737	177677	001142
22	022236	001020		
23	022240	010037	001136	
24	022244	062737	000012	001136
25	022252	012737	000100	001140
26	022260	062716	0C0002	
27	022264	112776	000100	000000
28	022272	012737	000022	001174
29	022300	000207		

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

```
.SBTTL SET OFFSET MODE SUBROUTINE  
:THIS SUBROUTINE EXECUTES AN OFFSET COMMAND AND VERIFIES THAT OFFSET  
:MODE SETS. THE DRIVE SHOULD BE IN DIAGNOSTIC MODE WHEN CALLING THE  
:SUBROUTINE, WHICH WILL LEAVE DMD ON. THE SUBROUTINE RETURNS TO THE  
:WORD FOLLOWING THE CALL UNLESS THERE IS AN ERROR, IN WHICH CASE IT  
:RETURNS TO THE SECOND WORD FOLLOWING THE CALL  
:  
:CALL: JSR PC,SETOM JUMP TO SUBROUTINE  
: BR ?? RETURN HERE IF NO ERROR  
: ERROR RETURN HERE IF ERROR  
:  
SETOM:  
MOV #DMD!MUR,RMMR1(R0) ;LOAD RMMR1  
MOV #0,RMER1(R0) ;LOAD RMER1  
MOV #0,RMER2(R0) ;LOAD RMER2  
MOV #OFFSET!GO, RMCS1(R0) ;LOAD RMCS1  
MOV RMDS(R0), $BDDAT ;STORE RMDS AT $BDDAT  
BIC #^COM,$BDDAT  
BNE 10$ ;BRANCH IF OFFSET ON  
MOV #OM,$GDDAT  
MOV R0,$BDADR  
ADD #RMDS,$BDADR  
ADD #2,(SP) ;MOVE RETURN ADDRESS TO ERROR  
MOVB #111,@(SP) ;WRITE ERROR NUMBER  
10$: RTS PC ;RETURN TO USER
```

```
1          .SBTTL CLEAR CONTROLLER SUBROUTINE
2
3          ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTERS,
4          ;AND DRIVES, THEN SELECTS THE DRIVE.
5          ;CALL:
6          ;      JSR      PC,CNTCLR      ;CALL TO ROUTINE
7
8          CNTCLR:
9          MOV      RO,-(SP)             ;;PUSH R0 ON STACK
10         MOV      $BASE,R0            ;RO = UNIBUS BASE ADDRESS
11         MOV      #CLR,RMCS2(R0)      ;CLEAR MASSBUS
12         MOVB     @TSTQUE,RMCS2(R0)   ;SELECT DEVICE UNDER TEST
13         MOV      (SP)+,R0            ;;POP STACK INTO R0
14         RTS      PC                  ;RETURN
```

1
2
3 022432
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 022432 004737 022404
20 022436 012760 0C0001 000024
21 022444 012704 000041
22
23 022450
24 022450 012760 005001 000024
25 022456 012760 001001 000024
26 022464 016005 000024
27 022470 032705 000040
28 022474 001023
29 022476 005304
30 022500 001363
31 022502 010037 001136
32 022506 062737 000024 001136
33 022514 005037 001142
34 022520 012737 000040 001140
35 022526 062716 000002
36 022532 112776 000030 000000
37 022540 000137 023302
38 022544
39 022544 012760 005001 000024
40 022552 012760 001001 000024
41 022560 016005 000024
42 022564 032705 000040
43 022570 001423
44 022572 005304
45 022574 001363
46 022576 010037 001136
47 022602 062737 000024 001136
48 022610 012737 000040 001142
49 022616 005037 001140
50 022622 062716 000002
51 022626 112776 000062 000000
52 022634 000137 023302
53
54
55 022640

.SBTTL ENABLE SEARCH SUBROUTINE

ENBSCH:

:THE REGISTER OUTPUT BUFFER SHOULD CONTAIN:
: RMDAO = DESIRED SECTOR AND TRACK ADDRESS
: RMDCO = DESIRED CYLINDER ADDRESS
: RMOFO = FORMAT, ECI, HCI
: RMBAD = BUFFER ADDRESS
: RMWCO = WORD COUNT
: RMCSTO = FUNCTION CODE

:NOTE: OFFSET IS NOT ENABLED WHEN USING THIS SUBROUTINE

:SET VOLUME VALID
:FIRST CLEAR -SET DIAGNOSTIC MODE-SYNCHRONIZE
:PROM STROBE

JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD,RMMR1(RO) ;LOAD RMMR1
MOV #33,R4 ;ALLOW UP TO 33 BIT CLOCKS
;TO SYNC PROM STROBE
5\$:
MOV #DMD!MUR!MCLK,RMMR1(RO) ;LOAD RMMR1
MOV #DMD!MUR,RMMR1(RO) ;LOAD RMMR1
MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
BIT #WC,R5 ;WAIT FOR PROM STROBE TO COME ON
BNE 6\$
DEC R4
BNE 5\$
MOV R0,\$BDADR ;PROM STROBE WONT SET
ADD #RMMR1,\$BDADR
CLR \$BDDAT
MOV #WC,\$GDDAT
ADD #2,(SP)
MOVB #30,@(SP)
JMP 60\$;REPORT ERROR
6\$:
MOV #DMD!MUR!MCLK,RMMR1(RO) ;LOAD RMMR1
MOV #DMD!MUR,RMMR1(RO) ;LOAD RMMR1
MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
BIT #WC,R5 ;WAIT FOR PROM STROBE TO GO OFF
BEQ 7\$
DEC R4
BNE 6\$
MOV R0,\$BDADR ;PROM STROBE WONT RESET
ADD #RMMR1,\$BDADR ;
MOV #WC,\$BDDAT
CLR \$GDDAT
ADD #2,(SP)
MOVB #62,@(SP)
JMP 60\$
;SECOND, CLOCK INDEX PULSE TO
: (1) CLEAR FORMAT CHANGE FLOP
: (2) CLEAR RTC FLOP
7\$:

```

56 022640 012760 001005 000024      MOV      #DMD.MUR.MI,RMMR1(RO) ;LOAD RMMR1
57 022646 012760 001001 000024      MOV      #DMD.MUR,RMMR1(RO) ;LOAD RMMR1
58
59 022654 012760 000000 000014      MOV      #0,RMER1(RO) ;LOAD RMFR1
60 022662 012760 000000 000042      MOV      #0,RMER2(RO) ;LOAD RMER2
61 022670 012760 000023 000000      MOV      #PACACK.GO, RMCS1(RO) ;LOAD RMCS1
62 022676 016037 000012 001142      MOV      RMD5(RO), $BDDAT ;STORE RMD5 AT $BDDAT
63 022704 042737 177677 001142      BIC      #^CVV,$BDDAT ;VOLUME VALID SHOULD BE SET
64 022712 001021 000000 000000      BNE      10$
65 022714 010037 001136 000000      MOV      RO,$BDADR ;SETUP ERROR MSG
66 022720 062737 000012 001136      ADD      #RMD5,$BDADR
67 022726 012737 000100 001140      MOV      #VV,$GDDAT
68 022734 062716 000002 000000      ADD      #2,(SP) ;WRITE ERROR NUMBER
69 022740 112776 000100 000000      MOV      #100,@(SP) ;
70 022746 012737 000022 001174      MOV      #PACACK,$TMP0
71 022754 000552 000000 000000      BR       60$
72
73
74 022756 013760 001420 000006      ;LOAD REGISTERS
75 022756 013760 001446 000034      10$: MOV      RMDAO,RMDA(RO) ;LOAD RMDA
76 022764 013760 001444 000032      MOV      RMDCO,RMDC(RO) ;LOAD RMDC
77 022772 013760 001414 000002      MOV      RMOFO,RMOF(RO) ;LOAD RMOF
78 023000 013760 001414 000002      MOV      RMIJO,RMIJ(RO) ;LOAD RMIJ
79 023006 013760 001416 000004      MOV      RMBAO,RMBA(RO) ;LOAD RMBA
80
81
82 023014 012760 041401 000024      ;ENABLE DEBUG CLOCK AND LOAD FUNCTION CODE
83 023022 012760 000000 000014      MOV      #DMD!MUR!DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
84 023030 012760 000000 000042      MOV      #0,RMER1(RO) ;LOAD RMER1
85 023036 013760 001412 000000      MOV      #0,RMER2(RO) ;LOAD RMER2
86
87
88 023044 012737 000310 001534      ;WAIT FOR 'RUN AND GO' TO SET
89 023052 004777 156460 000000      20$: MOV      #200,WATCH ;SET WATCHDOG TIMER VALUE
90 023056 016037 000024 001142      JSR      PC,@CLOCK ;START THE CLOCK
91 023064 042737 137777 001142      MOV      RMMR1(RO), $BDDAT ;STORE RMMR1 AT $BDDAT
92 023072 001023 000000 000000      BIC      #^CRG,$BDDAT
93 023074 005737 001534 000000      BNE      30$
94 023100 001366 000000 000000      TST      WATCH ;ANY TIME LEFT?
95 023102 004777 156432 000000      BNE      20$ ;YES-WAIT
96 023106 012737 040000 001140      JSR      PC,@STOPCL ;STOP THE CLOCK
97 023114 010037 001136 000000      MOV      #RG,$GDDAT ;SETUP ERROR MSG
98 023120 062737 000024 001136      MOV      RO,$BDADR
99 023126 062716 000002 000000      ADD      #RMMR1,$BDADR
100 023132 112776 000101 000000      ADD      #2,(SP) ;WRITE ERROR NUMBER IN
101 023140 000460 000000 000000      MOV      #101,@(SP) ;USE n'S CALL -
102 023142 004777 156372 000000      30$: BR       60$
103
104
105 023146 012704 000021 000000      ;STEP COMMAND SEQUENCER TO HEAD SEQUENCE AT LOCATION 156 (17 CLOCKS)
106 023152 012760 141401 000024      40$: MOV      #17,R4 ;R4=CLOCK COUNT
107 023160 012760 041401 000024      MOV      #DMD!MUR.DBEN!MOC!DBCK,RMMR1(RO) ;LOAD RMMR1
108 023166 005304 000000 000000      MOV      #DMD!MUR.DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
DEC      R4

```

```

109 023170 001370          BNE      40$
110
111          ;DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER TO PASS
112          ;TEST AT LOCATION 166
113 023172 012760 041001 000024      MOV      #DMD.MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
114 023200 012760 041401 000024      MOV      #DMD.MUR!DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
115
116          ;STEP SEQUENCER (35 CLOCKS) AND VERIFY SEARCH IS ENABLED
117 023206 012704 000043          MOV      #35.,R4 ;R4=CLOCK COUNT
118 023212          50$:
119 023212 012760 151401 000024      MOV      #DMD!MUR!DBEN.MOC!DBCK!MSEN,RMMR1(RO) ;LOAD RMMR1
120 023220 012760 051401 000024      MOV      #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
121 023226 005304          DEC      R4
122 023230 001370          BNE      50$
123 023232 016037 000024 001142      MOV      RMMR1(RO), $BDDAT ;STORE RMMR1 AT $BDDAT
124 023240 042737 173777 001142      BIC      #^CESRC,$BDDAT ;SEARCH SHOULD BE ENABLED
125 023246 001015          BNE      60$
126 023250 010037 001136          MOV      R0,$BDADR ;SETUP ERROR MESSAGE
127 023254 062737 000024 001136      ADD      #RMMR1,$BDADR
128 023262 012737 0C4000 001140      MOV      #ESRC,$GDDAT
129 023270 062716 000002          ADD      #2,(SP) ;WRITE ERROR NUMBER
130 023274 112776 000102 000000      MOVB    #102,a(SP)
131 023302          60$:
132 023302 000207          RTS      PC
  
```

```

1          .SBTTL  SECTOR COMPARE SUBROUTINE
2
3          ;THIS SUBROUTINE CONTINUES THE EXECUTION OF A DATA COMMAND
4          ;FROM WHERE SEARCH HAS BEEN ENABLED TO WHERE SECTOR COMPARE
5          ;IS SET.
6
7 023304    SCTCMP:
8
9          ;SET INDEX PULSE TO CLEAR FORMAT CHANGE FLOP
10         MOV     #DMD!MUR!DBEN!MOC!MSEN!MI,RMMR1(RO)      ;LOAD RMMR1
11 023312    012760 051401 000024      MOV     #DMD.MUR!DBEN!MOC!MSEN,RMMR1(RO)      ;LOAD RMMR1
12
13         ;WAIT AT LEAST 4 MS FOR 'RETURN TO CENTER LINE' ONE SHOT TO SET
14 023320    012737 000006 001534      MOV     #6,WATCH          ;SET WATCHDOG TIMER VALUE
15         023326    004777 156204      JSR     PC,@CLOCK        ;START THE CLOCK
16 023332    005737 001534      10$:   TST     WATCH
17 023336    001375      BNE     10$
18 023340    004777 156174      JSR     PC,@STOPCL      ;STOP THE CLOCK
19
20         ;DATA INPUT TO SEARCH ENABLE FLOP SHOULD BE ONE - CLOCK SECTOR PULSE TO
21         ;SET FLOP
22 023344    012760 051441 000024      MOV     #DMD!MUR!DBEN!MOC!MSEN!MS,RMMR1(RO)      ;LOAD RMMR1
23 023352    012760 051401 000024      MOV     #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO)      ;LOAD RMMR1
24
25         ;WITH SECTOR COMPARE HIGH, CLOCK SECTOR PULSE TO SET SECTOR COMPARE FLOP
26 023360    012760 051403 000024      MOV     #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO)      ;LOAD RMMR1
27 023366    012760 051443 000024      MOV     #DMD!MUR!DBEN!MOC!MSEN!MSC!MS,RMMR1(RO)  ;LOAD RMMR1
28 023374    012760 051403 000024      MOV     #DMD!MUR!DBEN!MOC!MSEN!MSC,RMMR1(RO)      ;LOAD RMMR1
29 023402    012760 051401 000024      MOV     #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO)      ;LOAD RMMR1
30 023410    000207      RTS     PC

```



```

1
2
3
4
5
6 023412
7
8
9 023412
10 023412 012760 055401 000024
11 023420 012760 051401 000024
12 023426 016005 000024
13 023432 032705 000040
14 023436 001765 -
15
16
17 023440 012704 000260
18 023444
19 023452 012760 055401 000024
20 023460 016005 051401 000024
21 023464 032705 000024
22 023470 001010 002000
23 023472 005304
24 023474 001363
25 023476 062716 000002
26 023502 117776 000035 000000
27 023510 000430
28
29
30
31 023512
32 023512 016005 000024
33 023516 032705 000040
34 023522 001007
35 023524 012760 055401 000024
36 023532 012760 051401 000024
37 023540 000414
38 023542
39 023546 016005 000024
40 023552 032705 000040
41 023554 001407
42 023554 012760 055401 000024
43 023562 012760 051401 000024
44 023570 000764
45 023572
46 023572 000207

```

```

:SBTTL SET LOOKING FOR SYNC SUBROUTINE
:THIS SUBROUTINE WILL SETUP THE DATA TIMING SEQUENCER
:ASSUMING SEARCH IS ENABLED, TO THE POINT WHERE 'PLFS' IS ACTIVE
SETLFS:
:PULSE BIT CLOCK UNTIL PROM STROBE SETS
10$:
MOV #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(RO) ;LOAD RMMR1
MOV #DMD.MUR!DBEN!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
BIT #WC,R5
BEQ 10$

:SET UP DATA SEQUENCER TO LOCATION 11. WHERE PLFS WILL SET
20$:
MOV #176.,R4 ;MAX NUMBER OF BIT CLOCK
MOV #DMD!MUR!DBEN!MOC!MSEN!MCLK,RMMR1(RO) ;LOAD RMMR1
MOV #DMD!MUR!DBEN!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
BIT #PLFS,R5 ;EXIT IF PLFS IS SET
BNE 30$
DEC R4 ;ERROR IF COUNT EXHAUSTED
BNE 20$
ADD #2,(SP)
MOV# #35,@(SP) ;CANT SET PLFS
BR 50$

:STEP THE MAIN REG CLOCK UNTIL THE TRAILING EDGE OF PROM STORBE
:IS DETECTED.
30$:
MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
BIT #WC,R5
BNE 40$
MOV #MR1AAA!MCLK,RMMR1(RO) ;LOAD RMMR1
MOV #MR1AAA,RMMR1(RO) ;LOAD RMMR1
BR 50$

40$:
MOV RMMR1(RO),R5 ;STORE RMMR1 AT R5
BIT #WC,R5
BEQ 50$
MOV #MR1AAA!MCLK,RMMR1(RO) ;LOAD RMMR1
MOV #MR1AAA,RMMR1(RO) ;LOAD RMMR1
BR 40$

50$:
RTS PC ;EXIT

```

```

1          .SBTTL  CLOCK SYNC SUBROUTINE
2
3          ;THIS SUBROUTINE WILL SIMULATE THE HEADER AND DATA SYNC
4          ;PATTERN BEING READ
5
6 023574    CLKSNC:
7
8          ;VERIFY THAT 'PLFS' IS ON
9 023574    016004    000024    MOV     RMMR1(R0),R4    ;STORE RMMR1 AT R4
10 023600    032704    002000    BIT     #PLFS,R4       ;LOOK FOR SYN SET ?
11 023604    001023          BNE     10$           ;BRANCH IF SO
12 023606    010437    001142    MOV     R4,$BDDAT
13 023612    042737    175777    001142    BIC     #^PLFS,$BDDAT
14 023620    012737    002000    001140    MOV     #PLFS,$GDDAT
15 023626    010037    001136          MOV     R0,$BDADR
16 023632    062737    000024    001136    ADD     #RMMR1,$BDADR
17 023640    062716    000002          ADD     #2,(SP)
18 023644    112776    000035    000000    MOVSB  #35,@(SP)
19 023652    000472          BR      60$
20 023654    012705    000021    10$:    MOV     #17.,R5        ;MAKE SURE PROM STROBE IS OFF
21 023660    032704    000040    20$:    BIT     #WC,R4
22 023664    001434          BEQ     30$
23 023666    012760    055401    000024    MOV     #MR1AAA!MCLK,RMMR1(R0) ;LOAD RMMR1
24 023674    012760    051401    000024    MOV     #MR1AAA,RMMR1(R0)      ;LOAD RMMR1
25 023702    016004    000024          MOV     RMMR1(R0),R4        ;STORE RMMR1 AT R4
26 023706    005305          DEC     R5
27 023710    001363          BNE     20$
28
29          ;ERROR CAN'T RESET PROM STROBE WITH LFS ACTIVE
30 023712    010437    001142          MOV     R4,$BDDAT          ;SETUP ERROR FOR USER
31 023716    042737    177737    001142    BIC     #^CWC,$BDDAT
32 023724    005037    001140          CLR     $GDDAT
33 023730    010037    001136          MOV     R0,$BDADR
34 023734    062737    000024    001136    ADD     #RMMR1,$BDADR
35 023742    062716    000002          ADD     #2,(SP)
36 023746    112776    000062    000000    MOVSB  #62,@(SP)
37 023754    000431          BR      60$
38
39          ;CLOCK THE SYNC PATTERN (00011001) THROUGH THE SHIFT REGISTER
40 023756    30$:    MOV     #014400,R4
41 023756    012704    014400          MOV     #16.,70$
42 023762    012737    000020    024042    40$:    MOV     #MR1AAA,R5        ;STROBE BIT COUNT
43 023770    012705    051401          CLC
44 023774    000241          ROR    R4                ;GENERATE REG VALUE
45 023776    006004          ROR    R4                ;WITH MAINTENANCE CLOCK ON
46 024000    103002          BCC   50$
47 024002    052705    002000          BIS   #MRD,R5            ;SET READ BIT PER PATTERN BIT
48
49 024006    50$:    MOV     R5,RMMR1(R0)      ;LOAD RMMR1
50 024012    052705    004000          BIS   #MCLK,R5
51 024016    010560    000024          MOV     R5,RMMR1(P0)      ;LOAD RMMR1
52 024022    042705    004000          BIC   #MCLK,R5            ;CLOCK ONE BIT
53 024026    010560    000024          MOV     R5,RMMR1(R0)      ;LOAD RMMR1
54 024032    005337    024042          DEC   70$
55 024036    001354          BNE   40$
56

```

57
58
59
60 024040
61 024040 000207
62
63 024042 000000

:CAN'T VERIFY SYNC CLOCK WAS DETECTED
:USER CAN DO SO BY STEPIING
:BIT CLOCK AND VERIFY PROM STROBE SETS WITHIN ONE WORD TIME
60\$: RTS PC
70\$: .WORD 0 :TEMPORARY STORAGE

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

:*****
:*SAVE R0-R5
:*CALL:
:* SAVREG
:*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
:*
:*TOP---(+16)
:* +2---(+18)
:* +4---R5
:* +6---R4
:* +8---R3
:*+10---R2
:*+12---R1
:*+14---R0

```

```

024044          $SAVREG:
024044 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
024046 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
024050 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
024052 010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
024054 010446      MOV     R4,-(SP)      ;;PUSH R4 ON STACK
024056 010546      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
024060 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
024064 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
024070 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PS OF CALL
024074 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PC OF CALL
024100 000002      RTI

```

*RESTORE R0-R5

*CALL:

* RESREG

\$RESREG:

```

024102          $RESREG:
024102 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PC OF CALL
024106 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PS OF CALL
024112 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
024116 012666 000022  MOV     (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
024122 012605      MOV     (SP)+,R5      ;;POP STACK INTO R5
024124 012604      MOV     (SP)+,R4      ;;POP STACK INTO R4
024126 012603      MOV     (SP)+,R3      ;;POP STACK INTO R3
024130 012602      MOV     (SP)+,R2      ;;POP STACK INTO R2
024132 012601      MOV     (SP)+,R1      ;;POP STACK INTO R1
024134 012600      MOV     (SP)+,R0      ;;POP STACK INTO R0
024136 000002      RTI

```

.SBITL BINARY TO ASCII AND TYPE ROUTINE

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

```

*CALL:
*      MOV      NUMBER,-(SP)      ;;NUMBER TO BE TYPED
*      TYPBN
*
024140 010146          $TYPBN: MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
024142 016601 000006      MOV      6(SP),R1      ;;GET THE INPUT NUMBER
024146 000261          SEC                          ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
024150 112737 000060 024212 1$: MCVB     #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
024156 006101          ROL      R1                ;;GET THIS BIT
024160 001406          BEQ      2$                ;;DONE?
024162 105537 024212      ADCB     $BIN           ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
024166 104401 024212      TYPE     ,$BIN         ;;GO TYPE THIS BIT
024172 000241          CLC                          ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
024174 000765          BR       1$                ;;GO DO THE NEXT BIT
024176 012601          2$:  MOV      (SP)+,R1      ;;POP THE STACK INTO R1
024200 016666 000002 000004  MOV      2(SP),4(SP)  ;;ADJUST THE STACK
024206 012616          MOV      (SP)+,(SP)
024210 000002          RTI                          ;;RETURN TO USER
024212      000      000      $BIN: .BYTE     0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
  
```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPF ROUTINE

```

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

```

```

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

```

```

024214          $TYPDS:
024214 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
024216 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
024220 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
024222 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
024224 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
024226 012746 020200  MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
024232 016605 000020  MOV      20(SP),R5      ;;GET THE INPUT NUMBER
024236 100004      BPL      1$          ;;BR IF INPUT IS POS.
024240 005405      NEG      R5          ;;MAKE THE BINARY NUMBER POS.
024242 112766 000055 000001  MOVB   #'-',1(SP)      ;;MAKE THE ASCII NUMBER NEG.
024250 005000      1$:      CLR      R0          ;;ZERO THE CONSTANT'S INDEX
024252 012703 024430      MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
024256 112723 000040      MOVB   #'',(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
024262 005002      2$:      CLR      R2          ;;CLEAR THE BCD NUMBER
024264 016001 024420      MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
024270 160105      3$:      SUB      R1,R5      ;;FORM THIS BCD DIGIT
024272 002402      BLT      4$          ;;BR IF DONE
024274 005202      INC      R2          ;;INCREASE THE BCD DIGIT BY 1
024276 000774      BR      3$
024300 060105      4$:      ADD      R1,R5      ;;ADD BACK THE CONSTANT
024302 005702      TST      R2          ;;CHECK IF BCD DIGIT=0
024304 001002      BNE      5$          ;;FALL THROUGH IF 0
024306 105716      TSTB   (SP)          ;;STILL DOING LEADING 0'S?
024310 100407      BMI      7$          ;;BR IF YES
024312 106316      5$:      ASLB   (SP)          ;;MSD?
024314 103003      BCC      6$          ;;BR IF NO
024316 116663 000001 177777  MOVB   1(SP),-1(R3)    ;;YES--SET THE SIGN
024324 052702 000060      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
024330 052702 000040      6$:      BIS      #'',R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024334 110223      MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024336 005720      TST      (R0)+      ;;JUST INCREMENTING
024340 020027 000010      CMP      R0,#10      ;;CHECK THE TABLE INDEX
024344 002746      BLT      2$          ;;GO DO THE NEXT DIGIT
024346 003002      BGT      8$          ;;GO TO EXIT
024350 010502      MOV      R5,R2      ;;GET THE LSD
024352 000764      BR      6$          ;;GO CHANGE TO ASCII
024354 105726      8$:      TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
024356 100003      BPL      9$          ;;BR IF NO
024360 116663 177777 177776  MOVB   -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
024366 105013      9$:      CLRB   (R3)      ;;SET THE TERMINATOR
024370 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
024372 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
024374 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
024376 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1

```

024400	012600		MOV	(SP)+,RO	::POP STACK INTO RO
024402	104401	C24430	TYPE	\$DBLK	::NOW TYPE THE NUMBER
024406	016666	C00002 000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
024414	012616		MOV	(SP)+,(SP)	
024416	000002		RTI		::RETURN TO USER
024420	023420		\$D*BL:	10000.	
024422	001750			1000.	
024424	000144			100.	
024426	000012			10.	
024430			\$DBLK:	.BLKW 4	

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

024440 017646 000000
024444 116637 000001 024663
024452 112637 024665
024456 062716 000002
024462 000406
024464 112737 000001 024663
024472 112737 000006 024665
024500 112737 000005 024662
024506 010346
024510 010446
024512 010546
024514 113704 024665
024520 005404
024522 062704 000006
024526 110437 024664
024532 113704 024663
024536 016605 000012
024542 005003
024544 006105 1$:
024546 000404 BR 3$
024550 006105 2$:
024552 006105 ROL R5
024554 006105 ROL R5
024556 010503 MOV R5,R3
024560 006103 3$:
024562 105337 024664 DECB $OMODE
024566 100016 BPL 7$
024570 042703 177770 BIC #177770,R3
024574 001002 BNE 4$
024576 005704 TST R4
024600 001403 BEQ 5$
024602 005204 4$: INC R4
  
```

```

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


024604	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
024610	052703	000040	5\$:	BIS	#' ,R3	::MAKE ASCII IF NOT ALREADY
024614	110337	024660		MOVB	R3,8\$::SAVE FOR TYPING
024620	104401	024660		TYPE	,8\$::GO TYPE THIS DIGIT
024624	105337	024662	7\$:	DECB	\$OCNT	::COUNT BY 1
024630	003347			RGT	2\$::BR IF MORE TO DO
024632	002402			BLT	6\$::BR IF DONE
024634	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
024636	000744			BR	2\$::GO DO THE LAST DIGIT
024640	012605		6\$:	MOV	(S?)+,R5	::RESTORE R5
024642	012604			MOV	(SP)+,R4	::RESTORE R4
024644	012603			MOV	(SP)+,R3	::RESTORE R3
024646	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
024654	012616			MOV	(SP)+,(SP)	
024656	000002			RTI		::RETURN
024660	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
024661	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
024662	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
024663	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
024664	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

```

024666 105737 001173 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
024672 100002 BPL 1$ ;;BR IF YES
024674 000000 HALT ;;HALT HERE IF NO TERMINAL
024676 000430 BR 3$ ;;LEAVE
024700 010046 1$: MOV RO,-(SP) ;;SAVE RO
024702 017600 000002 MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
024706 122737 000001 001242 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
024714 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
024716 132737 000100 001243 BITB #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
024724 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
024726 010037 024736 MOV RO,61$ ;;SETUP MESSAGE ADDRESS FOR APT
024732 004737 031300 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
024736 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
024740 132737 000040 001243 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
024746 001003 BNE 60$ ;;YES,SKIP TYPE OUT
024750 112046 2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
024752 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
024754 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
024756 012600 60$: MOV (SP)+,RO ;;RESTORE RO
024760 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
024764 000002 RTI ;;RETURN
024766 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
024772 001430 BEQ 8$
024774 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
025000 001006 BNE 5$
025002 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
025004 104401 TYPE ;;TYPE A CR AND LF
025006 001217 $CRLF
025010 105037 025216 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
025014 000755 BR 2$ ;;GET NEXT CHARACTER
025016 004737 025100 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
025022 123726 001172 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
025026 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
025030 013746 001170 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
025034 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
025040 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
025042 004737 025100 JSR PC,$TYPEC ;;GO TYPE A NULL
025046 105337 025216 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
025052 000770 BR 7$ ;;LOOP

```

:HORIZONTAL TAB PROCESSOR

```

025054 112716 000040      8$:  MOV#  #' , (SP)      ;; REPLACE TAB WITH SPACE
025060 004737 025100      9$:  JSR  PC, $TYPEC      ;; TYPE A SPACE
025064 132737 000007 025216 BIT#  #7, $CHARCNT      ;; BRANCH IF NOT AT
025072 001372          BNE  9$                ;; TAB STOP
025074 005726          TST  (SP)+          ;; POP SPACE OFF STACK
025076 000724          BR   2$                ;; GET NEXT CHARACTER
025100          $TYPEC:
025100 105777 154054      TSTB @ $TKS          ;; CHAR IN KYBD BUFFER?
025104 100022          BPL  10$          ;; BR IF NOT
025106 017746 154050      MOV  @ $TKB, -(SP)      ;; GET CHAR
025112 042716 177600      BIC  #177600, (SP)    ;; STRIP EXTRANEIOUS BITS
025116 122716 000023      CMPB # $XOFF, (SP)    ;; WAS CHAR XOFF
025122 001012          BNE  102$         ;; BR IF NOT
025124          101$:
025124 105777 154030      TSTB @ $TKS          ;; WAIT FOR CHAR
025130 100375          BPL  101$
025132 117716 154024      MOVB @ $TKB, (SP)      ;; GET CHAR
025136 042716 177600      BIC  #177600, (SP)    ;; STRIP IT
025142 122716 000021      CMPB # $XON, (SP)    ;; WAS IT XON?
025146 001366          BNE  101$          ;; BR IF NOT
025150          102$:
025150 005726          TST  (SP)+          ;; FIX STACK
025152          10$:
025152 105777 154006      TSTB @ $TPS          ;; WAIT UNTIL PRINTER IS READY
025156 100375          BPL  10$
025160 116677 000002 154000 MOVB 2(SP), @ $TPB    ;; LOAD CHAR TO BE TYPED INTO DATA REG.
025166 122766 000015 000002 CMPB #CR, 2(SP)      ;; IS CHARACTER A CARRIAGE RETURN?
025174 001003          BNE  1$                ;; BRANCH IF NO
025176 105037 025216      CLRB $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
025202 000406          BR   $TYPEX      ;; EXIT
025204 122766 000012 000002 1$: CMPB #LF, 2(SP)      ;; IS CHARACTER A LINE FEED?
025212 001402          BEQ  $TYPEX      ;; BRANCH IF YES
025214 105227          INCB (PC)+          ;; COUNT THE CHARACTER
025216 000000          $CHARCNT: .WORD 0      ;; CHARACTER COUNT STORAGE
025220 000207          $TYPEX: RFS  PC

```

.SBT'L SCOPE HANDLER ROUTINE

```

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

```

025222          $SCOPE:
025222 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
025224 032777 040000 153722 1$:      BIT      #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
025232 001402          BEQ      9$          ;;NO IF SW14=0
025234 000137 025664          JMP      $OVER          ;;JUMP OVER SCOPE ROUTINE
025240          9$:
;#####START OF CODE FOR THE XOR TESTER#####
025240 000416 $XTSTR: BR      6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
;;THIS INSTRUCTION TO A 'NOP' (NOP-240)
025242 013746 000004          MOV      @ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
025246 012737 025266 000004          MOV      #5$,@ERRVEC          ;;SET FOR TIMEOUT
025254 005737 177060          TST      @#177060          ;;TIME OUT ON XOR?
025260 012637 000004          MOV      (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
025264 000561          BR      $SVLAD          ;;GO TO THE NEXT TEST
025266 022626          5$:      CMP      (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
025270 012637 000004          MOV      (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
025274 000521          BR      7$          ;;LOOP ON THE PRESENT TEST
025276          6$:;#####END OF CODE FOR THE XOR TESTER#####
025276 032777 000400 153650          BIT      #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
025304 001421          BEQ      2$          ;;BR IF NO
025306 005046          CLR      -(SP)          ;;CLEAR A TEMP. LOCATION
025310 117716 153640          MOVB   @SWR,(SP)          ;;PICKUP THE DESIRED TEST NUMBER
025314 001414          BEQ      8$          ;;BRANCH IF BAD TEST NUMBER IN SWR
025316 022716 000024          CMP      #24,(SP)          ;;CHECK THE NUMBER IN THE SWR
025322 002411          BLT      8$          ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
025324 011637 001116          MOV      (SP),$TSTNM          ;;UPDATE THE TEST NUMBER
025330 005316          DEC      (SP)          ;;BACKUP BY ONE
025332 006316          ASL      (SP)          ;;SCALE THE TEST NUMBER AS AN INDEX
025334 062716 025702          ADD      #$$SW08TBL,(SP)          ;;FORM THE ADDRESS OF TEST POINTER
025340 013637 001122          MOV      @((SP)+,$LPADR)          ;;SET LOOP ADDRESS TO DESIRED TEST
025344 000547          BR      $OVER          ;;GO LOOP ON THE TEST
025346 005726          8$:      TST      (SP)+          ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
025350 105737 001117          2$:      TSTB   $ERFLG          ;;HAS AN ERROR OCCURRED?
025354 001502          BEQ      3$          ;;BR IF NO
025356 022737 177777 026342          CMP      #-1,CPSAVE          ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
025364 001455          BEQ      2003$          ;;KICK AROUND ROUTINE IF SO
025366 013746 000004          MOV      ERRVEC,-(SP)          ;;SAVE CONTENTS OF ERROR VECTOR
025372 012737 025410 000004          MOV      #2000$,ERRVEC          ;;SETUP 'TRAP' RETURN ADDRESS
025400 013737 177766 026342          MOV      177766,CPSAVE          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
025406 000406          BR      2001$
025410 012737 177777 026342 2000$:  MOV      #-1,CPSAVE          ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
025416 012716 025424          MOV      #2001$, (SP)          ;;SETUP RETURN ADDRESS
025422 000002          RTI
    
```

```

SCOPE HANDLER ROUTINE

025424 012637 000004      2001$: MOV      (SP)+,ERRVEC      ;;RESTORE CONTENTS OF ERROR VECTOR

025430 022737 177777 026342 2002$: CMP      #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
025436 001430          BEQ      2003$              ;;BRANCH IF SO
025440 032737 000001 026342      BIT      #BIT00,CPSAVE      ;;SEE IF THE POWER MONITOR BIT IS ON
025446 001424          BEQ      2003$              ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
025450 042737 000001 177766      BIC      #BIT00,177766     ;;CLEAR THE BIT FOUND TO BE SET
025456 013746 001154      MOV      SWR,-(SP)         ;;SAVE SWR ADDRESS
025462 017646 000000      MOV      @ (SP),-(SP)      ;;SAVE SWR VALUE
025466 012737 000176 001154      MOV      #176,SWR         ;;GET SOFTWARE SWR ADDRESS
025474 011677 153454      MOV      (SP),@SWR        ;;GET CURRENT SWR VALUE
025500 042777 001000 153446      BIC      #BIT09,@SWR       ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
025506 104177          EMT      177              ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
025510 012676 000000      MOV      (SP)+,@(SP)      ;;RESTORE SWR TO ORIGINAL VALUE
025514 012637 001154      MOV      (SP)+,SWR        ;;RESTORE SWR ADDRESS
025520          2003$:
025520 123737 001131 001117      CMPB     $ERMAX,$ERFLG    ;;MAX. ERRORS FOR THIS TEST OCCURRED?
025526 101015          BHI      3$                ;;BR IF NO
025530 032777 001000 153416      BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
025536 001404          BEQ      4$                ;;BR IF NO
025540 013737 001124 001122 7$:  MOV      $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
025546 000446          BR
025550          4$:
025554 005037 001206      CLRB     $ERFLG          ;;ZERO THE ERROR FLAG
025560 000415          CLR      $TIMES         ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
025562 032777 004000 153364 3$:  BIT      #BIT11,@SWR      ;;INHIBIT ITERATIONS?
025570 001011          BNE     1$                ;;BR IF YES
025572 005737 001230      TST     $PASS            ;;IF FIRST PASS OF PROGRAM
025576 001406          BEQ     1$                ;;INHIBIT ITERATIONS
025600 005237 001120      INC     $ICNT           ;;INCREMENT ITERATION COUNT
025604 023737 001206 001120      CMP     $TIMES,$:CNT     ;;CHECK THE NUMBER OF ITERATIONS MADE
025612 002024          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
025614 012737 000001 001120 1$:  MOV      #1,$:CNT        ;;REINITIALIZE THE ITERATION COUNTER
025622 013737 025700 001206      MOV     $MXCNT,$TIMES    ;;SET NUMBER OF ITERATIONS TO DO
025630 105237 001116      $SVLAD: INCB     $TSTNM    ;;COUNT TEST NUMBERS
025634 113737 001116 001226      MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
025642 011637 001122      MOV     (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
025646 011637 001124      MOV     (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
025652 005037 001210      CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
025656 112737 000001 001131      MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
025664 013777 001116 153264 $OVER: MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER
025672 013716 001122      MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
025676 000002          RTI
025700 000024      $MXCNT: 20.            ;;MAX. NUMBER OF ITERATIONS
025702          $SW08TBL:
          .REPT $TN-1
025702 005370          .WORD  TST1+2          ;;STARTING ADDRESS OF TEST 1
025704 005700          .WORD  TST2+2          ;;STARTING ADDRESS OF TEST 2
025706 006062          .WORD  TST3+2          ;;STARTING ADDRESS OF TEST 3
025710 006232          .WORD  TST4+2          ;;STARTING ADDRESS OF TEST 4
025712 006362          .WORD  TST5+2          ;;STARTING ADDRESS OF TEST 5
025714 007446          .WORD  TST6+2          ;;STARTING ADDRESS OF TEST 6
025716 010516          .WORD  TST7+2          ;;STARTING ADDRESS OF TEST 7
025720 010640          .WORD  TST10+2         ;;STARTING ADDRESS OF TEST 10
025722 010724          .WORD  TST11+2        ;;STARTING ADDRESS OF TEST 11
025724 011174          .WORD  TST12+2        ;;STARTING ADDRESS OF TEST 12
025726 011524          .WORD  TST13+2        ;;STARTING ADDRESS OF TEST 13

```

025730 012110
025732 012436
025734 013106
025736 013532
025740 013754
025742 014446
025744 015434
025746 016702
025750 020072

.WORD TST14+2
.WORD TST15+2
.WORD TST16+2
.WORD TST17+2
.WORD TST20+2
.WORD TST21+2
.WORD TST22+2
.WORD TST23+2
.WORD TST24+2

:: STARTING ADDRESS OF TEST 14
:: STARTING ADDRESS OF TEST 15
:: STARTING ADDRESS OF TEST 16
:: STARTING ADDRESS OF TEST 17
:: STARTING ADDRESS OF TEST 20
:: STARTING ADDRESS OF TEST 21
:: STARTING ADDRESS OF TEST 22
:: STARTING ADDRESS OF TEST 23
:: STARTING ADDRESS OF TEST 24

.SBTTL ERROP HANDLER ROUTINE

```

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

```

025752 105037 026344      $ERROR: CLRB      IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
025756 104410              CKSWR              ;;TEST FOR CHANGE IN SOFT-SWR
025760 105237 001117      7$:   INCB      $ERRFLG          ;;SET THE ERROR FLAG
025764 001775              BEQ          7$              ;;DON'T LET THE FLAG GO TO ZERO
025766 013777 001116 153162  MOV      $STMM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
025774 032777 002000 153152  BIT      #BIT10,@SWR      ;;BELL ON ERROR?
026002 001402              BEQ          1$              ;;NO - SKIP
026004 104401 001212              TYPE      $BELL            ;;RING BELL
026010 005237 001126      1$:   INC      $ERTTL          ;;COUNT THE NUMBER OF ERRORS
026014 011637 001132      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
026020 162737 000002 001132  SUB      #2,$ERRPC
026026 117737 153100 001130  MOVB    @ERRPC,$ITEMB    ;;STRIP AND SAVE THE ERROR ITEM CODE
026034 032777 001000 153112  BIT      #BIT09,@SWR      ;;SEE IF LOOP ON ERROR IS SET
026042 001060              BNE      1004$           ;;BRANCH AROUND ROUTINE IF SO
026044 122737 000177 001130  CPMB    #177,$ITEMB      ;;SEE IF THIS IS THE POWER FAIL CALL
026052 001454              BEQ      1004$           ;;BRANCH AROUND ROUTINE IF IT IS
026054 105737 026344      TSTB    IBSAVE           ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
026060 001047              BNE      1003$           ;;BRANCH IF SO
026062 022737 177777 026342  CMP      #-1,CPSAVE       ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
026070 001445              BEQ      1004$           ;;BRANCH IF SO
026072 013746 000004      MOV      ERRVEC,-(SP)     ;;SAVE CONTENTS OF ERROR VECTOR
026076 012737 026114 000004  MOV      #1000$,ERRVEC    ;;SETUP 'TRAP' RETURN ADDRESS
026104 013737 177766 026342  MOV      177766,CPSAVE    ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
026112 000406              BR          1001$
026114 012737 177777 026342 1000$: MOV      #-1,CPSAVE       ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
026122 012716 026130      MOV      #1001$,(SP)     ;;SETUP RETURN ADDRESS
026126 000002              RTI
026130 012637 000004      1001$: MOV      (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

026134 022737 177777 026342 1002$: CMP      #-1,CPSAVE       ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
026142 001420              BEQ      1004$           ;;BRANCH IF SO
026144 032737 000001 026342  BIT      #BIT00,CPSAVE    ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
026152 001414              BEQ      1004$           ;;BRANCH IF OK
026154 042737 000001 177766  BIC      #BIT00,177766    ;;CLEAR THE BIT FOUND SET
026162 113737 001130 026344  MOVB    $ITEMB,IBSAVE     ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
026170 112737 000177 001130  MOVB    #177,$ITEMB      ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
026176 000402              BR          1004$           ;;BRANCH OVER IBSAVE CLEARING

026200 105037 026344      1003$: CLRB      IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
026204 032777 020000 152742  1004$: BIT      #BIT13,@SWR    ;;SKIP TYPEOUT IF SET
026212 001004              BNE      20$              ;;SKIP TYPEOUTS
026214 004737 026346      JSR      PC,ERRYP        ;;GO TO USER ERROR ROUTINE
    
```

026220	104401	001217		TYPE	,\$SCLRF	
026224				20\$:		
026224	122737	000001	001242	CMPB	#APTENV,\$ENV	::RUNNING IN APT MODE
026232	001007			BNE	2\$::NO,SKIP APT ERROR REPORT
026234	113737	001130	026246	MOVB	\$ITEMB,21\$::SET ITEM NUMBER AS ERROR NUMBER
026242	004737	031310		JSR	PC,\$ATV4	::REPORT FATAL ERROR TO APT
026246	000			21\$:	.BYTE	0
026247	000				.BYTE	0
026250	000777			22\$:	BR	22\$
026252	105737	026344		2\$:	TSTB	IBSAVE
026256	001005				BNE	3\$
026260	005777	152670			TST	@SWR
026264	100002				BPL	3\$
026266	000000				HALT	
026270	104410				CKSWR	::TEST FOR CHANGE IN SOFT-SWR
026272				3\$:		
026272	032777	001000	152654	BIT	#BIT09,@SWR	::LOOP ON ERROR SWITCH SET?
026300	001402			BEQ	4\$::BR IF NO
026302	013716	001124		MOV	\$LPERR,(SP)	::FUDGE RETURN FOR LOOPING
026306	005737	001210		4\$:	TST	\$ESCAPE
026312	001402				BEQ	5\$
026314	013716	001210			MOV	\$ESCAPE,(SP)
026320				5\$:		
026320	022737	021530	000042	CMP	#SENDAD,@#42	::ACT-11 AUTO-ACCEPT?
026326	001001			BNE	6\$::BRANCH IF NO
026330	000000				HALT	::YES
026332				6\$:		
026332	105737	026344		TSTB	IBSAVE	::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
026336	001210			BNE	7\$::BRANCH BACK TO CALL ORIGINAL ERROR
026340	000002				RTI	::RETURN
026342	000000			PSAVE:	.WORD	0
026344	000000			IBSAVE:	.WORD	0
						::LOCATION TO SAVE CPU ERROR REG CONTENTS
						::LOCATION TO SAVE ITEM BYTE

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

```

.SBITL ERROR TYPEOUT ROUTINE

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

ERRRYP: SAVREG
        BIT     #SW13,@SWR      ;INHIBIT TYPEOUTS??
        BEQ    1$              ;NO!!
        JMP    27$             ;YES!.

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

;TYPE 'DRIVE TYPE' RMO5, RMO3 OR RMO2 FOR UNIT UNDER TEST
        MOV    $BASE,R0        ;GET RM BASE ADDRESS
        MOV    RMDT(R0),R0     ;GET DRIVE TYPE REGISTER
        BIC   #177740,R0      ;SAVE DRIVE TYPE BITS AND
        MOV    #RMO3,3$       ;GET ASCII DRIVE TYPE
        CMP   #24,R0          ;IS DEVICE AN RMO3 ?
        BEQ   2$              ;YES !!

        MOV    #RMO2,3$       ;SAVE ASCII DRIVE TYPE
        CMP   #25,R0          ;IS DEVICE AN RMO2 ?
        BEQ   2$              ;YES !!

        MOV    #RMO5,3$       ;SAVE ASCII DRIVE TYPE
        CMP   #27,R0          ;IS DEVICE AN RMO5 ?
        BNE   4$              ;NO !!
        TYPE  " - "           ;TYPE " - "
        TYPE  ,ERTY05         ;TYPE DRIVE TYPE
        TYPE  ,ERTY05         ;TYPE DRIVE TYPE
        .WORD 0               ;DRIVE TYPE MESSAGE IS STORED HERE

;TYPE TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER
4$:      CLR    TSTNMB          ;LOAD TEST NUMBER FOR
        MOV    $TESTN,TSTNMB
        TYPE  ,ERTY01          ;TYPE 'TST#'
        MOV    TSTNMB,-(SP)    ;SAVE TSTNMB FOR TYPEOUT
                                ;TYPE TEST NUMBER
                                ;GO TYPE--OCTAL ASCII
        TYPOS  ;TYPE 3 DIGIT(S)
        .BYTE  3
        .BYTE  0               ;SUPPRESS LEADING ZEROS
        CLR   ERRNMB          ;LOAD ERROR NUMBER FOR
        MOVB  $ITEMB,FRRNMB    ;TYPEOUT
    
```

```

026346 104414
026350 032777 020000 152576
026356 001402
026360 000137 027206

026400 104403
026402 003
026403 000

026404 013700 001276
026410 016000 000026
026414 042700 177740
026420 012737 032211 026472
026426 022700 000024
026432 001414

026434 012737 032204 026472
026442 022700 000025
026446 001406

026450 012737 032216 026472
026456 022700 000027
026462 001004
026464 104401 027257
026470 104401
026472 000000

026516 104403
026520 003
026521 000

026522 005037 027214
026526 113737 001130 027214
    
```

```

50 026534 001406 BEQ 5$ :SKIP IF NO ERROR CALLED
51 026536 104401 027237 TYPE ,ERTY02 :TYPE 'ERR#'
52 026542 013746 027214 MOV ERRNMB,-(SP) :SAVE ERRNMB FOR TYPEOUT
:TYPE ERROR NUMBER
026546 104403 TYPOS :GO TYPE--OCTAL ASCII
026550 003 .BYTE 3 :TYPE 3 DIGIT(S)
026551 000 .BYTE 0 :SUPPRESS LEADING ZEROS
53 026552 104401 027246 5$: TYPE ,ERTY03 :TYPE 'PC='
54 026556 013746 001132 MOV $ERRPC,-(SP) :SAVE $ERRPC FOR TYPEOUT
:TYPE PROGRAM COUNTER
026562 104403 TYPOS :GO TYPE--OCTAL ASCII
026564 006 .BYTE 6 :TYPE 6 DIGIT(S)
026565 001 .BYTE 1 :TYPE LEADING ZEROS
55
56 :GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
57 026566 005737 027214 6$: TST ERRNMB :WAS AN ERROR CALLED?
58 026572 001002 BNE 7$ :BR IF YES
59 026574 000137 027206 JMP 27$ :NO--EXIT
60
61 026600 104401 001217 7$: TYPE ,$CRLF :YES--TYPE CRLF
62 026604 105037 027220 CLRB BOTFLG :CLEAR BOT FLAG
63 026610 105037 027221 CLRB CHR CNT :CLEAR CHARACTER COUNTER
64 026614 013700 027214 MOV ERRNMB,R0 :R0 POINTS TO FIRST OF
65 026620 122700 000177 CMPB #177,R0 :SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
66 026624 001003 BNE 8$ :BRANCH IF NOT
67 026626 012700 027264 MOV #PFECB,R0 :MOVE POWER FAIL ERROR CALL TABLE TO R0
68 026632 000405 BR 9$
69 026634 006300 8$: ASL R0 :FOUR ENTRIES IN ERROR
70 026636 006300 ASL R0 :TABLE
71 026640 006300 ASL R0
72 026642 062700 001532 ADD #$ERRTB-8.,R0
73 026646 011001 9$: MOV (R0),R1 :R1 POINTS TO ERROR MESSAGE
74 :TABLE
75 026650 001507 BEQ 19$ :BRANCH IF NO ERROR MESSAGE
76
77 :TYPE THE ERROR MESSAGE
78 026652 012102 10$: MOV (R1)+,R2 :R2=ADDRESS OF MESSAGE STRING
79 026654 001505 BEQ 19$ :BRANCH IF END OF MESSAGE
80 026656 010237 027024 MOV R2,18$ :LOAD ADDRESS OF STRING
81 026662 005037 027216 CLR BOTADR :CLEAR BOT ADDRESS
82 026666 112203 11$: MOVB (R2)+,R3 :END OF STRING??
83 026670 001454 BEQ 17$ :YES!!
84 026672 122703 000015 CMPB #CR,R3 :CARRIAGE RETURN??
85 026676 001003 BNE 12$ :NO!!
86 026700 105037 027221 CLRB CHR CNT :YES--CLEAR CHAR COUNT
87 026704 000770 BR 11$ :GET NEXT CHARACTER
88 026706 122703 000012 12$: CMPB #LF,R3 :LINE FEED??
89 026712 001765 BEQ 11$ :YES--GET NEXT CHARACTER
90 026714 122703 000011 CMPB #HT,R3 :HORIZONTAL TAB??
91 026720 001007 BNE 14$ :NO!!
92 026722 105237 027221 13$: INCB CHR CNT :ADJUST CHARACTER COUNT
93 026726 132737 000007 027221 BITB #7,CHR CNT
94 026734 001372 BNE 13$
95 026736 000407 BR 15$
96 026740 105237 027221 14$: INCB CHR CNT :INCREMENT CHARACTER COUNT
97 026744 122703 000040 CMPB #' ,R3 :SPACE??
98 026750 001002 BNE 15$ :NO..
    
```

```

99 026752 010237 027216      MOV      R2,BOTADR      ;SAVE ADDRESS OF SPACE
100 026756 122737 C00100 027221 15$: CMPB    #64.,CHRCNT    ;END OF LINE??
101 026764 103340          BHI     11$            ;NO!!
102 026766 013704 027216      MOV      BOTADR,R4      ;GET ADDRESS OF LAST SPACE
103 026772 001007          BNE     16$            ;BRANCH IF SPACE DETECTED
104 026774 104401 001217      TYPE    ,$CRLF         ;TYPE CRLF
105 027000 105037 027221      CLRB    CHRCNT         ;CLEAR CHARACTER COUNT
106 027004 013702 027024      MOV      18$,R2        ;SET UP R2 FOR TESTING
107 027010 000726          BR      11$
108 027012 105044          CLRB    -(14)         ;REPLACE SPACE
109 027014 112737 177777 027220 16$: MOVB   #-1,BOTFLG    ;SET BOT FLAG
110 027022 104401          TYPE    ,TYPE ERROR MESSAGE STRING
111 027024 000000          18$: .WORD  ;STRING ADDRESS GOES HERE
112 027026 105737 027220      TSTB    BOTFLG        ;WAS STRING TRUNCATED??
113 027032 001707          BEQ     10$            ;NO!!
114 027034 104401 001217      TYPE    , $CRLF       ;YES-TYPE CRLF
115 027040 105037 027220      CLRB    BOTFLG        ;CLEAR BOT FLAG
116 027044 105037 027221      CLRB    CHRCNT        ;CLEAR CHARACTER COUNT
117 027050 013702 027216      MOV      BOTADR,R2     ;SETUP R2 FOR TESTING
118 027054 010237 027024      MOV      R2,18$        ;SETUP 18$ FOR TYPING
119 027060 112742 000040      MOVB    #'-',(R2)     ;RESTORE SPACE
120 027064 105722          TSTB    (R2)+         ;RESTORE R2
121 027066 000677          BR      11$            ;TYPE REST OF STRING
122
123
124 027070 016001 000002      ;TYPE ERROR HEADER AND ERROR DATA
125 027074 001444          19$: MOV      2(R0),R1    ;R1 POINTS TO ERROR HEADER TABLE
126 027076 104401 001217      BEQ     27$            ;BRANCH IF NO HEADER
127 027102 016002 000004      TYPE    , $CRLF       ;(ASSUME NO DATA)
128 027106 016003 000006      MOV      4(R0),R2      ;R2 POINTS TO DATA ADDRESS TABLE
129 027112 012137 027122      MOV      6(R0),R3      ;R3 POINTS TO FORMAT TABLE
130 027116 001433          20$: MOV      (R1)+,21$  ;PUT HEADER ADDRESS FOR TYPE
131          BEQ     27$            ;BRANCH IF END OF HEADERS
132          ;(ASSUME END OF DATA)
133 027120 104401          TYPE    , $CRLF       ;HEADER ADDRESS GOES HERE
134 027124 000000          21$: .WORD  0
135 027130 005702          TYPE    , $CRLF
136 027132 001767          TST     R2            ;DATA WITH HEADER??
137 027134 012204          BEQ     20$            ;NO!!
138 027136 012305          MOV     (R2)+,R4      ;R4 POINTS TO DATA ADDRESS
139 027140 105725          MOV     (R3)+,R5      ;R5 POINTS TO FORMAT
140 027142 100407          22$: TSTB    (R5)+         ;WHAT KIND OF DATA??
141 027144 001403          SMI     24$            ;BINARY
142 027146 013446          BEQ     23$            ;OCTAL
143 027150 104405          MOV     @ (R4)+,-(SP) ;DECIMAL
144 027152 000405          TYPDS
145 027154 013446          BR      25$
146 027156 104402          23$: MOV     @ (R4)+,-(SP)
147 027160 000402          TYPDC
148 027162 013446          BR      25$
149 027164 104406          24$: MOV     @ (R4)+,-(SP)
150 027166 005714          TYPBN
151 027170 001403          25$: TST     (R4)         ;MORE DATA??
152 027172 104401 027254          BEQ     26$            ;NO!!
153 027176 000760          TYPE    ,ERTY04      ;YES-TYPE 2 SPACES
154 027200 104401 001217      BR      22$            ;AND CONTINUE
155 027204 000742          26$: TYPE    , $CRLF     ;TYPE ONE BLANK LINE
          BR      20$            ;BEFORE NEXT HEADER

```

156	027206	104415		278:	RESREG		
157	027210	000207			RTS	PC	
158							
159	027212	000000		TSTNMB:	.WORD	0	;TEST NUMBER
160	027214	000000		ERRNMB:	.WORD	0	;ERROR NUMBER
161	027216	000000		BOTADR:	.WORD	0	;BEGINNING OF TEXT ADDRESS
162	027220	000		BOTFLG:	.BYTE	0	;BOT FLAG
163	027221	000		CHRCNT:	.BYTE	0	;CHARACTER COUNT
164							
165	027222	104	122	126	ERTY00:	.ASCIZ	@D'V#@
166	027227	054	040	124	ERTY01:	.ASCIZ	@, TEST#@
167	027237	054	040	105	ERTY02:	.ASCIZ	@, ERR#@
168	027246	054	040	120	ERTY03:	.ASCIZ	@, PC=@
169	027254	040	040	000	ERTY04:	.ASCIZ	@ @
170	027257	040	055	040	ERTY05:	.ASCIZ	@ - @
171					.EVEN		
172	027264	027274	027362	027400	PFECH:	PFECH1,PFECH2,PFECH3,PFECH4	;WORDS DEFINING TABLES BELOW
173	027274	027300	000000		PFECH1:	.+4,0	
174	027300	120	117	127		.ASCIZ	?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
175					.EVEN		
176	027362	027366	000000		PFECH2:	.+4,0	
177	027366	103	120	125		.ASCIZ	?CPUERREG?
178					.EVEN		
179	027400	027402			PFECH3:	.+2	
180	027402	026342	000000			.WORD	CPSAVE,0
181	027406	027410			PFECH4:	.+2	
182	027410	000	000			.BYTE	0,0

```

1
.SBTTL TTY INPUT ROUTINE
:*****
027412 000000 .ENABL LSB
027414 000000 $TKCNT: .WORD 0 ;;NUMBER OF ITEMS IN QUEUE
027416 000000 $TKQIN: .WORD 0 ;;INPUT POINTER
027420 027421 $TKQOUT: .WORD 0 ;;OUTPUT POINTER
$TKQSRT: .BLKB 1 ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
:
:*CALL:
:* JSR PC,$TKINT
:* RETURN
:
027422 005037 027412 $TKINT: CLR $TKCNT ;;CLEAR COUNT OF ITEMS IN QUEUE
027426 012737 027420 027414 MOV #TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
027434 013737 027414 027416 MOV $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
027442 012737 027472 000060 MOV #TKSRV,@TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
027450 012737 000200 000062 MOV #200,@TKVEC+2 ;;'BR' LEVEL 4
027456 005777 151500 TST @TKB ;;CLEAR DONE FLAG
027462 012777 000100 151470 MOV #100,@TKS ;;ENABLE TTY KEYBOARD INTERRUPT
027470 000207 RTS PC ;;RETURN TO CALLER

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (SHUT)
:
027472 117746 151464 $TKSRV: MOVB @TKB,-(SP) ;;PICKUP THE CHARACTER
027476 042716 177600 BIC #C177,(SP) ;;STRIP THE JUNK
027502 021627 000021 CMP (SP),#XON ;;IS IT A RANDOM XON?
027506 001002 BNE 30$ ;;BRANCH IF NO
027510 005726 TST (SP)+ ;;CLEAN RANDOM XON OFF STACK
027512 000002 RTI ;;RETURN
027514
30$:
027514 021627 000003 CMP (SP),#3 ;;IS IT A CONTROL C?
027520 001007 BNE 1$ ;;BRANCH IF NO
027522 104401 030620 TYPE ,SCNTLC ;;TYPE A CONTROL-C (^C)
027526 004737 027422 JSR PC,$TKINT ;;INIT THE KEYBOARD
027532 005726 TST (SP)+ ;;CLEAN UP STACK
027534 000137 030662 JMP SHUT ;;CONTROL C RESTART
027540 021627 000007 1$: CMP (SP),#7 ;;IS IT A CONTROL G?
027544 001004 BNE 2$ ;;BRANCH IF NO
027546 022737 000176 001154 CMP #SWREG,SWR ;;IS SOFT-SWR SELECTED?
027554 001500 BEQ 6$ ;;GO TO SWR CHANGE

2$:
027556
027556 022737 000001 027412 CMP #1,$TKCNT ;;IS THE QUEUE FULL?
027564 001004 BNE 3$ ;;BRANCH IF NO
027566 104401 001212 TYPE ,SBELL ;;RING THE TTY BELL

```

```

027572 005726          TST      (SP)+          ;;CLEAN CHARACTER OFF OF STACK
027574 000451          BR       5$              ;;EXIT
027576 021627 000023  3$:    CMP      (SP),#23          ;;IS IT A CONTROL-S?
027602 001021          BNE      32$              ;;BRANCH IF NO
027604 005077 151350  CLR      @STKS          ;;DISABLE TTY KEYBOARD INTERRUPTS
027610 005726          TST      (SP)+          ;;CLEAN CHAR OFF STACK
027612 105777 151342  31$:   TSTB     @STKS          ;;WAIT FOR A CHAR
027616 100375          BPL      31$              ;;LOOP UNTIL ITS THERE
027620 117746 151336  MOVB     @STKB,-(SP)      ;;GET THE CHARACTER
027624 042716 177600  BIC      #'C177,(SP)    ;;MAKE IT 7-BIT ASCII
027630 022627 000021  CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
027634 001366          BNE      31$              ;;BRANCH IF NO
027636 012777 000100 151314  MOV      #100,@STKS     ;;REENABLE TTY KEYBOARD INTERRUPTS
027644 000002          RTI                      ;;RETURN
027646 005237 027412  32$:   INC      $TKCNT        ;;COUNT THIS CHARACTER
027652 021627 000140  CMP      (SP),#140      ;;IS IT UPPER CASE?
027656 002405          BLT      4$              ;;BRANCH IF YES
027660 021627 000175  CMP      (SP),#175      ;;IS IT A SPECIAL CHAR?
027664 003002          BGT      4$              ;;BRANCH IF YES
027666 042716 000040  BIC      #40,(SP)       ;;MAKE IT UPPER CASE
027672 112677 177516  4$:   MOVB     (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
027676 005237 027414  INC      $TKQIN        ;;UPDATE THE POINTER
027702 023727 027414 027421  CMP      $TKQIN,$STKQEND ;;GO OFF THE END?
027710 001003          BNE      5$              ;;BRANCH IF NO
027712 012737 027420 027414  MOV      #$TKQSR,$STKQIN ;;RESET THE POINTER
027720 000002          RTI                      ;;RETURN
  
```

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

```

027722 022737 000176 001154  $CKSWR: CMP      #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
027730 001124          BNE      15$              ;;EXIT IF NOT
027732 105777 151222  TSTB     @STKS          ;;IS A CHAR WAITING?
027736 100121          BPL      15$              ;;IF NOT, EXIT
027740 117746 151216  MOVB     @STKB,-(SP)    ;;YES
027744 042716 177600  BIC      #'C177,(SP)    ;;MAKE IT 7-BIT ASCII
027750 021627 000007  CMP      (SP),#7        ;;IS IT A CONTROL-G?
027754 001300          BNE      2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                          ;;AND EXIT
  
```

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

```

027756 123727 001150 000001  6$:   CMPEB    $AUTOB,#1     ;;ARE WE RUNNING IN AUTO-MODE?
027764 001674          BEQ      2$              ;;BRANCH IF YES
027766 005726          TST      (SP)+          ;;CLEAR CONTROL-G OFF STACK
027770 004737 027422  JSR      PC,$TKINT      ;;FLUSH THE TTY INPUT QUEUE
027774 005077 151160  CLR      @STKS          ;;DISABLE TTY KEYBOARD INTERRUPTS
030000 112737 000001 001151  MOVB     #1,$INTAG      ;;SET INTERRUPT MODE INDICATOR

030006 104401 030632  $GTSWR: TYPE     ,$CNTLG      ;;ECHO THE CONTROL-G (^G)
030012 104401 030637  TYPE     ,$MSWR        ;;TYPE CURRENT CONTENTS
030016 013746 000176  MOV      SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
030022 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  
```

```

030024 104401 030650          TYPE      .SMNEW      ;;PROMPT FOR NEW SWR
030030 005046          CLR        -(SP)      ;;CLEAR COUNTER
030032 005046          CLR        -(SP)      ;;THE NEW SWR
030034 105777 151120      7$:      TSTB      @STKS      ;;CHAR THERE?
030040 100375          BPL        7$        ;;IF NOT TRY AGAIN

030042 117746 151114          MOVB      @STKB,-(SP)  ;;PICK UP CHAR
030046 042716 177600          BIC        #^C17,(SP)  ;;MAKE IT 7-BIT ASCII

030052 021627 000003          CMP        (SP),#3     ;;IS IT A CONTROL-C?
030056 001015          BNE        9$        ;;BRANCH IF NOT
030060 104401 030620          TYPE      .SCNTLC    ;;YES, ECHO CONTROL-C (^C)
030064 062706 000006          ADD        #6,SP      ;;CLEAN UP STACK
030070 123727 001151 000001  CMPB      $INTAG,#1    ;;REENABLE TTY KEYBOARD INTERRUPTS?
030076 001003          BNE        8$        ;;BRANCH IF NO
030100 012777 000100 151052  MOV        #100,@STKS  ;;ALLOW TTY KEYBOARD INTERRUPTS
030106 000137 030662      8$:      JMP        SHUT        ;;CONTROL-C RESTART

030112 021627 000025      9$:      CMP        (SP),#25    ;;IS IT A CONTROL-U?
030116 001005          BNE        10$       ;;BRANCH IF NOT
030120 104401 030625          TYPE      .SCNTLU    ;;YES, ECHO CONTROL-U (^U)
030124 062706 000006      20$:     ADD        #6,SP      ;;IGNORE PREVIOUS INPUT
030130 000737          BR         19$       ;;LET'S TRY IT AGAIN

030132 021627 000015      10$:     CMP        (SP),#15    ;;IS IT A <CR>?
030136 001022          BNE        16$       ;;BRANCH IF NO
030140 005766 000004          TST        4(SP)     ;;YES, IS IT THE FIRST CHAR?
030144 001403          BEQ        11$       ;;BRANCH IF YES
030146 016677 000002 151000  MOV        2(SP),@SWR  ;;SAVE NEW SWR
030154 062706 000006      11$:     ADD        #6,SP      ;;CLEAN UP STACK
030160 104401 001217      14$:     TYPE      .SCRLF    ;;ECHO <CR> AND <LF>
030164 123727 001151 000001  CMPB      $INTAG,#1    ;;RE-ENABLE TTY KBD INTERRUPTS?
030172 001003          BNE        15$       ;;BRANCH IF NOT
030174 012777 000100 150756  MOV        #100,@STKS  ;;RE-ENABLE TTY KBD INTERRUPTS
030202 000002          RTI         ;;RETURN
030204 004737 025100      16$:     JSR        PC,$TYPEC  ;;ECHO CHAR
030210 021627 000060          CMP        (SP),#60  ;;CHAR < 0?
030214 002420          BLT        18$       ;;BRANCH IF YES
030216 021627 000067          CMP        (SP),#67  ;;CHAR > 7?
030222 003015          BGT        18$       ;;BRANCH IF YES
030224 042726 000060          BIC        #60,(SP)+ ;;STRIP-OFF ASCII
030230 005766 000002          TST        2(SP)     ;;IS THIS THE FIRST CHAR
030234 001403          BEQ        17$       ;;BRANCH IF YES
030236 006316          ASL        (SP)     ;;NO, SHIFT PRESENT
030240 006316          ASL        (SP)     ;;  CHAR OVER TO MAKE
030242 006316          ASL        (SP)     ;;  ROOM FOR NEW ONE.
030244 005266 000002      17$:     INC        2(SP)     ;;KEEP COUNT OF CHAR
030250 056616 177776          BIS        -2(SP),(SP) ;;SET IN NEW CHAR
030254 000667          BR         7$        ;;GET THE NEXT ONE
030256 104401 001216      18$:     TYPE      .SQUES    ;;TYPE ?<CR><LF>
030262 000720          BR         20$     ;;SIMULATE CONTROL-U
.DSABL  LSB

```

```

: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *      RDCHR          ;; GET A CHARACTER FROM THE QUEUE
: *      RETURN HERE   ;; CHARACTER IS ON THE STACK
: *                   ;; WITH PARITY BIT STRIPPED OFF
:
030264 011646          SRDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC AND
030266 016666 000004 000002      MOV      4(SP),2(SP)      ;; THE PS
030274 005066 000004          CLR      4(SP)          ;; GET READY FOR A CHARACTER
030300 005046          CLR      -(SP)          ;; PUT NEW PS ON STACK
030302 012746 030310      MOV      #64$,-(SP)      ;; PUT NEW PC ON STACK
030306 000002          RTI                   ;; POP NEW PC AND PS
030310
030310 005737 027412      64$:   TST      $TKCNT          ;; WAIT ON A CHARACTER
030314 001775          1$:   BEQ      1$
030316 005337 027412      DEC      $TKCNT          ;; DECREMENT THE COUNTER
030322 117766 177070 000004      MOVB   @STKQOUT,4(SP)      ;; GET ONE CHARACTER
030330 005237 027416      INC      $TKQOUT          ;; UPDATE THE POINTER
030334 023727 027416 027421      CMP    $TKQOUT,#$TKQEND  ;; DID IT GO OFF OF THE END?
030342 001003          BNE     2$
030344 012737 027420 027416      MOV    #$TKQRT,$TKQOUT  ;; RESET THE POINTER
030352 000002          RTI                   ;; RETURN
:
: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:
: *      RDLIN          ;; INPUT A STRING FROM THE TTY
: *      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: *                   ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
:
030354 010346          SRDLIN: MOV      R3, -(SP)          ;; SAVE R3
030356 005046          CLR      -(SP)          ;; CLEAR THE RUBOUT KEY
030360 012703 030610      1$:   MOV      #$TTYIN,R3      ;; GET ADDRESS
030364 022703 030620      2$:   CMP      #$TTYIN+8.,R3    ;; BUFFER FULL?
030370 101456          BLOS   4$
030372 104411          RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
030374 112613          MOVB   (SP)+,(R3)      ;; GET CHARACTER
030376 122713 000177      10$:  CMPB   #177,(R3)      ;; IS IT A RUBOUT
030402 001022          BNE     5$
030404 005716          TST      (SP)          ;; IS THIS THE FIRST RUBOUT?
030406 001007          BNE     6$
030410 112737 000134 030606      MOVB   #' \ ,9$        ;; TYPE A BACK SLASH
030416 104401 030606      TYPE   ,9$
030422 012716 177777      MOV    #-1,(SP)        ;; SET THE RUBOUT KEY
030426 005303          6$:   DEC      R3            ;; BACKUP BY ONE
030430 020327 030610      CMP    R3,$$          ;; STACK EMPTY?
030434 103434          BLO   4$
030436 111337 030606      MOVB   (R3),9$        ;; SETUP TO TYPEOUT THE DELETED CHAR.
030442 104401 030606      TYPE   ,9$
030446 000746          BR     2$
030450 005716          5$:   TST      (SP)          ;; GO READ ANOTHER CHAR.
030452 001406          BEQ    7$
030454 112737 000134 030606      MOVB   #' \ ,9$        ;; RUBOUT KEY SET?
030462 104401 030606      TYPE   ,9$          ;; BR IF NO
030466 005016          CLR      (SP)          ;; TYPE A BACK SLASH
030470 122713 000025      7$:   CMPB   #25,(R3)      ;; CLEAR THE RUBOUT KEY
030474 001003          BNE     8$          ;; IS CHARACTER A CTRL U?
:
: *                   ;; BR IF NO

```



```

030476 104401 03062  TYPE      , $CNTLU      ;; TYPE A CONTROL 'U'
030502 000726          BR      1$           ;; GO START OVER
030504 122713 000022  8$:      CMPB     #22,(R3)  ;; IS CHARACTER A '^R'?
030510 001011          BNE     3$           ;; BRANCH IF NO
030512 105013          CLRB     (R3)        ;; CLEAR THE CHARACTER
030514 104401 001217  TYPE      , $CRLF      ;; TYPE A 'CR' & 'LF'
030520 104401 030610  TYPE      , $TTYIN     ;; TYPE THE INPUT STRING
030524 000717          BR      2$           ;; GO PICKUP ANOTHER CHACTER
030526 104401 001216  4$:      TYPE      , $QUES   ;; TYPE A '?'
030532 000712          BR      1$           ;; CLEAR THE BUFFER AND LOOP
030534 111337 030606  3$:      MOVB     (R3),9$    ;; ECHO THE CHARACTER
030540 104401 030606  TYPE      , 9$
030544 122723 000015  CMPB     #15,(R3)+    ;; CHECK FOR RETURN
030550 001305          BNE     2$           ;; LOOP IF NOT RETURN
030552 105063 177777  CLRB     -1(R3)      ;; CLEAR RETURN (THE 15)
030556 104401 001220  TYPE      , $LF       ;; TYPE A LINE FEED
030562 005726          TST     (SP)+    ;; CLEAN RUBOUT KEY FROM THE STACK
030564 012603          MOV     (SP)+,R3    ;; RESTORE R3
030566 011646          MOV     (SP)-,(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
030570 016666 0C0004 000002  MOV     4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
030576 012766 030610 000004  MOV     # $TTYIN,4(SP)
030604 000002          RTI           ;; RETURN
030606 000          9$:      .BYTE    0           ;; STORAGE FOR ASCII CHAR. TO TYPE
030607 000          .BYTE    0           ;; TERMINATOR
030610          $TTYIN: .BLKB    8           ;; RESERVE 8 BYTES FOR TTY INPUT
030620 136 103 015 $CNTLC: .ASCIZ  /^C/<15><12>  ;; CONTROL 'C'
030625 136 125 015 $CNTLU: .ASCIZ  /^U/<15><12>  ;; CONTROL 'U'
030632 136 107 015 $CNTLG: .ASCIZ  /^G/<15><12>  ;; CONTROL 'G'
030637 015 012 123 $MSWR: .ASCIZ  <15><12>/SWR = /
030650 040 040 116 $MNEW: .ASCIZ  / NEW = /
          .EVEN

2
3 030662 005737 000042  SHUT:    TST     @#42      ;; ANY MONITOR PRESENT ?
4 030666 001002          BNE     1$           ;; BR IF YES
5 030670 000137 002732  JMP     START        ;; GO TO START
6 030674 005037 001300  1$:     CLR     $DEVM    ;; FUDGE NO DRIVES IN MAP
7 030700 000137 021332  JMP     $EOP        ;; RETURN TO $EOP
  
```

1

.SBTTL READ AN OCTAL NUMBER FROM THE TTY

```

*****
*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
*CHANGE IT TO BINARY.
*CALL:
*      RDOCT          ;;READ AN OCTAL NUMBER
*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
*                   ;;HIGH ORDER BITS ARE IN $HIOCT

030704 011646          000004 000002 $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
030706 016666          000004 000002      MOV      4(SP),2(SP)      ;;INPUT NUMBER
030714 010046          000004 000002      MOV      R0,-(SP)        ;;PUSH R0 ON STACK
030716 010146          000004 000002      MOV      R1,-(SP)        ;;PUSH R1 ON STACK
030720 010246          000004 000002      MOV      R2,-(SP)        ;;PUSH R2 ON STACK
030722 104412          000004 000002 1$:  RDLIN          ;;READ AN ASCII LINE
030724 012600          000004 000002      MOV      (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
030726 005001          000004 000002      CLR      R1              ;;CLEAR DATA WORD
030730 005002          000004 000002      CLR      R2
030732 112046          000004 000002 2$:  MOVB      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
030734 001412          000004 000002      BEC      3$              ;;IF ZERO GET OUT
030736 006301          000004 000002      ASL      R1              ;;*2
030740 006102          000004 000002      ROL      R2
030742 006301          000004 000002      ASL      R1              ;;*4
030744 006102          000004 000002      ROL      R2
030746 006301          000004 000002      ASL      R1              ;;*8
030750 006102          000004 000002      ROL      R2
030752 042716          000004 000002 177770 BIC      #'C7,(SP)        ;;STRIP THE ASCII JUNK
030756 062601          000004 000002      ADD      (SP)+,R1        ;;ADD IN THIS DIGIT
030760 000764          000004 000002      BR       2$              ;;LOOP
030762 005726          000004 000002 3$:  TST      (SP)+          ;;CLEAN TERMINATOR FROM STACK
030764 010166          000012 000002      MOV      R1,12(SP)       ;;SAVE THE RESULT
030770 010237          000004 000002      MOV      R2,$HIOCT
030774 012602          000004 000002      MOV      (SP)+,R2        ;;POP STACK INTO R2
030776 012601          000004 000002      MOV      (SP)+,R1        ;;POP STACK INTO R1
031000 012600          000004 000002      MOV      (SP)+,R0        ;;POP STACK INTO R0
031002 000002          000004 000002      RTI
031004 000000          000004 000002 $HIOCT: .WORD 0          ;;HIGH ORDER BITS GO HERE
  
```

.SBTTL TRAP DECODER

```

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

```

```

031006 016646 000002 $TRAP: MOV 2(SP),-(SP) ;;ASSUME THE STATUS OF
031012 042716 000020 BIC #20,(SP) ;; THE CALLER--DO NOT ALLOW
031016 012746 031024 MOV #1$,-(SP) ;; T-BIT TRAPS
031022 000002 RTI ;;SET THE NEW STATUS
031024 010046 1$: MOV R0,-(SP) ;;SAVE R0
031026 016600 000002 MOV 2(SP),R0 ;;GET TRAP ADDRESS
031032 005740 TST -(R0) ;;BACKUP BY 2
031034 111000 MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
031036 006300 ASL R0 ;;POSITION FOR INDEXING
031040 016000 031060 MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
031044 000200 RTS R0 ;;GO TO ROUTINE

```

```

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

```

```

031046 011646 $TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
031050 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
031056 000002 RTI ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

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

```

```

: ROUTINE
:-----
031060 031046 $TRPAD: .WORD $TRAP2
031062 024666 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
031064 024464 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
031066 024440 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
031070 024500 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
031072 024214 $TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
031074 024140 $TYPBN ;;CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER

031076 030012 $GTSWR ;;CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING

031100 027722 $CKSWR ;;CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
031102 030264 $RDCHR ;;CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
031104 030354 $RDLIN ;;CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
031106 030704 $RDOCT ;;CALL=RDOCT TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
031110 024044 $SAVREG ;;CALL=SAVREG TRAP+14(104414) SAVE R0-R5 ROUTINE
031112 024102 $RESREG ;;CALL=RESREG TRAP+15(104415) RESTORE R0-R5 ROUTINE

```

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
031114 012737 031254 000024 $PWRDN: MOV # $ILLUP,@#PWRVEC ;;SET FOR FAST UP
031122 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
031130 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
031132 010146 MCV R1,-(SP) ;;PUSH R1 ON STACK
031134 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
031136 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
031140 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
031142 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
031144 017746 150004 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
031150 010637 031260 MOV SP,$SAVR6 ;;SAVE SP
031154 012737 031166 000024 MOV # $PWRUP,@#PWRVEC ;;SET UP VECTOR
031162 000000 HALT
031164 000776 BR -2 ;;HANG UP
*****
:POWER UP ROUTINE
031166 012737 031254 000024 $PWRUP: MOV # $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
031174 013706 031260 MOV $SAVR6,SP ;;GET SP
031200 005037 031260 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
031204 005237 031260 1$: INC $SAVR6 ;;WAIT FOR THE INC
031210 001375 BNE 1$ ;;OF WORD
031212 012677 147736 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
031216 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
031220 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
031222 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
031224 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
031226 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
031230 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
031232 012737 031114 000024 MOV # $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
031240 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
031246 104401 TYPE ;;REPORT THE POWER FAILURE
031250 031262 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
031252 000002 RTI
031254 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
031256 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
031260 000000 $SAVR6: 0 ;;PUT THE SP HERE
031262 015 012 120 $POWER: .ASCIZ <15><12>'POWER'
.EVEN

```

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
031272 112737 000001 031536 $ATY1:  MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
031300 112737 000001 031534 $ATY3:  MOVB  #1,$MFLG      ;;TO TYPE A MESSAGE
031306 000403
031310 112737 000001 031536 $ATY4:  MOVB  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
031316
031316 010046          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
031320 010146          MOV    R1,-(SF)      ;;PUSH R1 ON STACK
031322 105737 031534          TSTB  $MFLG      ;;SHOULD TYPE A MESSAGE?
031326 001450          BFO   5$          ;;IF NOT: BR
031330 122737 000001 001242      CMPB  #APTENV,$ENV   ;;OPERATING UNDER APT?
031336 001031          BNE   3$          ;;IF NOT: BR
031340 132737 000100 001243      BITB  #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
031346 001425          BEQ   3$          ;;IF NOT: BR
031350 017600          MOV    @4(SP),R0     ;;GET MESSAGE ADDR.
031354 062766 000002 000004      ADD    #2,4(SP)     ;;BUMP RETURN ADDR.
031362 005737 001222          1$:  -ST   $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
031366 001375          BNE   1$          ;;IF NOT: WAIT
031370 010037 001236          MOV    R0,$MSGAD    ;;PUT ADDR IN MAILBOX
031374 105720          2$:  TSTB  (R0)+     ;;FIND END OF MESSAGE
031376 001376          BNE   2$
031400 163700 001236          SUB    $MSGAD,R0    ;;SUB START OF MESSAGE
031404 006200          ASR    R0          ;;GET MESSAGE LNTH IN WORDS
031406 010037 001240          MOV    R0,$MSGLGT  ;;PUT LENGTH IN MAILBOX
031412 012737 000004 001222      MOV    #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.
031420 000413          BR    5$
031422 017637 000004 031446  3$:  MOV    @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
031430 062766 000002 000004      ADD    #2,4(SP)     ;;BUMP RETURN ADDRESS
031436 013746 177776          MOV    177776,-(SP) ;;PUSH 177776 ON STACK
031442 004737 024666          JSR   PC,$TYPF     ;;CALL TYPE MACRO
031446 000000          4$:  .WORD  0
031450          5$:
031450 105737 031536          10$: TSTB  $FFLG      ;;SHOULD REPORT FATAL ERROR?
031454 001416          BEQ   12$         ;;IF NOT: BR
031456 005737 001242          TST   $ENV       ;;RUNNING UNDER APT?
031462 001413          BEQ   12$         ;;IF NOT: BR
031464 005737 001222          11$: TST   $MSGTYPE   ;;FINISHED LAST MESSAGE?
031470 001375          BNE   11$        ;;IF NOT: WAIT
031472 017637 000004 001224      MOV    @4(SP),$FATAL ;;GET ERROR #
031500 062766 000002 000004      ADD    #2,4(SP)     ;;BUMP RETURN ADDR.
031506 005237 001222          INC   $MSGTYPE   ;;TELL APT TO TAKE ERROR
031512 105037 031536          12$: CLRB  $FFLG      ;;CLEAR FATAL FLAG
031516 105037 031535          CLRB  $LFLG      ;;CLEAR LOG FLAG
031522 105037 031534          CLRB  $MFLG      ;;CLEAR MESSAGE FLAG
031526 012601          MOV    (SP)+,R1    ;;POP STACK INTO R1
031530 012600          MOV    (SP)+,R0    ;;POP STACK INTO R0
031532 000207          RTS   PC         ;;RETURN
031534          000          $MFLG: .BYTE  0    ;;MESSG. FLAG
031535          000          $LFLG: .BYTE  0    ;;LOG FLAG
031536          000          $FFLG: .BYTE  0    ;;FATAL FLAG
          .EVEN
000200          APTSIZE = 200
000001          APTENV  = 001
000100          APTSPOOL= 100
000040          APTCSUP = 040

```

.SBTTL CONSOLE MESSAGES

2				
3	031540	075	000	EQUALS: .ASCIZ @=@
4	031542	101	114	ALL: .ASCIZ @ALL@<CRLF>
5	031547	040	077	040 QUES: .ASCIZ @ ? @
6	031553	054	040	000 COMMA: .ASCIZ @, @
7	031556	200	124	131 MSHELP: .ASCIZ <CRLF>@TYPE HELP TEXT (L) N ? @
8	031607	200	122	115 CNSL01: .ASCIZ <CRLF>@RMCS1=@
9	031617	040	114	111 CNSL02: .ASCIZ @ LIMITS - LO= 160000, HI= 17xxxx@<CRLF>
10	031661	122	115	126 CNSL03: .ASCIZ @RMVEC=@
11	031670	040	114	111 CNSL04: .ASCIZ @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
12	031724	200	124	131 CNSL07: .ASCII <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
3	032011	200	101	116 .ASCIZ <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETURN.@
14	032066	200		CNSL08: .ASCII <CRLF>
15	032067	040	077	111 CNSL09: .ASCIZ @ ?ILLEGAL INPUT@<CRLF>
16	032110	200	104	122 DRIVES: .ASCIZ <CRLF>/DRIVE(S) TO BE TESTED/
17	032137	116	117	116 NONE: .ASCIZ /NONE/
18	032144	200	104	122 MSDRVS: .ASCIZ <CRLF>/DRIVE(S): /
19	032160	104	122	111 MSGDRV: .ASCIZ /DRIVE/
20	032166	200	125	116 SYSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
21	032204	122	115	060 \$RM02: .ASCIZ /RM02/
22	032211	122	115	060 \$RM03: .ASCIZ /RM03/
23	032216	122	115	060 \$RM05: .ASCIZ /RM05/
24	032223	040	116	117 NOTRM: .ASCIZ @ NOT AN RM05/3/2@
25	032244	040	114	117 LODEV: .ASCIZ / LOAD DEVICE/
26	032261	040	116	117 NOTPRS: .ASCIZ / NOT PRESENT/
27	032276	040	116	117 NOTAVL: .ASCIZ / NOT AVAILABLE/
28	032315	040	117	106 UNTOFF: .ASCIZ / OFFLINE/
29	032326	040	117	116 UNTON: .ASCIZ / ONLINE/
30	032336	116	000	N: .ASCIZ /N/
31	032340	131	000	Y: .ASCIZ /Y/
32	032342	040		BLNKS4: .ASCII / /
33	032343	040		BLNKS3: .ASCII / /
34	032344	040		BLNKS2: .ASCII / /
35	032345	040	000	BLNKS1: .ASCIZ / /
36				.EVEN

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

```
.SBTTL FUNCTION CODE TABLE

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

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

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

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

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

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

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

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

; BIT 08 IS NOT USED.

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

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

; BIT 05 IS NOT USED.

; BIT 04 IS NOT USED.

; BIT 03 IS NOT USED.

; BIT 02 IS NOT USED.

; BIT 01 IS NOT USED.

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

FNCDTB: ;FUNCTION CODE TABLE

.WORD OPI ;NOP
```

Function Code	Table	Macro	Function Name
58	032352	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (2)
59	032354	132000	.WORD ATA:OPI:IVC:IAE ;SEEK
60	032356	130000	.WORD ATA:OPI:IVC ;RECALIBRATE
61	032360	020000	.WORD OPI ;DRIVE CLEAR
62	032362	030000	.WORD OPI:IVC ;RELEASE
63	032364	130000	.WORD OPI:ATA:IVC ;OFFSET
64	032366	130000	.WORD OPI:ATA:IVC ;RETURN TO CENTERLINE
65	032370	020000	.WORD OPI ;READ IN PRESET
66	032372	020000	.WORD OPI ;PACK ACKNOWLEDGE
67	032374	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (24)
68	032376	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (26)
69	032400	132000	.WORD ATA:OPI:IVC:IAE ;SEARCH
70	032402	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (32)
71	032404	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (34)
72	032406	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (36)
73	032410	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (40)
74	032412	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (42)
75	032414	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (44)
76	032416	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (46)
77	032420	073300	.WORD WCE:OPI:IVC:IAE:AOE:HCE:ECH ;WRITE CHECK DATA
78	032422	073300	.WORD WCE:OPI:IVC:IAE:AOE:HCE:ECH ;WRITE CHECK HEADER AND DATA
79	032424	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (54)
80	032426	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (56)
81	032430	037200	.WORD OPI:IVC:WLE:IAE:AOE:HCE ;WRITE DATA
82	032432	037000	.WORD OPI:IVC:WLE:IAE:AOE ;WRITE HEADER AND DATA
83	032434	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (64)
84	032436	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (66)
85	032440	033300	.WORD OPI:IVC:IAE:AOE:HCE:ECH ;READ DATA
86	032442	033300	.WORD OPI:IVC:IAE:AOE:HCE:ECH ;READ HEADER AND DATA
87	032444	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (74)
88	032446	130001	.WORD OPI:ATA:ILF:IVC ;ILLEGAL FUNCTION (76)

2
3 032450 001
4 032451 002
5 032452 004
6 032453 010
7 032454 020
8 032455 040
9 032456 100
10 032457 200
11

.SBTTL ATTENTION (ATA) TABLE

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

.EVEN

1					.SBTTL	ERROR MESSAGE TABLE	
2							
3	032460	04751	040562	000000	EMT1:	.WORD	EMS300,EMS1,0
4	032466	047167	047212	047237	EMT2:	.WORD	EMS301,EMS302,EMS303,EMS1,EMS304
5	032500	052427	051633	051773		.WORD	EMS511,EMS500,EMS501,EMS502,EMS503,0
6	032514	047167	047257	047212	EMT3:	.WORD	EMS301,EMS306,EMS302
7	032522	052427	052150	051773		.WORD	EMS511,EMS505,EMS501,EMS502,0
8	032534	047151	047212	047273	EMT4:	.WORD	EMS300,EMS302,EMS307,EMS2
9	032544	052427	052020	051773		.WORD	EMS511,EMS502,EMS501,EMS503,0
10	032556	047167	047334	047351	EMT5:	.WORD	EMS301,EMS310,EMS311
11	032564	052427	052020	051773		.WORD	EMS511,EMS502,EMS501,EMS503,EMS504
12	032576	047415	000000			.WORD	EMS312,0
13	032602	047167	047257	047351	EMT6:	.WORD	EMS301,EMS306,EMS311
14	032610	052427	052020	051773		.WORD	EMS511,EMS502,EMS501,EMS503,EMS504,0
15	032624	047167	047453	047212	EMT7:	.WORD	EMS301,EMS313,EMS302
16	032632	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,EMS504,EMS503,0
17	032646	047561	047602	047504	EMT10:	.WORD	EMS316,EMS317,EMS314
18	032654	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
19	032664	047561	047602	047533	EMT11:	.WORD	EMS316,EMS317,EMS315
20	032672	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
21	032702	047561	047622	047504	EMT12:	.WORD	EMS316,EMS320,EMS314
22	032710	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
23	032720	047561	047622	047533	EMT13:	.WORD	EMS316,EMS320,EMS315
24	032726	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
25	032736	047561	047642	047504	EMT14:	.WORD	EMS316,EMS321,EMS314
26	032744	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
27	032754	047561	047642	047533	EMT15:	.WORD	EMS316,EMS321,EMS315
28	032762	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
29	032772	047561	047662	047504	EMT16:	.WORD	EMS316,EMS322,EMS314
30	033000	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
31	033010	047561	047662	047533	EMT17:	.WORD	EMS316,EMS322,EMS315
32	033016	052427	051773	052020		.WORD	EMS511,EMS501,EMS502,0
33	033026	045142	050101	050157	EMT20:	.WORD	EMS71,EMS335,EMS340,EMS72,EMS377,EMS372
34	033042	052427	052070	000000		.WORD	EMS511,EMS503,0
35	033050	045142	050124	050157	EMT21:	.WORD	EMS71,EMS336,EMS340,EMS72,EMS400,EMS372
36	033064	052427	052070	000000		.WORD	EMS511,EMS503,0
37	033072	045277	050404	045347	EMT22:	.WORD	EMS73,EMS352,EMS74,EMS402,EMS70,EMS406
38	033106	052427	052070	052115		.WORD	EMS511,EMS503,EMS504,0
39	033116	045277	050404	045422	EMT23:	.WORD	EMS73,EMS352,EMS75,EMS402,EMS77,EMS406
40	033132	052427	052070	052115		.WORD	EMS511,EMS503,EMS504,0
41	033142	045277	050521	045422	EMT24:	.WORD	EMS73,EMS356,EMS75,EMS402,EMS77,EMS377
42	033156	052427	052070	052115		.WORD	EMS511,EMS503,EMS504,0
43	033166	045277	050101	050157	EMT25:	.WORD	EMS73,EMS335,EMS340,EMS76,EMS411
44	033200	052427	052070	000000		.WORD	EMS511,EMS503,0
45	033206	047167	047257	046541	EMT26:	.WORD	EMS301,EMS306,EMS252,EMS253,EMS327,EMS254
46	033222	052427	052070	051773		.WORD	EMS511,EMS503,EMS501,EMS502
47	033232	047770	047504	000000		.WORD	EMS330,EMS314,0
48	033240	050143	045277	051376	EMT27:	.WORD	EMS337,EMS73,EMS410,EMS76,EMS411
49	033252	052427	052070	000000		.WORD	EMS511,EMS503,0
50	033260	050143	045602	050170	EMT30:	.WORD	EMS337,EMS100,EMS341,EMS101
51	033270	052427	052070	052115		.WORD	EMS511,EMS503,EMS504,0
52	033300	047151	046541		EMT31:	.WORD	EMS300,EMS252
53	033304	052427	051773	052070		.WORD	EMS511,EMS501,EMS503,0
54	033314	045602	051016		EMT32:	.WORD	EMS100,EMS370
55	033320	052427	052115	000000		.WORD	EMS511,EMS504,0
56	033326	045602	051422		EMT33:	.WORD	EMS100,EMS412
57	033332	052427	052115	000000		.WORD	EMS511,EMS504,0

ERROR MESSAGE TABLE

58	033340	045716	051422		EMT34:	.WORD	EMS102,EMS412
59	033344	052427	052070	000000		.WORD	EMS511,EMS503,0
60	033352	045716	050124		EMT35:	.WORD	EMS102,EMS336
61	033356	052427	052070	000000		.WORD	EMS511,EMS503,0
62	033364	045602	050101	050157	EMT36:	.WORD	EMS100,EMS335,EMS340,EMS102,EMS334
63	033376	052427	052115	000000		.WORD	EMS511,EMS504,0
64	033404	045602	050101	050157	EMT37:	.WORD	EMS100,EMS335,EMS340,EMS102,EMS377,EMS365
65	033420	051435	046024	051467		.WORD	EMS413,EMS104,EMS415
66	033426	052427	052115	000000		.WORD	EMS511,EMS504,0
67	033434	045602	050124	050157	EMT40:	.WORD	EMS100,EMS336,EMS340,EMS416,EMS104,EMS415
68	033450	052427	052115	000000		.WORD	EMS511,EMS504,0
69	033456	045602	050124	050157	EMT41:	.WORD	EMS100,EMS336,EMS340,EMS73,EMS415,EMS402
70	033472	045716	051212			.WORD	EMS102,EMS400
71	033476	052427	052115	000000		.WORD	EMS511,EMS504,0
72	033504	045764	051227	047762	EMT42:	.WORD	EMS103,EMS401,EMS327,EMS370
73	033514	052427	052070	000000		.WORD	EMS511,EMS503,0
74	033522	051450	046024	051517	EMT43:	.WORD	EMS414,EMS104,EMS417
75	033530	052427	052115	000000		.WORD	EMS511,EMS504,0
76	033536	046051	051227	047762	EMT44:	.WORD	EMS105,EMS401,EMS327,EMS370
77	033546	052427	052070	000000		.WORD	EMS511,EMS503,0
78	033554	047151	046574		EMT45:	.WORD	EMS300,EMS253
79	033560	052427	051773	052070		.WORD	EMS511,EMS501,EMS503,0
80	033570	046112	051227	051517	EMT46:	.WORD	EMS106,EMS401,EMS417
81	033576	052427	052070	000000		.WORD	EMS511,EMS503,0
82	033604	051435	046157	051535	EMT47:	.WORD	EMS413,EMS107,EMS420,EMS417
83	033614	052427	052115	000000		.WORD	EMS511,EMS504,0
84	033622	046167	051435	051333	EMT50:	.WORD	EMS110,EMS413,EMS405,EMS107
85	033632	052427	052070	000000		.WORD	EMS511,EMS503,0
86	033640	050231	046226	051550	EMT51:	.WORD	EMS343,EMS111,EMS421
87	033646	052427	052115	000000		.WORD	EMS511,EMS504,0
88	033654	051450	051506	046157	EMT52:	.WORD	EMS414,EMS416,EMS107,EMS421
89	033664	052427	052115	000000		.WORD	EMS511,EMS504,0
90	033672	050007	040747	046732	EMT53:	.WORD	EMS331,EMS4,EMS256
91	033700	052427	051773	000000		.WORD	EMS511,EMS501,0
92	033706	046244	051435	051333	EMT54:	.WORD	EMS112,EMS413,EMS405,EMS607
93	033716	052427	052070	000000		.WORD	EMS511,EMS503,0
94	033724	046302	051227	047762	EMT55:	.WORD	EMS113,EMS401,EMS327,EMS370
95	033734	052427	052070	000000		.WORD	EMS511,EMS503,0
96	033742	046341	051227	051517	EMT56:	.WORD	EMS114,EMS401,EMS417
97	033750	052427	052070	000000		.WORD	EMS511,EMS503,0
98	033756	046762	050037		EMT57:	.WORD	EMS257,EMS322
99	033762	052427	052223	051773		.WORD	EMS511,EMS506,EMS501,0
100	033772	041000	050055		EMT60:	.WORD	EMS5,EMS333
101	033776	052427	052264	051773		.WORD	EMS511,EMS507,EMS501,0
102	034006	051435	046407	051535	EMT61:	.WORD	EMS413,EMS115,EMS420,EMS417
103	034016	052427	052115	000000		.WORD	EMS511,EMS504,0
104	034024	050322	045602	050170	EMT62:	.WORD	EMS346,EMS100,EMS341,EMS101
105	034034	052427	052115	000000		.WORD	EMS511,EMS504,0
106	034042	046302	051033		EMT63:	.WORD	EMS113,EMS371
107	034046	052427	052070	000000		.WORD	EMS511,EMS503,0
108	034054	051435	046424	050420	EMT64:	.WORD	EMS413,EMS116,EMS353
109	034062	052427	052115	000000		.WORD	EMS511,EMS504,0
110	034070	051565	050420		EMT65:	.WORD	EMS422,EMS353
111	034074	052427	052115	000000		.WORD	EMS511,EMS504,0
112	034102	041321	050124	050177	EMT66:	.WORD	EMS12,EMS336,EMS342
113	034110	052427	051773	000000		.WORD	EMS511,EMS501,0
114	034116	047151	041050		EMT67:	.WORD	EMS300,EMS6

115	034122	052427	051773		.WORD	EMS511,EMS501
116	034126	041114	050055	000000	.WORD	EMS7,EMS333,0
117	034134	047151	041050	047762	EMT70:	EMS300,EMS6,EMS327,EMS7
118	034144	052427	051773	052115	.WGRD	EMS511,EMS501,EMS504,0
119	034154	041050	050101	050157	EMT71:	EMS6,EMS335,EMS340,EMS10,EMS333,EMS342
120	034170	052427	051773	052020	.WORD	EMS511,EMS501,EMS502,0
121	034200	041050	050124	050157	EMT72:	EMS6,EMS336,EMS340,EMS10,EMS334,EMS342
122	034214	052427	051773	052020	.WORD	EMS511,EMS501,EMS502,0
123	034224	047167	047016	047237	EMT73:	EMS301,EMS260,EMS303,EMS11
124	034234	052427	051773	052020	.WORD	EMS511,EMS501,EMS502,0
125	034244	050231	050245	050177	EMT74:	EMS343,EMS344,EMS342,0
126	034254	047151	041377		EMT75:	EMS300,EMS13
127	034260	052427	052070	000000	.WORD	EMS511,EMS503,0
128	034266	050322	041377	050274	EMT76:	EMS346,EMS13,EMS345
129	034274	052427	052070	000000	.WORD	EMS511,EMS503,0
130	034302	050143	041377	050274	EMT77:	EMS337,EMS13,EMS345
131	034310	052427	052070	000000	.WORD	EMS511,EMS503,0
132	034316	047167	047453	046627	EMT100:	EMS301,EMS313,EMS254,EMS347,EMS13
133	034330	052427	052070	000000	.WORD	EMS511,EMS503,0
134	034336	050322	041446	050170	EMT101:	EMS346,EMS14,EMS341,EMS15
135	034346	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0
136	034356	050143	045074		EMT102:	EMS337,EMS70
137	034362	052427	052070	000000	.WORD	EMS511,EMS503,0
138	034370	047167	047453	046627	EMT103:	EMS301,EMS313,EMS254,EMS347,EMS15
139	034402	052427	052070		.WORD	EMS511,EMS503
140	034406	041446	050037	000000	.WORD	EMS14,EMS332,0
141	034414	050322	041652	050170	EMT104:	EMS346,EMS17,EMS341,EMS16
142	034424	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0
143	034434	050143	041652	050170	EMT105:	EMS337,EMS17,EMS341,EMS16
144	034444	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0
145	034454	047167	047453	046627	EMT106:	EMS301,EMS313,EMS254,EMS347,EMS16
146	034466	052427	052070		.WORD	EMS511,EMS503
147	034472	041652	050037	000000	.WORD	EMS17,EMS332,0
148	034500	050322	041713	050170	EMT107:	EMS346,EMS20,EMS341,EMS21
149	034510	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0
150	034520	041713	050366	041757	EMT110:	EMS20,EMS351,EMS21,EMS350,EMS22,EMS315
151	034534	052427	051773	000000	.WORD	EMS511,EMS501,0
152	034542	041713	050071	050361	EMT111:	EMS20,EMS334,EMS350,EMS22,EMS333
153	034554	052427	051773	000000	.WORD	EMS511,EMS501,0
154	034562	050143	041713	050170	EMT112:	EMS337,EMS20,EMS341,EMS21
155	034572	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0
156	034602	050143	041713	050170	EMT113:	EMS337,EMS20,EMS341,EMS21,EMS350,EMS22,EMS334
157	034620	052427	051773	000000	.WORD	EMS511,EMS501,0
158	034626	041713	050404	041757	EMT114:	EMS20,EMS352,EMS21,EMS350,EMS22,EMS333
159	034642	052427	051773	000000	.WORD	EMS511,EMS501,0
160	034650	047167	047453	046627	EMT115:	EMS301,EMS313,EMS254,EMS347,EMS21
161	034662	052427	052070		.WORD	EMS511,EMS503
162	034666	041713	050037	000000	.WORD	EMS20,EMS332,0
163	034674	050322	042103	050170	EMT116:	EMS346,EMS23,EMS341,EMS24
164	034704	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0
165	034714	050143	042103	050170	EMT117:	EMS337,EMS23,EMS341,EMS24
166	034724	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0
167	034734	047167	047453	046627	EMT120:	EMS301,EMS313,EMS254,EMS347,EMS24
168	034746	052427	052070		.WORD	EMS511,EMS503
169	034752	042103	050037	000000	.WORD	EMS23,EMS332,0
170	034760	050322	042240	050170	EMT121:	EMS346,EMS25,EMS341,EMS26
171	034770	052427	052070	051773	.WORD	EMS511,EMS503,EMS501,0

172	035000	050143	042240	050170	EMT122:	.WORD	EMS337,EMS25,EMS341,EMS26
173	035010	052427	052070	051773		.WORD	EMS511,EMS503,EMS501,0
174	035020	047167	047453	046627	EMT123:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS26
175	035032	052427	052070			.WORD	EMS511,EMS503
176	035036	042240	050037	000000		.WORD	EMS25,EMS332,0
177	035044	047151	042375	047273	EMT124:	.WORD	EMS300,EMS27,EMS307,EMS2
178	035054	052427	052070	000000		.WORD	EMS511,EMS503,0
179	035062	050007	042375	050420	EMT125:	.WORD	EMS331,EMS27,EMS353
180	035070	052427	052070			.WORD	EMS511,EMS503
181	035074	042441	047533	000000		.WORD	EMS30,EMS315,0
182	035102	050143	042375	050170	EMT126:	.WORD	EMS337,EMS27,EMS341,EMS30
183	035112	052427	052070	000000		.WORD	EMS511,EMS503,0
184	035120	047167	047453	046627	EMT127:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS30
185	035132	052427	052070			.WORD	EMS511,EMS503
186	035136	042375	050037	000000		.WORD	EMS27,EMS332,0
187	035144	042521	050444	046437	EMT130:	.WORD	EMS31,EMS354,EMS250
188	035152	052427	052115	052070		.WORD	EMS511,EMS504,EMS503,0
189	035162	042572	050444	046437	EMT131:	.WORD	EMS32,EMS354,EMS250
190	035170	052427	052115	052070		.WORD	EMS511,EMS504,EMS503,0
191	035200	050477	042652	046437	EMT132:	.WORD	EMS355,EMS33,EMS250,EMS341,EMS30
192	035212	052427	052115	000000		.WORD	EMS511,EMS504,0
193	035220	050477	042702	046437	EMT133:	.WORD	EMS355,EMS34,EMS250,EMS341,EMS30
194	035232	052427	052115	000000		.WORD	EMS511,EMS504,0
195	035240	050007	040747	046670	EMT134:	.WORD	EMS331,EMS4,EMS255
196	035246	052427	052115	000000		.WORD	EMS511,EMS504,0
197	035254	042731	050541	050563	EMT135:	.WORD	EMS35,EMS357,EMS360,EMS15
198	035264	052427	051773	000000		.WORD	EMS511,EMS501,0
199	035272	047047	050637		EMT136:	.WORD	EMS261,EMS362
200	035276	052427	052070	000000		.WORD	EMS511,EMS503,0
201	035304	047151	042773	047273	EMT137:	.WORD	EMS300,EMS36,EMS307,EMS2
202	035314	052427	051773	000000		.WORD	EMS511,EMS501,0
203	035322	050477	043023	046670	EMT140:	.WORD	EMS355,EMS37,EMS255,EMS341,EMS30
204	035334	052427	052115	000000		.WORD	EMS511,EMS504,0
205	035342	050322	043055	050274	EMT141:	.WORD	EMS346,EMS40,EMS345
206	035350	052427	052115	000000		.WORD	EMS511,EMS504,0
207	035356	050143	043055	050170	EMT142:	.WORD	EMS337,EMS40,EMS341,EMS30
208	035366	052427	052115	000000		.WORD	EMS511,EMS504,0
209	035374	050660	047334	043133	EMT143:	.WORD	EMS363,EMS310,EMS41
210	035402	052427	051773	000000		.WORD	EMS511,EMS501,0
211	035410	050143	043133	050170	EMT144:	.WORD	EMS337,EMS41,EMS341,EMS252,EMS327,EMS253
212	035424	052427	051773	000000		.WORD	EMS511,EMS501,0
213	035432	043133	050444	050675	EMT145:	.WORD	EMS41,EMS354,EMS364,EMS252,EMS365,EMS253
214	035446	052427	051773	000000		.WORD	EMS511,EMS501,0
215	035454	047167	047257	042773	EMT146:	.WORD	EMS301,EMS306,EMS36
216	035462	052427	051773	052070		.WORD	EMS511,EMS501,EMS503,0
217	035472	050723	043201		EMT147:	.WORD	EMS366,EMS42
218	035476	052427	052070	000000		.WORD	EMS511,EMS503,0
219	035504	050746	050420	050716	EMT150:	.WORD	EMS367,EMS353,EMS365,EMS42,EMS354,EMS3
220	035520	052427	052070	000000		.WORD	EMS511,EMS503,0
221	035526	050143	042773		EMT151:	.WORD	EMS337,EMS36
222	035532	052427	051773	000000		.WORD	EMS511,EMS501,0
223	035540	043263	050444	042773	EMT152:	.WORD	EMS43,EMS354,EMS36
224	035546	052427	051773	000000		.WORD	EMS511,EMS501,0
225	035554	050746	050420	050716	EMT153:	.WORD	EMS367,EMS353,EMS365,EMS36,EMS370
226	035566	052427	052070	000000		.WORD	EMS511,EMS503,0
227	035574	050746	050420	050716	EMT154:	.WORD	EMS367,EMS353,EMS365,EMS36,EMS371
228	035606	052427	052070	000000		.WORD	EMS511,EMS503,0

229	035614	047151	043334	047273	EMT155:	.WORD	EMS300,EMS44,EMS307,EMS2
230	035624	052427	052070	000000		.WORD	EMS511,EMS503,0
231	035632	050746	050420	050716	EMT156:	.WORD	EMS367,EMS353,EMS365,EMS44,EMS354,EMS3
232	035646	052427	052070	000000		.WORD	EMS511,EMS503,0
233	035654	047151	043375	047273	EMT157:	.WORD	EMS300,EMS45,EMS307,EMS2
234	035664	052427	052070	000000		.WORD	EMS511,EMS503,0
235	035672	050746	050420	050716	EMT160:	.WORD	EMS367,EMS353,EMS365,EMS45,EMS354,EMS3
236	035706	052427	052070	051773		.WORD	EMS511,EMS503,EMS501
237	035714	042731	050055	000000		.WORD	EMS35,EMS333,0
238	035722	047151	043452	047273	EMT161:	.WORD	EMS300,EMS46,EMS307,EMS2
239	035732	052427	052070	000000		.WORD	EMS511,EMS503,0
240	035740	050143	043452	050420	EMT162:	.WORD	EMS337,EMS46,EMS353
241	035746	052427	052070	051773		.WORD	EMS511,EMS503,EMS501,0
242	035756	042731	050101	050143	EMT163:	.WORD	EMS35,EMS335,EMS337,EMS41,EMS334,EMS372
243	035772	052427	051773	000000		.WORD	EMS511,EMS501,0
244	036000	043534	050101	050143	EMT164:	.WORD	EMS47,EMS335,EMS337,EMS41,EMS335,EMS372
245	036014	052427	051773	000000		.WORD	EMS511,EMS501,0
246	036022	050143	042731	047762	EMT165:	.WORD	EMS337,EMS35,EMS327,EMS47
247	036032	052427	051773			.WORD	EMS511,EMS501
248	036036	043133	050055	051062		.WORD	EMS41,EMS333,EMS372,0
249	036046	047151	043534	047273	EMT166:	.WORD	EMS300,EMS47,EMS307,EMS2
250	036056	052427	051773	052070		.WORD	EMS511,EMS501,EMS503,0
251	036066	043574	050101	050157	EMT167:	.WORD	EMS50,EMS335,EMS340,EMS36,EMS333
252	036100	052427	051773	052070		.WORD	EMS511,EMS501,EMS503,0
253	036110	050143	042731		EMT170:	.WORD	EMS337,EMS35
254	036114	052427	051773	000000		.WORD	EMS511,EMS501,0
255	036122	043574	042702	040702	EMT171:	.WORD	EMS50,EMS34,EMS3
256	036130	052427	051773	000000		.WORD	EMS511,EMS501,0
257	036136	047167	047257	043643	EMT172:	.WORD	EMS301,EMS306,EMS51
258	036144	052427	051773	052115		.WORD	EMS511,EMS501,EMS504,0
259	036154	050746	050420	050716	EMT173:	.WORD	EMS367,EMS353,EMS365,EMS47,EMS354,EMS3
260	036170	052427	051773	000000		.WORD	EMS511,EMS501,0
261	036176	047151	046437	047762	EMT174:	.WORD	EMS300,EMS250,EMS327,EMS255,EMS327,EMS256
262	036212	050170	052526			.WORD	EMS341,EMS600
263	036216	052427	051773	000000		.WORD	EMS511,EMS501,0
264	036224	047151	046732	050170	EMT175:	.WORD	EMS300,EMS256,EMS341,EMS600
265	036234	052427	051773	000000		.WORD	EMS511,EMS501,0
266	036242	047151	046437	050170	EMT176:	.WORD	EMS300,EMS250,EMS341,EMS600
267	036252	052427	052115	000000		.WORD	EMS511,EMS504,0
268	036260				EMT177:		
269	036260	050143	043712	050170	EMT200:	.WORD	EMS337,EMS52,EMS341,EMS601
270	036270	052427	051773	000000		.WORD	EMS511,EMS501,0
271	036276	050322	043712	050170	EMT201:	.WORD	EMS346,EMS52,EMS341,EMS602
272	036306	052427	051773	000000		.WORD	EMS511,EMS501,0
273	036314	050322	043643	050170	EMT202:	.WORD	EMS346,EMS51,EMS341,EMS602
274	036324	052427	051773	000000		.WORD	EMS511,EMS501,0
275	036332	050322	043712	050274	EMT203:	.WORD	EMS346,EMS52,EMS345,EMS373,EMS255
276	036344	052427	052115	051773		.WORD	EMS511,EMS504,EMS501,0
277	036354	050322	043712	050170	EMT204:	.WORD	EMS346,EMS52,EMS341,EMS27
278	036364	052427	052115	051773		.WORD	EMS511,EMS504,EMS501,0
279	036374	043753	050444	040702	EMT205:	.WORD	EMS53,EMS354,EMS3
280	036402	052427	052070	052020		.WORD	EMS511,EMS503,EMS502,EMS510,0
281	036414	050143	044014	051120	EMT206:	.WORD	EMS337,EMS54,EMS374,EMS250,EMS327,EMS255
282	036430	047762	040702			.WORD	EMS327,EMS3
283	036434	052427	052115	051773		.WORD	EMS511,EMS504,EMS501,0
284	036444	044014	050444	040702	EMT207:	.WORD	EMS54,EMS354,EMS3
285	036452	052427	051773	000000		.WORD	EMS511,EMS501,0

1	040364	052761	000000	EHT1:	.WORD	EH1,0
2	040370	053037	000000	EHT2:	.WORD	EH2,0
3	040374	053054	000000	EHT5:	.WORD	EH5,0
4	040400	053112	000000	EHT7:	.WORD	EH7,0
5	040404	053131	000000	EHT47:	.WORD	EH47,0
6	040410	053157	000000	EHT52:	.WORD	EH52,0
7	040414	053255	000000	EHT57:	.WORD	EH57,0
8	040420	053313	000000	EHT61:	.WORD	EH61,0
9	040424	053351	000000	EHT65:	.WORD	EH65,0
10	040430	053425	000000	EHT71:	.WORD	EH71,0
11	040434	053045	000000	EHT74:	.WORD	EH3,0
12	040440	053502	000000	EHT115:	.WORD	EH115,0
13	040444	053577	000000	EHT130:	.WORD	EH130,0
14	040450	053715	000000	EHT132:	.WORD	EH132,0
15	040454	054013	000000	EHT142:	.WORD	EH142,0
16	040460	054111	000000	EHT145:	.WORD	EH145,0
17	040464	054226	000000	EHT150:	.WORD	EH150,0
18	040470	054324	000000	EHT213:	.WORD	EH213,0
19	040474	054421	000000	EHT220:	.WORD	EH220,0

1	040500	054460	EDT1:	.WORD	ED1
2	040502	054470	EDT2:	.WORD	ED2
3	040504	054474	EDT5:	.WORD	ED5
4	040506	054502	EDT47:	.WORD	ED47
5	040510	054512	EDT52:	.WORD	ED52
6	040512	054524	EDT57:	.WORD	ED57
7	040514	054532	EDT61:	.WORD	ED61
8	040516	054544	EDT65:	.WORD	ED65
9	040520	054554	EDT71:	.WORD	ED71
10	040522	054470	EDT74:	.WORD	ED2
11	040524	054564	EDT115:	.WORD	ED115
12	040526	054576	EDT130:	.WORD	ED130
13	040530	054564	EDT132:	.WORD	ED115
14	040532	054612	EDT220:	.WORD	ED220

FR
VI
DY
7

1	040562	116	117	116	EMS1:	.ASCIZ	@NONEXISTENT DEVICE 'NED' (RMCS2, BIT 12) a
2	040633	103	117	116	EMS2:	.ASCIZ	@CONTROLLER CLEAR 'CLR' (RMCS2, BIT 05) a
3	040702	106	125	116	EMS3:	.ASCIZ	@FUNCTION CODE (RMCS1, BITS 01 - 05) a
4	040747	125	116	125	EMS4:	.ASCIZ	@UNUSED BIT POSITIONS OF a
5	041000	104	105	126	EMS5:	.ASCIZ	@DEVICE AVAILABLE 'DVA' (RMCS1, BIT 11) a
6	041050	120	101	122	EMS6:	.ASCIZ	@PARTIY ERROR 'PAR' (RMER1, BIT 03) a
7	041114	104	101	124	EMS7:	.ASCIZ	@DATA PARITY ERROR 'DPE' (RMER2, BIT 03) a
8	041165	120	101	122	EMS10:	.ASCIZ	@PARITY TEST 'PAT' (RMCS2, BIT 04) a
9	041230	115	101	123	EMS11:	.ASCII	@MASSBUS CONTROL BUS PARITY ERROR 'MCPF' a
10	041300	050	122	115		.ASCIZ	a(RMCS1, BIT 13) a
11	041321	111	114	114	EMS12:	.ASCIZ	@ILLEGAL REGISTER ERROR 'ILR' (RMER1, BIT 01) a
12	041377	104	111	101	EMS13:	.ASCIZ	@DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 00) a
13	041446	115	105	104	EMS14:	.ASCIZ	@MEDIUM ON LINE 'MOL' (RMDS, BIT 12) a
14	041513	115	101	111	EMS15:	.ASCIZ	@MAINTENANCE UNIT READY 'MUR' (RMMR1, BIT 09) a
15	041571	115	101	111	EMS16:	.ASCIZ	@MAINTENANCE WRITE PROTECT 'MWP' (RMMR1, BIT 03) a
16	041652	127	122	111	EMS17:	.ASCIZ	@WRITE LOCK 'WRL' (RMDS, BIT 11) a
17	041713	104	105	126	EMS20:	.ASCIZ	@DEVICE CHECK 'DVC' (RMER2, BIT 07) a
18	041757	115	101	111	EMS21:	.ASCIZ	@MAINTENANCE DRIVE FAULT 'MDF' (RMMR1, BIT 06) a
19	042036	125	116	123	EMS22:	.ASCIZ	@UNSAFE STATUS 'UNS' (RMER1, BIT 14) a
20	042103	123	105	105	EMS23:	.ASCIZ	@SEEK INCOMPLETE STATUS 'SKI' (RMER2, BIT 14) a
21	042161	115	101	111	EMS24:	.ASCIZ	@MAINTENANCE SEEK ERROR 'MSER' (RMMR1, BIT 07) a
22	042240	120	117	123	EMS25:	.ASCIZ	@POSITIONING IN PROGRESS 'PIP' (RMDS, BIT 13) a
23	042316	115	101	111	EMS26:	.ASCIZ	@MAINTENANCE ON CYLINDER 'MOC' (RMMR1, BIT 08) a
24	042375	105	116	104	EMS27:	.ASCIZ	@END OF BLOCK 'EBL' (RMMR1, BIT 13) a
25	042441	104	111	101	EMS30:	.ASCIZ	@DIAGNOSTIC END OF BLOCK 'DEBL' (RMMR1, BIT 13) a
26	042521	114	101	123	EMS31:	.ASCIZ	@LAST SECTOR STATUS 'LS' (RMMR1, BIT 02) a
27	042572	114	101	123	EMS32:	.ASCIZ	@LAST SECTOR/TRACK STATUS 'LST' (RMMR1, BIT 01) a
28	042652	123	105	103	EMS33:	.ASCIZ	@SECTOR ADDRESS BITS OF a
29	042702	124	122	101	EMS34:	.ASCIZ	@TRACK ADDRESS BITS OF a
30	042731	126	117	114	EMS35:	.ASCIZ	@VOLUME VALID 'VV' (RMDS, BIT 06) a
31	042773	107	117	040	EMS36:	.ASCIZ	@GO BIT (RMCS1, BIT 00) a
32	043023	103	131	114	EMS37:	.ASCIZ	@CYLINDER ADDRESS BITS OF a
33	043055	114	101	123	EMS40:	.ASCIZ	@LAST BLOCK TRANSFERRED, 'LBT' (RMDS, BIT 10) a
34	043133	103	117	115	EMS41:	.ASCIZ	@COMPOSITE ERROR 'ERR' (RMDS, BIT 14) a
35	043201	103	117	115	EMS42:	.ASCIZ	@COMMAND SEQUENCER TEST BIT 'TST' (RMMR2, BIT 12) a
36	043263	104	122	111	EMS43:	.ASCIZ	@DRIVE READY STATUS 'DRY' (RMDS, BIT 07) a
37	043334	103	117	116	EMS44:	.ASCIZ	@CONTINUE 'CONT' (RMMR1, BIT 06) a
38	043375	111	116	126	EMS45:	.ASCIZ	@INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) a
39	043452	114	117	123	EMS46:	.ASCIZ	@LOSS OF SYSTEM CLOCK ERROR 'LSC' (RMER2, BIT 11) a
40	043534	117	103	103	EMS47:	.ASCIZ	@OCCUPIED 'OCC' (RMMR1, BIT 15) a
41	043574	111	114	114	EMS50:	.ASCIZ	@ILLEGAL FUNCTION 'ILF' (RMER1, BIT 0) a
42	043643	117	106	106	EMS51:	.ASCIZ	@OFFSET DIRECTION 'OFD' (RMOF, BIT 07) a
43	043712	117	106	106	EMS52:	.ASCIZ	@OFFSET MODE 'OM' (RMDS, BIT 00) a
44	043753	122	125	116	EMS53:	.ASCIZ	@RUN AND GO 'RG' (RMMR1, BIT 14) a
45	044014	111	116	126	EMS54:	.ASCIZ	@INVALID ADDRESS ERROR 'IAE' (RMER1, BIT 10) a
46	044071	101	104	104	EMS55:	.ASCIZ	@ADDRESS OVERFLOW ERROR 'AOE' (RMER1, BIT 09) a
47	044147	122	105	107	EMS56:	.ASCII	@REGISTER MODIFICATION REFUSED ERROR a
48	044213	042	122	115		.ASCIZ	a'RMR' (RMER1, BIT 02) a
49	044242	104	122	111	EMS57:	.ASCIZ	@DRIVE REQUEST REQUIRED STATUS 'DRQ' (RMDT, BIT 11) a
50	044326	120	122	117	EMS60:	.ASCIZ	@PROGRAMMABLE STATUS 'PGM' (RMDS, BIT 09) a
51	044400	104	122	111	EMS61:	.ASCIZ	@DRIVE PRESENT STATUS 'DPR' (RMDS, BIT 08) a
52	044453	120	117	122	EMS62:	.ASCIZ	@PORT REQUEST FLOP 'RQA,RQB' (RMMR2, BITS 15,14) a
53	044534	101	124	124	EMS63:	.ASCIZ	@ATTENTION 'ATA' (RMDS, BIT 15) a
54	044574	127	122	111	EMS64:	.ASCIZ	@WRITE LOCK ERROR 'WLE' (RMER1, BIT 11) a
55	044644	105	130	103	EMS65:	.ASCIZ	@EXCEPTION 'REX' (RMMR1, BIT 12) a
56	044705	111	116	126	EMS66:	.ASCIZ	@INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) a
57	044762	124	101	107	EMS67:	.ASCIZ	@TAG BUS (RMMR2, BITS 00-09) OR TAG CONTROL a

58	045036	114	111	116		.ASCIZ	@LINES (RMMR2, BITS 10,11,13) @
59	045074	123	105	101	EMS70:	.ASCIZ	@SEARCH ENABLE 'ESRC' (RMMR1, BIT 11) @
60							
61	045142	117	120	105	EMS71:	.ASCIZ	@OPERATION INCOMPLETE ERROR 'OPI'(RMR1 BIT13) @
62	045221	104	111	123	EMS72:	.ASCIZ	@DISABLE SEARCH TIMEOUT 'MSEN' (RMMR1 BIT 12) @
63	045277	104	122	111	EMS73:	.ASCIZ	@DRIVE TIMING ERROR 'DTE'(RMR1,BIT 12) @
64	045347	115	101	111	EMS74:	.ASCIZ	@MAINTENANCE INDEX PULSE 'MI'(RMMR1 BIT 2) @
65	045422	115	101	111	EMS75:	.ASCIZ	@MAINTENANCE SECTOR PULSE 'MS'(RMMR1 BIT 5) @
66	045476	106	117	122	EMS76:	.ASCIZ	@FORMAT BIT 'FMT16' (RMOF,BIT12) @
67	045537	123	105	103	EMS77:	.ASCIZ	@SECTOR COMPARE 'MSC'(RMMR1 BIT 1) @
68	045602	120	122	117	EMS100:	.ASCIZ	@PROM STROBE 'WC' (RMMR1 BIT 5) @
69	045642	115	101	111	EMS101:	.ASCIZ	@MAINTENANCE BIT CLOCK 'MCLK' (RMMR1 BIT11) @
70	045716	114	117	117	EMS102:	.ASCIZ	@LOOKING FOR SYNC 'PLFS'(RMMR1,BIT10) @
71	045764	127	122	111	EMS103:	.ASCIZ	@WRITE GATE 'BB00' (RMMR2 BIT0) @
72	046024	110	105	101	EMS104:	.ASCIZ	@HEADER SYNC PATTERN @
73	046051	110	105	101	EMS105:	.ASCIZ	@HEADER AREA 'PHA' (RMMR1 BIT 7) @
74	046112	105	116	101	EMS106:	.ASCIZ	@ENABLE CRC OUT 'ECRC' (RMMR1 BIT 9) @
75	046157	110	105	101	EMS107:	.ASCIZ	@HEADER @
76	046167	122	105	101	EMS110:	.ASCIZ	@READ GATE 'BB01'(RMMR2 BIT 1) @
77	046226	110	105	101	EMS111:	.ASCIZ	@HEADER ERROR @
78	046244	104	101	124	EMS112:	.ASCIZ	@DATA TIMING SEQUENCER OUTPUT @
79	046302	104	101	124	EMS113:	.ASCIZ	@DATA AREA 'PDA' (RMMR1 BIT 8) @
80	046341	105	116	101	EMS114:	.ASCIZ	@ENABLE ECC OUT 'EECC' (RMMR1, BIT 4) @
81	046407	105	103	103	EMS115:	.ASCIZ	@ECC PATTERN @
82	046424	105	103	103	EMS116:	.ASCIZ	@ECC ERROR @
83	046437	104	111	123	EMS250:	.ASCIZ	@DISK ADDRESS REGISTER (RMDA) @
84	046475	103	117	116	EMS251:	.ASCIZ	@CONTROL STATUS REGISTER #1 (RMCS1) @
85	046541	105	122	122	EMS252:	.ASCIZ	@ERROR REGISTER #1 (RMR1) @
86	046574	105	122	122	EMS253:	.ASCIZ	@ERROR REGISTER #2 (RMR2) @
87	046627	115	101	111	EMS254:	.ASCIZ	@MAINTENANCE REGISTER #1 (RMMR1) @
88	046670	104	105	123	EMS255:	.ASCIZ	@DESIRED CYLINDER REGISTER (RMDC) @
89	046732	117	106	106	EMS256:	.ASCIZ	@OFFSET REGISTER (RMOF) @
90	046762	104	122	111	EMS257:	.ASCIZ	@DRIVE TYPE REGISTER (RMDT) @
91	047016	110	117	114	EMS260:	.ASCIZ	@HOLDING REGISTER (RMRH) @
92	047047	123	105	122	EMS261:	.ASCIZ	@SERIAL NUMBER REGISTER (RMSN) @
93	047106	101	124	124	EMS262:	.ASCIZ	@ATTENTION SUMMARY REGISTER (RMSA) @
94							
95	047151	103	101	116	EMS300:	.ASCIZ	@CANNOT CLEAR @
96	047167	103	101	116	EMS301:	.ASCIZ	@CANNOT WRITE/READ @
97	047212	101	116	131	EMS302:	.ASCIZ	@ANY DEVICE REGISTER @
98	047237	127	111	124	EMS303:	.ASCIZ	@WITHOUT @
99	047250	105	122	122	EMS304:	.ASCIZ	@ERROR @
100	047257	101	040	117	EMS306:	.ASCIZ	@A ONE FROM @
101	047273	125	123	111	EMS307:	.ASCIZ	@USING MASSBUS INITIALIZE, I.E., @
102	047334	101	040	132	EMS310:	.ASCIZ	@A ZERO FROM @
103	047351	105	126	105	EMS311:	.ASCIZ	@EVERY DEVICE REGISTER BIT POSITION @
104	047415	124	110	105	EMS312:	.ASCIZ	@THE FOLLOWING BITS ARE STUCK @
105	047453	101	040	123	EMS313:	.ASCIZ	@A SHIFTING ONE BIT FROM @
106	047504	101	120	120	EMS314:	.ASCIZ	@APPEARS STUCK AT ZERO @
107	047533	101	120	120	EMS315:	.ASCIZ	@APPEARS STUCK AT ONE @
108	047561	122	105	107	EMS316:	.ASCIZ	@REGISTER SELECT @
109	047602	061	040	050	EMS317:	.ASCIZ	@1 (1,2,4,8,16) @
110	047622	062	040	050	EMS320:	.ASCIZ	@2 (1,2,4,8,16) @
111	047642	064	040	050	EMS321:	.ASCIZ	@4 (1,2,4,8,16) @
112	047662	070	040	050	EMS322:	.ASCIZ	@8 (1,2,4,8,16) @
113	047702	101	114	114	EMS323:	.ASCIZ	@ALL ONES FROM @
114	047721	101	114	114	EMS324:	.ASCIZ	@ALL ZEROS FROM @

1	054460	001140	001142	001136	ED1:	.WORD	\$GDDAT,\$BDDAT,\$RDADR,0
2	054470	001136	000000		ED2:	.WORD	\$RDADR,0
3	054474	001140	001142	000000	ED5:	.WORD	\$GDDAT,\$BDDAT,0
4	054502	001174	001176	001200	ED47:	.WORD	\$TMP0,\$TMP1,\$TMP2,0
5	054512	001174	054636	001176	ED52:	.WORD	\$TMP0,BUFFER,\$TMP1,BUFFER+2,0
6	054524	001142	001136	000000	ED57:	.WORD	\$BDDAT,\$BDADR,0
7	054532	001200	001202	001174	ED61:	.WORD	\$TMP2,\$TMP3,\$TMP0,\$TMP1,0
8	054544	001140	001142	001174	ED65:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,0
9	054554	001140	001142	001450	ED71:	.WORD	\$GDDAT,\$BDDAT,RMHRO,0
10	054564	001140	001142	001136	ED115:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,\$TMP0,0
11	054576	001140	001142	001136	ED130:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,\$TMP0,\$TMP1,0
12	054612	001142	001136		ED220:	.WORD	\$BDDAT,\$BDADR
13							
14	054616	000	000	000	EF1:	.BYTE	0,0,0
15	054621	000			EF2:	.BYTE	0
16	054622	000	000		EF5:	.BYTE	0,0
17	054624	000	000	000	EF57:	.BYTE	0,0,0,0
18	054630	000	000	000	EF130:	.BYTE	0,0,0,0,0
19							
20					.EVEN		
21	054636				BUFFER:		
22	054636				BUFOUR:	.BLKW	258.
23	055642				BUFTWO:	.BLKW	258.
24		054636			.=BUFFER		
25							
26	054636				HELP:		
27	054636	200			.ASCII	<CRLF>	
28	054637	200			.ASCII	<CRLF>	
29	054640	114	111	123	.ASCII	@LIST OF TESTS@<CRLF>	
30	054656	055	055	055	.ASCII	@-----@<CRLF>	
31	054674	124	061	011	.ASCII	@T1 TRANSFER TEST@<CRLF>	
32	054715	124	062	011	.ASCII	@T2 CTOD TEST@<CRLF>	
33	054732	124	063	011	.ASCII	@T3 MASSBUS INITIALIZE TEST@<CRLF>	
34	054765	124	064	011	.ASCII	@T4 CLEAR STUCK ACTIVE TEST@<CRLF>	
35	055020	124	065	011	.ASCII	@T5 TRISTATE TRANSFER TEST@<CRLF>	
36	055052	124	066	011	.ASCII	@T6 REGISTER SELECT TEST@<CRLF>	
37	055102	124	067	011	.ASCII	@T7 DRIVE TYPE TEST@<CRLF>	
38	055125	124	061	060	.ASCII	@T10 DEVICE AVAILABLE TEST@<CRLF>	
39	055157	124	061	061	.ASCII	@T11 SEARCH TIMEOUT TEST@<CRLF>	
40	055207	124	061	062	.ASCII	@T12 SET DTE TEST@<CRLF>	
41	055230	124	061	063	.ASCII	@T13 FORMAT CHANGE TEST@<CRLF>	
42	055257	124	061	064	.ASCII	@T14 PROM STROBE TEST@<CRLF>	
43	055304	124	061	065	.ASCII	@T15 SYNC WORD COUNT INHIBIT TEST@<CRLF>	
44	055345	124	061	066	.ASCII	@T16 SYNC DETECTION TEST@<CRLF>	
45	055375	124	061	067	.ASCII	@T17 ABORT SYNC DETECTION TEST@<CRLF>	
46	055433	124	062	060	.ASCII	@T20 SYNC GENERATION TEST@<CRLF>	
47	055464	124	062	061	.ASCII	@T21 WRITE HEADER TEST@<CRLF>	
48	055512	124	062	062	.ASCII	@T22 HEADER COMPARE TEST @<CRLF>	
49	055543	124	062	063	.ASCII	@T23 ECC GENERATION TEST@<CRLF>	
50	055573	124	062	064	.ASCII	@T24 ECC DETECTION TEST@<CRLF>	
51	055622	200			.ASCII	<CRLF>	
52	055623	117	120	105	.ASCII	@OPERATIONAL SWITCH SETTINGS@<CRLF>	
53	055657	055	055	055	.ASCII	@-----@<CRLF>	
54	055713	123	127	111	.ASCII	@SWITCH USE@<CRLF>	
55	055737	055	055	055	.ASCII	@-----@<CRLF>	
56	055774	040	040	061	.ASCII	@ 15 HALT ON ERROR@<CRLF>	
57	056020	040	040	061	.ASCII	@ 14 LOOP ON TEST@<CRLF>	

BI
BI
BI
BI
BL
BL
BL
BC
BC
BF
BS
BU

BU
BU
CC
CH
CH
CK
CL
CL
CM
CN
CN
CN
CN
CN
CO
CO
CP
CR

CR

CY
DB
DB

DB
DC
DD
DE
DI
DI

SYMBOL TABLE

EMS27	042375	EMS361	050611	EMS52	043712	EMT124	035044	EMT206	036414
EMS3	040702	EMS362	050637	EMS53	043753	EMT125	035062	EMT207	036444
EMS30	042441	EMS363	050660	EMS54	044014	EMT126	035102	EMT21	033050
EMS300	047151	EMS364	050675	EMS55	044071	EMT127	035120	EMT210	036460
EMS301	047167	EMS365	050716	EMS56	044147	EMT13	032720	EMT211	036476
EMS302	047212	EMS366	050723	EMS57	044242	EMT130	035144	EMT212	036514
EMS303	047237	EMS367	050746	EMS6	041050	EMT131	035162	EMT213	036526
EMS304	047250	EMS37	043023	EMS60	044326	EMT132	035200	EMT214	036556
EMS306	047257	EMS370	051016	EMS600	052526	EMT133	035220	EMT215	036570
EMS307	047273	EMS371	051033	EMS601	052556	EMT134	035240	EMT216	036610
EMS31	042521	EMS372	051062	EMS602	052576	EMT135	035254	EMT217	036624
EMS310	047334	EMS373	051114	EMS603	052637	EMT136	035272	EMT22	033072
EMS311	047351	EMS374	051120	EMS604	052660	EMT137	035304	EMT220	036640
EMS312	047415	EMS375	051151	EMS605	052705	EMT14	032736	EMT221	036656
EMS313	047453	EMS376	051165	EMS606	052723	EMT140	035322	EMT222	036674
EMS314	047504	EMS377	051201	EMS607	052743	EMT141	035342	EMT223	036712
EMS315	047533	EMS4	040747	EMS61	044400	EMT142	035356	EMT224	036730
EMS316	047561	EMS40	043055	EMS62	044453	EMT143	035374	EMT225	036750
EMS317	047602	EMS400	051212	EMS63	044534	EMT144	035410	EMT226	036772
EMS32	042572	EMS401	051227	EMS64	044574	EMT145	035432	EMT227	037010
EMS320	047622	EMS402	051244	EMS65	044644	EMT146	035454	EMT23	033116
EMS321	047642	EMS403	051253	EMS66	044705	EMT147	035472	EMT230	037026
EMS322	047662	EMS404	051314	EMS67	044762	EMT15	032754	EMT231	037044
EMS323	047702	EMS405	051333	EMS7	041114	EMT150	035504	EMT232	037060
EMS324	047721	EMS406	051343	EMS70	045074	EMT151	035526	EMT233	037102
EMS325	047741	EMS407	051356	EMS71	045142	EMT152	035540	EMT234	037124
EMS326	047752	EMS41	043133	EMS72	045221	EMT153	035554	EMT235	037152
EMS327	047762	EMS410	051376	EMS73	045277	EMT154	035574	EMT236	037172
EMS33	042652	EMS411	051405	EMS74	045347	EMT155	035614	EMT237	037224
EMS330	047770	EMS412	051422	EMS75	045422	EMT156	035632	EMT24	033142
EMS331	050007	EMS413	051435	EMS76	045476	EMT157	035654	EMT240	037244
EMS332	050037	EMS414	051450	EMS77	045537	EMT16	032772	EMT241	037270
EMS333	050055	EMS415	051467	EMTVEC=	000030	EMT160	035672	EMT242	037314
EMS334	050071	EMS416	051506	EMT1	032460	EMT161	035722	EMT243	037340
EMS335	050101	EMS417	051517	EMT10	032646	EMT162	035740	EMT244	037364
EMS336	050124	EMS42	043201	EMT100	034316	EMT163	035756	EMT245	037402
EMS337	050143	EMS420	051535	EMT101	034336	EMT164	036000	EMT246	037420
EMS34	042702	EMS421	051550	EMT102	034356	EMT165	036022	EMT247	037444
EMS340	050157	EMS422	051565	EMT103	034370	EMT166	036046	EMT25	033166
EMS341	050170	EMS43	043263	EMT104	034414	EMT167	036066	EMT250	037462
EMS342	050177	EMS44	043334	EMT105	034434	EMT17	033010	EMT251	037506
EMS343	050231	EMS45	043375	EMT106	034454	EMT170	036110	EMT252	037532
EMS344	050245	EMS46	043452	EMT107	034500	EMT171	036122	EMT253	037556
EMS345	050274	EMS47	043534	EMT11	032664	EMT172	036136	EMT254	037574
EMS346	050322	EMS5	041000	EMT110	034520	EMT173	036154	EMT255	037612
EMS347	050340	EMS50	043574	EMT111	034542	EMT174	036176	EMT256	037636
EMS35	042731	EMS500	051633	EMT112	034562	EMT175	036224	EMT257	037654
EMS350	050361	EMS501	051773	EMT113	034602	EMT176	036242	EMT26	033206
EMS351	050366	EMS502	052020	EMT114	034626	EMT177	036260	EMT260	037700
EMS352	050404	EMS503	052070	EMT115	034650	EMT2	032466	EMT261	037724
EMS353	050420	EMS504	052115	EMT116	034674	EMT20	033026	EMT262	037750
EMS354	050444	EMS505	052150	EMT117	034714	EMT200	036260	EMT263	037766
EMS355	050477	EMS506	052223	EMT12	032702	EMT201	036276	EMT264	040004
EMS356	050521	EMS507	052264	EMT120	034734	EMT202	036314	EMT265	040030
EMS357	050541	EMS51	043643	EMT121	034760	EMT203	036332	EMT266	040046
EMS36	042773	EMS510	052354	EMT122	035000	EMT204	036354	EMT267	040064
EMS360	050563	EMS511	052427	EMT123	035020	EMT205	036374	EMT27	033240

EMT270	040100	EQUALS	031540	ILRG60=	000060	NSA	=	100000	RIP	=	000020		
EMT271	040124	ERR	=	040000	ILRG62=	000062	OCC	=	100000	RELEASE=	000012		
EMT272	040150	ERRNMB	027214	ILRG64=	000064	OFD	=	000200	RMAS	=	000016		
EMT273	040166	ERROR	-	104000	ILRG66=	000066	OFFSET=	000014	RMASI	001354			
EMT274	040204	ERRTYP	026346	ILRG70=	000070	OM	=	000001	RMASO	001430			
EMT275	040222	ERRVEC=	000004	ILRG72=	000072	OPE	=	020000	RMBA	=	000004		
EMT276	040242	ERTY00	027222	ILRG74=	000074	OPI	=	020000	RMBAE	=	000050		
EMT277	040260	ERTY01	027227	ILRG76=	000076	OR	=	000200	RMBAE1	001406			
EMT3	032514	ERTY02	027237	IOTVEC=	000020	PACACK=	000022		RMBAEO	001462			
EMT30	033260	ERTY03	027246	IPCK0	=	000001	PACACK=	000022	RMBAI	001342			
EMT300	040304	ERTY04	027254	IPCK1	=	000002	PAR	=	000010	RMBAO	001416		
EMT301	040330	ERTY05	027257	IPCK2	=	000004	PAT	=	000020	RMCS1	=	000000	
EMT302	040346	ESRC	=	004000	IPCK3	=	000010	PCLOCK	022010	RMCS1I	001336		
EMT31	033300	FER	=	000020	IR	=	000100	PCOUNT	022076	RMCS10	001412		
EMT32	033314	FMT16	=	010000	IVC	=	010000	PDA	=	000400	RMCS2	=	000010
EMT33	033326	FNCDTB	032350	LBC	=	002000	PFECH	027264	RMCS2I	001346			
EMT34	033340	FNCMSK=	000077	LBT	=	002000	PFECH1	027274	RMCS20	001422			
EMT35	033352	F0	=	000002	LCLOCK	022026	PFECH2	027362	RMCS3	=	000052		
EMT36	033364	F1	=	000004	LCOUNT	022076	PFECH3	027400	RMCS3I	001410			
EMT37	033404	F2	=	000010	LF	=	000012	PFECH4	027406	RMCS30	001464		
EMT4	032534	F3	=	000020	LODEV	032244	PGE	=	002000	RMDA	=	000006	
EMT40	033434	F4	=	000040	LS	=	000004	PGM	=	001000	RMDAI	001344	
EMT41	033456	GETBUF	001336	LSC	=	004000	PHA	=	000200	RMDAO	001420		
EMT42	033504	GO	=	000001	LST	=	000002	PIP	=	020000	RMDB	=	000022
EMT43	033522	GTSWR	=	104407	LSTOP	022140	PIRQ	=	177772	RMDBI	001360		
EMT44	033536	HCE	000200	LSTRK	001334	PIRQVE=	000240		RMDBO	001434			
EMT45	033554	HCI	=	002000	MCLK	=	004000	PLCLK	022034	RMDC	=	000034	
EMT46	033570	HCRC	=	000400	MCPE	=	020000	PLFS	=	002000	RMDCI	001372	
EMT47	033604	HELP	054636	MDF	=	000100	PLSTP	022144	RMDCO	001446			
EMT5	032556	HT	=	000011	MDPE	=	000400	PRO	=	000000	RMDS	=	000012
EMT50	033622	IAE	=	002000	MJ	=	000004	PR1	=	000040	RMDSI	001350	
EMT51	033640	IBSAVE	026344	MOC	=	000400	PR2	=	000100	RMDSO	001424		
EMT52	033674	IDXMSK=	000077	MOH	=	020000	PR3	=	000140	RMDT	=	000026	
EMT53	033672	IE	=	000100	MOL	=	010000	PR4	=	000200	RMDTI	001364	
EMT54	033706	ILF	=	000001	MRD	=	002000	PR5	=	000240	RMDTO	001440	
EMT55	033724	ILF02	=	000002	MR1AAA=	051401	PR6	=	000300	RMEC1	=	000044	
EMT56	033742	ILF24	=	000024	MS	=	000040	PR7	=	000340	RMEC1I	001402	
EMT57	033756	ILF26	=	000026	MSC	=	000002	PS	=	177776	RMEC10	001456	
EMT6	032602	ILF30	000030	MSDRVS	032144	PSEL	=	002000	RMEC2	=	000046		
EMT60	033772	ILF32	=	000032	MSE	=	100000	PSTOP	022132	RMEC2I	001404		
EMT61	034006	ILF34	=	000034	MSEN	=	010000	PSW	=	177776	RMEC20	001460	
EMT62	034024	ILF36	=	000036	MSER	=	000200	PUTBUF	001412	RMER1	=	000014	
EMT63	034042	ILF40	=	000040	MSGDRV	032160	PWRVEC=	000024	RMER1I	001352			
EMT64	034054	ILF42	=	000042	MSHELP	031556	QUES	031547	RMER10	001426			
EMT65	034070	ILF44	=	000044	MUR	=	001000	RD	=	000070	RMER2	=	000042
EMT66	034102	ILF46	=	000046	MWD	=	000010	RDCHR	=	104411	RMER2I	001400	
EMT67	034116	ILF54	=	000054	MWP	=	000010	RDLIN	=	104412	RMER20	001454	
EMT7	032624	ILF56	=	000056	MXF	=	001000	RDOCT	=	104413	RMR	=	000036
EMT70	034134	ILF64	=	000064	N	032336	RDY	=	000200	RMR1	001374		
EMT71	034154	ILF66	=	000066	NDTMSK=	115760	READY	005200	RMR10	001450			
EMT72	034200	ILF74	=	000074	NED	=	010000	READY1	005226	RMLA	=	000020	
EMT73	034224	ILF76	=	000076	NEM	=	004000	RECAL	=	000006	RMLA1	001356	
EMT74	034244	ILR	=	000002	NONE	032137	RESREG=	104415	RMLAO	001432			
EMT75	034254	ILRG50=	000050	NOP	=	000000	RESVEC=	000010	RMR1	=	000024		
EMT76	034266	ILRG52=	000052	NOTAVL	032276	RFX	=	010000	RMR1I	001362			
EMT77	034302	ILRG54=	000054	NOTPRS	032261	RG	=	040000	RMR10	001436			
FNBSCH	022432	ILRG56=	000056	NOTRM	032223	RH	=	000072	RMR2	000040			

RMMR21	001376	SW05	= 000040	TST5	006360	\$CNTLU	030625	\$LPVEC	001516
RMMR20	001452	SW06	= 000100	TST6	007444	\$CPUOP	001250	\$MADR1	001254
RMOF	= 000032	SW07	= 000200	TST7	010514	\$CRLF	001217	\$MADR2	001260
RMOFI	001370	SW08	= 000400	TYPBN	= 104406	\$DBLK	024430	\$MADR3	001264
RMOFO	001444	SW09	= 001000	TYPDS	= 104405	\$DDW0	001306	\$MADR4	001270
RMR	= 000004	SW1	= 000002	TYPE	= 104401	\$DDW1	001310	\$MAIL	001222
RMSN	= 000030	SW10	= 002000	TYPOC	= 104402	\$DDW2	001312	\$MAMS1	001252
RMSNI	001366	SW11	= 004000	TYPON	= 104404	\$DDW3	001314	\$MAMS2	001256
RMSND	001442	SW12	= 010000	TYPOS	= 104403	\$DDW4	001316	\$MAMS3	001262
RMWIC	= 000002	SW13	= 020000	UNS	= 040000	\$DDW5	001320	\$MAMS4	001266
RMWICI	001340	SW14	= 040000	UNMSK	= 000007	\$DDW6	001322	\$MBADR	001102
RMWICO	001414	SW15	= 100000	UNTOFF	032315	\$DDW7	001324	\$MFLG	031534
RQA	= 100000	SW2	= 000004	UNTON	032326	\$DEVCT	001232	\$MNEW	030650
RQB	= 040000	SW3	= 000010	UPE	= 020000	\$DEVN	001300	\$MSGAD	001236
RTC	= 000016	SW4	= 000020	USE	= 040000	\$DOAGN	021540	\$MSGLG	001240
R6	= 0000006	SW5	= 000040	U0	= 000001	\$DTBL	024420	\$MSGTY	001222
R7	= 0000007	SW6	= 000100	U1	= 000002	\$ENDAD	021530	\$MSWR	030637
SADMSK	= 000377	SW7	= 000200	U2	= 000004	\$ENDCT	021366	\$MTYP1	001253
SAVREG	= 104414	SW8	= 000400	VV	= 000100	\$ENULL	021544	\$MTYP2	001257
SA1	= 000001	SW9	= 001000	WATCH	001534	\$ENV	001242	\$MTYP3	001263
SA16	= 000020	SYSTAT	032166	WC	= 000040	\$ENVN	001243	\$MTYP4	001267
SA2	= 000002	TADMSK	= 177400	WCD	= 000050	\$EOP	021332	\$MXCNT	025700
SA4	= 000004	TAG	= 020000	WCE	= 040000	\$EOPCT	021360	\$NULL	001170
SAB	= 000010	TAGADR	= 001114	WCEHI	= 010000	\$EOSP	021274	\$NWTST	= 000001
SC	= 100000	TAP	= 040000	WCELO	= 004000	\$ERFLG	001117	\$OCNT	024662
SCOPE	000004	TA1	= 000400	WCF	= 000040	\$ERMAX	001131	\$OMODE	024664
SCTCMP	023304	TA16	= 010000	WCH	= 000052	\$ERROR	025752	\$OVER	025664
SCTMSK	= 003700	TA2	= 001000	WD	= 000060	\$ERRPC	001132	\$PASS	001230
SCO	000100	TA4	= 002000	WH	= 000062	\$ERRTB	001542	\$PASTM	001106
SC1	= 000200	TAB	= 004000	WLE	= 004000	\$ERTTL	001126	\$POWER	031262
SC2	= 000400	TBITVE	= 000014	WRL	= 004000	\$ESCAP	001210	\$PSW	001530
SC3	= 001000	TIME	001532	XNUDC	= 176000	\$ETABL	001242	\$PWDRN	031114
SC4	002000	TKVEC	= 000060	XNUER2	= 001567	\$ETEND	001326	\$PWDRM	031250
SEARCH	= 000030	TPVEC	= 000064	XNUOF	= 161577	\$FATAL	001224	\$PWDRUP	031166
SEEK	= 000004	TRAPVE	= 000034	XSIZ	003766	\$FFLG	031536	\$QUES	001216
SETLFS	023412	TRE	= 040000	XXDP	001332	\$FILLC	001172	\$RDCHR	030264
SETOM	022302	TRTVEC	= 000014	Y	032340	\$FILLS	001171	\$RDLIN	030354
SETVV	022160	TST	= 010000	\$APTHD	001100	\$GDADR	001134	\$RDOCT	030704
SHUT	030662	TSTNMB	027212	\$ATYC	031316	\$GDDAT	001140	\$RDSZ	= 000010
SIZCLK	021550	TSTQUE	001466	\$ATY1	031272	\$GET42	021520	\$RESRE	024102
SKI	= 040000	TST1	005366	\$ATY3	031300	\$GTSWR	030012	\$RMO2	032204
SNGPRT	= 020024	TST10	010636	\$ATY4	031310	\$GT42P	021514	\$RMO3	032211
STACK	= 001100	TST11	010722	\$AUTOB	001150	\$HD	= 000000	\$RMO5	032216
STANDA	004322	TST	011172	\$BASE	001276	\$HIBTS	001100	\$RTNAD	021540
START	002732	TST13	011522	\$BDADR	001136	\$HIOCT	031004	\$SAVRE	024044
START1	002722	TST14	012106	\$BDDAT	001142	\$ICNT	001120	\$SAVR6	031260
START2	002736	TST15	012434	\$BELL	001212	\$ILLUP	031254	\$SCOPE	025222
STKLMT	= 177774	TST16	013104	\$BIN	024212	\$INTAG	001151	\$SETUP	= 000137
STOPCL	001540	TST17	013530	\$CDW1	001302	\$ITEMB	001130	\$STUP	= 177777
SWR	001154	TST2	005676	\$CDW2	001304	\$LF	001220	\$SVLAD	025630
SWREG	000176	TST20	013752	\$CHARC	025216	\$LFLG	031535	\$SVPC	= 000210
SW0	= 000001	TST21	014444	\$CKSWR	027722	\$LLCSR	001522	\$SWR	= 167400
SW00	= 000001	TST22	015432	\$CMTAG	001114	\$LVEC	001524	\$SWREG	001244
SW01	= 000002	TST23	016700	\$CM3	= 000000	\$LPADR	001122	\$SWRMK	= 000000
SW02	= 000004	TST24	020070	\$CM4	= 000005	\$LPCSB	001514	\$SWOBT	025702
SW03	= 000010	TST3	006060	\$CNTLC	030620	\$LPCSR	001512	\$TESTN	001226
SW04	= 000020	TST4	006230	\$CNTLG	030632	\$LPERR	001124	\$TIMES	001206

BIT6	4-682#														
BIT7	4-682#	9-72													
BIT8	4-682#														
BIT9	4-682#														
BLNKS1	36-35#														
BLNKS2	9-134	10-34	10-48	36-34#											
BLNKS3	36-33#														
BLNKS4	9-83	36-32#													
BOTADR	30-81*	30-99*	30-102	30-117	30-161#										
BOTFLG	30-62*	30-109*	30-112	30-115*	30-162#										
BPTVEC	4-682#														
BSE	4-875#														
BUFFER	13-560	13-631	13-686	13-812	13-923	13-:01	13-:43	13-:61	13-:35	13-?65	13-?67	13-?82	45-5	45-5	
	45-21#	45-24													
BUF ONE	13-817	13-C56	45-22#												
BUF TWO	45-23#														
CC	4-867#														
CH	4-868#														
CHGADR	7-0#	9-14*	9-17*	10-25	10-27*										
CHRCNT	30-63*	30-86*	30-92*	30-93	30-96*	30-100	30-105*	30-116*	30-163#						
CKSWR	28-1	29-1	29-1	33-1#											
CLKSNC	13-204	13-831	13-878	22-6#											
CLOCK	7-0#	13-569	13-588	15-13*	15-27*	15-39*	19-88	20-14							
CLR	4-940#	9-85	11-57	18-10											
CMNSTA	9-143	11-2#													
CNSLO1	10-32	36-8#													
CNSLO2	10-40	36-9#													
CNSLO3	10-44	36-10#													
CNSLO4	10-54	36-11#													
CNSLO7	10-61	36-12#													
CNSLO8	10-92	36-14#													
CNSLO9	36-15#														
CNTCLR	13-7	13-28	13-75	13-108	13-116	13-139	13-162	13-213	13-262	13-365	13-381	13-403	13-425	13-442	
	13-464	13-480	13-495	13-545	13-608	13-677	13-748	16-14	18-8#	19-19					
COMMA	10-86	11-6	11-14	36-6#											
CONT	4-828#														
CPSAVE	28-1	28-1	28-1	28-1*	28-1*	29-1	29-1	29-1	29-1	29-1	29-1#	29-1*	29-1*	30-180	
CR	4-682#	10-73	10-84	27-1	27-1	30-84	43-118	43-178	43-179	43-180	43-181	43-182	43-183	43-184	
	43-185	43-186	43-187	43-188	43-189	43-189	43-190								
CRLF	4-682#	9-6	9-28	9-28	9-42	9-53	9-59	9-60	11-30	27-1	27-1	36-4	36-7	36-8	
	36-9	36-11	36-12	36-13	36-14	36-15	36-16	36-18	36-20	44-1	44-3	44-5	44-9	44-11	
	44-14	44-16	44-18	44-20	44-22	44-24	44-26	44-28	44-30	44-32	45-27	45-28	45-29	45-30	
	45-31	45-32	45-33	45-34	45-35	45-36	45-37	45-38	45-39	45-40	45-41	45-42	45-43	45-44	
	45-45	45-46	45-47	45-48	45-49	45-50	45-51	45-52	45-53	45-54	45-55	45-56	45-57	45-58	
	45-59	45-60	45-61	45-62	45-63	45-64	45-65	45-66	45-67	45-68	45-69	45-70	45-71		
CYLSK	4-857#	13-80													
DBCK	4-802#	19-106	19-119												
DBEN	4-803#	4-835	13-587	13-641	13-642	13-656	13-657	13-658	13-661	13-662	13-696	13-697	13-698	13-699	
	13-715	13-718	13-719	13-720	13-723	13-724	13-829	13-830	13-856	13-857	13-872	13-873	13-890	13-891	
	13-901	13-902	19-82	19-106	19-107	19-113	19-114	19-119	19-120	20-10	20-11	20-21	20-22	20-25	
	20-26	20-27	20-28	21-10	21-11	21-18	21-19								
DBL	4-958#														
DCK	4-764#	4-781	13-C44												
DDISP	4-682#	6-0	9-23												
DEBL	4-804#														
DISPLA	6-0#	9-23*	9-23*	28-1*	29-1*										
DISPRE	5-1#	9-23													

PA
PA
PA
PC
PC
PD
PF
PF
PF
PG
PG
PH
PI
PI
PL
PL
PR
PR
PR
PR
PR
PR
PS
PS
PS
PU
PW
QU
R6
R7
RD
RD
RD
RD
RE
RE
RE
RE
RG
RH
RI
RL

EMS370	39-54	39-72	39-76	39-94	39-225	39-347	39-363	39-379	39-395	43-150#				
EMS371	39-106	39-227	39-397	43-151#										
EMS372	39-33	39-35	39-242	39-244	39-248	39-315	39-327	39-331	39-336	43-152#				
EMS373	39-275	39-292	43-153#											
EMS374	39-281	43-154#												
EMS375	39-303	43-155#												
EMS376	39-305	39-307	39-311	39-331	43-156#									
EMS377	39-33	39-41	39-64	39-323	39-327	39-331	39-340	39-349	39-356	39-365	39-372	39-381	39-392	39-407
	43-157#													
EMS4	39-90	39-195	43-4#											
EMS40	39-205	39-207	43-33#											
EMS400	39-35	39-70	39-336	43-158#										
EMS401	39-72	39-76	39-80	39-94	39-96	39-325	39-332	39-336	39-345	39-361	39-377	39-384	39-401	39-410
	43-159#													
EMS402	39-37	39-39	39-41	39-69	39-340	39-349	39-356	39-365	39-372	39-381	39-392	39-407	43-160#	
EMS403	39-340	39-349	39-356	39-365	39-372	39-381	39-392	39-407	43-161#					
EMS404	39-343	39-352	39-375	39-399	39-405	43-162#								
EMS405	39-84	39-92	39-338	39-343	39-345	39-347	39-352	39-354	39-359	39-361	39-363	39-368	39-370	39-375
	39-377	39-379	39-384	39-386	39-390	39-395	39-397	39-399	39-401	39-403	39-405	39-410	43-163#	
EMS406	39-37	39-39	39-338	39-354	39-370	39-390	43-164#							
EMS407	39-388	43-165#												
EMS41	39-209	39-211	39-213	39-242	39-244	39-248	39-321	39-323	43-34#					
EMS410	39-48	43-166#												
EMS411	39-43	39-48	43-167#											
EMS412	39-56	39-58	43-168#											
EMS413	39-65	39-82	39-84	39-92	39-102	39-108	43-169#							
EMS414	39-74	39-88	43-170#											
EMS415	39-65	39-67	39-69	43-171#										
EMS416	39-67	39-88	43-172#											
EMS417	39-74	39-80	39-82	39-96	39-102	43-173#								
EMS42	39-217	39-219	43-35#											
EMS420	39-82	39-102	43-174#											
EMS421	39-86	39-88	43-175#											
EMS422	39-110	43-176#												
EMS43	39-223	43-36#												
EMS44	39-229	39-231	43-37#											
EMS45	39-233	39-235	43-38#											
EMS46	39-238	39-240	43-39#											
EMS47	39-244	39-246	39-249	39-259	43-40#									
EMS5	39-100	39-299	43-5#											
EMS50	39-251	39-255	43-41#											
EMS500	39-5	43-178#												
EMS501	39-5	39-7	39-9	39-11	39-14	39-16	39-18	39-20	39-22	39-24	39-26	39-28	39-30	39-32
	39-46	39-53	39-79	39-91	39-99	39-101	39-113	39-115	39-118	39-120	39-122	39-124	39-135	39-142
	39-144	39-149	39-151	39-153	39-155	39-157	39-159	39-164	39-166	39-171	39-173	39-198	39-202	39-210
	39-212	39-214	39-216	39-222	39-224	39-236	39-241	39-243	39-245	39-247	39-250	39-252	39-254	39-256
	39-258	39-260	39-263	39-265	39-270	39-272	39-274	39-276	39-278	39-283	39-285	39-293	39-296	39-298
	39-300	39-302	39-304	39-306	39-308	39-310	39-312	39-314	39-316	39-318	39-320	39-322	39-324	39-326
	39-328	39-333	39-335	39-337	43-181#									
EMS502	39-5	39-7	39-9	39-11	39-14	39-16	39-18	39-20	39-22	39-24	39-26	39-28	39-30	39-32
	39-46	39-120	39-122	39-124	39-280	39-330	39-333	39-402	43-182#					
EMS503	39-5	39-9	39-11	39-14	39-16	39-34	39-36	39-38	39-40	39-42	39-44	39-46	39-49	39-51
	39-53	39-59	39-61	39-73	39-77	39-79	39-81	39-85	39-93	39-95	39-97	39-107	39-127	39-129
	39-131	39-133	39-135	39-137	39-139	39-142	39-144	39-146	39-149	39-155	39-161	39-164	39-166	39-168
	39-171	39-173	39-175	39-178	39-180	39-183	39-185	39-188	39-190	39-200	39-216	39-218	39-220	39-226
	39-228	39-230	39-232	39-234	39-236	39-239	39-241	39-250	39-252	39-280	39-312	39-326	39-328	39-330
	39-333	39-335	39-337	39-339	39-341	39-344	39-346	39-348	39-350	39-353	39-355	39-357	39-360	39-362

EMT10	8-25	39-17#
EMT100	39-132#	
EMT101	39-134#	
EMT102	8-199	39-136#
EMT103	39-138#	
EMT104	39-141#	
EMT105	39-143#	
EMT106	39-145#	
EMT107	39-148#	
EMT11	8-28	39-19#
EMT110	39-150#	
EMT111	39-152#	
EMT112	39-154#	
EMT113	39-156#	
EMT114	39-158#	
EMT115	39-160#	
EMT116	39-163#	
EMT117	39-165#	
EMT12	8-31	39-21#
EMT120	39-167#	
EMT121	39-170#	
EMT122	39-172#	
EMT123	39-174#	
EMT124	39-177#	
EMT125	39-179#	
EMT126	39-182#	
EMT127	39-184#	
EMT13	8-34	39-23#
EMT130	39-187#	
EMT131	39-189#	
EMT132	39-191#	
EMT133	39-193#	
EMT134	39-195#	
EMT135	39-197#	
EMT136	39-199#	
EMT137	39-201#	
EMT14	8-37	39-25#
EMT140	39-203#	
EMT141	39-205#	
EMT142	39-207#	
EMT143	39-209#	
EMT144	39-211#	
EMT145	39-213#	
EMT146	39-215#	
EMT147	39-217#	
EMT15	8-40	39-27#
EMT150	39-219#	
EMT151	39-221#	
EMT152	39-223#	
EMT153	39-225#	
EMT154	39-227#	
EMT155	39-229#	
EMT156	39-231#	
EMT157	39-233#	
EMT16	8-43	39-29#
EMT160	39-235#	
EMT161	39-238#	

EMT162	39-240#	
EMT163	39-242#	
EMT164	39-244#	
EMT165	39-246#	
EMT166	39-249#	
EMT167	39-251#	
EMT17	8-46	39-31#
EMT170	8-193	39-253#
EMT171	39-255#	
EMT172	39-257#	
EMT173	8-202	39-259#
EMT174	8-205	39-261#
EMT175	8-208	39-264#
EMT176	8-211	39-266#
EMT177	39-268#	
EMT2	8-6	39-4#
EMT20	8-49	39-33#
EMT200	8-217	39-269#
EMT201	39-271#	
EMT202	39-273#	
EMT203	39-275#	
EMT204	39-277#	
EMT205	39-279#	
EMT206	39-281#	
EMT207	39-284#	
EMT21	8-52	39-35#
EMT210	39-286#	
EMT211	39-288#	
EMT212	39-290#	
EMT213	39-292#	
EMT214	39-295#	
EMT215	39-297#	
EMT216	39-299#	
EMT217	39-301#	
EMT22	8-55	39-37#
EMT220	39-303#	
EMT221	39-305#	
EMT222	39-307#	
EMT223	39-309#	
EMT224	39-311#	
EMT225	39-313#	
EMT226	39-315#	
EMT227	39-317#	
EMT23	8-58	39-39#
EMT230	39-319#	
EMT231	39-321#	
EMT232	39-323#	
EMT233	39-325#	
EMT234	39-327#	
EMT235	39-329#	
EMT236	39-331#	
EMT237	39-334#	
EMT24	8-61	30-41#
EMT240	39-336#	
EMT241	39-338#	
EMT242	39-340#	
EMT243	39-343#	

EMT244	39-345#	
EMT245	39-347#	
EMT246	39-349#	
EMT247	39-352#	
EMT25	8-64	39-43#
EMT250	39-354#	
EMT251	39-356#	
EMT252	39-359#	
EMT253	39-361#	
EMT254	39-363#	
EMT255	39-365#	
EMT256	39-368#	
EMT257	39-370#	
EMT26	8-67	39-45#
EMT260	39-372#	
EMT261	39-375#	
EMT262	39-377#	
EMT263	39-379#	
EMT264	39-381#	
EMT265	39-384#	
EMT266	39-386#	
EMT267	39-388#	
EMT27	8-70	39-48#
EMT270	39-390#	
EMT271	39-392#	
EMT272	39-395#	
EMT273	39-397#	
EMT274	39-399#	
EMT275	8-196	39-401#
EMT276	8-150	39-403#
EMT277	39-405#	
EMT3	8-9	39-6#
EMT30	8-73	39-50#
EMT300	39-407#	
EMT301	39-410#	
EMT302	8-214	39-412#
EMT31	8-76	39-52#
EMT32	8-79	39-54#
EMT33	8-82	39-56#
EMT34	8-85	39-58#
EMT35	8-88	39-60#
EMT36	8-91	39-62#
EMT37	8-94	39-64#
EMT4	8-12	39-8#
EMT40	8-97	39-67#
EMT41	8-100	39-69#
EMT42	8-103	39-72#
EMT43	8-106	8-109
EMT44	39-76#	39-74#
EMT45	8-112	39-78#
EMT46	8-115	39-80#
EMT47	8-118	39-82#
EMT5	8-15	39-10#
EMT50	8-121	39-84#
EMT51	8-124	39-86#
EMT52	8-127	39-88#
EMT53	39-90#	

LS*OP	15-28	15-84#												
LSIRK	7-0#	11-55*	11-63*	13-:58	13-:	13-:11	13-:33	13->88	13->91	13-?79	13-B14			
MCLK	4-806#	13-755	13-766	13-780	13-794	13-829	13-856	13-872	13-890	13-901	13-945	13-960	13-982	13-984
	13-:23	13-:84	13-:96	13-:18	13-:41	13-:85	13-<11	13-<36	13-<53	13-<77	13-<98	13-:55	13-:76	13-:98
	13->20	13->39	13->63	13->72	13->79	13->98	13-?04	13-?17	13-?43	13-A19	13-A41	13-A70	13-A89	13-B85
	13-C03	13-C05	13-C71	13-C90	19-23	19-37	21-10	21-18	21-35	21-41	22-23	22-50	22-52	
MCPE	4-921#													
MDF	4-811#													
MDPE	4-937#													
MI	4-814#	13-619	13-680	13-715	19-56	20-10								
MOC	4-809#	4-835	13-587	13-641	13-642	13-656	13-657	13-658	13-661	13-662	13-696	13-697	13-698	13-699
	13-715	13-718	13-719	13-720	13-723	13-724	13-829	13-830	13-856	13-857	13-872	13-873	13-890	13-891
	13-901	13-902	19-82	19-106	19-107	19-114	19-119	19-120	20-10	20-11	20-21	20-22	20-25	20-26
	20-27	20-28	21-10	21-11	21-:8	21-19								
MOH	4-841#													
MOL	4-753#	9-110												
MR1AAA	4-835#	13-943	13-945	13-946	13-960	13-961	13-976	13-:18	13-:23	13-:24	13-:84	13-:85	13-:96	13-:97
	13-:18	13-:19	13-:41	13-:42	13-:85	13-:86	13-<11	13-<12	13-<36	13-<37	13-<53	13-<54	13-<77	13-<78
	13-<98	13-<99	13-:55	13-:56	13-:76	13-:77	13-:98	13-:99	13->20	13->21	13->39	13->40	13->63	13->64
	13->72	13->98	13-?17	13-?18	13-?43	13-?44	13-A19	13-A20	13-A41	13-A42	13-A70	13-A71	13-A89	13-A90
	13-B85	13-B86	13-B97	13-C71	13-C72	13-C90	13-C91	21-35	21-36	21-41	21-42	22-23	22-24	22-43
MRD	4-807#	13-943	13-945	13-946	13-980	13->76	13-?02	13-B85	13-B86	13-C01	22-47			
MS	4-812#	13-641	13-657	13-661	13-697	13-719	13-723	13-:18	13-:23	13-:24	20-21	20-26		
MSC	4-815#	13-656	13-657	13-658	13-696	13-697	13-698	13-718	13-719	13-720	20-25	20-26	20-27	
MSDRVS	10-64	36-18#												
MSE	4-887#													
MSEN	4-805#	4-835	13-641	13-642	13-656	13-657	13-658	13-661	13-662	13-696	13-697	13-698	13-699	13-715
	13-718	13-719	13-720	13-723	13-724	13-829	13-830	13-856	13-857	13-872	13-873	13-890	13-891	13-901
	13-902	19-119	19-120	20-10	20-11	20-21	20-22	20-25	20-26	20-27	20-28	21-10	21-11	21-18
	21-19													
MSER	4-810#													
MSGDRV	11-68	36-19#												
MSHELP	10-11	36-7#												
MUR	4-808#	4-835	13-587	13-641	13-642	13-656	13-657	13-658	13-661	13-662	13-696	13-697	13-698	13-699
	13-715	13-718	13-719	13-720	13-723	13-724	13-829	13-830	13-856	13-857	13-872	13-873	13-890	13-891
	13-901	13-902	16-16	17-14	19-23	19-24	19-37	19-38	19-56	19-57	19-82	19-106	19-107	19-113
	19-114	19-119	19-120	20-10	20-11	20-21	20-22	20-25	20-26	20-27	20-28	21-10	21-11	21-18
	21-19													
MWD	4-831#	13-:10	13-:36	13-<50	13-<95	13-A37								
MWP	4-813#													
MXF	4-936#													
N	10-19	36-30#												
NDTMSK	4-781#													
NED	4-933#	9-88	13-9	13-25	13-30	13-44								
NEM	4-934#													
NONE	11-7	36-17#												
NOP	4-698#													
NOTAVL	9-117	36-27#												
NOTPRS	9-114	36-26#												
NOTRM	9-108	36-24#												
NSA	4-839#													
OCC	4-819#													
OFD	4-852#													
OFFSET	4-704#	17-17												
OPM	4-760#	17-19	17-21											
OPE	4-877#													
OPI	4-766#	13-578	13-595	13-597	37-57	37-58	37-59	37-60	37-61	37-62	37-63	37-64	37-65	37-66

	13-901	13-902	13-943	13-945	13-946	13-960	13-961	13-981	13-983	13-985	13-:18	13-:23	13-:24	13-:84
	13-:85	13-:96	13-:97	13-:18	13-:19	13-:41	13-:42	13-:85	13-:86	13-:11	13-:12	13-:36	13-:37	13-:53
	13-<54	13-<77	13-<78	13-<98	13-<99	13-=55	13-=56	13-=76	13-=77	13-=98	13-=99	13->20	13->21	13->39
	13->40	13->63	13->64	13->78	13->80	13-?03	13-?05	13-?17	13-?18	13-?43	13-?44	13-A19	13-A20	13-A41
	13-A42	13-A70	13-A71	13-A89	13-A90	13-B85	13-B86	13-C02	13-C04	13-C06	13-C71	13-C72	13-C90	13-C91
	16-15	16-16	17-14	19-20	19-23	19-24	19-37	19-38	19-56	19-57	19-82	19-106	19-107	19-113
	19-114	19-119	19-120	20-10	20-11	20-21	20-22	20-25	20-26	20-27	20-28	21-10	21-11	21-18
	21-19	21-35	21-36	21-41	21-42	22-23	22-24	22-49	22-51	22-53				
PUTMR2	4-444#	13-387												
PUTOF	4-452#	13-81	13-168	13-175	13-218	13-224	13-266	13-383	13-405	13-444	13-468	13-496	19-77	
PUTSN	4-404#	13-386	13-429											
PUTWC	4-317#	19-78												
PUTX	4-301#													
REPORT	4-682#													
RGBFMC	4-12#	7-0	7-0											
SCTCMP	4-632#	13-819	13-930	13-:08	13-:50	13-:75	13-=43	13-?98	13-B25					
SETLFS	4-643#	13-933	13-:11	13-@01	13-B28									
SETOM	4-609#													
SETPRI	4-682#	11-49	15-64	15-87	31-1									
SETTRA	33-1	33-1	33-1	33-1	33-1	33-1	33-1	33-1	33-1	33-1	33-1	33-1	33-1	33-i#
SETUP	4-682#	9-23												
SETVV	4-598#													
SKIP	4-682#													
SLASH	4-682#													
STARS	4-682#	5-5	5-8	5-8	5-8	6-0	6-0	6-0	13-1	13-1	13-73	13-73	13-106	13-106
	13-137	13-137	13-159	13-159	13-326	13-326	13-518	13-518	13-543	13-543	13-554	13-554	13-601	13-601
	13-671	13-671	13-746	13-746	13-806	13-806	13-917	13-917	13-995	13-995	13-:37	13-:37	13-:54	13-:54
	13--29	13- 29	13-?75	13-?75	13-B10	13-B10	14-20	23-1	24-1	25-1	26-1	27-1	28-1	29-1
	31-1	31-1	31-1	31-1	31-1	32-1	33-1	34-1	34-1	35-1				
SWRSU	4-682#	9-23	9-23#											
TAGS	4-46#	6-0												
TRMTRP	33-1#													
TYPBIN	4-682#													
TYPDEC	4-682#	14-20	14-20											
TYPNAM	4-670#	4-682#	9-28											
TYPNUM	4-682#													
TYPOCS	4-682#	9-82	11-16	11-69	30-22	30-47	30-52	30-54						
TYPOCT	4-682#	10-35	31-1											
TYPTXT	4-682#	9-6	9-42	9-53	9-59	9-60	11-30	14-20	14-20					