

.REM \

IDENTIFICATION

PRODUCT CODE: AC-T106A-MC  
 PRODUCT NAME: CZRMB00 RMB0 DISKLESS TEST, PT 1  
 PRODUCT DATE: APRIL 1, 1982  
 MAINTAINER: CX DIAGNOSTIC GROUP  
 AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1982 DIGITAL EQUIPMENT CORPORATION

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

CONTENTS

-----

1  
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. XSDP COMPATIBILITY
  - 7. OPERATING SYSTEM COMPATIBILITY
- 6. TEST DESCRIPTION

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

## 1.0 INTRODUCTION

### 1.1 ABSTRACT

THE RM80 DISKLESS DIAGNOSTIC IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL AND DIAGNOSTIC MEANS TO VERIFY THE OPERABILITY OF THE RM80 DISK SUBSYSTEM EXCLUDING AND INDEPENDENTLY OF THE STORAGE MODULE DRIVE. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

TO DETECT ERRORS AND FAULTS IN THE RM70 MASSBUS CONTROLLER;

TO DETECT ERRORS AND FAULTS IN THE RM80 MASSBUS ADAPTER;

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

### 1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM80 DISK SUBSYSTEM, EXCLUDING THE STORAGE MODULE DISK DRIVE AND THE RM70 MASSBUS CONTROLLER.

## 2.0 OPERATING REQUIREMENTS

### 2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL, IS REQUIRED TO LOAD AND EXECUTE THE RM80 DISKLESS DIAGNOSTIC.

PDP-11/70 PROCESSOR  
20K MEMORY  
KW11-L OR KW11-P CLOCK  
PROGRAM LOADING DEVICE  
TERMINAL  
RM70 CONTROLLER  
1 TO 8 RM80 DISK DRIVES

### 2.2 MEDIA REQUIREMENTS

NONE

### 2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

NONE

## 3.0 OPERATING PROCEDURE

58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
4

### 3.1 LOADING

THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA

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

### 3.2 SWITCH OPTIONS

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

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

THE LOW ORDER 8 SWITCHES ARE USED IN CONJUNCTION WITH SW08 TO SPECIFY A PARTICULAR TEST WHICH THE PROGRAM WILL LOOP ON.

### 3.3 STARTING

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

### 3.4 HALTING

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

### 3.5 RESTARTING

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

## 4.0 OPERATOR INTERFACE

### 4.1 PROGRAM ID

THE PROGRAM TYPES ITS NAME AND MAINDEC NUMBER THE FIRST TIME IT

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

IS STARTED AFTER BEING LOADED

#### 4.2 CONSOLE DIALOGUE

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

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

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

##### EXAMPLE 1

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

##### EXAMPLE 2

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

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

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

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

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

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

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

'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 QUEUE. THE END OF PASS REPORT IS AS FOLLOWS.

'END OF PASS ?'

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

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

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

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

THE FOLLOWING PRINTOUT IS AN ERROR MESSAGE IN THIS PROGRAM:

```
DRV# 0 - RM80, TEST# 25, ERR# 63, PC=017566
ILLEGAL REGISTER ERROR "ILR" (RMER1, BIT 01) SHOULD BE SET
DURING REGISTER TRANSFER
PROBABLE FAULT(S):
(NOT INCLUDING CABLES OR CONNECTORS)
IF MODULE, M7686,
```

```
EXPCTD RECEVD TEST
STATUS STATUS PEGSTR
000002 000000 176750
```

#### 4.7 EXECUTION TIME

PASS 1 OF THE PROGRAM TAKES ABOUT 20 SECONDS. PASS 2 AND SUBSEQUENT PASSES TAKE 60 SECONDS.

#### 5.0 ENVIRONMENTAL SUPPORT

##### 5.1 PROCESSOR COMPATIBILITY

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

##### 5.2 DUAL PORT CONFIGURATIONS

THE RM80 DISKLESS DIAGNOSTIC IS NOT EXECUTABLE ON RM80 SUBSYSTEMS HAVING THE DUAL PORT OPTION UNLESS THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B) AND NOT TO THE PROGRAMMABLE POSITION (A/B).

##### 5.3 MEMORY PARITY HARDWARE

286 MEMORY PARITY HARDWARE WILL NOT BE USED DURING THE EXECUTION OF  
287 THE RM80 DISKLESS DIAGNOSTIC.  
288

289  
290  
291 5.4 MEMORY MANAGEMENT HARDWARE  
292

293 MEMORY MANAGEMENT HARDWARE WILL NOT BE USED DURING THE RM80  
294 DISKLESS DIAGNOSTIC.  
295

296  
297  
298 5.5 ACT11, APT11 COMPATIBILITY  
299

300 THE RM80 DISKLESS DIAGNOSTIC PROGRAM IS COMPATIBLE WITH ACT11  
301 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES. FURTHER, THE PROGRAM WILL  
302 EXECUTE A QUICK PASS DURING THE FIRST PASS IN SUPPORT OF QUICK VERIFY  
303 MODE.  
304

305  
306 5.6 XXDP COMPATIBILITY  
307

308 THE RM80 DISKLESS DIAGNOSTIC PROGRAM IS COMPATIBLE WITH XXDP  
309 IN DUMP AND CHAIN MODES.  
310

311  
312 5.7 OPERATING SYSTEM COMPATIBILITY  
313

314 THE PROGRAM IS NOT REQUIRED TO BE COMPATIBLE WITH ANY OPERATING  
315 SYSTEM.  
316

317  
318 6.0 TEST DESCRIPTION  
319

320 THE PROGRAM IS DESIGNED IN A BOTTOM UP MANNER SUCH THAT EACH TEST  
321 GENERALLY USES A MORE COMPLEX SUBSET OF HARDWARE THAN THE PREVIOUS  
322 TEST.  
323

324 MODULE CALLOUT IS PREDICATED ON THE ASSUMPTION THAT EARLIER TESTS  
325 HAVE BEEN COMPLETED WITHOUT ERROR AND THAT ERRORS ARE DUE TO SINGLE,  
326 NONTRANSIENT HARDWARE FAILURES.  
327

328 THE 'RM80 DISKLESS DIAGNOSTIC' CAN BE EXECUTED USING AN RH70  
329 MASSBUS CONTROLLER.  
330

331 UNLESS SPECIFIED BY THE OPERATOR OR BY THE ENVIRONMENT TABLE THE  
332 TEST IS REPEATED FOR EACH POSSIBLE DEVICE STARTING WITH DEVICE 0.  
333

334 THE MODULES WHICH MAY BE CALLED OUT DURING THE EXECUTION OF THE  
335 TEST ARE AS FOLLOWS:  
336

337 IF  
338 CS  
339 DS  
340  
341  
342



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

MASSEUS MODULE

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

TEST 1 TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE RM80 CAN COMPLETE A REGISTER TRANSFER ON THE MASSBUS, AND, IN PARTICULAR, TO VERIFY THAT 'TRANSFER' IS NOT STUCK IN AN INACTIVE STATE.

PROCEDURE:

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

PROBABLE FAULT:

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

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

TEST 2 CTOD TEST

PURPOSE:

TO VERIFY THAT DATA CAN BE TRANSFERRED TO AND FROM THE RM80 USING THE CONTROL BUS AND, IN PARTICULAR, TO VERIFY THAT 'CONTROLLER TO DEVICE' HAS NOT FAILED.

PROCEDURE:

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

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

## PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

## 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

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

TEST 5 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

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

ONE. THE ZERO REGISTER IS READ BACK AND IF THE SELECT LINE IS STUCK AT ZERO, THE ZERO REGISTER WILL CONTAIN ONES. THE PROCESS IS REPEATED TO DETECT A STUCK AT ONE FAULT, EXCEPT IN THIS CASE, THE ONES REGISTER IS WRITTEN FIRST.

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

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

TEST 7 DRIVE TYPE TEST

PURPOSE:

TO TEST THE 'DRIVE TYPE' REGISTER, RMDT.

PROCEDURE:

THE PROGRAM READS RMDT AND VERIFIES THAT THE RESULT CORRESPONDS TO A SINGLE PORT OR DUAL PORT RM80 DRIVE.

PROBABLE FAULT:

1. IF MODULE

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 HOLDING REGISTER TRANSFER TEST

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

PURPOSE:

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

PROCEDURE:

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

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

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

PROBABLE FAULT:

- 1. IF MODULE

TEST 12 CONTROL STATUS #1 TRANSFER TEST

PURPOSE:

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

PROCEDURE:

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

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

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

PROBABLE FAULT:

- 1. IF MODULE

TEST 13 ERROR REGISTER #1 TRANSFER TEST

PURPOSE:

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

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

PROCEDURE:

THIS TEST WRITES ONES IN ERROR REGISTER 1, RMER1, THEN  
WRITES ZEROS AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ONE.  
'UNSAFE' IS NOT TESTED DURING THIS TEST. IN ORDER TO LIMIT THE  
PROBABLE FAULTS TO ONE OR TWO MODULES, THE TEST IS EXECUTED IN 3

PARTS WITH EACH PART TESTING THOSE BITS WHOSE PRESET FUNCTIONS  
ARE DERIVED FROM THE SAME MODULE.

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

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

PROBABLE FAULT:

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

TEST 14 CLEAR OFFSET STUCK ACTIVE TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

1. IF MODULE
2. DS MODULE

TEST 15 OFFSET REGISTER TRANSFER 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 THE OFFSET REGISTER IS NOT STUCK AT ONE, STUCK AT ZERO, AND THAT THERE IS NO ADJACENT BIT INTERFERENCE.

PROCEDURE:

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

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

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

PROBABLE FAULT:

1. IF MODULE
2. DS MODULE

TEST 16 ERROR REGISTER #2 TRANSFER TEST

PURPOSE:

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

PROCEDURE:

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

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

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

PROBABLE FAULT:

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

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

## TEST 17 SERIAL NUMBER TEST

## PURPOSE:

TO VERIFY THAT THE SERIAL NUMBER CAN BE READ.

## PROCEDURE:

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

## PROBABLE FAULT:

1. CS MODULE

## TEST 20 CONTROL BUS PARITY DETECTION TEST

## PURPOSE:

TO TEST THE RMB0'S PARITY CHECKING LOGIC FOR THE MASSEUS ASYNCHRONOUS CONTROL BUS.

## PROCEDURE:

THIS TEST WRITES A SHIFTING ONE BIT DATA PATTERN IN THE DISK ADDRESS REGISTER USING 'PAT' TO CONTROL THE STATE OF THE PARITY BIT. 'PAR' STATUS, BIT 03 OF RMR1, IS CHECKED AFTER EACH PATTERN IS TRANSFERRED. NOTE THE FOLLOWING TABLE SHOWS A SET OF TEST PATTERNS THAT COULD BE USED INSTEAD OF A SHIFTING ONE BIT PATTERN.

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

## PROBABLE FAULT:

1. IF MODULE
2. ASSYNCHRONOUS MASSBUS MODULE

## TEST 21 CONTROL BUS PARITY GENERATION TEST

## PURPOSE:



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

TO TEST THE RMBO'S PARITY GENERATING LOGIC FOR THE MASSBUS ASYNCHRONOUS CONTROL BUS.

PROCEDURE:

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

DATA PATTERN	MCPE
000000	0
056747	0
135672	0
163135	0

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

TEST 22 RMDA, RMDC FAULT TEST

PURPOSE:

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

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

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

'EBL' STUCK, WHICH WOULD INHIBIT THE CLOCK FUNCTION

PROCEDURE:

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

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

## 3. CS MODULE

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

## TEST 23 DISK ADDRESS TRANSFER TEST

## PURPOSE:

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

## PROCEDURE:

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

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

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

## PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

## TEST 24 DESIRED CYLINDER TRANSFER TEST

## PURPOSE:

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

## PROCEDURE:

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

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

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

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

PROBABLE FAULT:

- 1. DS MODULE
- 2. IF MODULE

TEST 25 ILLEGAL REGISTER TEST

PURPOSE:

TO TEST ILLEGAL REGISTER ERROR DETECTION IN THE RM80.

PROCEDURE:

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

PROBABLE FAULT:

- 1. IF MODULE
- 2. ASYNCHRONOUS MASSBUS MODULE

TEST 26 RESET GO BY INIT TEST

PURPOSE:

TO VERIFY THAT GO CAN BE RESET BY INITIALIZE.

PROCEDURE:

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

PROBABLE FAULT:

- 1. CS MODULE
- 2. IF MODULE

TEST 27 DIAGNOSTIC MODE TEST

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

PURPOSE:

TO VERIFY THAT 'DIAGNOSTIC MODE', BIT 0 OF RPPR1, IS NOT STUCK AT ONE OR ZERO.

PROCEDURE:

THE RM80 IS INITIALIZED AND 'DMD' IS CHECKED FOR ZERO. 'DMD' IS WRITTEN WITH ONE AND READ TO VERIFY THAT IT IS NOT STUCK AT ZERO, THEN WRITTEN WITH ZERO AND READ TO VERIFY THAT IT IS NOT STUCK AT ONE.

PROBABLE FAULT:

- 1. CS MODULE
- 2. IF MODULE

TEST 30 MOL TEST

PURPOSE:

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

PROCEDURE:

AFTER INITIALIZING THE SUBSYSTEM, THE TEST SETS 'DIAGNOSTIC MODE' AND READS THE DRIVE STATUS REGISTER, RMD5, EXPECTING MOL, BIT 12 TO BE ZERO. 'MAINTENANCE UNIT READY', BIT 7 OF RPPR1, IS SET AND MOL SHOULD BE ONE. THE TEST THEN WRITES A ZERO IN MUR AND READS RMD5, VERIFYING THAT 'MEDIUM ON LINE' IS ZERO.

PROBABLE FAULT:

- 1. CS MODULE
- 2. IF MODULE

TEST 31 WRITE LOCK TEST

PURPOSE:

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

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE PROGRAM SETS MWP, BIT 03 OF

1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083

RMR1, AND READS RMD5 TO VERIFY THAT WRL, BIT 11 IS SET. THEN  
RMP IS RESET AND WRL SHOULD BE ZERO.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

TEST 32 DRIVE FAULT TEST

PURPOSE:

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

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE PROGRAM SETS MDF, BIT 06 OF  
RMR1, AND READS RMR3 TO VERIFY THAT DVC, BIT 07 IS SET RMR1 IS  
ALSO READ AND UNS, BIT 14 SHOULD ALSO BE SET. THEN MDF IS RESET  
AND DVC AND UNS SHOULD BE RESET.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

TEST 33 SEEK ERROR TEST

PURPOSE:

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

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE TEST SETS MSER, BIT 07 OF  
RMR1 AND READS RMR3 TO VERIFY THAT SKI, BIT 14 IS SET. MSER IS  
RESET AND SKI SHOULD RESET.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140

## TEST 34 PIP TEST

## PURPOSE:

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

## PROCEDURE:

DIAGNOSTIC MODE IS SET THEN MOC BIT 08 OF RMPR1 IS SET AND PIP, BIT 13 OF RMD5, SHOULD BE ZERO. MOC IS THEN RESET AND PIP SHOULD BE ONE.

## PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

## TEST 35 EBL TEST

## PURPOSE:

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

## PROCEDURE:

THE PROGRAM SETS DIAGNOSTIC MODE AND VERIFIES THAT EBL IS RESET. THEN IT SETS DEBL AND VERIFIES THAT EBL IS SET. FINALLY, THE TEST TRANSFERS A SHIFTING ONE BIT TO RMPR1, AND CHECKS FOR DEBL BEING SET BY AN ADJACENT BIT.

## PROBABLE FAULT:

1. CS MODULE

## TEST 36 LAST SECTOR, LAST TRACK TEST

## PURPOSE:

TO VERIFY THE DESIRED TRACK/SECTOR PLA ON THE DS MODULE USING RMPR1, BITS 01 AND 02.

## PROCEDURE:

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

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

PROBABLE FAULT:

1. DS MODULE
2. LS MODULE

#### TEST 37 RMDA COUNT TEST

PURPOSE:

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

PROCEDURE:

THE TEST INCREMENTS RMDA USING DIAGNOSTIC END OF BLOCK 'DEBL' AND VERIFIES THE RESULT IN 18 BIT FORMAT, 16 BIT FORMAT AND 16 BIT FORMAT WITH 'SSEI' SET.

PROBABLE FAULT:

1. DS MODULE

#### TEST 40 RMDC COUNT TEST

PURPOSE:

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

PROCEDURE:

THE PROGRAM INCREMENTS RMDC USING DIAGNOSTIC END OF BLOCK 'DEBL', AND VERIFIES THE RESULT IN 18 BIT FORMAT, 16 BIT FORMAT AND 16 BIT FORMAT WITH 'SSEI' SET.

PROBABLE FAULT:

1. DS MODULE

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

## TEST 41 LBT TEST

## PURPOSE:

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

## PROCEDURE:

THE TEST USES DIAGNOSTIC EBL TO SET LBT, AND TRANSFERS TO RMDA TO RESET LBT. THE RESULTS ARE VERIFIED IN 18 BIT FORMAT, 16 BIT FORMAT AND 16 BIT FORMAT WITH 'SSEI' SET.

## PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

## TEST 42 COMPOSITE ERROR TEST

## PURPOSE:

TO TEST 'COMPOSITE ERROR', BIT 14 OF RMD5.

## PROCEDURE:

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

## PROBABLE FAULT:

1. IF MODULE

## TEST 43 WRITE GO TEST

## PURPOSE:

TO VERIFY THAT GO CAN BE SET.

## PROCEDURE:

THE TEST ENABLES THE DEBUG CLOCK, THEN TRANSFERS A NOP



1255 FUNCTION CODE AND GO BIT TO RMCS1, VERIFYING THAT GO SETS. ALL  
1256 FUNCTION CODES ARE TESTED.

1257  
1258 PROBABLE FAULT:

- 1259  
1260 1. IF MODULE  
1261  
1262 2. CS MODULE  
1263

1264  
1265  
1266  
1267  
1268 TEST 44 BRANCH MULTIPLEXOR TEST

1269  
1270 PURPOSE:

1271 TO VERIFY THAT THE OUTPUT OF THE COMMAND SEQUENCER BRANCH  
1272 MULTIPLEXOR DOES NOT HAVE A FAULT.

1273  
1274 PROCEDURE:

1275 WITH DEBUG CLOCK ENABLED, THE TEST USES VARIOUS FUNCTION  
1276 CODES AND REGISTER CONDITIONS TO ADDRESS THE TEST BIT MULTIPLEXOR  
1277 SUCH THAT THE TEST BIT, BIT12 OF RMMR2, CAN BE CHECKED FOR A  
1278 STUCK FAULT.

1279  
1280 PROBABLE FAULT:

- 1281  
1282 1. CS MODULE  
1283

1284  
1285  
1286  
1287  
1288  
1289  
1290 TEST 45 SET/RESET GO TEST

1291  
1292 PURPOSE:

1293 TO VERIFY THAT GO CAN BE SET AND RESET.

1294  
1295 PROCEDURE:

1296 THE SUBSYSTEM IS INITIALIZED AND PUT IN DIAGNOSTIC MODE WITH  
1297 'DEBUG CLOCK ENABLE', BIT 14 OF RMMR1 SET. CERTAIN FUNCTION CODES  
1298 ARE WRITTEN IN RMCS1 AND THE PROGRAM READS RMCS1 TO VERIFY THAT  
1299 GO IS SET. RMD5 IS ALSO READ TO VERIFY THAT 'DRY' IS RESET.  
1300 THEN THE PROGRAM STEPS THE DEBUG CLOCK USING BIT 15 OF RMMR1 AND  
1301 VERIFIES THAT 'GO' RESETS AND 'DRY' SETS. USING A FUNCTION CODE  
1302 THAT RESETS GO AT A DIFFERENT PROM ADDRESS. THE TEST FAILS IF GO  
1303 DOES NOT SET OR CANNOT BE RESET BY THE COMMAND SEQUENCER. THE  
1304 TEST ALSO FAILS IF 'DRIVE READY' IS NOT THE COMPLIMENT OF GO.

1305  
1306 PROBABLE FAULT:

- 1307  
1308 1. CS MODULE  
1309  
1310  
1311

1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368

2. IF MODULE

TFST 46 END 1 RESET GO TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

- 1. CS MODULE

TEST 47 SET PULSE TEST

PURPOSE:

TO VERIFY THAT THE COMMAND SEQUENCER CAN GENERATE SET PULSE.

PROCEDURE:

WITH DEBUG CLOCK 9LED, THE TEST STEPS THE COMMAND SEQUENCER THROUGH PART. F VARIOUS FUNCTION CODES AND CHECKS CONTINUE, BIT 06 OF RMMR1 TO DETERMINE IF SET PULSE IS BEING GENERATED.

PROBABLE FAULT:

- 1. CS MODULE

TEST 50 SET/RESET IVC TEST

PURPOSE:

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

PROCEDURE:

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

THE PROGRAM RESETS VOLUME VALID USING 'MAINTENANCE UNIT READY', BIT09 OF RMR1, THEN LOADS THE FUNCTION CODE AND GO IN RMC51. EACH FUNCTION CODE IS TESTED AND 'IVC', BIT 12 OF RMR2 IS CHECKED.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

TEST 51 SET LSC TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

TEST 52 DECODE TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

1. IF MODULE

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

TEST 53 SET/RESET VOLUME VALID TEST

PURPOSE:

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

PROCEDURE:

USING 'MAINTENANCE UNIT READY', BIT 9 OF RMR1, THIS TEST FORCES A ZERO TO ONE TRANSITION OF UNIT READY AND VERIFIES THAT VOLUME VALID, BIT 6 OF RMD5 IS ZERO. THEN THE TEST EXECUTES A PACK ACKNOWLEDGE COMMAND, VERIFYING THAT VV SETS. THE PROCEDURE IS REPEATED WITH A READ IN PRESET COMMAND.

PROBABLE FAULT:

- 1. IF MODULE

TEST 54 ILLEGAL FUNCTION TEST

PURPOSE:

TO TEST ILLEGAL FUNCTION ERROR IN THE RM80.

PROCEDURE:

WITH DIAGNOSTIC CLOCK ENABLED TO INHIBIT THE COMMAND SEQUENCER, THIS TEST VERIFIES THAT 'ILF', BIT 0 OF RMR1, IS OFF FOR LEGAL FUNCTION CODES AND ON FOR ILLEGAL FUNCTIONCODES. THE STATUS OF THE 'GO' BIT IS IGNORED.

PROBABLE FAULT:

- 1. IF MODULE

TEST 55 OCCUPIED TEST

PURPOSE:

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

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

## PROCEDURE:

FOR EACH DATA TRANSFER COMMAND, 'DCC', BIT 15 OF RMPR1 SHOULD BE ONE, DEBUG CLOCK IS ENABLED TO PREVENT GO FROM RESETTING BEFORE STATUS IS SAMPLED.

## PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

## TEST 56 READ IN PRESET TEST

## PURPOSE:

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

## PROCEDURE:

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

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

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

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

THE TEST FAILS IF NONF OF THE PRESET TERMS ARE ZERO AFTER THE RIP COMMAND.

## PROBABLE FAULT:

1. IF MODULE
2. DS MODULE

## TEST 57 RIP/RMOF TEST

## PURPCSE:

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

TO VERIFY THAT 'READ IN PRESET' RESETS FMT16, ECI, HCI AND SSEI BITS 09, 10, 11 AND 12 OF RMOF.

PROCEDURE:

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

PROBABLE FAULT:

1. IF MODULE
2. DS MODULE

TEST 60 RMDA/RMDC/RIP TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

1. DS MODULE

TEST 61 OFFSET COMMAND TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

1. IF MODULE

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

TEST 62 RETURN TO CENTER TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

- 1. IF MODULE

TEST 63 RMDC CLEAR OFFSET TEST

PURPOSE:

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

PROCEDURE:

THE TEST EXECUTES A) OFFSET COMMAND, WRITES RMDC, AND VERIFIES THAT OM, BIT 00 OF RMD5 IS ZERO.

PROBABLE FAULT:

- 1. DS MODULE
- 2. IF MODULE

TEST 64 EBL CLEAR OFFSET TEST

PURPOSE:

TO VERIFY THAT OFFSET MODE CLEARS WHEN HEAD SWITCHING OCCURS.

PROCEDURE:

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

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

PROBABLE FAULT:

1. DS MODULE

#### TEST 65 RUN AND GO TEST

PURPOSE:

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

PROCEDURE:

THE RMB0 IS INITIALIZED AND A DATA TRANSFER COMMAND WITH GO SET IS WRITTEN IN RMCS1. 'RUN AND GO', BIT 14 OF REPORT SHOULD BE ONE FOR EACH DATA COMMAND. THE DEBUG CLOCK IS ENABLED SO THAT GO DOES NOT RESET BEFORE STATUS IS TESTED.

PROBABLE FAULT:

1. CS MODULE
2. SYNCHRONOUS MASSBUS MODULE

#### TEST 66 SET IAE TEST

PURPOSE:

TO VERIFY THAT INVALID ADDRESS ERROR CAN SET.

PROCEDURE:

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

PROBABLE FAULT:

1. DS MODULE
2. IF MODULE

#### TEST 67 SEARCH, SEEK, READ, WRITE TEST



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

## PURPOSE:

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

## PROCEDURE:

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

## PROBABLE FAULT:

1. IF MODULE

## TEST 70 INVALID TRACK/SECTOR TEST

## PURPOSE:

TO VERIFY THAT INVALID TRACK AND SECTOR ADDRESSES ARE DETECTED.

## PROCEDURE:

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

## PROBABLE FAULT:

1. DS MODULE
2. TRACK ADDRESS OPTION JUMPER

## TEST 71 INVALID CYLINDER TEST

## PURPOSE:

TO VERIFY THAT INVALID CYLINDER ADDRESSES ARE DETECTED.

## PROCEDURE:

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

## PROBABLE FAULTS:

1. DS MODULE

## 2. CYLINDER ADDRESS OPTION JUMPER

1758  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824

## TEST 72 SET AOE TEST

## PURPOSE:

TO VERIFY THAT ADDRESS OVERFLOW ERROR IS DETECTED.

## PROCEDURE:

THE TEST LOADS THE ADDRESS OF THE LAST SECTOR IN RMDA AND RMDC, THEN INITIATES A DATA COMMAND WITH DEBUG CLOCK ENABLED. END OF BLOCK IS FORCED TO INCREMENT THE SECTOR ADDRESS, AND THE TEST VERIFIES THAT 'AOE' IS SET, IN 18 BIT FORMAT, 16 BIT FORMAT AND 16 BIT FORMAT WITH 'SSEI' SET.

## PROBABLE FAULT:

1. DS MODULE

## TEST 73 SET RMR TEST

## PURPOSE:

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

## PROCEDURE:

'DEBUG CLOCK ENABLE' IS SET TO INHIBIT THE COMMAND SEQUENCER, THEN A NOP COMMAND AND GO BIT IS WRITTEN IN RMCST. WITHOUT STEPPING THE DEBUG CLOCK, THE TEST WRITES RMR AND RMA5, WHICH SHOULD NOT SET RMR STATUS. THEN RMDA IS WRITTEN AND RMR STATUS, BIT 02 OF RMR1, SHOULD BE ONE.

## PROBABLE FAULT:

1. IF MODULE

## TEST 74 PGM STATUS CHECK

## PURPOSE:

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

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

PROCEDURE:

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

PROBABLE FAULT:

1. IF MODULE

TEST 75 DVA/DPR STATUS CHECK

PURPOSE:

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

PROCEDURE:

DVA AND DPR ARE TESTED AND BOTH SHOULD BE ON.

PROBABLE FAULT:

1. IF MODULE

TEST 76 PORT REQUEST TEST, PART 1

PURPOSE:

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

PROCEDURE:

THE TEST EXECUTES A RELEASE COMMAND, THEN, ASSUMING THE PORT IS RELEASED, IT READS RMCS1, THEN READS RMPR2 AND VERIFIES THAT ONE OF THE PORT REQUEST FLOPS IS SET.

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

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

TEST 77 PORT REQUEST TEST, PART 2

PURPOSE:

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

PROCEDURE:

THE TEST EXECUTES A RELEASE COMMAND THEN WRITES RMAS AND READS RMR2, VERIFYING THAT ONE OF THE REQUEST FLOPS IS SET.

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

TEST 100 PORT REQUEST TEST, PART 3

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

TEST 101 RELEASE TEST

PURPOSE:

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

PROCEDURE:

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

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

## PROBABLE FAULT:

1. IF MODULE

## TEST 102 WRITE ATA TEST

## PURPOSE:

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

## PROCEDURE:

THE PROGRAM RESETS AND SETS UNIT READY WHICH SHOULD CAUSE AN ATTENTION, THEN WRITES THE ATTENTION SUMMARY REGISTER AND VERIFIES THAT ATTENTION IS RESET.

## PROBABLE FAULT:

1. IF MODULE
2. CS MODULE

## TEST 103 RESET ATA BY GO TEST

## PURPOSE:

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

## PROCEDURE:

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

## PROBABLE FAULT:

1. IF MODULE

## TEST 104 UNIT READY ATA TEST

## PURPOSE:

1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052

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

PROCEDURE:

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

PROBABLE FAULT:

- 1. IF MODULE

TEST 105 ERROR ATA TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

- 1. IF MODULE

TEST 106 REGISTER TRANSFER ATA TEST

PURPOSE:

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

PROCEDURE:

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

PROBABLE FAULT:

- 1. IF MODULE

2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109

## TEST 107 P SET ATA TEST

## PURPOSE:

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

## PROCEDURE:

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

## PROBABLE FAULT:

1. IF MODULE

## TEST 110 SET WLE TEST

## PURPOSE:

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

## PROCEDURE:

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

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

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

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

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

## PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

## TEST 111 EXCEPTION TEST

## PURPOSE:

2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166

TO VERIFY THAT 'REX' OF RMR1 IS RESET AFTER THE CONTROLLER IS INITIALIIZED AND SET WHEN AN ERROR IS DETECTED DURING A DATA TRANSFER COMMAND.

PROCEDURE:

THE PROGRAM WILL INITIALIZE THE MASSBUS ('REX' SHOULD BE CLEAR) AND THEN EXECUTE THE WRITE DATA COMMAND USING THE MAINTANCE DEBUG CLOCK. WHILE THE COMMAND IS BEING EXECUTED (RUN AND GO SET), THE PROGRAM CAUSES A 'RMR' ERRGR, BY TRYING TO WRITE THE RMR1 REGISTER ('RFX' SHOULD BE SET).

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

TEST 112 RECALIBRATE TEST

PURPOSE:

TO VERIFY THAT 'OPI' SETS, IF UNIT READY DROPS DURING RECALIBRATE COMMAND EXECUTION.

TO VERIFY THAT THE RECALIBRATE COMMAND ABORTS DURING COMMAND EXECUTION.

TO VERIFY THAT 'OPI' SETS, IF ON CYLINDER LATCH DOES NOT CLEAR

TO VERIFY THAT 'ATA' SETS, IF THE DRIVE COMPLETES THE RECALIBRATE COMMAND.

TO VERIFY THAT THE RECALIBRATE COMMAND ABORTS AFTER EXECUTION DURING A WAIT LOOP.

TO VERIFY THE TAG BUS DURING A RECALIBRATE COMMAND.

PROCEDURE:

THE PROGRAM EXECUTES THE FOLLOWING COMMANDS USING THE MAINTANCE DEBUG CLOCK AND EXPECTS THE RESULTS FOLLOWING EACH COMMAND.

EXECUTE RECALIBRATE COMMAND, DROP UNIT READY AND VERIFY THAT 'OPI' IS SET.

EXECUTE RECALIBRATE COMMAND, SET DRIVE FAULT ('MDF' IN RMR1) TO CAUSE COMMAND ABORT AND VERIFY THAT 'GO' IS RESET.

EXECUTE RECALIBRATE COMMAND, VERIFY THAT 'OPI' IS SET WHEN ON CYLINDER LATCH IS NOT CLEARED.



2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223

EXECUTE RECALIBRATE COMMAND, DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER AGAIN AND VERIFY THAT 'ATA' IS SET.

EXECUTE RECALIBRATE COMMAND, DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER RESET AND VERIFY THAT 'GO' IS STILL SET.

EXECUTE RECALIBRATE COMMAND AND VERIFY THAT THE TAG BUS IS CORRECT ACCORDING A PRE-DETERMINED TABLE.

PROBABLE FAULT:

- 1. CS MODULE

TEST 113 SEEK TEST

PURPOSE:

TO VERIFY THAT 'DPI' SETS, IF UNIT READY DROPS DURING SEEK COMMAND EXECUTION.

TO VERIFY THAT THE SEEK COMMAND ABORTS DURING COMMAND EXECUTION.

TO VERIFY THAT 'DPI' SETS, IF ON CYLINDER LATCH DOES NOT CLEAR.

TO VERIFY THAT 'ATA' SETS, IF THE DRIVE COMPLETES THE SEEK COMMAND.

TO VERIFY THAT THE SEEK COMMAND ABORTS AFTER EXECUTION DURING A WAIT LOOP.

TO VERIFY THE TAG BUS DURING A SEEK COMMAND.

PROCEDURE:

THE PROGRAM EXECUTES THE FOLLOWING COMMANDS USING THE MAINTANCE DEBUG CLOCK AND EXPECTS THE RESULTS FOLLOWING EACH COMMAND.

EXECUTE SEEK COMMAND, DROP UNIT READY AND VERIFY THAT 'DPI' IS SET.

EXECUTE SEEK COMMAND, SET DRIVE FAULT ('DF' IN RMR1) TO CAUSE COMMAND ABORT AND VERIFY THAT 'GO' IS RESET.

EXECUTE SEEK COMMAND, VERIFY THAT 'DPI' IS SET WHEN ON CYLINDER LATCH IS NOT CLEARED.

EXECUTE SEEK COMMAND, DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER AGAIN AND VERIFY THAT 'ATA' IS SET.

EXECUTE SEEK COMMAND, DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER RESET AND VERIFY THAT 'GO' IS STILL SET.

EXECUTE SEEK COMMAND AND VERIFY THAT THE TAG BUS IS CORRECT ACCORDING A PRE-DETERMINED TABLE.

2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231  
2232  
2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245  
2246  
2247  
2248  
2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259  
2260  
2261  
2262  
2263  
2264  
2265  
2266  
2267  
2268  
2269  
2270  
2271  
2272  
2273  
2274  
2275  
2276  
2277  
2278  
2279  
2280

PROBABLE FAULT:

- 1. CS MODULE

TEST 114 SEARCH TEST

TO VERIFY THAT 'DPI' SETS, IF UNIT READY DROPS DURING SEARCH COMMAND EXECUTION.

TO VERIFY THAT THE SEARCH COMMAND ABORTS DURING COMMAND EXECUTION.

TO VERIFY THAT 'DPI' SETS, IF ON CYLINDER LATCH DOES NOT CLEAR.

TO VERIFY THAT 'ATA' SETS, IF THE DRIVE COMPLETES THE SEARCH COMMAND.

TO VERIFY THAT THE SEARCH COMMAND ABORTS AFTER EXECUTION DURING A WAIT LOOP.

TO VERIFY THAT SEARCH COMMAND ABORTS DURING SECTOR COMPARE LOOP

TO VERIFY THE TAG BUS DURING A SEARCH COMMAND.

PROCEDURE:

THE PROGRAM EXECUTES THE FOLLOWING COMMANDS USING THE MAINTANCE DEBUG CLOCK AND EXPECTS THE RESULTS FOLLOWING EACH COMMAND.

EXECUTE SEARCH COMMAND, DROP UNIT READY AND VERIFY THAT 'DPI' IS SET.

EXECUTE SEARCH COMMAND, SET DRIVE FAULT ('MDF' IN RPPR1) TO CAUSE COMMAND ABORT AND VERIFY THAT 'GO' IS RESET.

EXECUTE SEARCH COMMAND, VERIFY THAT 'DPI' IS SET WHEN ON CYLINDER LATCH IS NOT CLEARED.

EXECUTE SEARCH COMMAND, DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER AGAIN AND VERIFY THAT 'ATA' IS SET.

EXECUTE SEARCH COMMAND, DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER RESET AND VERIFY THAT 'GO' IS STILL SET.

EXECUTE SEARCH COMMAND, WHILE IN SECTOR COMPARE LOOP, SET DRIVE FAULT ('MDF' IN RPPR1) TO CAUSE COMMAND ABORT AND VERIFY THAT 'ATA' IS SET.

EXECUTE SEARCH COMMAND AND VERIFY THAT THE TAG BUS IS CORRECT ACCORDING A PRE-DETERMINED TABLE.

PROBABLE FAULT:

- 1. CS MODULE

2281  
2282  
2283  
2284  
2285  
2286  
2287  
2288  
2289  
2290  
2291  
2292  
2293  
2294  
2295  
2296  
2297  
2298  
2299  
2300  
2301  
2302  
2303  
2304  
2305  
2306  
2307  
2308  
2309  
2310  
2311  
2312  
2313  
2314  
2315  
2316  
2317  
2318  
2319  
2320  
2321  
2322  
2323  
2324  
2325  
2326

## TEST 115 SEARCH TIMEOUT TEST

## PURPOSE:

TO VERIFY THAT 'DPI' SETS, IF 'MSEN' (SEARCH TIMEOUT ENABLE) IS DROPPED DURING SEARCH COMMAND EXECUTION.

## PROCEDURE:

EXECUTE SEARCH COMMAND, VERIFY THAT 'DPI' IS SET WHEN 'MSEN' IS CLEARED.

## PROBABLE FAULT:

1. CS MODULE

## TEST 116 - 120 DATA COMMAND TESTS (1, 2, 3)

## PURPOSE:

TO VERIFY THE COMMAND SEQUENCER DURING DATA COMMANDS.

## PROCEDURE:

THIS TEST, LIKE RECALIBRATE, SEEK, AND SEARCH TESTS, USES THE MAINTENANCE REGISTER TO SIMULATE DRIVE CONDITIONS AND FORCE THE COMMAND SEQUENCER THROUGH EACH BRANCH PATH. ADDITIONAL ITEMS WHICH ARE TESTED INCLUDE OFFSET PLUS AND MINUS ON THE TAG BUS AND 'ENABLE SEARCH', BIT 11 OF RMPR1.

## PROBABLE FAULT:

1. CS MODULE

1  
695  
696

```

:*LAST REVISION 07-AUG-81
*TITLE CZRNBAO RMB0 DSKLS PT1
*COPYRIGHT (C) 1982
*DIGITAL EQUIPMENT CORPORATION
*COLORADO SPGS., CO. 80919
*
*PROGRAM BY MIKE LEAVITT
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81

```

697

```

.SBTTL OPERATIONAL SWITCH SETTINGS

```

698

```

*
*      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<?:0>
*      7              TN128
*      6              TN64
*      5              TN32
*      4              TN16
*      3              TN8
*      2              TN4
*      1              TN2
*      0              TN1

```

699  
700

```

.SBTTL BASIC DEFINITIONS

```

```

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

```

```

000011  :*MISCELLANEOUS DEFINITIONS
000012  HT = 11             ;;CODE FOR HORIZONTAL TAB
000015  LF = 12             ;;CODE FOR LINE FEED
000200  CR = 15             ;;CODE FOR CARRIAGE RETURN
177776  CRLF = 200         ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776  PS = 177776        ;;PROCESSOR STATUS WORD
177774  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

```

```

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

```

000005	R5	=	85	::	GENERAL REGISTER
000006	R6	=	86	::	GENERAL REGISTER
000007	R7	=	87	::	GENERAL REGISTER
000006	SP	=	86	::	STACK POINTER
000007	PC	=	87	::	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
ERRVEC = 4           ;;TIME OUT AND OTHER ERRORS
RESVEC = 10          ;;RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC = 14         ;;'T' BIT
TRTVEC = 14         ;;TRACE TRAP
BPTVEC = 14         ;;BREAKPOINT TRAP (BPT)
IOTVEC = 20          ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC = 24          ;;POWER FAIL
EMTVEC = 30          ;;EMULATOR TRAP (EMT) **ERROR**
TRAPVEC = 34        ;;"TRAP" TRAP
TKVEC = 60           ;;TTY KEYBOARD VECTOR
TPVEC = 64           ;;TTY PRINTER VECTOR
PIRQVEC = 240        ;;PROGRAM INTERRUPT REQUEST VECTOR
    
```

701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729

.SBTTL RM80 REGISTER BIT DEFINITIONS

;\*RMCS1 CONTROL STATUS REGISTER

```

DVA = BIT11          ;DEVICE AVAILABLE-READ ONLY
F4 = BIT05           ;FUNCTION CODE
F3 = BIT04           ;FUNCTION CODE
F2 = BIT03           ;FUNCTION CODE
F1 = BIT02           ;FUNCTION CODE
F0 = BIT01           ;FUNCTION CODE
GO = BIT00           ;GO BIT
FNCSK = 000077      ;FUNCTION CODE MASK
    
```

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

```

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

732	000030	ILF30 = 000030	: ILLEGAL COMMAND
	000032	ILF32 = 000032	: ILLEGAL COMMAND
	000034	ILF34 = 000034	: ILLEGAL COMMAND
	000036	ILF36 = 000036	: ILLEGAL COMMAND
	000040	ILF40 = 000040	: ILLEGAL COMMAND
	000042	ILF42 = 000042	: ILLEGAL COMMAND
	000044	ILF44 = 000044	: ILLEGAL COMMAND
	000046	ILF46 = 000046	: ILLEGAL COMMAND
733	000050	WCD = 000050	: WRITE CHECK DATA COMMAND
734	000052	WCH = 000052	: WRITE CHECK HEADER AND DATA
735	000054	ILF54 = 000054	: ILLEGAL COMMAND
736	000056	ILF56 = 000056	: ILLEGAL COMMAND
737	000060	WD = 000060	: WRITE DATA COMMAND
738	000062	WH = 000062	: WRITE HEADER AND DATA COMMAND
739	000064	ILF64 = 000064	: ILLEGAL COMMAND
740	000066	ILF66 = 000066	: ILLEGAL COMMAND
741	000070	RD = 000070	: READ DATA COMMAND
742	000072	RH = 000072	: READ HEADER AND DATA COMMAND
743	000074	ILF74 = 000074	: ILLEGAL COMMAND
744	000076	ILF76 = 000076	: ILLEGAL COMMAND
745			
746		: *RMDA DISK ADDRESS REGISTER	
747			
748		: TRACK ADDRESS DEFINITIONS	
749	004000	TA8 = BIT11	: TRACK ADDRESS 8.
750	002000	TA4 = BIT10	: TRACK ADDRESS 4
751	001000	TA2 = BIT09	: TRACK ADDRESS 2
752	000400	TA1 = BIT08	: TRACK ADDRESS 1
753			
754		: SECTOR ADDRESS DEFINITIONS	
755	000020	SA16 = BIT04	: SECTOR ADDRESS 16.
756	000010	SAR = BIT03	: SECTOR ADDRESS 8.
757	000004	SA4 = BIT02	: SECTOR ADDRESS 4
758	000002	SA2 = BIT01	: SECTOR ADDRESS 2
759	000001	SA1 = BIT00	: SECTOR ADDRESS 1
760			
761		: TRACK & SECTOR MASKS	
762	177400	TADMSK = 177400	: TRACK ADDRESS MASK
763	000377	SADMSK = 000377	: SECTOR ADDRESS MASK
764			
765		: *RMDS DRIVE STATUS REGISTER	
766			
767	100000	ATA = BIT15	: ATTENTION ACTIVE
768	040000	ERR = BIT14	: COMPOSITE ERROR
769	020000	PIP = BIT13	: POSITIONING IN PROGRESS
770	010000	MOL = BIT12	: MEDIUM ON LINE
771	004000	WRL = BIT11	: WRITE LOCK
772	002000	LBT = BIT10	: LAST BLOCK TRANSFERRED
773	001000	PGM = BIT09	: PROGRAMMABLE
774	000400	DPR = BIT08	: DRIVE PRESENT
775	000200	DRY = BIT07	: DRIVE READY
776	000100	VV = BIT06	: VOLUME VALID
777	000001	OM = BIT00	: OFFSET MODE ACTIVE
778			
779		: *RMER1 ERROR REGISTER #1	
780			
781	100000	DCK = BIT15	: DATA CHECK ERROR

782	040000	UNS	= BIT14	:DRIVE UNSAFE
783	020000	OPI	= BIT13	:OPERATION INCOMPLETE
784	010000	DTE	= BIT12	:DRIVE TIMING ERROR
785	004000	WLE	= BIT11	:WRITE LOCK ERROR
786	002000	IAE	= BIT10	:INVALID ADDRESS ERROR
787	001000	AOE	= BIT09	:ADDRESS OVERFLOW ERROR
788	000400	HCRC	= BIT08	:HEADER CRC ERROR
789	000200	HCE	= BIT07	:HEADER COMPARE ERROR
790	000100	ECH	= BIT06	:ECC 'WORD' ERROR
791	000040	WCF	= BIT05	:WRITE CLOCK FAILURE
792	000020	FER	= BIT04	:FORMAT ERROR
793	000010	PAR	= BIT03	:PARITY ERROR
794	000004	RMR	= BIT02	:REGISTER MODIFICATION REFUSED
795	000002	ILR	= BIT01	:ILLEGAL REGISTER
796	000001	ILF	= BIT00	:ILLEGAL FUNCTION
797				
798	115760	NDTMSK	= DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER	
799				: 'NDTMSK' IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
800				: COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
801				
802				: *RMAS ATTENTION SUMMARY REGISTER
803				
804	000377	ATNMSK	= 377	: MASK FOR ATTENTION BITS
805				
806				: *RMLA LOOK AHEAD REGISTER
807				
808	002000	SC4	= BIT10	: SECTOR COUNT = 16
809	001000	SC3	= BIT09	: SECTOR COUNT = 8
810	000400	SC2	= BIT08	: SECTOR COUNT = 4
811	000200	SC1	= BIT07	: SECTOR COUNT = 2
812	000100	SC0	= BIT06	: SECTOR COUNT = 1
813				
814	003700	SCTMSK	= 003700	: SECTOR COUNT MASK
815				
816				: *RMR1 MAINTENANCE REGISTER #1
817				
818				: WRITE ONLY BITS
819	100000	DBCK	= BIT15	: DEBUG CLOCK
820	040000	DBEN	= BIT14	: DEBUG CLOCK ENABLE
821	020000	DEBL	= BIT13	: DIAGNOSTIC END OF BLOCK
822	010000	MSEN	= BIT12	: SEARCH TIMEOUT ENABLE
823	004000	MCLK	= BIT11	: MAINTENANCE CLOCK
824	002000	MRD	= BIT10	: READ DATA
825	001000	MLR	= BIT09	: UNIT READY
826	000400	MOC	= BIT08	: ON CYLINDER
827	000200	MSER	= BIT07	: SEEK ERROR
828	000100	PDF	= BIT06	: DRIVE FAULT
829	000040	MS	= BIT05	: SECTOR PULSE
830	000010	MWP	= BIT03	: WRITE PROTECT
831	000004	MI	= BIT02	: INDEX PULSE
832	000002	MSC	= BIT01	: SECTOR COMPARE
833	000001	MDI	= BIT00	: DIAGNOSTIC MODE
834				
835				: READ ONLY BITS
836	100000	OCC	= BIT15	: OCCUPIED
837	040000	RG	= BIT14	: RUN AND GO
838	020000	EBL	= BIT13	: END OF BLOCK



839	010000	REX	= BIT12	: EXCEPTION
840	004000	ESRC	= BIT11	: ENABLE SEARCH
841	002000	PLFS	= BIT10	: LOOKING FOR SYNC
842	001000	ECRC	= BIT09	: ENABLE CRC OUT
843	000400	PDA	= BIT08	: DATA AREA
844	000200	PHA	= BIT07	: HEADER AREA
845	000100	CONT	= BIT06	: CONTINUE
846	000040	WC	= BIT05	: WORD CLOCK
847	000020	ECC	= BIT04	: ENABLE ECC OUT
848	000010	MWD	= BIT03	: WRITE DATA BIT
849	000004	LS	= BIT02	: LAST SECTOR
850	000002	LST	= BIT01	: LAST SECTOR AND TRACK
851	000001	DMD	= BIT00	: DIAGNOSTIC MODE
852	051401	MRIAAA	= DMD!MUR!DBEN!MOC!MSEN	
853				
854		:*RMDT	DRIVE TYPE REGISTER	
855				
856	100000	NSA	= BIT15	: NOT SECTOR ADDRESSED = 0
857	040000	TAP	= BIT14	: TAPE DRIVE = 0
858	020000	MOV	= BIT13	: MOVING HEAD = 1
859	004000	DRQ	= BIT11	: DRIVE REQUEST REQUIRED
860				
861	020026	SNGPRT	= 020026	: SINGLE PORT DRIVE TYPE
862	024026	DULPRT	= 024026	: DUAL PORT DRIVE TYPE
863				
864		:*RMOF	OFFSET REGISTER	
865				
866	010000	FMT16	= BIT12	: 16 BIT WORD FORMAT
867	004000	ECI	= BIT11	: ECC INHIBIT
868	002000	HCI	= BIT10	: HEADER COMPARE INHIBIT
869	001000	SSEI	= BIT09	: SKIP SECTOR ERROR INHIBIT
870	000200	OFD	= BIT07	: OFFSET FORWARD
871	160577	XNUOF	= 160577	: UNUSED BITS OF RMOF
872				
873		:*RMDC	DESIRED CYLINDER ADDRESS REGISTER	
874				
875	001777	CYLSK	= 001777	: MASK FOR CYLINDER ADDRESS
876	176000	XNUDC	= 176000	: UNUSED BITS OF RMDC
877				
878		:*RMR2	MAINTENANCE REGISTER #2	
879				
880		:READ ONLY BITS		
881	100000	RA	= BIT15	: PORT A REQUEST
882	040000	RB	= BIT14	: PORT B REQUEST
883	020000	TAG	= BIT13	: TAG CONTROL
884	010000	TST	= BIT12	: COMMAND SEQUENCE TEST BIT
885	004000	CC	= BIT11	: CONTROL OR CYLINDER TAG
886	002000	CH	= BIT10	: CONTROL OR HEAD TAG
889	001000	BB09	= BIT09	: TAG BUS
	000400	BB08	= BIT08	: TAG BUS
	000200	BB07	= BIT07	: TAG BUS
	000100	BB06	= BIT06	: TAG BUS
	000040	BB05	= BIT05	: TAG BUS
	000020	BB04	= BIT04	: TAG BUS
	000010	BB03	= BIT03	: TAG BUS
	000004	BB02	= BIT02	: TAG BUS
	000002	BB01	= BIT01	: TAG BUS

```

890          000001          BB00      = BIT00          ;TAG BUS
891
892          ;*RMR2 ERROR REGISTER 2
893          100000          GSE       = BIT15          ;BAD SECTOR ERROR
894          040000          SKI       = BIT14          ;SEEK INCOMPLETE
895          020000          OPE       = BIT13          ;OPERATOR PLUG ERROR
896          010000          IVC       = BIT12          ;INVALID COMMAND ERROR
897          004000          LSC       = BIT11          ;LOSS OF SYSTEM CLOCK
898          002000          LBC       = BIT10          ;LOSS OF BIT CLOCK
899          000200          DVC       = BIT07          ;DEVICE CHECK
900          000040          SSE       = BIT05          ;SKIP SECTOR ERROR
901          000010          DPE       = BIT03          ;DATA PARITY ERROR
902          001527          XNMR2     = 001527          ;UNUSED BITS OF RMR2
903
904          .SBTTL PROGRAM MNEMONICS
905
906          100000          MSE       = BIT15          ;MANUFACTURING DETECTED SECTOR ERROR
907          040000          USE       = BIT14          ;USER DETECTED SECTOR ERROR
908          020000          SSF       = BIT13          ;SKIP SECTOR FAILURE
909
910          .SBTTL RMB0 REGISTER INDEX VALUES
911
912          000000          RMCS1     = 00          ;CONTROL STATUS REGISTER #1
913          000006          RMDA     = 06          ;DISK ADDRESS REGISTER
914          000012          RMDS     = 12          ;DRIVE STATUS REGISTER
915          000014          RMR1     = 14          ;ERROR REGISTER #1
916          000016          RMAS     = 16          ;ATTENTION SUMMARY REGISTER
917          000020          RPLA     = 20          ;LOOK AHEAD REGISTER
918          000024          RMR1     = 24          ;MAINTENANCE REGISTER
919          000026          RMDT     = 26          ;DRIVE TYPE REGISTER
920          000030          RPSN     = 30          ;SERIAL NUMBER REGISTER
921          000032          RPOF     = 32          ;OFFSET REGISTER
922          000034          RMDC     = 34          ;DESIRED CYLINDER REGISTER
923          000036          RMR1     = 36          ;HOLDING REGISTER
924          000040          RMR2     = 40          ;MAINTENANCE REGISTER #2
925          000042          RMR2     = 42          ;ERROR REGISTER #2
926          000044          RPEC1    = 44          ;ECC POSITION REGISTER
927          000046          RPEC2    = 46          ;ECC PATTERN REGISTER
930          000050          ILRG50   = 50          ;ILLEGAL REGISTER 50
931          000052          ILRG52   = 52          ;ILLEGAL REGISTER 52
932          000054          ILRG54   = 54          ;ILLEGAL REGISTER 54
933          000056          ILRG56   = 56          ;ILLEGAL REGISTER 56
934          000060          ILRG60   = 60          ;ILLEGAL REGISTER 60
935          000062          ILRG62   = 62          ;ILLEGAL REGISTER 62
936          000064          ILRG64   = 64          ;ILLEGAL REGISTER 64
          000066          ILRG66   = 66          ;ILLEGAL REGISTER 66
          000070          ILRG70   = 70          ;ILLEGAL REGISTER 70
          000072          ILRG72   = 72          ;ILLEGAL REGISTER 72
          000074          ILRG74   = 74          ;ILLEGAL REGISTER 74
          000076          ILRG76   = 76          ;ILLEGAL REGISTER 76
931
932          000077          IDXMSK    = 77          ;MASK FOR REGISTER INDEX NUMBER
933
934          .SBTTL RH CONTROLLER REGISTER BIT DEFINITIONS
935
936
  
```

```

937          ;*RMCS1 CONTROL STATUS REGISTER #1
938
939          100000      SC      = BIT15      .SPECIAL CONDITION-READ ONLY
940          040000      TRE      = BIT14      .TRANSFER ERROR
941          020000      MCPE     = BIT13      .MASSBUS CONTROL BUS PARITY ERROR-READ ONLY
942          002000      FSEL     = BIT10      .PORT B SELECT
943          001000      A17      = BIT09      .ADDRESS EXTENSION
944          000400      A16      = BIT08      .ADDRESS EXTENSION
945          000200      RDY      = BIT07      .READY-READ ONLY
946          000100      IE       = BIT06      .INTERRUPT ENABLE
947
948          ;*RMCS2 RM CONTROL STATUS REGISTER #2
949
950          100000      DLT       = BIT15      .DATA LATE-READ ONLY
951          040000      WCE       = BIT14      .WRITE CHECK ERROR-READ ONLY
952          020000      UPE       = BIT13      .UNIBUS PARITY ERROR
953          010000      NED       = BIT12      .NONEXISTANT DRIVE-READ ONLY
954          004000      NEM       = BIT11      .NONEXISTANT MEMORY-READ ONLY
955          002000      PGE       = BIT10      .PROGRAM ERROR-READ ONLY
956          001000      MXF       = BIT09      .MISSED TRANSFER
957          000400      MDPE      = BIT08      .MASSBUS DATA BUS PARITY ERROR-READ ONLY
958          000200      OR        = BIT07      .OUTPUT READY-READ ONLY
959          000100      IR        = BIT06      .INPUT READY-READ ONLY
960          000040      CLR       = BIT05      .CONTROLLER CLEAR
961          000020      PAT       = BIT04      .PARITY TEST
962          000010      BAI       = BIT03      .UNIBUS ADDRESS INCREMENT INHIBIT
963          000004      U2        = BIT02      .UNIT SELECT
964          000002      U1        = BIT01      .UNIT SELECT
965          000001      U0        = BIT00      .UNIT SELECT
966
967          ;UNIT SELECT MASK
968
969          000007      UNTMASK = 7          ;UNIT SELECT MASK
970
971          ;*RMCS3 RM70 CONTROL STATUS REGISTER #3
972
973          100000      APE       = BIT15      .ADDRESS PARITY ERROR
974          040000      DPEHI     = BIT14      .DATA PARITY ERROR HIGH WORD
975          020000      DPELO     = BIT13      .DATA PARITY ERROR LOW WORD
976          010000      WCEHI     = BIT12      .WRITE CHECK ERROR HIGH WORD
977          004000      WCELO     = BIT11      .WRITE CHECK ERROR LOW WORD
978          002000      DBL       = BIT10      .DOUBLE WORD TRANSFER
979          000100      IE        = BIT06      .INTERRUPT ENABLE
980          000010      IPCK3     = BIT03      .INVERT PARITY CHECK
981          000004      IPCK2     = BIT02      .INVERT PARITY CHECK
982          000002      IPCK1     = BIT01      .INVERT PARITY CHECK
983          000001      IPCK0     = BIT00      .INVERT PARITY CHECK
984
985          .SBTTL RM CONTROLLER REGISTER INDEX VALUES
986
987          000000      RMCS1     = 00          ;CONTROL, STATUS REGISTER #1
988          000002      RMWC      = 02          ;WORD COUNT REGISTER
989          000004      RMBR      = 04          ;BUS ADDRESS REGISTER
990          000010      RMCS2     = 10          ;CONTROL, STATUS REGISTER #2
991          000022      RMDB      = 22          ;DATA BUFFER
992          000050      RMBAE     = 50          ;BUS ADDRESS EXTENSION
993          000052      RMCS3     = 52          ;CONTROL, STATUS REGISTER #3
  
```

994  
995  
996  
997

176700  
120254

ABASE = 176700  
AVECT1 = 120254

:UNIBUS ADDRESS  
:UNIBUS VECTOR ADDRESS AND PRIORITY

```

1          .SBTTL TRAP CATCHER
          .=0
          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,MALT'
          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
          .=174
000174 000174 000000  DISREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
000176 000000 000000  SWREG:  .WORD 0      ;;SOFTWARE SWITCH REGISTER

          .SBTTL STARTING ADDRESS(ES)
2          000200 000137 004652  JMP      @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
3          000204 000137 004642  JMP      @#START1    ;;CHANGE RM/RM BUS ADDRESS
4
5          .SBTTL ACT11 HOOKS
          ;*****
          ;HOOKS REQUIRED BY ACT11
          $SVPC=.      ;SAVE PC
          .=46
000046 000210 054302  $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
          .=52
000052 000000 000210  .WORD 0      ;;2)SET LOC.52 TO ZERO
          .=$SVPC      ;; RESTORE PC

11         .=1100
12         .SBTTL APT PARAMETER BLOCK
13         ;*****
          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
          ;*****
          . $X=.      ;;SAVE CURRENT LOCATION
000024 000024 000200  .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
          .=44      ;;FOR APT START UP
000044 000044 001100  .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
          $APTHDR      ;;POINT TO APT HEADER BLOCK
          .=$X      ;;RESET LOCATION COUNTER
          ;*****
          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
          ;INTERFACE SPEC.

001100 001100 000000  $APTHC:
001100 001222 000000  $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102 001222 000000  $MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104 000024 000000  $STMT:  .WORD 20.    ;;RUN TIM OF LONGEST TEST
001106 000024 000000  $PASTM: .WORD 20.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110 000024 000000  $UNITM: .WORD 20.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112 000042 000000  .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
14         TAGADR=.
  
```

0

.SBTTL COMMON TAGS

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

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

.SBTTL APT MAILBOX-ETABLE

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

001230	000000	SPASS:	.WORD	APASS	::PASS COUNT
001232	000000	SDEVCT	.WORD	ADEVCT	::DEVICE COUNT
001234	000000	SUNIT	.WORD	AUNIT	::I/O UNIT NUMBER
001236	000000	SMSGAD	.WORD	AMSGAD	::MESSAGE ADDRESS
001240	000000	SMSGLG	.WORD	AMSGLG	::MESSAGE LENGTH
001242		SETABLE			::APT ENVIRONMENT TABLE
001242	000	SENV:	.BYTE	AENV	::ENVIRONMENT BYTE
001243	000	SEVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
001244	000000	SSWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
001246	000000	SUSR:	.WORD	AUSR	::USER SWITCHES
001250	000000	SCPUOP:	.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	SMAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001253	000	SMTYP1:	.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	SMADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
		*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
001256	000	SMAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001257	000	SMTYP2:	.BYTE	AMTYP2	::MEM. TYPE,BLK#2
001260	000000	SMADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001262	000	SMAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001263	000	SMTYP3:	.BYTE	AMTYP3	::MEM. TYPE,BLK#3
001264	000000	SMADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001266	000	SMAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001267	000	SMTYP4:	.BYTE	AMTYP4	::MEM. TYPE,BLK#4
001270	000000	SMADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001272	120254	SVECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001274	000000	SVECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001276	176700	SBASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001300	000000	SDEVN:	.WORD	ADEVN	::DEVICE MAP
001302	000000	SCDW1:	.WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001304	000000	SCDW2:	.WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001306	000000	SDDW0:	.WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
001310	000000	SDDW1:	.WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
001312	000000	SDDW2:	.WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
001314	000000	SDDW3:	.WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
001316	000000	SDDW4:	.WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
001320	000000	SDDW5:	.WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
001322	000000	SDDW6:	.WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
001324	000000	SDDW7:	.WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
001326		SETEND:			
		.MEXIT			

.SBTTL USER DEFINED TAGS

001326 000000  
 001330 000000  
 001332 000000

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

001334 000  
 001335 015

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

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

001336

GETBUF:

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

;REGISTER INPUT BUFFER

RMCS1I: .WORD 0 ;CONTROL, STATUS REGISTER #1  
 RMC1I: .WORD 0 ;WORD COUNT REGISTER  
 RMBAI: .WORD 0 ;BUS ADDRESS REGISTER  
 RMDAI: .WORD 0 ;DISK ADDRESS REGISTER  
 RMCS2I: .WORD 0 ;CONTROL, STATUS REGISTER #2  
 RMDSI: .WORD 0 ;DRIVE STATUS REGISTER  
 RMER1I: .WORD 0 ;ERROR REGISTER #1  
 RMASI: .WORD 0 ;ATTENTION SUMMARY REGISTER  
 RMLAI: .WORD 0 ;LOOK AHEAD REGISTER  
 RMCBI: .WORD 0 ;DATA BUFFER  
 RMR1I: .WORD 0 ;MAINTENANCE REGISTER #1  
 RMDTI: .WORD 0 ;DRIVE TYPE REGISTER  
 RMSNI: .WORD 0 ;SERIAL NUMBER REGISTER  
 RMOFI: .WORD 0 ;OFFSET REGISTER  
 RMDCI: .WORD 0 ;DESIRED CYLINDER REGISTER  
 RMR1I: .WORD 0 ;HOLDING REGISTER  
 RMR2I: .WORD 0 ;MAINTENANCE REGISTER #2  
 RMER2I: .WORD 0 ;ERROR REGISTER #2  
 RMECI: .WORD 0 ;ECC POSITION REGISTER  
 RMECI: .WORD 0 ;ECC PATTERN REGISTER  
 RMBAEI: .WORD 0 ;BUS ADDRESS EXTENSION REGISTER  
 RMCS3I: .WORD 0 ;CONTROL, STATUS REGISTER #3

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

001412

PUTBUF:

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

;REGISTER OUTPUT BUFFER

RMCS1O: .WORD 0 ;CONTROL, STATUS REGISTER #1  
 RMC1O: .WORD 0 ;WORD COUNT REGISTER  
 RMBAO: .WORD 0 ;BUS ADDRESS REGISTER  
 RMDAO: .WORD 0 ;DISK ADDRESS REGISTER  
 RMCS2O: .WORD 0 ;CONTROL, STATUS REGISTER #2  
 RMDSO: .WORD 0 ;DRIVE STATUS REGISTER  
 RMER1O: .WORD 0 ;ERROR REGISTER #1  
 RMASO: .WORD 0 ;ATTENTION SUMMARY REGISTER  
 RMLAO: .WORD 0 ;LOOK AHEAD REGISTER  
 RMDBO: .WORD 0 ;DATA BUFFER  
 RMR1O: .WORD 0 ;MAINTENANCE REGISTER #1



001440	000000	RMDTO	.WORD	0	:DRIVE TYPE REGISTER
001442	000000	RMSNC:	.WORD	0	:SERIAL NUMBER REGISTER
001444	000000	RMOFO:	.WORD	0	:OFFSET REGISTER
001446	000000	RMDCO:	.WORD	0	:DESIRED CYLINDER REGISTER
001450	000000	RPHRO:	.WORD	0	:HOLDING REGISTER
001452	000000	RPMR20:	.WORD	0	:MAINTENANCE REGISTER #2
001454	000000	RPER20:	.WORD	0	:ERROR REGISTER #2
001456	000000	RPEC10:	.WORD	0	:ECC POSITION REGISTER
001460	000000	RPEC20:	.WORD	0	:ECC PATTERN REGISTER
001462	000000	RMBAE0:	.WORD	0	:BUS ADDRESS EXTENSION REGISTER
001464	000000	RMCS30:	.WORD	0	:CONTROL, STATUS REGISTER #3

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

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

:PUT TAGS HERE

.SBTTL ERROR POINTER TABLE

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

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

1 001542  
 2  
 3

\$ERRTB:  
 ;ERROR 1            CANNOT CLEAR NED STATUS  
                   EMT1  
                   EHT1  
                   EDT1  
                   EFT1

001542 064376  
 001544 072262  
 001546 072362  
 001550 072410

4  
 5  
 6

;ERROR 2            CANNOT READ OR WRITE ANY DEVICE REG WITHOUT NED  
                   EMT2  
                   EHT2  
                   EDT2  
                   EFT2

001552 064404  
 001554 072266  
 001556 072364  
 001560 072412

7  
 8  
 9

;ERROR 3            CANNOT WRITE/READ ONES TO ANY DEVICE REGISTER  
                   EMT3  
                   0  
                   0  
                   0

001562 064432  
 001564 000000  
 001566 000000  
 001570 000000

10  
 11  
 12

;ERROR 4            CANNOT CLEAR ANY DEVICE REGISTER BITS W/MASSEL INIT  
                   EMT4  
                   0  
                   0  
                   0

001572 064452  
 001574 000000  
 001576 000000  
 001600 000000

13  
 14  
 15

;ERROR 5            CANNOT WRITE/READ ZEROS TO ALL BIT POSITIONS  
                   EMT5  
                   EHT5  
                   EDT5  
                   EFT5

001602 064474  
 001604 072272  
 001606 072366  
 001610 072414

16  
 17

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

18	001612 064520 001614 072272 001616 072366 001620 072414	EMT6 EMT5 EDT5 EFT5	
19			
20			:ERROR 7 CANNOT WRITE/READ SHIFTING ON BIT TO ALL BIT POSITIONS OF DEVICE REGISTERS
21			
22	001622 064542 001624 072276 001626 072366 001630 072414	EMT7 EMT7 EDT5 EFT5	
23			
24			:ERROR 10 REGISTER SELECT 1 APPEARS S-A-0
25	001632 064564 001634 000000 001636 000000 001640 000000	EMT10 0 0 0	
26			
27			:ERROR 11 REGISTER SELECT 1 APPEARS S-A-1
28	001642 064602 001644 000000 001646 000000 001650 000000	EMT11 0 0 0	
29			
30			:ERROR 12 REGISTER SELECT 2 APPEARS S-A-0
31	001652 064620 001654 000000 001656 000000 001660 000000	EMT12 0 0 0	
32			
33			:ERROR 13 REGISTER SELECT 2 APPEARS S-A-1
34	001662 064636 001664 000000 001666 000000 001670 000000	EMT13 0 0 0	
35			
36			:ERROR 14 REGISTER SELECT 4 APPEARS S-A-0
37	001672 064654 001674 000000 001676 000000 001700 000000	EMT14 0 0 0	
38			
39			:ERROR 15 REGISTER SELECT 4 APPEARS S-A-1

ERROR POINTER TABLE

40	001702	064672	EMT15	
	001704	000000	0	
	001706	000000	0	
	001710	000000	0	
41				
42			:ERROR 16	REGISTER SELECT 8 APPEARS S-A-0
43				
	001712	064710	EMT16	
	001714	000000	0	
	001716	000000	0	
	001720	000000	0	
44				
45			:ERROR 17	REGISTER SELECT 8 APPEARS S-A-1
46				
	001722	064726	EMT17	
	001724	000000	0	
	001726	000000	0	
	001730	000000	0	
47				
48			:ERROR 20	CANT WRITE ZEROS RMDA
49				
	001732	064744	EMT20	
	001734	072262	EHT1	
	001736	072362	EDT1	
	001740	072410	EFT1	
50				
51			:ERROR 21	CANT WRITE ONES RMDA
52				
	001742	064764	EMT21	
	001744	072262	EHT1	
	001746	072362	EDT1	
	001750	072410	EFT1	
53				
54			:ERROR 22	BIT INTERFERENCE IN WRITING/READING RMDA
55				
	001752	065004	EMT22	
	001754	072262	EHT1	
	001756	072362	EDT1	
	001760	072410	EFT1	
56				
57			:ERROR 23	CANT WRITE ZEROS RMCS1
58				
	001762	065020	EMT23	
	001764	072262	EHT1	
	001766	072362	EDT1	
	001770	072410	EFT1	
59				
60			:ERROR 24	CANT WRITE ONES RMCS1
61				

	001772	065040		EMT24
	001774	072262		EHT1
	001776	072362		EDT1
	002000	072410		EFT1
62				
63			.ERROR 25	BIT INTERFERENCE IN WRITING/READING RMCS1
64				
	002002	065060		EMT25
	002004	072262		EHT1
	002006	072362		EDT1
	002010	072410		EFT1
65				
66			:ERROR 26	MBA CLR L IS STUCK ACTIVE
67				
	002012	065074		EMT26
	002014	000000		O
	002016	000000		C
	002020	000000		O
68				
69			:ERROR 27	CANNOT CLEAR RMER1-PAR,RMR,ILF,ILR
70				
	002022	065126		EMT27
	002024	072262		EHT1
	002026	072362		EDT1
	002030	072410		EFT1
71				
72			:ERROR 30	CANNOT CLEAR RMER1-DCK,IAE,ADE,HCRC,HCE,ECH,WCF,FER
73				
	002032	065140		EMT30
	002034	072262		EHT1
	002036	072362		EDT1
	002040	072410		EFT1
74				
75			:ERROR 31	CANNOT CLEAR RMER1-OPI,DTE
76				
	002042	065154		EMT31
	002044	072262		EHT1
	002046	072362		EDT1
	002050	072410		EFT1
77				
78			:ERROR 32	CANNOT WRITE 0 IN RMER1-PAR,RMR,ILF,ILR
79				
	002052	065170		EMT32
	002054	072262		EHT1
	002056	072362		EDT1
	002060	072410		EFT1
80				
81			:ERROR 33	CANNOT WRITE 0 IN RMER1-DCK,IAE,ADE,HCRC,HCE,ECH,WCF,FER
82				
	002062	065204		EMT33

002064	072262		EHT1
002066	072362		EDT1
00207C	072410		EFT1
83			
84		:ERROR 34	CANNOT WRITE 0 IN RMER1-OPI,DTE
85			
002072	065222		EMT34
002074	072262		EHT1
002076	072362		EDT1
002100	072410		EFT1
86			
87		:ERROR 35	CANNOT WRITE 1 IN RMER1
88			
002102	065240		EMT35
002104	072262		EHT1
002106	072362		EDT1
002110	072410		EFT1
89			
90		:ERROR 36	CANNOT WRITE SHIFTING 1 IN RMER1
91			
002112	065254		EMT36
002114	072262		EHT1
002116	072362		EDT1
002120	072410		EFT1
92			
93		:ERROR 37	CANNOT WRITE ZEROS IN RMDC
94			
002122	065270		EMT37
002124	072262		EHT1
002126	072362		EDT1
002130	072410		EFT1
95			
96		:ERROR 40	CANNOT WRITE ONES IN RMDC
97			
002132	065304		EMT40
002134	072262		EHT1
002136	072362		EDT1
002140	072410		EFT1
98			
99		:ERROR 41	BIT INTERFERENCE IN WRITING/READING RMDC
100			
002142	065320		EMT41
002144	072262		EHT1
002146	072362		EDT1
002150	072410		EFT1
101			
102		:ERROR 42	CANNOT WRITE 1'S IN RMDC OR RMDA
103			
002152	065340		EMT42
002154	000000		0

	002156	000000	0	
	002160	000000	0	
104				
105			:ERROR 43	CANNOT CLEAR RMCS1-FUNCTION CODE
106				
	002162	065364	EMT43	
	002164	072262	EHT1	
	002166	072362	EDT1	
	002170	072410	EFT1	
107				
108			:ERROR 44	UNUSED BITS OF RMER2 NOT ZERO
109				
	002172	065376	EMT44	
	002174	072262	EHT1	
	002176	072362	EDT1	
	002200	072410	EFT1	
110				
111			:ERROR 45	CANNOT CLEAR RMER2-OPE,IVC,LSC
112				
	002202	065412	EMT45	
	002204	072262	EHT1	
	002206	072362	EDT1	
	002210	072410	EFT1	
113				
114			:ERROR 46	CANNOT CLEAR RMER2-LBC,DPE
115				
	002212	065426	EMT46	
	002214	072262	EHT1	
	002216	072362	EDT1	
	002220	072410	EFT1	
116				
117			:ERROR 47	CANNOT WRITE ZEROS RMER2-OPE,IVC,LSC
118				
	002222	065442	EMT47	
	002224	072262	EHT1	
	002226	072362	EDT1	
	002230	072410	EFT1	
119				
120			:ERROR 50	CANNOT WRITE ZEROS RMER2-LBC,DPE
121				
	002232	065464	EMT50	
	002234	072262	EHT1	
	002236	072362	EDT1	
	002240	072410	EFT1	
122				
123			:ERROR 51	CANNOT WRITE CNES RMER2
124				
	002242	065506	EMT51	
	002244	072262	EHT1	
	002246	072362	EDT1	

002250	072410	EFT1	
125			
126		:ERROR	52 CANNOT WRITE SHIFTING ONES RMR2
127			
002252	065526	EMT52	
002254	072262	EHT1	
002256	072362	EDT1	
002260	072410	EFT1	
128			
129		:ERROR	53 UNUSED BITS OF RMOF ARE NOT ZERO
130			
002262	065542	EMT53	
002264	072262	EHT1	
002266	072362	EDT1	
002270	072410	EFT1	
131			
132		:ERROR	54 CANNOT WRITE ZEROS RMOF
133			
002272	065556	EMT54	
002274	072262	EHT1	
002276	072362	EDT1	
002300	072410	EFT1	
134			
135		:ERROR	55 CANNOT WRITE ONES RMOF
136			
002302	065576	EMT55	
002304	072262	EHT1	
002306	072362	EDT1	
002310	072410	EFT1	
137			
138		:ERROR	56 CANNOT WRITE SHIFTING ONES RMOF
139			
002312	065616	EMT56	
002314	072262	EHT1	
002316	072362	EDT1	
002320	072410	EFT1	
140			
141		:ERROR	57 DEVICE IS NOT AN RMB0
142			
002322	065632	EMT57	
002324	072302	EHT57	
002326	072370	EDT57	
002330	072416	EFT57	
143			
144		:ERROR	60 DEVICE AVAILABLE IS NOT SET
145			
002332	065646	EMT60	
002334	072262	EHT1	
002336	072362	EDT1	
002340	072410	EFT1	



146			
147		:ERROR 61	CANNOT WRITE ZEROS RMHR
148			
	002342	065662	EMT61
	002344	072262	EHT1
	002346	072362	EDT1
	002350	072410	EFT1
149			
150		:ERROR 62	CANNOT WRITE ONES RMHR
151			
	002352	065702	EMT62
	002354	072262	EHT1
	002356	072362	EDT1
	002360	072410	EFT1
152			
153		:ERROR 63	CANNOT WRITE SHIFTING ONES RMHR
154			
	002362	065722	EMT63
	002364	072262	EHT1
	002366	072362	EDT1
	002370	072410	EFT1
155			
156		:ERROR 64	CANNOT CLEAR ILR STATUS
157			
	002372	065736	EMT64
	002374	072262	EHT1
	002376	072362	EDT1
	002400	072410	EFT1
158			
159		:ERROR 65	ILR ERROR SHOULD NOT BE SET
160			
	002402	065750	EMT65
	002404	072306	EHT65
	002406	072372	EDT65
	002410	072420	EFT65
161			
162		:ERROR 66	ILR ERROR SHOULD BE SET
163			
	002412	065764	EMT66
	002414	072306	EHT65
	002416	072372	EDT65
	002420	072420	EFT65
164			
165		:ERROR 67	CANNOT CLEAR PAR STATUS-DPE IS RESET
166			
	002422	066000	EMT67
	002424	072262	EHT1
	002426	072362	EDT1
	002430	072410	EFT1

167		:ERROR 70	CANNOT CLEAR PAR AND DPE STATUS
168			
169			
	002432	066016	EMT70
	002434	072262	EHT1
	002436	072362	EDT1
	002440	072410	EFT1
170			
171		:ERROR 71	'PAR' ERROR SHOULD NOT BE SET-'PAT' IS OFF
172			
	002442	066036	EMT71
	002444	072312	EHT71
	002446	072374	EDT71
	002450	072422	EFT71
173			
174		:ERROR 72	'PAR' ERROR SHOULD BE SET-'PAT' IS ON
175			
	002452	066062	EMT72
	002454	072312	EHT71
	002456	072374	EDT71
	002460	072422	EFT71
176			
177		:ERROR 73	'MCPE' ERROR SHOULD NOT BE SET
178			
	002462	066106	EMT73
	002464	072312	EHT71
	002466	072374	EDT71
	002470	072422	EFT71
179			
180		:ERROR 74	UNEXPECTED BUS TIMEOUT
181			
	002472	066126	EMT74
	002474	072316	EHT74
	002476	072376	EDT74
	002500	072424	EFT74
182			
183		:ERROR 75	CANT CLEAR 'DMD'
184			
	002502	066136	EMT75
	002504	072262	EHT1
	002506	072362	EDT1
	002510	072410	EFT1
185			
186		:ERROR 76	CANT WRITE ZERO 'DMD'
187			
	002512	066150	EMT76
	002514	072252	EHT1
	002516	072362	EDT1
	002520	072410	EFT1
188			

189		:ERROR 77	CANT WRITE ONE 'DMD'
190			
	002522 066164		EMT77
	002524 072262		EMT1
	002526 072362		EDT1
	002530 072410		EFT1
191			
192		:ERROR 100	DMD SET BY WRONG BIT
193			
	002532 066200		EMT100
	002534 072312		EMT71
	002536 072372		EDT65
	002540 072420		EFT65
194			
195		:ERROR 101	CANT CLEAR 'MOL' IN DIAGNOSTIC MODE
196			
	002542 066220		EMT101
	002544 072262		EMT1
	002546 072362		EDT1
	002550 072410		EFT1
197			
198		:ERROR 102	CANT SET 'MOL' IN DIAGNOSTIC MODE
199			
	002552 066240		EMT102
	002554 072262		EMT1
	002556 072362		EDT1
	002560 072410		EFT1
200			
201		:ERROR 103	'MUR' SET BY WRONG BIT
202			
	002562 066260		EMT103
	002564 072312		EMT71
	002566 072372		EDT65
	002570 072420		EFT65
203			
204		:ERROR 104	CANT RESET 'WRL' IN DIAGNOSTIC MODE
205			
	002572 066304		EMT104
	002574 072262		EMT1
	002576 072362		EDT1
	002600 072410		EFT1
206			
207		:ERROR 105	CANT SET 'WRL' IN DIAGNOSTIC MODE
208			
	002602 066324		EMT105
	002604 072262		EMT1
	002606 072362		EDT1
	002610 072410		EFT1
209			
210		:ERROR 106	'MWP' SET BY WRONG BIT

211	002612	066344	EMT106
	002614	072312	EHT71
	002616	072372	EDT65
	002620	072420	EFT65
212			
213			:ERROR 107 CANT RESET 'DVC' USING 'MDVC'
214	002622	066370	EMT107
	002624	072262	EHT1
	002626	072362	EDT1
	002630	072410	EFT1
215			
216			:ERROR 110 'DVC' IS RESET BUT 'UNS' IS SET
217	002632	066410	EMT110
	002634	072262	EHT1
	002636	072362	EDT1
	002640	072410	EFT1
218			
219			:ERROR 111 'DVC' IS SET BUT 'UNS' IS NOT SET
220	002642	066432	EMT111
	002644	072262	EHT1
	002646	072362	EDT1
	002650	072410	EFT1
221			
222			:ERROR 112 CANT SET 'DVC' USING MDVC'
223	002652	066452	EMT112
	002654	072262	EHT1
	002656	072362	EDT1
	002660	072410	EFT1
224			
225			:ERROR 113 'DVC' IS RESET BUT 'UNS' IS SET
226	002662	066472	EMT113
	002664	072262	EHT1
	002666	072362	EDT1
	002670	072410	EFT1
227			
228			:ERROR 114 'DVC' IS SET BUT 'UNS' IS NOT SET
229	002672	066516	EMT114
	002674	072262	EHT1
	002676	072362	EDT1
	002700	072410	EFT1
230			
231			:ERROR 115 'MDF' IS SET BY WRONG BIT
232			

002702	066540	EMT115
002704	072322	EHT115
002706	072400	EDT115
002710	072426	EFT115
233		
234		:ERROR 116 CANT RESEI 'SKI' USING 'MSER'
235		
002712	066564	EMT116
002714	072262	EHT1
002716	072362	EDT1
002720	072410	EFT1
236		
237		:ERROR 117 CANT SET 'SKI' USING 'MSER'
238		
002722	066604	EMT117
002724	072262	EHT1
002726	072362	EDT1
002730	072410	EFT1
239		
240		:ERROR 120 'SKI' SET BY WRONG BIT
241		
002732	066624	EMT120
002734	072322	EHT115
002736	072400	EDT115
002740	072426	EFT115
242		
243		:ERROR 121 CANT RESET 'PIP' USING 'MOC'
244		
002742	066650	EMT121
002744	072262	EHT1
002746	072362	EDT1
002750	072410	EFT1
245		
246		:ERROR 122 CANT SET 'PIP' USING 'MOC'
247		
002752	066670	EMT122
002754	072262	EHT1
002756	072362	EDT1
002760	072410	EFT1
248		
249		:ERROR 123 'MOC' SET BY WRONG BIT
250		
002762	066710	EMT123
002764	072322	EHT115
002766	072400	EDT115
002770	072426	EFT115
251		
252		:ERROR 124 CANT CLEAR 'EBL'
253		
002772	066734	EMT124

002774	072262	EMT1	
002776	072362	EDT1	
003000	072410	EFT1	
254			
255		:ERROR	125 'EBL' NOT ZERO IN DIAGNOSTIC MODE
256			
003002	066752	EMT125	
003004	072262	EMT1	
003006	072362	EDT1	
003010	072410	EFT1	
257			
258		:ERROR	126 CANT SET 'EBL' USING 'DEBL'
259			
003012	066772	EMT126	
003014	072262	EMT1	
003016	072362	EDT1	
003020	072410	EFT1	
260			
261		:ERROR	127 'DEBL' SET BY WRONG BIT
262			
003022	067010	EMT127	
003024	072322	EMT115	
003026	072400	EDT115	
003030	072426	EFT115	
263			
264		:ERROR	130 'LS' NOT CORRECT ACCORDING TO RMDA
265			
003032	067034	EMT130	
003034	072326	EMT130	
003036	072402	EDT130	
003040	072430	EFT130	
266			
267		:ERROR	131 'LST' NOT CORRECT ACCORDING TO RMDA
268			
003042	067052	EMT131	
003044	072326	EMT130	
003046	072402	EDT130	
003050	072430	EFT130	
269			
270		:ERROR	132 CANNOT INCREMENT SECTOR ADDRESS USING 'DEBL'
271			
003052	067070	EMT132	
003054	072332	EMT132	
003056	072404	EDT132	
003060	072432	EFT132	
272			
273		:ERROR	133 CANNOT INCREMENT TRACK ADDRESS USING 'DEBL'
274			
003062	067110	EMT133	
003064	072332	EMT132	

003066	072404	EDT132
003070	072432	EFT132
275		
276		:ERROR 134 UNUSED BITS OF RMDC NOT ZERO
277		
003072	067130	EMT134
003074	072262	EHT1
003076	072362	EDT1
003100	072410	EFT1
278		
279		:ERROR 135 'VV' NOT RESET BY UNIT READY
280		
003102	067144	EMT135
003104	072262	EHT1
003106	072362	EDT1
003110	072410	EFT1
281		
282		:ERROR 136 SERIAL NUMBER IS INCONSISTENT
283		
003112	067162	EMT136
003114	072262	EHT1
003116	072362	EDT1
003120	072410	EFT1
284		
285		:ERROR 137 CANT CLEAR 'GO' BIT
286		
003122	067174	EMT137
003124	072262	EHT1
003126	072362	EDT1
003130	072410	EFT1
287		
288		:ERROR 140 CANT INCREMENT CYLINDER USING 'DEBL'
289		
003132	067212	EMT140
003134	072332	EHT132
003136	072404	EDT132
003140	072432	EFT132
290		
291		:ERROR 141 CANT RESET 'LBT' BY WRITING RMDA
292		
003142	067232	EMT141
003144	072336	EHT142
003146	072404	EDT132
003150	072432	EFT132
293		
294		:ERROR 142 CANT SET 'LBT' USING 'DEBL'
295		
003152	067246	EMT142
003154	072336	EHT132
003156	072404	EDT132

003160	072432	EFT132	
296			
297		:ERROR 143	CANT READ ZERO FROM COMP ERROR
298			
003162	067264	EMT143	
003164	072262	EHT1	
003166	072362	EDT1	
003170	072410	EFT1	
299			
300		:ERROR 144	CANT SET COMP ERROR WITH RMER1 OR RMER2
301			
003172	067300	EMT144	
003174	072262	EHT1	
003176	072362	EDT1	
003200	072410	EFT1	
302			
303		:ERROR 145	COMP ERROR DID NOT SET
304			
003202	067322	EMT145	
003204	072342	EHT145	
003206	072402	EDT130	
003210	072430	EFT130	
305			
306		:ERROR 146	CANT SET 'GO' BIT
307			
003212	067344	EMT146	
003214	072262	EHT1	
003216	072362	EDT1	
003220	072410	EFT1	
308			
309		:ERROR 147	CANT READ A ONE FROM 'TST'
310			
003222	067362	EMT147	
003224	072262	EHT1	
003226	072362	EDT1	
003230	072410	EFT1	
311			
312		:ERROR 150	'TST' IS INCORRECT FOR THE FUNCTION CODE
313			
003232	067374	EMT150	
003234	072346	EHT150	
003236	072400	EDT115	
003240	072426	EFT115	
314			
315		:ERROR 151	CANT SET THE 'GO' BIT
316			
003242	067416	EMT151	
003244	072262	EHT1	
003246	072362	EDT1	
003250	072410	EFT1	



317				
318			:ERROR	152 'DRY' NOT THE COMPLEMENT OF 'GO'
319	003252	067430		EMT152
	003254	072262		EHT1
	003256	072362		EDT1
	003260	072410		EFT1
320				
321			:ERROR	153 'GO' RESET EARLY
322	003262	067444		EMT153
	003264	072262		EHT1
	003266	072362		EDT1
	003270	072410		EFT1
323				
324			:ERROR	154 'GO' DIDNT RESET ON TIME
325	003272	067464		EMT154
	003274	072262		EHT1
	003276	072362		EDT1
	003300	072410		EFT1
326				
327			:ERROR	155 CANT CLEAR CONTINUE
328	003302	067504		EMT155
	003304	072262		EHT1
	003306	072362		EDT1
	003310	072410		EFT1
329				
330			:ERROR	156 CONTINUE IS INCORRECT FOR THE FUNCTION CODE
331	003312	067522		EMT156
	003314	072366		EHT150
	003316	072400		EDT115
	003320	072426		EFT115
332				
333			:ERROR	157 CANT CLEAR IVC
334	003322	067544		EMT157
	003324	072262		EHT1
	003326	072362		EDT1
	003330	072410		EFT1
335				
336			:ERROR	160 IVC IS INCORRECT FOR THE FUNCTION CODE
337	003332	067562		EMT160
	003334	072366		EHT150
	003336	072400		EDT115
	003340	072426		EFT115

338			
339			
340			
	003342	067612	EMT161
	003344	072262	EHT1
	003346	072362	EDT1
	003350	072410	EFT1
341			
342			
343			
	003352	067630	EMT162
	003354	072262	EHT1
	003356	072362	EDT1
	003360	072410	EFT1
344			
345			
346			
	003362	067640	EMT163
	003364	072262	EHT1
	003366	072362	EDT1
	003370	072410	EFT1
347			
348			
349			
	003372	067670	EMT164
	003374	072262	EHT1
	003376	072362	EDT1
	003400	072410	EFT1
350			
351			
352			
	003402	067712	EMT165
	003404	000000	0
	003406	000000	0
	003410	000000	0
353			
354			
355			
	003412	067736	EMT166
	003414	072262	EHT1
	003416	072362	EDT1
	003420	072410	EFT1
356			
357			
358			
	003422	067756	EMT167
	003424	072262	EHT1
	003426	072362	EDT1
	003430	072410	EFT1
359			

:ERROR 161 CANT CLEAR LSC

:ERROR 162 CANT SET LSC

:ERROR 163 COMMAND DECODE WAS ENABLED WITH COMP ERROR SET

:ERROR 164 COMMAND DECODE WAS ENABLED WITH COMP ERROR SET

:ERROR 165 DECODE DOES NOT SET

:ERROR 166 CANT CLEAR OCCUPIED

:ERROR 167 ILF SET WITHOUT GI BIT

360			:ERROR 170	CANT SET VOLUME VALID
361	003432	070000		EMT170
	003434	072346		EMT150
	003436	072400		EDT115
	003440	072426		EFT115
362			:ERROR 171	ILF IS INCORRECT
363				EMT171
364	003442	070012		EMT150
	003444	072346		EDT115
	003446	072400		EFT115
	003450	072426		
365			:ERROR 172	CANT SET OFFSET DIRECTION BIT
366				EMT172
367	003452	070026		EMT1
	003454	072262		EDT1
	003456	072362		EFT1
	003460	072410		
368			:ERROR 173	OCCUPIED IS INCORRECT FOR FUNCTION CODE
369				EMT173
370	003462	070044		EMT150
	003464	072346		EDT115
	003466	072400		EFT115
	003470	072426		
371			:ERROR 174	READ IN PRESET DIDNT CLEAR RMDA, RMDC OR RMOF
372				EMT174
373	003472	070066		0
	003474	000000		0
	003476	000000		0
	003500	000000		
374			:ERROR 175	READ IN PRESET DIDNT CLEAR RMOF
375				EMT175
376	003502	070116		EMT1
	003504	072262		EDT1
	003506	072362		EFT1
	003510	072410		
377			:ERROR 176	READ IN PRESET DIDNT CLEAR RMDA
378				EMT176
379	003512	070136		EMT1
	003514	072262		EDT1
	003516	072362		EFT1
	003520	072410		
380			:ERROR 177	--RESERVED FOR POWER MONITOR BIT FAILURE--
381				

382	003522	000000	0	
	003524	000000	0	
	003526	000000	0	
	003530	000000	0	
383				
384				:ERROR 200 CANT SET OFFSET MODE BY OFFSET COMMAND
385	003532	070156	EMT200	
	003534	072262	EHT1	
	003536	072362	EDT1	
	003540	072410	EFT1	
386				
387				:ERROR 201 CANT RESET OFFSET MODE BY RTC COMMAND
388	003542	070174	EMT201	
	003544	072262	EHT1	
	003546	072362	EDT1	
	003550	072410	EFT1	
389				
390				:ERROR 202 CANT RESET OFD BY RTC COMMAND
391	003552	070212	EMT202	
	003554	072262	EHT1	
	003556	072362	EDT1	
	003560	072410	EFT1	
392				
393				:ERROR 203 CANT RESET OM BY RMDC
394	003562	070230	EMT203	
	003564	072262	EHT1	
	003566	072362	EDT1	
	003570	072410	EFT1	
395				
396				:ERROR 204 CANT RESET OM BY EBL
397	003572	070252	EMT204	
	003574	072262	EHT1	
	003576	072362	EDT1	
	003600	072410	EFT1	
398				
399				:ERROR 205 RUN AND GO NOT CORRECT FOR FUNCTION CODE
400	003602	070272	EMT205	
	003604	072362	EHT1150	
	003606	072400	EDT115	
	003610	072426	EFT115	
401				
402				:ERROR 206 CANT SET IAE ERROR
403				

	003612	070312		EMT206
	003614	000000		0
	003616	000000		0
	003620	000000		0
404				
405			:ERROR 207	IAE IS INCORRECT FOR FUNCTION CODE
406				
	003622	070342		EMT207
	003624	072345		EHT150
	003626	072400		EDT115
	003630	072426		EFT115
407				
408			:ERROR 210	IAE IS INCORRECT FOR RMDA
409				
	003632	070356		EMT210
	003634	072322		EHT115
	003636	072400		EDT115
	003640	072426		EFT115
410				
411			:ERROR 211	IAE IS INCORRECT FOR RMDC
412				
	003642	070374		EMT211
	003644	072322		EHT115
	003646	072400		EDT115
	003650	072426		EFT115
413				
414			:ERROR 212	CANT SET AOE
415				
	003652	070412		EMT212
	003654	072336		EHT142
	003656	072404		EDT132
	003660	072432		EFT132
416				
417			:ERROR 213	RMR SET WHEN WRITING RMAS OR RMCS
418				
	003662	070424		EMT213
	003664	072352		EHT213
	003666	072400		EDT115
	003670	072426		EFT115
419				
420			:ERROR 214	CANT SET RMR
421				
	003672	070454		EMT214
	003674	072352		EHT213
	003676	072400		EDT115
	003700	072426		EFT115
422				
423			:ERROR 215	DRQ IS 0 AND PGM IS 1
424				
	003702	070466		EMT215

003704	072262			EMT1
003706	072362			EDT1
003710	072410			EFT1
425				
426		:ERROR	216	DVA IS NOT SET
427				
003712	070506			EMT216
003714	072262			EHT1
003716	072362			EDT1
003720	072410			EFT1
428				
429		:ERROR	217	DPR IS NOT SET
430				
003722	070522			EMT217
003724	072262			EHT1
003726	072362			EDT1
003730	072410			EFT1
431				
432		:ERROR	220	CANT SET PORT REQUEST BY READING RMCS1
433				
003732	070536			EMT220
003734	072356			EHT220
003736	072406			EDT220
003740	072434			EFT220
434				
435		:ERROR	221	CANT SET PORT REQUEST BY WRITING RMAS
436				
003742	070554			EMT221
003744	072356			EHT220
003746	072406			EDT220
003750	072434			EFT220
437				
438		:ERROR	222	CANT SET PORT REQUEST BY WRITING RMDA
439				
003752	070572			EMT222
003754	072356			EHT220
003756	072406			EDT220
003760	072434			EFT220
440				
441		:ERROR	223	CANT RESET PORT REQUEST BY RELEASE COMMAND
442				
003762	070610			EMT223
003764	072356			EHT220
003766	072406			EDT220
003770	072434			EFT220
443				
444		:ERROR	224	CANT CLEAR ATA BY RMAS
445				
003772	070626			EMT224
003774	072262			EHT1

003776	072362	EDT1	
004000	072410	EFT1	
446			
447		:ERROR	225 ATA IS RESET BUT RMAS NOT ZERO
448			
004002	070646	EMT225	
004004	072262	EHT1	
004006	072362	EDT1	
004010	072410	EFT1	
449			
450		:ERROR	226 CANT RESET ATA BY GO
451			
004012	070670	EMT226	
004014	072262	EHT1	
004016	072362	EDT1	
004020	072410	EFT1	
452			
453		:ERROR	227 ATA NOT SET BY UNIT READY
454			
004022	070706	EMT227	
004024	072262	EHT1	
004026	072362	EDT1	
004030	072410	EFT1	
455			
456		:ERROR	230 ATA NOT SET BY UNIT READY
457			
004032	070724	EMT230	
004034	072262	EFT1	
004036	072362	EDT1	
004040	072410	EFT1	
458			
459		:ERROR	231 ATA NOT SET BY COMP ERROR
460			
004042	070742	EMT231	
004044	072262	EHT1	
004046	072362	EDT1	
004050	072410	EFT1	
461			
462		:ERROR	232 ATA SET/DID NOT SET WHEN REGISTER WRITTEN
463		:	WHILE COMP ERROR WAS SET
464			
004052	070756	EMT232	
004054	072352	EHT213	
004056	072400	EDT115	
004060	072426	EFT115	
465			
466		:ERROR	233 ATA NOT SET BY COMMAND SEQUENCER
467			
004062	071000	EMT233	
004064	072346	EHT150	

004066	072400	EDT115	
004070	072426	EFT115	
468			
469		:ERROR	234 WLE INCORRECT ACCORDING TO FUNCTION CODE
470			
004072	071022	EMT234	
004074	072346	EMT150	
004076	072400	EDT115	
004100	072426	EFT115	
471			
472		:ERROR	235 CANT CLEAR EXCEPTION
473			
004102	071050	EMT235	
004104	072262	EMT1	
004106	072362	EDT1	
004110	072410	EFT1	
474			
475		:ERROR	234 CANT SET EXCEPTION
476			
004112	071070	EMT236	
004114	072322	EMT115	
004116	072400	EDT115	
004120	072426	EFT115	
477			
478		:ERROR	237 CANT CLEAR IVC
479			
004122	071122	EMT237	
004124	072262	EMT1	
004126	072362	EDT1	
004130	072410	EFT1	
480			
481		:ERROR	240 CANT SET IVC
482			
004132	071122	EMT237	
004134	072346	EMT150	
004136	072400	EDT115	
004140	072426	EFT115	
483			
484		:ERROR	241 OPI NOT SET DURING RECALIBRATE
485			
004142	071166	EMT241	
004144	072262	EMT1	
004146	072362	EDT1	
004150	072410	EFT1	
486			
487		:ERROR	242 RECALIBRATE DID NOT ABORT WHEN DRIVE FAULT SET
488			
004152	071212	EMT242	
004154	072262	EMT1	
004156	072362	EDT1	



004160	072410	EFT1	
489			
490		:ERROR 243	OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
491		:	DROPPED DURING RECALIBRATE
492			
004162	071236	EMT243	
004164	072262	EHT1	
004166	072362	EDT1	
004170	072410	EFT1	
493			
494		:ERROR 244	ATA NOT SET DURING RECALIBRATE
495			
004172	071262	EMT244	
004174	072262	EHT1	
004176	072362	EDT1	
004200	072410	EFT1	
496			
497		:ERROR 245	GO RESET EARLY DURING RECALIBRATE
498			
004202	071300	EMT245	
004204	072262	EHT1	
004206	072362	EDT1	
004210	072410	EFT1	
499			
500		:ERROR 246	GO NOT RESET DURING RECALIBRATE
501			
004212	071316	EMT246	
004214	072262	EHT1	
004216	072362	EDT1	
004220	072410	EFT1	
502			
503		:ERROR 247	INCORRECT TAG BUS DURING RECALIBRATE
504			
004222	071342	EMT247	
004224	072262	EHT1	
004226	072362	EDT1	
004230	072410	EFT1	
505			
506		:ERROR 250	OPI SHOULD HAVE SET DURING SEEK BECAUSE UNIT
507		:	READY DROPPED
508			
004232	071360	EMT250	
004234	072262	EHT1	
004236	072362	EDT1	
004240	072410	EFT1	
509			
510		:ERROR 251	SEEK DID NOT ABORT WHEN DRIVE FAULT SET
511			
004242	071404	EMT251	
004244	072262	EHT1	

004246	072362	EDT1	
004250	072410	EFT1	
512			
513		:ERROR	252 OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
514		:	DROPPED DURING SEEK
515			
004252	071430	EMT252	
004254	072262	EHT1	
004256	072362	EDT1	
004260	072410	EFT1	
516			
517		:ERROR	253 ATA NOT SET DURING SEEK
518			
004262	071454	EMT253	
004264	072262	EHT1	
004266	072362	EDT1	
004270	072410	EFT1	
519			
520		:ERROR	254 GO RESET EARLY DURING SEEK
521			
004272	071472	EMT254	
004274	072262	EHT1	
004276	072362	EDT1	
004300	072410	EFT1	
522			
523		:ERROR	255 GO DID NOT RESET DURING SEEK
524			
004302	071510	EMT255	
004304	072262	EHT1	
004306	072362	EDT1	
004310	072410	EFT1	
525			
526		:ERROR	256 INCORRECT TAG BUS DURING SEEK
527			
004312	071534	EMT256	
004314	072262	EHT1	
004316	072362	EDT1	
004320	072410	EFT1	
528			
529		:ERROR	257 OPI NOT SET DURING SEARCH
530			
004322	071552	EMT257	
004324	072262	EHT1	
004326	072362	EDT1	
004330	072410	EFT1	
531			
532		:ERROR	260 SEARCH DID NOT ABORT WHEN DRIVE FAULT SET
533			
004332	071576	EMT260	
004334	072262	EHT1	

004336	072362	EDT1	
004340	072410	EFT1	
534			
535		:ERROR	261 OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
536		:	DROPPED DURING SEARCH
537			
004342	071622	EMT261	
004344	072262	EHT1	
004346	072362	EDT1	
004350	072410	EFT1	
538			
539		:ERROR	262 ATA NOT SET DURING SEARCH
540			
004352	071646	EMT262	
004354	072262	EHT1	
004356	072362	EDT1	
004360	072410	EFT1	
541			
542		:ERROR	263 GO RESET EARLY DURING SEARCH
543			
004362	071664	EMT263	
004364	072262	EHT1	
004366	072362	EDT1	
004370	072410	EFT1	
544			
545		:ERROR	264 GO DID NOT RESET DURING SEARCH
546			
004372	071702	EMT264	
004374	072262	EHT1	
004376	072362	EDT1	
004400	072410	EFT1	
547			
548		:ERROR	265 SEARCH ENABLE DIDNT SET DURING SEARCH
549			
004402	071726	EMT265	
004404	072262	EHT1	
004406	072362	EDT1	
004410	072410	EFT1	
550			
551		:ERROR	266 INCORRECT TAG BUS DURING SEARCH
552			
004412	071714	EMT266	
004414	072262	EHT1	
004416	072362	EDT1	
004420	072410	EFT1	
553			
554		:ERROR	267 OPI NOT SET BY SEARCH TIMEOUT
555			
004422	071762	EMT267	
004424	072262	EHT1	

004426	072362	ED11
004430	072410	EFT1
556		
557	:ERROR 270	GO NOT SET DURING DATA COMMAND
558		
004432	071776	EMT270
004434	072262	EHT1
004436	072362	EDT1
004440	072410	EFT1
559		
560	:ERROR 271	DATA COMMAND DID NOT ABORT WHEN DRIVE FAULT SET
561		
004442	072022	EMT271
004444	072262	EHT1
004446	072362	EDT1
004450	072410	EFT1
562		
563	:ERROR 272	EBL RESET EARLY DURING DATA COMMAND
564		
004452	072046	EMT272
004454	072262	EHT1
004456	072362	EDT1
004460	072410	EFT1
565		
566	:ERROR 273	EBL DIDNT RESET ON TIME DURING DATA COMMAND
567		
004462	072064	EMT273
004464	072262	EHT1
004466	072362	EDT1
004470	072410	EFT1
568		
569	:ERROR 274	GO NOT RESET DURING DATA COMMAND
570		
004472	072102	EMT274
004474	072262	EHT1
004476	072362	EDT1
004500	072410	EFT1
571		
572	:ERROR 275	RUN AND GO NOT SET DURING DATA COMMAND
573		
004502	072120	EMT275
004504	072262	EHT1
004506	072362	EDT1
004510	072410	EFT1
574		
575	:ERROR 276	INCORRECT TAG BUS DURING DATA COMMAND
576		
004512	072140	EMT276
004514	072262	EHT1
004516	072362	EDT1

004520	072410	EFT1	
577			
578		:ERROR 277	OPI NOT SET DURING DATA COMMAND WHEN ON
579		:	CYLINDER DIDNT DROP
580			
004522	072156	EMT277	
004524	072262	EHT1	
004526	072362	EDT1	
004530	072410	EFT1	
581			
582		:ERROR 300	DATA COMMAND DID NOT ABORT WHEN SEEK ERROR SET
583			
004532	072202	EMT300	
004534	072262	EHT1	
004536	072362	EDT1	
004540	072410	EFT1	
584			
585		:ERROR 301	SEARCH NOT ENABLED DURING DATA COMMAND
586			
004542	072226	EMT301	
004544	072262	EHT1	
004546	072362	EDT1	
004550	072410	EFT1	
587			
588		:ERROR 302	READ IN PRESET DIDNT CLEAR RMDC
589			
004552	072244	EMT302	
004554	072262	EHT1	
004556	072362	EDT1	
004560	072410	EFT1	
590		:PUT ERROR TABLE HERE	

```

1                                     ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 004562 011600 BADTMO: MOV (SP),R0 ;SAVE PC WHERE THE TIME OUT OCCURED
4 004564 005740 TST -(R0) ;ADJUST PC -2
5 004566 022626 CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
6 004570 104401 004576 TYPE ,65$ ;:TYPE ASCIZ STRING
004574 000417 BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
64$:
7 004634 010046 MOV R0,-(SP) ;SETUP FOR TYPING OUT PC
8 004636 104402 TYPOC
9 004640 000240 NOP ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
10 ;TO STOP ON UNEXPECTED TIMEOUT.
11
12 .SBTTL START OF PROGRAM
13
14 004642 012737 177777 001330 START1: MOV #-1,CHGADR ;CHANGE RH/RM BUS ADDRESS
15 004650 000402 BR START2
16
17 004652 005037 001330 START: CLR CHGADR ;NO CHANGE IN ADDRESS
18 004656 000240 START2: NOP
19 004660 005227 000000 INC #0 ;TTY LOOP, WAIT FOR INCREMENT
20 004664 001375 BNE -4 ;OF WORD
21 004666 000005 RESET ;RESET THE WORLD
22
23 .SBTTL INITIALIZE THE COMMON TAGS
;:CLEAR THE COMMON TAGS (SCMTAG) AREA
004670 012706 001114 MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
004674 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
004676 022706 001154 CMP #SWR,R6 ;:DONE?
004702 001374 BNE -6 ;:LOOP BACK IF NO
004704 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
;:INITIALIZE A FEW VECTORS
004710 012737 056370 000020 MOV #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
004716 012737 000340 000022 MOV #340,@IOTVEC+2 ;:LEVEL 7
004724 012737 057310 000020 MOV #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
004732 012737 000340 000022 MOV #340,@EMTVEC+2 ;:LEVEL 7
004740 012737 062314 000034 MOV #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
004746 012737 000340 000036 MOV #340,@TRAPVEC+2 ;:LEVEL 7
004754 012737 062422 000024 MOV #SPURON,@PURVEC ;:POWER FAILURE VECTOR
004762 012737 000340 000026 MOV #340,@PURVEC+2 ;:LEVEL 7
004770 013737 054140 054132 MOV SENDCT,SEOPCT ;:SETUP END-OF-PROGRAM COUNTER
004776 005037 001206 CLR TIMES ;:INITIALIZE NUMBER OF ITERATIONS
005002 005037 001210 CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
005006 112737 000001 001131 MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
005014 012737 005014 001122 MOV #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
005022 012737 005022 001124 MOV #,$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.
005030 013746 000004 MOV @ERRVEC,-(SP) ;:SAVE ERROR VECTOR
005034 012737 005070 000004 MOV #64$,@ERRVEC ;:SET UP ERROR VECTOR
005042 012737 177570 001154 MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
005050 012737 177570 001156 MOV #DISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
005056 022777 177777 174070 CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
005064 001012 BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
;:AND THE HARDWARE SWR IS NOT = -1
005066 000403 BR 65$ ;:BRANCH IF NO TIMEOUT
  
```

```

005070 012716 005076      64$:  MOV    #65$, (SP)      ;;SET UP FOR TRAP RETURN
005074 000002
005076 012737 000176 001154 65$:  MOV    #SWREG, SWR      ;;POINT TO SOFTWARE SWR
005104 012737 000174 001156 66$:  MOV    #DISPRÉG, DISPLAY
005112 012637 000004      66$:  MOV    (SP)+, @ERRVEC    ;;PESTORE ERROR VECTOR

005116 005037 001230      CLR    SPASS              ;;CLEAR PASS COUNT
005122 132737 000200 001243  BITB   #APTSIZE, SCNVM    ;;TEST USER SIZE UNDER APT
005130 001403      BEQ    67$              ;;YES, USE NON-APT SWITCH
005132 012737 001244 001154 67$:  MOV    #SSWREG, SWR      ;;NO, USE APT SWITCH REGISTER
005140

24 005140 012737 004562 000004 67$:  'TIMEOUT' TRAF VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 005140 012737 004562 000004 67$:  MOV    #BADTMO, ERRVEC   ;;SETUP FOR UNEXPECTED TIMEOUT
26 005146 012737 000300 000006 67$:  MOV    #PR6, ERRVEC+2   ;;LEVEL 6
27
28
.SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
005154 005227 177777      INC    #-1              ;;FIRST TIME?
005160 001032      BNE    68$              ;;BRANCH IF NO
005162 022737 054302 000042  CMP    #SENDAD, @#42    ;;ACT-11?
005170 001426      BEQ    68$              ;;BRANCH IF YES
005172 104401 005200      TYPE   69$              ;;TYPE ASCIZ STRING
005176 000423      BR     68$              ;;GET OVER THE ASCIZ
69$:  .ASCIZ <CRLF>@CZRMBAL - RMB0 DISKLESS TEST, PT 1@<CRLF>
68$:
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
005246 005737 000042      TST    @#42             ;;ARE WE RUNNING UNDER XXDP/ACT?
005252 001012      BNE    70$              ;;BRANCH IF YES
005254 123727 001242 000001  CMPB   $ENV, #1         ;;ARE WE RUNNING UNDER APT?
005262 001406      BEQ    70$              ;;BRANCH IF YES
005264 023727 001154 000176  CMP    SWR, #SWREG      ;;SOFTWARE SWITCH REG SELECTED?
005272 001005      BNE    71$              ;;BRANCH IF NO
005274 104407      GTSWR                    ;;GET SOFT-SWR SETTINGS
005276 000403      BR     71$
005300 112737 000001 001150 70$:  MOVB   #1, $AUTOB      ;;SET AUTO-MODE INDICATOR
005306 71$:

29
30
31
32
33 005306 005037 001332      CLR    XXDP              ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
34 005312 122737 000016 000041  CMPB   #16, @#41       ;;LOADED FROM AN RMB0 ?
35 005320 001121      BNE    3$               ;;BR IF NOT
36 005322 013737 000040 001332  MOV    @#40, XXDP        ;;GET DEVICE INDICATOR AND NUMBER
37 005330 122737 000007 001332  CMPB   #7, XXDP         ;;IS IT A VALID NUMBER ?
38 005336 103002      BHIS   1$              ;;YES
39 005340 105037 001332      CLRB   XXDP            ;;NO, DEFAULT TO DRIVE 0
40 005344 005737 000042 1$:  TST    @#42             ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
41 005350 001425      BEQ    2$              ;;BR IF NEITHER
42 005352 104401 005360      TYPE   73$              ;;TYPE ASCIZ STRING
005356 000412      BR     72$              ;;GET OVER THE ASCIZ
73$:  .ASCIZ <CRLF>/NOT TESTING DRIVE /
72$:

43 005404 005046      CLR    -(SP)            ;;CLEAR WORD ON STACK
44 005406 113716 001332      MOVB   XXDP, (SP)      ;;GET DRIVE ADDRESS
45 005412 104403      TYPOS                    ;;TYPE THE ADDRESS
46 005414 001      .BYTE  1              ;;ONLY 1 CHARACTER
    
```

```

47 005415 000 .BYTE 0 ;SUPRESS LEADING ZEROS
48 005416 104401 001217 TYPE $CRLF ;CR-LF
49 005422 000460 BR $$ ;GET NUMBER OF DRIVES
50
51 005424 005227 177777 2$: INC #-1 ;FIRST TIME THRU HERE ?
52 005430 001055 BNE 3$ ;NO
53 005432 104401 005440 TYPE 75$ ;:TYPE ASCIZ STRING
005436 000410 BR 74$ ;:GET OVER THE ASCIZ
;:75$: .ASCIZ <CRLF>/TO TEST DRIVE /
74$:
54 005460 005046 CLR -(SP) ;CLEAR WORD ON STACK
55 005462 113716 001332 MOVB XXDP,(SP) ;GET DRIVE ADDRESS
56 005466 104403 TYPOS ;TYPE DRIVE ADDRESS
57 005470 001 .BYTE 1 ;ONLY 1 CHARACTER
58 005471 000 .BYTE 0 ;SUPRESS LEADING ZEROS
59 005472 104401 005500 TYPE 76$ ;:TYPE ASCIZ STRING
005476 000432 BR $$ ;:GET OVER THE ASCIZ
;:76$: .ASCIZ /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
76$:
63 005564 ;:CHECK FOR AUTO MODE OR STANDALONE MODE
64 ;:CHECK FOR AUTO MODE OR STANDALONE MODE
65 005564 005037 001326 CLR AUTSIZ ;LET AUTO DRIVE SIZING OCCUR
66 005570 005737 000042 TST 2042 ;RUNNING IN AUTO MODE ?
67 005574 001537 BEQ STANDALONE ;BR IF NO
68 005576 012737 000377 001300 MOV #377,$DEVN ;SET DEVICE MAP FOR ALL DRIVES
69
70 ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
71 005604 XSIZ:
72 005604 132737 000200 001243 BITB #BIT7,$ENVN ;SIZING ALLOWED ?
73 005612 001124 BNE 12$ ;NO
74
75 005614 005001 CLR R1 ;START FROM DRIVE 0
76 005616 013700 001276 MOV $BASE,R0 ;LOAD THE BASE ADDRESS
77 005622 104401 063474 TYPE ,SYSTAT ;TYPE 'UNIT STATUS:'
78
79 005626 136137 063740 001300 1$: BITB ATATBL(R1),$DEVN ;IS DEVICE PRESENT IN MAP ?
80 005634 001507 BEQ 11$ ;BR IF NO
81 005636 104401 001217 TYPE $CRLF ;CR-LF
82 005642 010146 MOV R1,-(SP) ;:SAVE R1 FOR TYPEDOUT
005644 104403 TYPOS ;:GO TYPE--OCTAL ASCII
005646 002 .BYTE 2 ;:TYPE 2 DIGIT(S)
005647 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
83 005650 104401 063632 TYPE ,BLNKS4 ;TYPE 4 BLANKS
84
85 005654 012760 000040 000010 MOV #CLR,RMCS2(R0) ;CLEAR MASS BUS
86 005662 010160 000010 MOV R1,RMCS2(R0) ;LOAD THE DRIVE ADDRESS
87 005666 005760 000012 TST RMD5(R0) ;ACCESS DRIVE REGISTER
88 005672 032760 010090 000010 BIT #RD,RMCS2(R0) ;IS DRIVE PRESENT ?
89 005700 001027 BNE 3$ ;BR IF NO
90 005702 032760 004000 000000 BIT #DVA,RMCS1(R0) ;IS DRIVE AVAILABLE ?
91 005710 001426 BEQ 4$ ;BR IF NO
92 005712 012737 063512 006052 MOV #RM80,10$ ;ASSUME RM80 DEVICE
93 005720 022760 020026 000026 CMP #20026,RMDT(R0) ;SINGLE PORT RM80 ?
94 005726 001407 BEQ 2$ ;BR IF YES
95 005730 022760 024026 000026 CMP #24026,RMDT(R0) ;DUAL PORT RM80 ?
96 005736 001403 BEQ 2$ ;BR IF YES
97 005740 104401 063517 TYPE ,NOTRM ;DRIVE NOT AN RM80
    
```



98	005744	000443			BR	11\$		:CHECK NEXT DRIVE
99	005746	032760	010000	000012	2\$: BIT	#MOL,RMDS(RO)		:IS MEDIUM ON LINE ?
100	005754	001415			BEQ	6\$		:BR IF NO
101	005756	000417			BR	7\$		
102								
103	005760	104401	063551		3\$: TYPE	.NOTPRG		:DRIVE NOT PRESENT
104	005764	000402			BR	5\$		:CHECK NEXT DRIVE
105								
106	005766	104401	063566		4\$: TYPE	.NOTAVL		:DRIVE NOT AVAILABLE
107	005772	005737	001326		5\$: TST	AUTSIZ		:AUTO SIZING ON ?
108	005776	001026			BNE	11\$		:BR IF NO
109	006000	146137	063740	001300	BICB	ATNTBL(R1),SDEVN		:CLEAR DEVICE FROM BIT MAP
110	006006	000422			BR	11\$		:CHECK NEXT DRIVE
111								
112	006010	104401	063605		6\$: TYPE	.UNTOFF		:DRIVE OFFLINE
113	006014	000413			BR	9\$		:PRINT DRIVE TYPE
114								
115	006016	005737	001332		7\$: TST	XXDP		:LOADED FROM RM80 ?
116	006022	001406			BEQ	8\$		:NO
117	006024	123701	001332		CMPB	XXDP,R1		:IS THIS THE DRIVE ?
118	006030	001360			BNE	5\$		:BR IF NO
119	006032	104401	063534		TYPE	.LODEV		:DRIVE IS LOAD DEVICE
120	006036	000755			BR	5\$		
121								
122	006040	104401	063616		8\$: TYPE	.UNTON		:DRIVE ONLINE
123	006044	104401	063634		9\$: TYPE	.BLNKS2		:TYPE 2 BLANKS
124	006050	104401			TYPE			:PRINT DRIVE TYPE
125	006052	000000			10\$: .WORD	0		:MESSAGE ADDRESS HERE
126								
127	006054	005201			11\$: INC	R1		:INCREMENT THE DRIVE ADDRESS
128	006056	020127	000007		CMP	R1,#7		:ALL DRIVES ARE CHECKED ?
129	006062	003661			BLE	1\$		:BRANCH IF NOT
130								
131	006064	104401	001217		12\$: TYPE	.SCRLF		:CR-LF
132	006070	000137	006612		JMP	CMNSTART		:JUMP TO COMMON START

```

1          .SBTTL  STANDALONE INPUT ROUTINES
2
3          STANDALONE:
4 006074   004737   060730      JSR      PC,$KINT      :INITIALIZE CONSOLE
5
6 006100   005227   177777      INC      #-1           :FIRST TIME THRU HERE
7 006104   001023                BNE      2$           :BR IF NO
8
9          :SEE IF OPERATOR WANTS HELP TEXT
10
11 006106   104401   063064      TYPE     ,MSHELP      :WANT HELP ?
12 006112   104411                RDCMR                :GET RESPONSE
13 006114   012637   001176      MOV      (SP)+,$TMP1   :SAVE AND ECHO RESPONSE
14 006120   123727   001176      CMPB    $TMP1,#'Y     :WAS IT A YES RESPONSE ?
15 006126   001005                BNE      1$           :NO
16 006130   104401   001176      TYPE     , $TMP1      :TYPE 'Y'
17 006134   104401   104536      TYPE     ,HELP        :YES - TYPE HELP TEXT
18 006140   000414                BR       3$           :
19 006142   104401   063626      1$:     TYPE     ,N      :TYPE 'N'
20 006146   104401   001217      TYPE     , $CRLF      :CR-LF
21 006152   000407                BR       3$           :
22
23          :SEE IF USER WANTS TO CHANGE UNIBUS ADDRESS
24 006154                2$:
25 006154   005737   001330      TST     CHGADR        :CHANGE RH/RM BUS ADDRESS ?
26 006160   001475                BEQ     7$           :BR IF NO
27 006162   005037   001330      CLR     CHGADR        :NO CHANGE NEXT TIME
28 006166   104401   001217      TYPE     , $CRLF      :CR-LF
29
30          :DIALOGUE TO CHANGE THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY
31 006172                3$:
32 006172   104401   063115      TYPE     ,CNSLO1      :TYPE CURRENT BUS ADDRESS
33 006176   013746   001276      MOV     $BASE,-(SP)   :SAVE $BASE FOR TYPEOUT
34 006202   104402                TYPDC                :GO TYPE--OCTAL ASCII(ALL DIGITS)
35 006204   104401   063634      TYPE     ,BLNKS2     :TYPE 2 BLANKS
36 006210   101113                RDOCT                :GET NEW BUS ADDRESS
37 006212   012637   001176      MOV     (SP)+,$TMP1   :CARRIAGE RETURN ?
38 006216   001412                BEQ     5$           :YES-SKIP TO NEXT ENTRY
39 006220   022737   160000      CMP     #160000,$TMP1 :BASE ADDRESS IN I/O PAGE ?
40 006226   101403                BLOS   4$           :YES
41 006230   104401   063125      TYPE     ,CNSLO2     :TYPE WARNING MESSAGE
42 006234   000756                BR      3$           :TRY AGAIN
43 006236   013737   001176      MOV     $TMP1,$BASE   :STORE NEW BUS ADDRESS
44
45 006244   104401   063167      5$:     TYPE     ,CNSLO3
46 006250   005046                CLR     -(SP)
47 006252   113716   001272      MOVB   $VECT1,(SP)   :GET CURRENT VECTOR ADDRESS
48 006256   104402                TYPDC                :TYPE 2 BLANKS
49 006260   104401   063634      TYPE     ,BLNKS2     :GET NEW VECTOR ADDRESS
50 006266   012637   001176      RDOCT                :CARRIAGE RETURN?
51 006272   001430                BEQ     7$           :YES-SKIP TO NEXT ENTRY
52 006274   022737   001000      CMP     #1000,$TMP1  :VECTOR ADDRESS < 1000 ?
53 006302   101003                BHI   6$           :YES!!
54 006304   104401   063176      TYPE     ,CNSLO4     :TYPE WARNING MESSAGE
55 006310   000755                BR      5$           :RETRY
56 006312   113700   001272      6$:     MOVB   $VECT1,R0     :GET CURRENT VECTOR ADDRESS
    
```

```

57 006316 042700 177400      BIC      #^C<377>,R0      ;CLEAR SIGN EXTENSION
58 006322 005720              TST      (R0)+           ;INCREMENT R0 BY 2
59 006324 010060 177776      MOV      R0,-2(R0)       ;SETUP TRAP CATCHER ADDRESS AND
60 006330 005010              CLR      (R0)           ;PUT HALT IN TRAP ADDRESS +2
61 006332 013700 001176      MOV      STMP1,R0        ;GET NEW VECTOR ADDRESS
62 006336 012720 054702      MOV      #IRP,(R0)+      ;GET ADDRESS FOR INTERRUPT VECTOR
63 006342 113710 001273      MOVSB   SVECT1+1,(R0)    ;GET PRIORITY LEVEL
64 006346 113737 001176 001272  MOVSB   STMP1,SVECT1     ;STORE NEW VECTOR ADDRESS
65
66                               ;DIALOGUE TO INPUT DEVICE NUMBERS
67 006354 005227 177777      7$: INC      #-1         ;FIRST TIME THRU ?
68 006360 001002              B'IE    BS             ;BR IF NO
69 006362 104401 063232      TYPE    .CMSL07        ;TYPE INPUT INSTRUCTIONS
70 006366 104401 001217      8$: TYPE    .SCLF       ;CR-LF
71 006372 005037 001300      9$: CLR      $DEVN      ;CLEAR DEVICE MAP
72 006376 104401 063452      TYPE    .MSDRVS        ;TYPE 'DRIVE(S): '
73 006402 104411              RDCHR
74 006404 012637 001176      MOV      (SP)+,STMP1     ;GET RESPONSE
75 006410 023727 001176 000101  CMP      STMP1,#'A       ;IS INPUT 'A' ?
76 006416 001007              BNE     10$            ;NO
77 006420 104401 063050      TYPE    .ALL           ;YES, TYPE 'ALL' AND GO
78 006424 012737 000377 001300  MOV      #377,$DEVN      ;SET DEVICE MAP FOR ALL DRIVES
79 006432 000137 005604      JMP      XSIZ           ;AUTO SIZE
80
81 006436 023727 001176 000015 10$: CMP      STMP1,#CR     ;CARRIAGE RETURN ?
82 006444 001436              BEQ     12$            ;YES
83 006446 104401 001176      TYPE    .STMP1         ;ECHO RESPONSE
84 006452 023727 001176 000060  CMP      STMP1,#'0       ;NUMBER < 0 ?
85 006460 002430              BLT     12$            ;YES
86 006462 023727 001176 000067  CMP      STMP1,#'7       ;NUMBER > 7 ?
87 006470 003427              BLE     13$            ;NO
88 006472 000423              BR      12$            ;ILLEGAL INPUT
89
90 006474 104411              11$: RDCHR
91 006476 012637 001176      MOV      (SP)+,STMP1     ;GET RESPONSE
92 006502 023727 001176 000015  CMP      STMP1,#CR     ;CARRIAGE RETURN ?
93 006510 001432              BEQ     14$            ;YES
94 006512 104401 063061      TYPE    .COMMA         ;TYPE ','
95 006516 104401 001176      TYPE    .STMP1         ;ECHO RESPONSE
96 006522 023727 001176 000060  CMP      STMP1,#'0       ;NUMBER < 0 ?
97 006530 002404              BLT     12$            ;YES
98 006532 023727 001176 000067  CMP      STMP1,#'7       ;NUMBER > 7 ?
99 006540 003403              BLE     13$            ;NO
100 006542 104401 063374      12$: TYPE    .CNSL08     ;TYPE '' ?ILLEGAL INPUT''
101 006546 000711              BR      9$             ;RETRY
102
103 006550 013701 001176      13$: MOV      STMP1,R1     ;R1 = DRIVE NUMBER
104 006554 042701 177770      BIC      #^C7,R1
105 006560 156137 063740 001300  BISH    ATNBL(R1),$DEVN ;SET DEVICE IN MAP
106 006566 122737 000377 001300  CMPB    #377,$DEVN      ;DONE ?
107 006574 101337              BHI     11$            ;NO
108 006576 005237 001326      14$: INC      AUTSIZ     ;DO NOT AUTO SIZE WHEN TYPING DRIVE STATUS
109 006602 104401 001217      TYPE    .SCLF         ;CR-LF
110 006606 000137 005604      JMP      XSIZ           ;GO SIZE DEVICES

```

ASSEMBLE 'ES' QUE FROM DEVICE MAP

```

MONITARY
2 006612 104401 063416
3 006616 013700 001300
4 006622 001004
5 006624 104401 063061
6 006630 104401 063445
7 006634 012701 001470
8 006640 010137 001466
9 006644 012702 000001
10 006650 005003
11 006652 030200
12 006654 001413
13 006656 104401 063061
14 006662 010311
15 006664 010340
16 006666 104403
    006670 001
    006671 000
17 006672 116361 063740 000001
18 006700 062701 000002
19 006704 006302
20 006706 105702
21 006710 001402
22 006712 005203
23 006714 000756
24 006716 005011
25 006720 104401 001217
26
27
28 006724 004737 054322
29 006730 000425
30 006732 104401 006740
    006736 000413
    006766
31 006766 005737 000042
32 006772 001002
33 006774 000137 004652
34 007000 000137 054272
35 007004 000413
36
37 007006 000240
38 007010 105737 001300
39 007014 001007
40 007016 005737 000042
41 007022 001002
42 007024 000137 004652
43 007030 000137 054272
44 007034
45
46 007034 105037 001116
47 007040 005037 001206
48 007044 004737 060730
49 007050 012746 000000
    007051 012746 007062
    007061 000002

```

```

TYPE ,DRIVES TYPE 'DRIVE S... BE TESTED'
MOV $DEVN,R0 R0 = DEVICE MAP
BNE 18 BR IF DRIVES TO 'ES'
TYPE ,COMMA TYPE ' '
TYPE ,NONE TYPE 'NONE'
MOV #TSTQUE+2,R1 R1 = ADDRESS OF FIRST ENTRY IN QUE
MOV R1,TSTQUE INITIALIZE ENTRY POINTER
MOV #1,R2 R2 = DEVICE POINTER
CLR R3 R3 = DEVICE NUMBER
BIT R2,R0 ;S THIS DEVICE IN MAP ?
BEQ 38 NO
TYPE ,COMMA TYPE ' '
MOV R3,(R1) YES - ENTER DEVICE NUMBER IN QUE
MOV R3,-(SP) ;SAVE R3 FOR TYPEOUT
TYPPOS ;GO TYPE--OCTAL ASCII
.BYTE 1 ;TYPE 1 DIGIT(S)
.BYTE 0 ;SUPPRESS LEADING ZEROS
MOVB ATNTBL(R3),1(R1) ENTER ATTENTION BIT IN QUE
ADD #2 R1 ;ADVANCE ENTRY POINTER
ASL R2 ;ADVANCE DEVICE POINTER
TSTB R2 ;DONE ALL DEVICES ?
BEQ 48 ;YES
INC R3 ;ADVANCE DEVICE NUMBER
BR ;ENTER NEXT DEVICE
48: CLR (R1) ;TERMINATE TEST QUE
TYPE ,$CRLF ;TYPE CR-LF

;SIZE FOR CLOCK
JSR PC,SIZCLK ;SEE IF CLOCK PRESENT
BR 68 ;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 2#42 ;ANY MONITOR PRESENT ?
BNE 58 ;BR IF YES
JMP START ;JUMP TO START
58: JMP $GET42 ;RETURN CONTROL TO MONITOR
68: BR READY1

READY: NOP ;READY TO START TEST
TSTB $DEVN ;ANY DRIVES IN MAP ?
BNE 28 ;BR IF YES
TST 2#42 ;ANY MONITOR PRESENT ?
BNE 18 ;BR IF YES
JMP START ;JUMP TO START
18: JMP $GET42 ;RETURN CONTROL TO MONITOR
28:

READY1 CLR# $TSTNM ;RESET TEST NUMBER
CLR $TIMES ;INITIALIZE NUMBER OF ITERATIONS
ISR PC,$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

```

```

007062
50 007062 117737 172400 001234 64$: MOV B @TSTQUE,$UNIT ;LOAD DRIVE NUMBER
51 ;CLEAR MASSBUS CONTROLLER, SELECT DRIVE AND SET LAST TRACK ADDRESS
52 MOV $BASE,RO ;RO = UNIBUS ADDRESS
53 007070 013700 001276 MOV #CLR,RMCS2(RO) ;CLEAR MASSBUS
54 007074 012760 000040 000010 MOV @TSTQUE,RMCS2(RO) ;SELECT DEVICE UNDER TEST
55 007102 117760 172360 000010 MOV B @TAB:TA4:TA1,LSTRK ;SET LAST TRACK = 13.
56 007110 012737 006400 001334 MOV #TAB:TA4:TA1,LSTRK ;SET LAST TRACK = 13.
57 ;TYPE DRIVE NUMBER TO BE TESTED($UNIT)
58 TYPE $CRLF ;CR-LF
59 007116 104401 001217 TSTB $DEVN ;ANY DRIVES IN MAP ?
60 007122 105737 001300 BEQ 1$ ;BR IF NO
61 007126 001406 TYPE MSGDRV ;TYPE 'DRIVE'
62 007130 104401 063466 MOV $UNIT,-(SP) ;;SAVE $UNIT FOR TYPEOUT
63 007134 013746 001234 ;;TYPE DRIVE NUMBER
007140 104403 TYPUS ;;GO TYPE--OCTAL ASCII
007142 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
007143 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
64 007144 005004 1$: CLR R4 ;THESE TWO LOOPS ARE ADDED TO
65 007146 005304 DEC R4 ;WAIT FOR TTY
66 007150 001376 BNE .-2
67 007152 005304 DEC R4
68 007154 001376 BNE .-2
  
```

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

.SBTTL REGISTER AND STORAGE USAGE

:REGISTER ASSIGNMENTS

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

:STORAGE ASSIGNMENTS

:\$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  
: \$STSN - TEST NUMBER  
: \$UNIT = NUMBER OF DEVICE BEING TESTED  
: \$RGINBF = THE REGISTER INPUT BUFFER HAS A STORAGE LOCATION FOR  
: EACH REGISTER, AND IS USED WHEN READING STATUS AND  
: CONTROL DATA  
: \$RGOTBF = THE REGISTER OUTPUT BUFFER HAS A STORAGE LOCATION FOR  
: EACH REGISTER, AND IS USED FOR ASSEMBLING DATA TO BE  
: WRITTEN IN REGISTERS

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

\*\*\*\*\*  
 \*TEST 1 TRANSFER TEST  
 \*\*\*\*\*

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' ERROP -

```

48 ;ADVANCE THE REGISTER INDEX AND REPEAT THE TEST FOR THE NEXT
49 ;AVAILABLE DEVICE REGISTER
50 40$:
51 007402 062702 000002 ADD #2,R2 ;ADVANCE TO NEXT REGISTER
52 007406 022702 000002 CMP #RMWC,R2 ;IS THIS RMWC??
53 007412 001773 BEQ 40$ ;YES - TRY NEXT REGISTER
54 007414 022702 000004 CMP #RMBA,R2 ;IS THIS RMBA??
55 007420 001770 BEQ 40$ ;YES - TRY NEXT REGISTER
56 007422 022702 000010 CMP #RMCS2,R2 ;IS THIS RMCS2??
57 007426 001765 BEQ 40$ ;YES - TRY ANOTHER REGISTER
58 007430 022702 000016 CMP #RMAS,R2 ;IS THIS RMAS ??
59 007434 001762 BEQ 40$ ;YES - TRY ANOTHER REGISTER
60 007436 022702 000022 CMP #RMD8,R2 ;IS THIS RMD8??
61 007442 001757 BEQ 40$ ;YES - TRY ANOTHER REGISTER
62 007444 022702 000046 CMP #RMEC2,R2 ;IS THIS A LEGAL REGISTER
63 007450 103257 BHIS 10$ ;YES - TRY THIS REGISTER
64
65 ;GOT 'NONEXISTENT DEVICE' ERROR FOR EVERY REMOTE REGISTER ADDRESS
66 50$:
67 007452 013737 001276 001136 MOV $BASE,$BDADR ;STORE BASE ADDRESS
68 007460 104002 EMT 2
69 007462 000137 054046 60$: JMP $EOSP ;GO SELECT NEXT DEVICE
70
71 70$:
72
73
;*****
;*TEST 2 CTOD TEST
;*****
TST2:
007466 SCOPE ;SCOPE CALL
007466 000004 NOP
007470 000240 MOV #STACK,SP ;LOAD THE STACK POINTER
007472 012706 001100 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
007476 013700 001276 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
007502 013701 001466 MOV #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
007506 012737 000002 001226 JSR #C,CNTCLR ;GO CLEAR CONTROLLER
74
75 007514 004737 055156 ;WRITE ONES IN REMOTE REGISTERS
76
77
78 007520 012760 000076 000000 MOV #1LF76,RMCS1(R0) ;LOAD RMCS1
79 007526 012760 177777 000006 MOV #-1,RMDA(R0) ;LOAD RMDA
80 007534 012760 001777 000034 MOV #CYLSK,RMDC(R0) ;LOAD RMDC
81 007542 012760 017200 000032 MOV #CXNUOF,RMOF(R0) ;LOAD RMOF
007550 012760 017200 000032 MOV #CXNUOF,RMOF(R0) ;LOAD RMOF AGAIN TO SET SSEI
82
83 ;READ REMOTE REGISTERS TWICE
84 007556 012702 000001 10$: MOV #1,R2
85 007562
007562 016037 000000 001336 MOV RMCS1(R0),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
86 007570 016037 000006 001344 MOV RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
87 007576 016037 000034 001372 MOV RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
88 007604 016037 000032 001370 MOV RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
89 007612 005302 DEC R2
90 007614 100767 BPL 10$
91
92 ;SEE IF ANY ONE BITS CAME BACK

```



93	007616	042737	177701	001336
94	007624	001014		
95	007626	005737	001344	
96	007632	001011		
97	007634	042737	176000	001372
98	007642	001005		
99	007644	042737	160577	001370
100	007652	001001		

```

BIC #*CILF76,RMCS11 :IS RMCS1 0??
BNE 20$ :NO!!
TST RMDA1 :IS RMDA 0??
BNE 20$ :NO!!
BIC #*XNUDC,RMDC1 :IS RMDC 0??
BNE 20$ :NO!!
BIC #*XNUOF,RMOF1 :IS RMOF 0 ??
BNE 20$ :NO!!

```

101				
102				
103	007654	104003		
104	007656			
105				
106				

```

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

```

```

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

```

107	007656	000004		
108	007660	000240		
109	007662	012706	001100	
110	007666	013700	001276	
111	007672	013701	001466	
112	007676	012737	000003	001226

```

TST3:
SCOPE :SCOPE CALL
NOP
MOV #STACK,SP :LOAD THE STACK POINTER
MOV $BASE,R0 :R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 :R1 = POINTER TO DEVICE
MOV #3,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

```

108	007704	004737	055156	
109				
110				
111	007710	012760	000076	000000
112	007716	012760	177777	000014
113	007724	012760	177777	000042

```

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

```

114				
115				
116	007732	004737	055156	
117				
118				
119	007736	016037	000000	001336
120	007744	016037	000014	001352
121	007752	016037	000042	001400

```

;INITIALIZE MASSBUS WITH A CLEAR
JSR PC,CNTCLR :GO CLEAR CONTROLLER
;READ THE REGISTERS THAT WERE WRITTEN
MOV RMCS1(RO),RMCS11 :STORE RMCS1 IN INPUT BUFFER
MOV RMER1(RO),RMER11 :STORE RMER1 IN INPUT BUFFER
MOV RMER2(RO),RMER21 :STORE RMER2 IN INPUT BUFFER

```

122				
123				
124	007760	052737	177701	001336
125	007766	052737	001527	001400
126	007774	022737	177777	001336
127	010002	001011		
128	010004	022737	177777	001352
129	010012	001005		
130	010014	022737	177777	001400
131	010022	001001		

```

;SEE IF ANY REGISTER BITS WERE CLEARED
BIS #*CILF76,RMCS11 :SET ANY BIT NOT WRITTEN
BIS #*XNUER2,RMER21
CMP #-1,RMCS11 :ANY ZEROS IN RMCS1??
BNE 10$ :YES!!
CMP #-1,RMER11 :ANY ZEROS IN RMER1??
BNE 10$ :YES!!
CMP #-1,RMER21 :ANY ZEROS IN RMER2??
BNE 10$

```

132				
133				
134	010024	104004		
135	010026			
136				
137				

```

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

```

```

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

```

```

*****
TST4:
010026          010026 000004          SCOPE          ;SCOPE CALL
010030          010030 000240          NOP
010032          012706 001100          MOV #STACK,SP  ;LOAD THE STACK POINTER
010036          013700 001276          MOV SBASE,RO   ;RO = UNIBUS ADDRESS
010042          013701 001466          MOV TSTQUE,R1  ;R1 = POINTER TO DEVICE
010046          012737 000004 001226  MOV #4,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
138
139 010054 004737 055156          JSR PC,CNTCLR  ;GO CLEAR CONTROLLER
140
141          ;WRITE ONES IN TEST REGISTERS
142 010060 012760 177777 000014  MOV #1,RMER1(RO) ;LOAD RMER1
143 010066 012760 177777 000042  MOV #1,RMER2(RO) ;LOAD RMER2
144 010074 012760 000001 000024  MOV #DMD,RMR1(RO) ;LOAD RMR1
145
146          ;READ TEST REGISTERS AND SEE IF ANY BITS ARE ON
147 010102 016037 000014 001352  MOV RMR1(RO),RMR1I ;STORE RMR1 IN INPUT BUFFER
148 010110 016037 000042 001400  MOV RMR2(RO),RMR2I ;STORE RMR2 IN INPUT BUFFER
149 010116 016037 000024 001362  MOV RMR1(RO),RMR1I ;STORE RMR1 IN INPUT BUFFER
150 010124 042737 040000 001352  BIC #UNS,RMR1I   ;DONT ACCEPT UNSAFE
151 010132 001011          BNE 10$        ;BRANCH IF ANY OTHER BITS ON
152 010134 042737 040200 001400  BIC #SKI!DVC,RMR2I ;DONT ACCEPT SKI OR DVC
153 010142 001005          BNE 10$        ;BRANCH IF ANY OTHER BITS ON
154 010144 032737 000001 001362  BIT #DMD,RMR1I   ;BRANCH IF DMD IS ON
155 010152 001001          BNE 10$
156 010154 104026          EMT 26
157 010156
158
159
*****
*TEST 5          TRISTATE TRANSFER TEST
*****
TST5:
010156          010156 000004          SCOPE          ;SCOPE CALL
010160          010160 000240          NOP
010162          012706 001100          MOV #STACK,SP  ;LOAD THE STACK POINTER
010166          013700 001276          MOV SBASE,RO   ;RO = UNIBUS ADDRESS
010172          013701 001466          MOV TSTQUE,R1  ;R1 = POINTER TO DEVICE
010176          012737 000005 001226  MOV #5,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
160
161 010204 005002          CLR R2         ;CLEAR ERROR FLAGS
162 010206 004737 055156          JSR PC,CNTCLR  ;GO CLEAR CONTROLLER
163
164          ;WRITE ONES IN SELECTED REGISTERS
165 010212 012760 000076 000000  MOV #1LF76,RMCS1(RO) ;LOAD RMCS1
166 010220 012760 177777 000006  MOV #1,RMDA(RO)   ;LOAD RMDA
167 010226 012760 177777 000014  MOV #1,RMER1(RO)  ;LOAD RMER1
168 010234 012760 177777 000032  MOV #1,RMOF(RO)   ;LOAD RMOF
010242 012760 177777 000032  MOV #1,RMOF(RO)   ;LOAD RMOF AGAIN TO SET SSEI
169 010250 012760 177777 000042  MOV #1,RMER2(RO)  ;LOAD RMER2
170
171          ;WRITE ZEROS IN SELECTED REGISTERS
172 010256 012760 000000 000000  MOV #0,RMCS1(RO)  ;LOAD RMCS1
173 010264 012760 000000 000006  MOV #0,RMDA(RO)   ;LOAD RMDA
174 010272 012760 000000 000014  MOV #0,RMER1(RO)  ;LOAD RMER1
175 010300 012760 000000 000032  MOV #0,RMOF(RO)   ;LOAD RMOF
  
```

```

176 010306 012760 000000 000034      MOV      #0,RMDC(RO)      ;LOAD RMDC
177 010314 012760 000000 000042      MOV      #0,RMER2(RO)    ;LOAD RMER2
178
179                                     ;READ BACK ALL REGISTERS
180 010322 016037 000000 001336      MOV      RMCS1(RO),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
181 010330 016037 000006 001344      MOV      RMDA(RO),RMDAI  ;STORE RMDA IN INPUT BUFFER
182 010336 016037 000014 001352      MOV      RMER1(RO),RMER1I ;STORE RMER1 IN INPUT BUFFER
183 010344 016037 000032 001370      MOV      RMOF(RO),RMOFI  ;STORE RMOF IN INPUT BUFFER
184 010352 016037 000034 001372      MOV      RMDC(RO),RMDCI  ;STORE RMDC IN INPUT BUFFER
185 010360 016037 000042 001400      MOV      RMER2(RO),RMER2I ;STORE RMER2 IN INPUT BUFFER
186
187                                     ;CHECK EACH REGISTER CONTENT FOR ZERO BITS WRITTEN & READ
188 010366 012702 177777              MOV      #-1,R2          ;ACCUMULATE ZEROS IN R2
189 010372 052737 177701 001336      BIS      #^C1LF76,RMCS1I ;SET ALL BITS NOT WRITTEN
190 010400 052737 160577 001370      BIS      #XNUOF,RMOFI
191 010406 052737 176000 001372      BIS      #XNUDC,RMDCI
192 010414 052737 001527 001400      BIS      #XNUER2,RMER2I
193 010422 005137 001336              COM      RMCS1I          ;COMPLEMENT REGISTER CONTENTS
194 010426 005137 001344              COM      RMDAI
195 010432 005137 001352              COM      RMER1I
196 010436 005137 001370              COM      RMOFI
197 010442 005137 001372              COM      RMDCI
198 010446 005137 001400              COM      RMER2I
199 010452 043702 001336              BIC      RMCS1I,R2      ;ACCUMULATE ALL ZERO BITS
200 010456 043702 001344              BIC      RMDAI,R2
201 010462 043702 001352              BIC      RMER1I,R2
202 010466 043702 001370              BIC      RMOFI,R2
203 010472 043702 001372              BIC      RMDCI,R2
204 010476 043702 001400              BIC      RMER2I,R2
205 010502 001407                      BEQ      10$            ;BRANCH IF EACH BIT IS ZERO
206
207                                     ;ONE OR MORE BIT POSITIONS ARE NOT ZERO
208 010504 010237 001142              MOV      R2,$BDDAT      ;SAVE RESULT FOR TYPE
209 010510 005037 001140              CLR      $GDDAT         ;LOAD EXPECTED RESULT
210 010514 104005                      EMT      5
211 010516 052702 000001              BIS      #BIT0,R2      ;SET ERROR FLAG
212
213 010522 004737 055156              10$:    JSR      PC,ENTCLM   ;GO CLEAR CONTROLLER
214
215                                     ;PRESET SELECTED REGISTERS TO ZEROS
216                                     ;(ASSUME RMCS1, RMER1, RMER2 WERE CLEARED BY INIT)
217 010526 012760 000000 000006      MOV      #0,RMDA(RO)    ;LOAD RMDA
218 010534 012760 000000 000032      MOV      #0,RMOF(RO)    ;LOAD RMOF
219 010542 012760 000000 000034      MOV      #0,RMDC(RO)    ;LOAD RMDC
220
221                                     ;WRITE ONES IN SELECTED REGISTERS
222 010550 012760 000076 000000      MOV      #1LF76,RMCS1(RO) ;LOAD RMCS1
223 010556 012760 177777 000006      MOV      #-1,RMDA(RO)   ;LOAD RMDA
224 010564 012760 017200 000032      MOV      #^CXNUOF,RMOF(RO) ;LOAD RMOF
225 010572 012760 017200 000032      MOV      #^CXNUOF,RMOF(RO) ;LOAD RMOF AGAIN TO SET SSEI
226 010600 012760 001777 000034      MOV      #^CXNUDC,RMDC(RO) ;LOAD RMDC
227 010614 012760 176250 000042      MOV      #-1,RMER1(RO)  ;LOAD RMER1
228                                     ;LOAD RMER2
229                                     ;LOAD RMER2
230 010622 016037 000000 001336      MOV      RMCS1(RO),RMCS1I ;STORE RMCS1 IN INPUT BUFFER
    
```

```

231 010630 016037 000006 001344      MOV      RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
232 010636 016037 000032 001370      MOV      RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
233 010644 016037 000034 001372      MOV      RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
234 010652 016037 000014 001352      MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
235 010660 016037 000042 001400      MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
236
237                                     ;CHECK EACH REGISTER CONTENT FOR ONE BITS WRITTEN & READ
238 J10666 042737 177701 001336      BIC      #*CILF76,RMCS1I  ;CLEAR ALL BITS NOT WRITTEN
239 010674 042737 160577 001370      BIC      #XNUOF,RMOFI
240 010702 042737 176000 001372      BIC      #XNUDC,RMDCI
241 010710 042737 001527 001400      BIC      #XNUER2,RMER2I
242 010716 005002                                     CLR      R2                ;ACCUMULATE ONES IN R2
243 010720 053702 001336      BIS      RMCS1I,R2        ;ACCUMULATE ALL ONE BITS
244 010724 053702 001344      BIS      RMDAI,R2
245 010730 053702 001370      BIS      RMOFI,R2
246 010734 053702 001372      BIS      RMDCI,R2
247 010740 053702 001352      BIS      RMER1I,R2
248 010744 053702 001400      BIS      RMER2I,R2
249 010750 022702 177777      CMP      #-1,R2          ;SEE IF EACH BIT POSITION WAS ONE
250 010754 001410      BEQ     20$              ;BRANCH IF NONE STUCK
251
252                                     ;ONE OR MORE BIT POSITIONS ARE NOT ONE
253 010756 010237 001142                                     MOV      R2,$DDDAT        ;SAVE RESULT FOR TYPE
254 010762 012737 177777 001140      MOV      #-1,$GDDAT      ;EXPECTED RESULT
255 010770 104006                                     EMT     6
256 010772 052702 000002      BIS      #BIT1,R2        ;SET ERROR FLAG
257 010776                                     20$:
258 010776 005702                                     TST     R2                ;ANY ERRORS DETECTED ??
259 J11000 001130                                     BNE     30$              ;YES - DONT DO BIT TEST
260 011002 012702 000001      MOV      #1,R2          ;R2=BIT POSITION
261
262 011006                                     25$:
263 011006 004737 055156      JSR     PC,CNTCLR       ;GO CLEAR CONTROLLER
264
265                                     ;WRITE THE BIT PATTERN IN SELECTED DEVICE REGISTERS
266 011012 010260 000006      MOV      R2,RMDA(R0)    ;LOAD RMDA
267 011016 010260 000032      MOV      R2,RMOF(R0)    ;LOAD RMOF
268 011022 010260 000032      MOV      R2,RMOF(R0)    ;LOAD RMOF AGAIN TO SET SSEI
269 011026 010260 000034      MOV      R2,RMDC(R0)    ;LOAD RMDC
270 011032 010260 000014      MOV      R2,RMER1(R0)   ;LOAD RMER1
271 011036 010260 000042      MOV      R2,RMER2(R0)   ;LOAD RMER2
272
273                                     ;READ BACK THE REGISTERS
274 011042 016037 000006 001344      MOV      RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
275 011050 016037 000032 001370      MOV      RMOF(R0),RMOFI ;STORE RMOF IN INPUT BUFFER
276 011056 016037 000034 001372      MOV      RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
277 011064 016037 000014 001352      MOV      RMER1(R0),RMER1I ;STORE RMER1 IN INPUT BUFFER
278 011072 016037 000042 001400      MOV      RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
279
280                                     ;CHECK REGISTER CONTENTS FOR CORRECT PATTERN
281 011100 005003                                     CLR      R3                ;R3=ACCUMULATED ONE BIT
282 011102 012704 177777      MOV      #-1,R4        ;R4=ACCUMULATED ZERO BITS
283 011106 013705 001344      MOV      RMDAI,R5      ;GET ANY GOOD BITS FROM RMDA
284 011112 050503      BIS      R5,R3
285 011114 005105      COM     R5
286 011116 040504      BIC     R5,R4
287 011120 013705 001370      MOV      RMOFI,R5      ;GET GOOD BITS FROM RMOF
    
```

```

286 011124 042705 160577      BIC    #XNUOF,R5
287 011130 050503              BIS    R5,R3
288 011132 005105              COM    R5
289 011134 042705 160577      BIC    #XNUOF,R5
290 011140 040504              BIC    R5,R4
291 011142 013705 001372      MOV    RMDC!,R5          ;GET GOOD BITS FROM RMDC
292 011146 042705 176000      BIC    #XNUDC,R5
293 011152 050503              BIS    R5,R3
294 011154 005105              COM    R5
295 011156 042705 176000      BIC    #XNUDC,R5
296 011162 040504              BIC    R5,R4
297 011164 013705 001352      MOV    RMER1!,R5        ;GET GOOD BITS FROM RMER1
298 011170 050503              BIS    R5,R3
299 011172 005105              COM    R5
300 011174 040504              BIC    R5,R4
301 011176 013705 001400      MOV    RMER2!,R5        ;GET GOOD BITS FROM RMER2
302 011202 042705 001527      BIC    #XNUMER2,R5
303 011206 050503              BIS    R5,R3
304 011210 005105              COM    R5
305 011212 042705 001527      BIC    #XNUMER2,R5
306 011216 040504              BIC    R5,R4
307 011220 010205              MOV    R2,R5            ;RESET ALL ONES IN R3 EXCEPT
308 011222 005105              COM    R5                ;FOR THE TEST BIT
309 011224 040503              BIC    R5,R3
310 011226 040204              BIC    R2,R4            ;RESET TEST BIT IN R4
311 011230 050403              BIS    R4,R3            ;COMBINE ACCUMULATED 1'S + 0'S
312 011232 020302              CMP    R3,R2            ;IS PATTERN OK??
313 011234 001406              BEQ    26$              ;YES!!
314 011236 010237 001140      MOV    R2,$GGDAT        ;SAVE TEST PATTERN
315 011242 010337 001142      MOV    R3,$BDDAT        ;SAVE RESULT
316 011246 104007              EMT    7
317 011250 000404              BR     30$              ;SKIP TO NEXT
318
319
320 011252
321 011252 006302
322 011254 001402
323 011256 000137 011006
324 011262
325
331
332
    ;ADVANCE R2 TO THE NEXT PATTERN AND REPEAT TEST
26$:
    ASL    R2              ;SHIFT THE BIT
    BEQ    30$             ;EXIT IF DONE
30$:
    ;*****
    ;*TEST 6      REGISTER SELECT TEST
    ;*
    ;*NOTE: REGISTER SELECT 16 IS TESTED BY THE "ILR" TEST
    ;*
    ;*****
TST6:
    SCOPE                                ;SCOPE CALL
    NOP
    MOV    #STACK,SP                    ;LOAD THE STACK POINTER
    MOV    $BASE,R0                      ;R0 = UNIBUS ADDRESS
    MOV    TSTQUE,R1                     ;R1 = POINTER TO DEVICE
    MOV    #6,$TESTN                      ;:SET TEST NUMBER IN APT MAIL BOX
    
```

333  
 334 :THE FOLLOWING TABLE GIVES MASSBUS REGISTER SELECT VALUES FOR  
 335 :EACH DEVICE REGISTER

ADDRESS	REGISTER NAME	REG SEL (16,8,4,2,1)
336		
337		
338		
339		
340	RMCS1	00000
341	RMDS	00001
342	RMER1	00010
343	RMMA1	00011
344	RMA5	00100
345	RMDA	00101
346	RMDT	00110
347	RMLA	00111
348	RMSN	01000
349	RMOF	01001
350	RMDC	01010
351	RMHR	01011
352	RMMA2	01100
353	RMER2	01101
354	RMEC1	01110
355	RMEC2	01111

: EACH REGISTER SELECT LINE IS TESTED FOR A STUCK AT ONE,  
: STUCK AT ZERO FAULT. AS AN EXAMPLE, TO TEST REG SEL 1.  
: FOR S-A-0, RMER1 IS WRITTEN WITH ZEROS. THEN THE REGISTER  
: THAT HAS THE SAME SELECT VALUE, EXCEPT FOR THE SELECT LINE  
: BEING TESTED, IS WRITTEN WITH ONES. IN THIS EXAMPLE,  
: RMMA1 IS WRITTEN WITH ONES. IF SELECT LINE 1 IS S-A-0,  
: THE ALL ONES WORD WILL BE WRITTEN IN RMER1, AND RMER1  
: WILL NOT BE 0 WHEN READ BACK.

366 011310 005002  
367 011312 012703 177777  
368  
369  
370 011316 004737 055156  
371 011322 010260 000014  
372 011326 010260 000034  
373 011332 010360 000024  
374 011336 010360 000036  
375 011342 016037 000014 001352  
376 011350 016037 000034 001372  
377 011356 020337 001352  
378 011362 001007  
379 011364 052737 176000 001372  
380 011372 020337 001372  
381 011376 001001  
382 011400 104010  
383  
384

```
CLR R2 :R2= ZEROS SOURCE  
MOV #-1,R3 :R3= ONES SOURCE  
  
:TEST REG SEL 1 FOR S-A-0  
JSR PC,INTCLR :GO CLEAR CONTROLLER  
MOV R2,RMER1(RO) :LOAD RMER1  
MOV R2,RMDC(RO) :LOAD RMDC  
MOV R3,RMA1(RO) :LOAD RMA1  
MOV R3,RMA5(RO) :LOAD RMA5  
MOV RMER1(RO),RMER1I :STORE RMER1 IN INPUT BUFFER  
MOV RMDC(RO),RMDCI :STORE RMDC IN INPUT BUFFER  
CMP R3,RMER1I  
BNE 10$  
BIS #RMDCI,RMDCI  
CMP R3,RMDCI  
BNE 10$  
EMT 10
```

385 011402  
386 011402 004737 055156  
387 011406 010260 000006  
388 011412 010260 000032  
011416 010260 000032  
389 011422 010260 000042  
390 011426 010360 000016  
391 011432 010360 000030

```
:TEST REG SEL 1 FOR S-A-1  
10$:  
JSR PC,INTCLR :GO CLEAR CONTROLLER  
MOV R2,RMDA(RO) :LOAD RMDA  
MOV R2,RMOF(RO) :LOAD RMOF  
MOV R2,RMOF(RO) :LOAD RMOF AGAIN TO SET SSE1  
MOV R2,RMER2(RO) :LOAD RMER2  
MOV R3,RMA5(RO) :LOAD RMA5  
MOV R3,RMSN(RO) :LOAD RMSN
```

```

392 011436 010360 000040      MOV      R3,RMR2(RO)      ;LOAD RMR2
393 011442 016037 000006 001344  MOV      RMDA(RO),RMDAI   ;STORE RMDA IN INPUT BUFFER
394 011450 016037 000032 001370  MOV      RMOF(RO),RMOFI   ;STORE RMOF IN INPUT BUFFER
395 011456 016037 000042 001400  MOV      RMR2(RO),RMR2I   ;STORE RMR2 IN INPUT BUFFER
396 011464 020337 001344      CMP      R3,RMDAI
397 011470 001015      BNE     20$
398 011472 052737 160577 001370  BIS      #XNUOF,RMOFI
399 011500 020337 001370      CMP      R3,RMOFI
400 011504 001007      BNE     20$
401 011506 052737 001527 001400  BIS      #XNUER2,RMR2I
402 011514 020337 001400      CMP      R3,RMR2I
403 011520 001001      BNE     20$
404 011522 104011      EMT      11
    
```

:TEST REG SEL 2 FOR S-A-0

```

405
406
407 011524      20$:
408 011524 004737 055156      JSR      PC,CNTCLR       ;GO CLEAR CONTROLLER
409 011530 010260 000006      MOV      R2,RMDA(RO)     ;LOAD RMDA
410 011534 010260 000032      MOV      R2,RMOF(RO)     ;LOAD RMOF
      011540 010260 000032      MOV      R2,RMOF(RO)     ;LOAD RMOF AGAIN TO SET SSEI
411 011544 010260 000042      MOV      R2,RMR2(RO)     ;LOAD RMR2
412 011550 010360 000020      MOV      R3,RPLA(RO)     ;LOAD RPLA
413 011554 010360 000036      MOV      R3,RMR(RO)      ;LOAD RMR
414 011560 010360 000046      MOV      R3,RMEC2(RO)    ;LOAD RMEC2
415 011564 016037 000006 001344  MOV      RMDA(RO),RMDAI   ;STORE RMDA IN INPUT BUFFER
416 011572 016037 000032 001370  MOV      RMOF(RO),RMOFI   ;STORE RMOF IN INPUT BUFFER
417 011600 016037 000042 001400  MOV      RMR2(RO),RMR2I   ;STORE RMR2 IN INPUT BUFFER
418 011606 020337 001344      CMP      R3,RMDAI
419 011612 001015      BNE     30$
420 011614 052737 160577 001370  BIS      #XNUOF,RMOFI
421 011622 020337 001370      CMP      R3,RMOFI
422 011626 001007      BNE     30$
423 011630 052737 001527 001400  BIS      #XNUER2,RMR2I
424 011636 020337 001400      CMP      R3,RMR2I
425 011642 001001      BNE     30$
426 011644 104012      EMT      12
    
```

:TEST REG SEL 2 FOR S-A-1

```

427
428
429 011646      30$:
430 011646 004737 055156      JSR      PC,CNTCLR       ;GO CLEAR CONTROLLER
431 011652 010260 000014      MOV      R2,RMR1(RO)     ;LOAD RMR1
432 011656 010260 000034      MOV      R2,RMDC(RO)     ;LOAD RMDC
433 011662 012760 000076 000000  MOV      #ILF76,RMCS1(RO) ;LOAD RMCS1
434 011670 010360 000030      MOV      R3,RMSN(RO)     ;LOAD RMSN
435 011674 016037 000014 001352  MOV      RMR1(RO),RMR1I   ;STORE RMR1 IN INPUT BUFFER
436 011702 016037 000034 001372  MOV      RMDC(RO),RMDCI   ;STORE RMDC IN INPUT BUFFER
437 011710 052737 177701 001352  BIS      #CILF76,RMR1I
438 011716 020337 001352      CMP      R3,RMR1I
439 011722 001007      BNE     40$
440 011724 052737 176000 001372  BIS      #XNUDC,RMDCI
441 011732 020337 001372      CMP      R3,RMDCI
442 011736 001001      BNE     40$
443 011740 104013      EMT      13
    
```

:TEST REG SEL 4 FOR S-A-0

```

444
445
446 011742      40$:
447 011742 004737 055156      JSR      PC,CNTCLR       ;GO CLEAR CONTROLLER
    
```

```

448 011746 010260 000014      MOV      R2,RMER1(RO)      :LOAD RMER1
449 011752 010260 000032      MOV      R2,RMDF(RO)       :LOAD RMDF
      011756 010260 000032      MOV      R2,RMDF(RO)       :LOAD RMDF AGAIN TO SET SSEI
450 011762 010260 000034      MOV      R2,RMDC(RO)       :LOAD RMDC
451 011766 010360 000026      MOV      R3,RMDT(RO)       :LOAD RMDT
452 011772 010360 000042      MOV      R3,RMER2(RO)      :LOAD RMER2
453 011776 010360 000044      MOV      R3,RMEC1(RO)      :LOAD RMEC1
454 012002 016037 000014 001352  MOV      RMER1(RO),RMER1I    :STORE RMER1 IN INPUT BUFFER
455 012010 016037 000032 001370  MOV      RMDF(RO),RMDFI     :STORE RMDF IN INPUT BUFFER
456 012016 016037 000034 001372  MOV      RMDC(RO),RMDCI     :STORE RMDC IN INPUT BUFFER
457 012024 020337 001352      CMP      R3,RMER1I
458 012030 001015      BNE     50$
459 012032 052737 160577 001370  BIS     #XNUCF,RMDFI
460 012040 020337 001370      CMP      R3,RMDFI
461 012044 001007      BNE     50$
462 012046 052737 176000 001372  BIS     #XNUDC,RMDCI
463 012054 020337 001372      CMP      R3,RMDCI
464 012060 001001      BNE     50$
465 012062 104014      EMT     14
    
```

:TEST REG SEL 4 FOR S-A-1

```

466 50$:
467
468 012064 004737 055156      JSR     PC,CNTCLR          :GO CLEAR CONTROLLER
469 012064 010260 000006      MOV      R2,RMDA(RO)       :LOAD RMDA
470 012070 010260 000042      MOV      R2,RMER2(RO)      :LOAD RMER2
471 012074 010360 000012      MOV      R3,RMDS(RO)       :LOAD RMDS
472 012100 010360 000032      MOV      R3,RMDF(RO)       :LOAD RMDF
473 012104 010360 000032      MOV      R3,RMDF(RO)       :LOAD RMDF AGAIN TO SET SSEI
      012110 010360 000032      MOV      R3,RMDF(RO)
474 012114 016037 000006 001344  MOV      RMDA(RO),RMDAI     :STORE RMDA IN INPUT BUFFER
475 012122 016037 000042 001400  MOV      RMER2(RO),RMER2I   :STORE RMER2 IN INPUT BUFFER
476 012130 020337 001344      CMP      R3,RMDAI
477 012134 001007      BNE     60$
478 012136 052737 001527 001400  BIS     #XNUMER2,RMER2I
479 012144 020337 001400      CMP      R3,RMER2I
480 012150 001001      BNE     60$
481 012152 104015      EMT     15
    
```

:TEST REG SEL 8 FOR S-A-0

```

482 60$:
483
484 012154 004737 055156      JSR     PC,CNTCLR          :GO CLEAR CONTROLLER
485 012154 010260 000014      MOV      R2,RMER1(RO)      :LOAD RMER1
486 012160 010260 000006      MOV      R2,RMDA(RO)       :LOAD RMDA
487 012164 010260 000034      MOV      R3,RMDC(RO)       :LOAD RMDC
488 012170 010360 000042      MOV      R3,RMER2(RO)      :LOAD RMER2
489 012174 010360 000014 001352  MOV      RMER1(RO),RMER1I    :STORE RMER1 IN INPUT BUFFER
490 012200 016037 000006 001344  MOV      RMDA(RO),RMDAI     :STORE RMDA IN INPUT BUFFER
491 012206 016037 000006      CMP      R3,RMER1I
492 012214 020337 001352      BNE     70$
493 012220 001004      CMP      R3,RMDAI
494 012222 020337 001344      BNE     70$
495 012226 001001      BNE     70$
496 012230 104016      EMT     16
    
```

:TEST REG SEL 8 FOR S-A-1

```

497 70$:
498
499 012232 004737 055156      JSR     PC,CNTCLR          :GO CLEAR CONTROLLER
500 012232 010260 000032      MOV      R2,RMDF(RO)       :LOAD RMDF
501 012236 010260 000032      MOV      R2,RMDF(RO)       :LOAD RMDF AGAIN TO SET SSEI
      012242 010260 000032
    
```



```

502 012246 010260 00034  MOV R2,RMDC,RO LOAD RMDC
503 012252 010260 00004  MOV R2,RMER2(RO) LOAD RMER2
504 012256 010360 00001  MOV R3,RMDS(RO) LOAD RMDS
505 012262 010360 00001  MOV R3,RMER1(RO) LOAD RMER1
506 012266 010360 000006  MOV R3,RMDA(RO) LOAD RMDA
507 012272 016037 000032 001370  MOV RMDF(RO),RMDFI STORE RMDF IN INPUT BUFFER
508 012280 016037 000034 001372  MOV RMDC(RO),RMDCI STORE RMDC IN INPUT BUFFER
509 012286 016037 000242 001400  MOV RMER2(RO),RMER2I STORE RMER2 IN INPUT BUFFER
510 012294 052737 160577 001370  BIS #XNUOF,RMDFI
511 012302 001015  BNE B0$
512 012310 022737 176000 001372  CMP #XNUDC,RMDCI
513 012318 020337 001372  CMP R3,RMDCI
514 012326 001007  BNE B0$
515 012334 052737 001527 001400  BIS #XNUMER2,RMER2I
516 012342 020337 001400  CMP R3,RMER2I
517 012350 001001  BNE B0$
518 012354 104017  EMT 17
519 012356  B0$
520
521
522

```

```

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

```

```

012356  TST7:
012356 000004  SCOPE :SCOPE CALL
012360 000240  NOP
012362 012706 001100  MOV #STACK,SP :LOAD THE STACK POINTER
012366 013700 001276  MOV SBASE,RO :RO = UNIBUS ADDRESS
012372 013701 001466  MOV TSTQUE,R1 :R1 = POINTER TO DEVICE
012377 012737 000007 001226  MOV #7,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
525
526 012404 016037 000026 001142  MOV RMDT(RO),SBDDAT :STORE RMDT AT SBDDAT
527 012412 022737 020026 001142  CMP #SNGPRT,$BDDAT :SINGLE PORT RM80??
528 012420 001414  BEQ 10$ :YES!!
529 012422 022737 024026 001142  CMP #DULPRT,$BDDAT :DUAL PORT RM80??
530 012430 001410  BEQ 10$ :YES!!
531 012432 010037 001136  MOV RO,$BDDADR :LOAD BAD ADDRESS
532 012436 062737 000026 001136  ADD #RMDT,$BDDADR
533 012444 104057  EMT 57
534
535 012446 000137 054046  JMP $EOSP .GO TO NEXT DEVICE
536 012452 10$:
537
538

```

```

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

```

```

012452  TST10:
012452 000004  SCOPE :SCOPE CALL
012454 000240  NOP
012456 012706 001100  MOV #STACK,SP :LOAD THE STACK POINTER
012462 013700 001276  MOV SBASE,RO :RO = UNIBUS ADDRESS
012466 013701 001466  MOV TSTQUE,R1 :R1 = POINTER TO DEVICE
012472 012737 000010 001226  MOV #10,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
539
540 012500 004737 055156  JSR PC,CNTCLR :GO CLEAR CONTROLLER
541 012504 016037 000000 001142  MOV RMCS1(RO),SBDDAT :STORE RMCS1 AT SBDDAT

```

542 012512 042737 173777 001142  
 543 012520 001000  
 544 012522 012737 004000 001140  
 545 012530 010037 001136  
 546 012534 104060  
 547 012536  
 548  
 549

BIC #^CDVA,\$BDDAT ;CLEAR ALL BUT DVA  
 BNE 10\$ ;BRANCH IF DVA SET  
 MOV #DVA,\$GDDAT ;SETUP EXPECTED  
 MOV R0,\$BDADR ;SETUP REG ADDRESS  
 EMT 60

10\$:

\*\*\*\*\*  
 :\*TEST 11 HOLDING REGISTER TRANSFER TEST  
 \*\*\*\*\*

\*\*\*\*\*  
 TST11:  
 \*\*\*\*\*

012536  
 012536 000004  
 012540 000240  
 012542 012706 001100  
 012546 013700 001276  
 012552 013701 001466  
 012556 012737 000011 001226

SCOPE ;SCOPE CALL  
 NOP  
 MOV #STACK,SP ;LOAD THE STACK POINTER  
 MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS  
 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE  
 MOV #11,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

550  
 551 012564 004737 055156  
 552 012570 005003  
 553 012572 010037 00136  
 554 012576 062737 00036 001136  
 555

JSR PC,CNTCLR ;GO CLEAR CONTROLLER  
 CLR R3 ;CLEAR ERROR FLAGS  
 MOV R0,\$BDADR ;SETUP REGISTER ADDRESS  
 ADD #RMHR,\$BDADR

:WRITE ONES THEN ZEROS IN RMHR AND CHECK FOR S-A-1 BITS.  
 :NOTE THAT IT IS NECESSARY TO WRITE SOME OTHER REGISTER IN  
 :ORDER TO WRITE THE HOLDING REGISTER, AND RMDA IS USED FOR THIS  
 :PURPOSE.

560 012604 012760 177777 000006  
 561 012612 012760 000000 000006  
 562 012620 016037 000036 001142  
 563 012626 005137 001142  
 564 012632 001405  
 565 012634 005037 001140  
 566 012640 104061  
 567 012642 052703 000001  
 568  
 569

MOV #-1,RMDA(R0) ;LOAD RMDA  
 MOV #0,RMDA(R0) ;LOAD RMDA  
 MOV RMHR(R0),\$BDDAT ;STORE RMHR AT \$BDDAT  
 COM \$BDDAT ;ANY ERROR??  
 BEQ 10\$ ;NO!!  
 CLR \$GDDAT ;LOAD EXPECTED  
 EMT 61  
 BIS #BIT0,R3 ;SET ERROR FLAGS

:WRITE ZEROS THEN ONES IN RMHR AND CHECK FOR S-A-0 BITS.  
 10\$:

570 012646  
 571 012646 012760 000000 000006  
 572 012654 012760 177777 000006  
 573 012662 016037 000036 001142  
 574 012670 005137 001142  
 575 012674 012737 177777 001140  
 576 012702 023737 001140 001142  
 577 012710 001403  
 578 012712 104062  
 579 012714 052703 000002  
 580  
 581

MOV #0,RMDA(R0) ;LOAD RMDA  
 MOV #-1,RMDA(R0) ;LOAD RMDA  
 MOV RMHR(R0),\$BDDAT ;STORE RMHR AT \$BDDAT  
 COM \$BDDAT ;RMHR IS COMPLEMENTED WHEN READ  
 MOV #-1,\$GDDAT ;SETUP EXPECTED  
 CMP \$GDDAT,\$BDDAT ;ANY ERROR??  
 BEQ 20\$ ;NO!!  
 EMT 62  
 BIS #BIT1,R3 ;SET ERROR FLAG

:IF NO PREVIOUS ERRORS, WRITE AND READ SHIFTING ONE BIT PATTERN.  
 20\$:

582 012720  
 583 012720 005703  
 584 012722 00025  
 585 012724 012702 000001  
 586 012730  
 587 012730 012760 000000 000006  
 588 012736 010260 000006

TST R3 ;ANY FLAGS SET??  
 BNE 50\$ ;YES!!  
 MOV #1,R2 ;R2=DATA PATTERN

30\$:

MOV #0,RMDA(R0) ;LOAD RMDA  
 MOV R2,RMDA(R0) ;LOAD RMDA

```

589 012742 016037 000036 001142      MOV      RMHR(RO),SBDDAT ;STORE RMHR AT SBDDAT
590 012750 005137 001142      COM      SBDDAT          ;RMHR IS COMPLEMENTED
591 012754 023702 001142      CMP      SBDDAT,R2      ;ANY ERROR??
592 012760 001404      BEQ      40$            ;NO!!
593 012762 010237 001142      MOV      R2,$GDDAT      ;SETUP EXPECTED
594 012766 104063      EMT      63
595 012770 000402      BR       50$            ;DO NOT COLLECT ALL ERRORS
596
597 012772 006302      40$:    ASL      R2          ;SHIFT TO NEXT PATTERN
598 012774 001355      BNE      30$            ;CONTINUE IF NOT DONE
599
600 012776      50$:
601
602

```

\*\*\*\*\*  
 :\*TEST 12 CONTROL STATUS #1 TRANSFER TEST  
 \*\*\*\*\*

```

012776      TST12:
012776 000004      SCOPE          ;SCOPE CALL
013000 000240      NOP
013002 012706 001100      MOV      #STACK,SP     ;LOAD THE STACK POINTER
013006 013700 001276      MOV      $BASE,R0      ;R0 = UNIBUS ADDRESS
013012 013701 001466      MOV      TSTQUE,R1     ;R1 = POINTER TO DEV:CE
013016 012737 000012 001226      MOV      #12,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

```

```

603
604 013024 005003      CLR      R3            ;R3 = ERROR INDICATOR
605 013026 004737 055156      JSR      PC,CNTCLR     ;GO CLEAR CONTROLLER
606
607      ;WRITE ONES IN RMCS1, BITS 01-05, THEN CLEAR. READ AND
608      ;CHECK FOR S-A-1 BITS.
609 013032 012760 000076 000000      MOV      #ILF76,RMCS1(RO) ;LOAD RMCS1
610
611      JSR      PC,CNTCLR     ;GO CLEAR CONTROLLER
612 013040 004737 055156      MOV      RMCS1(RO),SBDDAT ;STORE RMCS1 AT SBDDAT
613 013044 016037 000000 001142      BIC      #*CILF76,$BDDAT
614 013052 042737 177701 001142      BEQ      5$
615 013060 001410      CLR      $GDDAT
616 013062 005037 001140      MOV      RO,$BDADR
617 013066 010037 001136      ADD      #RMCS1,$BDADR
618 013072 062737 000000 001136      EMT      43
619

```

620 ;WRITE ONES IN RMCS1, BITS 01-05, THEN WRITE ZEROS. READ AND CHECK FOR  
 621 ;S-A-1 BITS.  
 622 5\$:

```

623 013102 012760 000076 000000      MOV      #ILF76,RMCS1(RO) ;LOAD RMCS1
624 013110 012760 000000 000000      MOV      #0,RMCS1(RO)    ;LOAD RMCS1
625 013116 016037 000000 001142      MOV      RMCS1(RO),SBDDAT ;STORE RMCS1 AT SBDDAT
626 013124 042737 177701 001142      BIC      #*CILF76,$BDDAT
627 013132 001412      BEQ      10$
628 013134 005037 001140      CLR      $GDDAT
629 013140 010037 001136      MOV      RO,$BDADR
630 013144 062737 000000 001136      ADD      #RMCS1,$BDADR
631 013152 104023      EMT      23
632 013154 052703 000001      BIS      #BIT0,R3        ;SET ERROR FLAG
633

```

634 ;WRITE ZEROS IN RMCS1, THEN ONES, READ AND CHECK S-A-0 BITS.  
 635 10\$:

```

636 013160 012760 000000 000000      MOV      #0,RMCS1(R0)      ;LOAD RMCS1
637 013166 012760 000076 000000      MOV      #ILF76,RMCS1(R0) ;LOAD RMCS1
638 013174 016037 000000 001142      MOV      RMCS1(R0),%BDDAT ;STORE RMCS1 AT %BDDAT
639 013202 042737 177701 001142      BIC      #^CILF76,%BDDAT
640 013210 012737 000076 001140      MOV      #ILF76,%GDDAT
641 013216 023737 001140 001142      CMP      %GDDAT,%BDDAT
642 013224 001410                      BEQ      20$
643 013226 010037 001136 001136      MOV      R0,%BDADR
644 013232 062737 000000 001136      ADD      #RMCS1,%BDADR
645 013240 104024                      EMT      24
646 013242 052703 000002                      BIS      #BIT1,R3          ;SET ERROR FLAG
647
648
649 013246 005703                      ;WRITE A SHIFTING ONE BIT PATTERN IN RMCS1, READ AND CHECK FOR STUCK BITS.
650 013246 001035                      20$:
651 013250 001035                      TST      R3                ;OMIT IF ANY ERRORS
652 013252 012702 000002                      BNE      50$
653 013256 010203                      MOV      #2,R2            ;R2 = TEST PATTERN
654 013256 0427C 177701                      30$:
655 013260 012760 000000 000000      MOV      R2,R3            ;R3 = EXPECTED RESULT, BITS 1-5
656 013264 012760 000000 000000      BIC      #^CILF76,R3
657 013272 010260 000000 000000      MOV      #0,RMCS1(R0)    ;LOAD RMCS1
658 013276 016037 000000 001142      MOV      R2,RMCS1(R0)    ;LOAD RMCS1
659 013304 042737 177701 001142      MOV      RMCS1(R0),%BDDAT ;STORE RMCS1 AT %BDDAT
660 013312 020337 001142                      BIC      #^CILF76,%BDDAT
661 013316 001410                      CMP      R3,%BDDAT
662 013320 010337 001140                      BEQ      40$
663 013324 010037 001136                      MOV      R3,%GDDAT
664 013330 062737 000000 001136      MOV      R0,%BDADR
665 013336 104025                      ADD      #RMCS1,%BDADR
666 013340 006302                      EMT      25
667 013342 001345                      40$:
668 013344                      ASL      R2                ;SHIFT TO NEXT BIT
669                      BNE      30$              ;CONTINUE IF R2 NOT ZERO
670

```

\*\*\*\*\*  
 ;\*TEST 13 ERROR REGISTER #1 TRANSFER TEST  
 \*\*\*\*\*

```

013344
013344 006704                      TST13:
013346 00C '0                          SCOPE                      ;SCOPE CALL
013350 012700 001100                      NOP
013354 013700 001276                      MOV      #STACK,SP        ;LOAD THE STACK POINTER
013360 013701 001466                      MOV      %BASE,R0         ;R0 = UNIBUS ADDRESS
013364 012737 000013 001226      MOV      TSTQUE,R1        ;R1 = POINTER TO DEVICE
671                      MOV      #13,%STESTN      ;;SET TEST NUMBER IN APT MAIL BOX
672 013372 005003                      CLR      R3                ;CLEAR ERROR FLAG
673
674                      ;WRITE ONES IN RMFR1, CLEAR AND CHECK FOR S-A-1 BITS
675 013374 012760 177777 000014      MOV      #-1,RMER1(R0)    ;LOAD RMFR1
676
677 013402 04737 055156                      JSR      PC,CNTCLR        ;GO CLEAR CONTROLLER
678 013406 016037 000014 001352      MOV      RMFR1(R0),RMFR1 ;STOPE RMFR1 IN INPUT BUFFER
679 013414 013737 001352 001142      MOV      RMFR1,%BDDAT
680 013422 042737 177760 001142      BIC      #^C<PAR!RMR!ILF!ILR>,%BDDAT
681 013430 001410                      BEQ      10$
682 013432 005037 001140                      CLR      %GDDAT

```

683	013436	010037	001136		MOV	RO, \$BDADR
684	013442	062737	000014	001136	ADD	#RMER1, \$BDADR
685	013450	104027			EMT	27
686	013452					
687	013452	013737	001352	001142	10\$: MOV	RMER1, \$BDDAT
688	013460	042737	074017	001142	BIC	#^C<DCK!IAE!AOE!HCRC!HCE!ECH!WCF!FER>, \$BDDAT
689	013466	001410			BEQ	20\$
690	013470	005037	001140		CLR	\$GDDAT
691	013474	010037	001136		MOV	RO, \$BDADR
692	013500	062737	000014	001136	ADD	#RMER1, \$BDADR
693	013506	104030			EMT	30
694	013510					
695	013510	013737	001352	001142	20\$: MOV	RMER1, \$BDDAT
696	013516	042737	147777	001142	BIC	#^C<OPI!DTE>, \$BDDAT
697	013524	001410			BEQ	30\$
698	013526	005037	001140		CLR	\$GDDAT
699	013532	010037	001136		MOV	RO, \$BDADR
700	013536	062737	000014	001136	ADD	#RMER1, \$BDADR
701	013544	104031			EMT	31
702						
703						
704	013546					
705	013546	012760	177777	000014	30\$: MOV	#-1, RMER1(RO) ;LOAD RMER1
706	013554	012760	000000	000014	MOV	#0, RMER1(RO) ;LOAD RMER1
707	013562	016037	000014	001352	MOV	RMER1(RO), RMER1 ;STORE RMER1 IN INPUT BUFFER
708	013570	013737	001352	001142	MOV	RMER1, \$BDDAT
709	013576	042737	177770	001142	BIC	#^C<RMR!ILF!ILR>, \$BDDAT
710	013604	001412			BEQ	40\$
711	013606	005037	001140		CLR	\$GDDAT
712	013612	010037	001136		MOV	RO, \$BDADR
713	013616	062737	000014	001136	ADD	#RMER1, \$BDADR
714	013624	104032			EMT	32
715	013626	052703	000001		BIS	#BIT0, R3 ;SET ERROR FLAG
716	013632	013737	001352	001142	40\$: MOV	RMER1, \$BDDAT
717	013640	042737	074017	001142	BIC	#^C<DCK!IAE!AOE!HCRC!HCE!ECH!WCF!FER>, \$BDDAT
718	013646	001412			BEQ	50\$
719	013650	005037	001140		CLR	\$GDDAT
720	013654	010037	001136		MOV	RO, \$BDADR
721	013660	062737	000014	001136	ADD	#RMER1, \$BDADR
722	013666	104033			EMT	33
723	013670	052703	000001		50\$: BIS	#BIT0, R3 ;SET ERROR FLAG
724	013674					
725	013674	013737	001352	001142	MOV	RMER1, \$BDDAT
726	013702	042737	147777	001142	BIC	#^C<OPI!DTE>, \$BDDAT
727	013710	001412			BEQ	60\$
728	013712	005037	001140		CLR	\$GDDAT
729	013716	010037	001136		MOV	RO, \$BDADR
730	013722	062737	000014	001136	ADD	#RMER1, \$BDADR
731	013730	104034			EMT	34
732	013732	052703	030000		BIS	#BIT, R3
733						
734						
735	013736					
736	013736	012760	000000	000014	60\$: MOV	#0, RMER1(RO) ;LOAD RMER1
737	013744	012760	177777	000014	MOV	#-1, RMER1(RO) ;LOAD RMER1
738	013752	016037	000014	001142	MOV	RMER1(RO), \$BDDAT ;STORE RMER1 AT \$BDDAT
739	013760	012737	177777	001140	MOV	#-1, \$GDDAT

WRITE ONES THEN ZEROS IN RMER1, READ AND CHECK FOR S-A-1 BITS

30\$:  
 40\$:  
 50\$:  
 60\$:

WRITE ZEROS THEN ONES IN RMER1, READ AND CHECK FOR S-A-0 BITS

60\$:  
 MOV #0, RMER1(RO) ;LOAD RMER1  
 MOV #-1, RMER1(RO) ;LOAD RMER1  
 MOV RMER1(RO), \$BDDAT ;STORE RMER1 AT \$BDDAT  
 MOV #-1, \$GDDAT

740 013766 023737 001140 001142  
 741 013774 001410  
 742 013776 010037 001136  
 743 014002 062737 000014 001136  
 744 014010 104035  
 745 014012 052703 000002  
 746  
 747  
 748  
 749 014016  
 750 014016 005703  
 751 014020 001042  
 752 014022 012702 000001  
 753 014026  
 754 014026 004737 055156  
 755 014032 010260 000014  
 756 014036 016037 000014 001142  
 757 014044 032702 000010  
 758 014050 001003  
 759 014052 042737 000010 001142  
 760 014060 032702 040000  
 761 014064 001003  
 762 014066 042737 040000 001142  
 763 014074 020237 001142  
 764 014100 001410  
 765 014102 010237 001140  
 766 014106 010037 001136  
 767 014112 062737 000014 001136  
 768 014120 104036  
 769  
 770 014122 006302  
 771 014124 001340  
 772 014126  
 773  
 774

```

;WRITE A SHIFTING 1 BIT IN RMER1 AND CHECK FOR STUCK BITS
;NOTE: DONT TEST UNSAFE OR PARITY
70$:
TST R3 ;SKIP THIS PART IF ANY ERRORS
BNE 120$
MOV #1,R2 ;R2 - TEST PATTERN
80$:
JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV R2,RMER1(RO) ;LOAD RMER1
MOV RMER1(RO),%BDDAT ;STORE RMER1 AT %BDDAT
BIT #PAR,R2 ;DONT TEST PAR = 0
BNE 90$
BIC #PAR,%BDDAT
90$:
BIT #UNS,R2 ;DONT TEST UNS = 0
BNE 100$
BIC #UNS,%BDDAT
100$:
CMP R2,%BDDAT
BEQ 110$
MOV R2,%GDDAT
MOV RO,%BDADR
ADD #RMER1,%BDADR
FMT 36
110$:
ASL R2 ;SHIFT TO NEXT BIT
BNE 80$ ;CONTINUE IF R2 NOT ZERO
120$:

```

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

014126  
 014126 000004  
 014130 000240  
 014132 012706 001100  
 014135 013700 001276  
 014142 013701 001466  
 014145 012737 000014 G01226  
 775  
 776 014154 004737 055156  
 777 014160 012760 177777 000032  
 014166 012760 177777 000032  
 778 014174 016037 000032 001142  
 779 014202 042737 177577 001142  
 780 014210 001011  
 781 014212 012737 000200 001140  
 782 014220 010037 001136  
 783 014224 062737 000032 001136  
 784 014232 104172  
 785 014234

```

TST14:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV %BASE,RO ;RO = UNIBUS ADDRESS
MOV TSTQUE,P ;R1 = POINTER TO DEV. CE
MOV #14,ST% ;SET TEST NUMBER IN APT MAIL BOX
775:
JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #-1,RMOF(RO) ;LOAD RMOF
MOV #-1,RMOF(RO) ;LOAD RMOF AGAIN TO SET SSEI
MOV RMOF(RO),%BDDAT ;STORE RMOF AT %BDDAT
BIC #%CFD,%BDDAT
BNE 10$ ;BRANCH IF OFD IS A ONE
MOV #OFD,%GDDAT ;SETUP ERROR MESSAGE
MOV RO,%BDADR
ADD #RMOF,%BDADR
FMT 172
10$:
;END OF TEST

```



```

831 014472 012737 017200 001140      MOV      #^XNUOF,%GDDAT ;SETUP EXPECTED STATUS
832 014500 023737 001140 001142      CMP      %GDDAT,%SDDAT
833 014506 001403                      BEQ      40$              ;BRANCH IF NO ERROR
834 014510 104055                      EMT
835 014512 052703 000002              BIS      #BIT1,R3
836
837                                     ;IF NO PREVIOUS ERRORS, TEST RMOF WITH SHIFTING ONE BIT PATTERN
838                                     ;IN 18 BIT MODE.
839 014516                                40$:
840 014516 005703                      TST      R3              ;ANY ERROR??
841 014520 001062                      BNE      100$           ;YES!!
842 014522 012702 000001              MOV      #1,R2          ;STARTING DATA PATTERN
843 014526                                50$:
844 014526 012760 000000 000032      MOV      #0,RMOF(R0)    ;LOAD RMOF
845 014534 010260 000032              MOV      R2,RMOF(R0)    ;LOAD RMOF
                                014540 010260 000032      MOV      R2,RMOF(R0)    ;LOAD RMOF AGAIN TO SET SSEI
846 014544 016037 000032 001142      MOV      RMOF(R0),%SDDAT ;STORE RMOF AT %SDDAT
847 014552 010203                      MOV      R2,R3          ;SETUP EXPECTED RESULT
848 014554 010337 001140              MOV      R3,%GDDAT
849 014560 042703 161577              BIC      #XNUOF,%SSEI,R3 ;CLEAR UNUSED BITS
850 014564 020337 001142              CMP      R3,%SDDAT      ;COMPARE EXPECTED & RECEIVED
851 014570 001401                      BEQ      60$              ;BRANCH IF NO ERROR
852 014572 104056                      EMT
853
854 014574 006302                      60$:
855 014576 001353                      ASL      R2              ;SHIFT TO NEXT BIT
                                BNE      50$              ;CONTINUE IF NOT DONE
856
857                                     ;TEST RMOF WITH SHIFTING ONE BIT PATTERN IN 16 BIT MODE
858 014600                                70$:
859 014600 012702 000001              MOV      #1,R2          ;STARTING DATA PATTERN
860 014604                                80$:
861 014604 012760 000000 000032      MOV      #0,RMOF(R0)    ;LOAD RMOF
862 014612 052702 010000              BIS      #FMT16,R2      ;SET 16 BIT MODE WITH ROTATING PATTERN
863 014616 010260 000032              MOV      R2,RMOF(R0)    ;LOAD RMOF
                                014622 010260 000032      MOV      R2,RMOF(R0)    ;LOAD RMOF AGAIN TO SET SSEI
864 014626 016037 000032 001142      MOV      RMOF(R0),%SDDAT ;STORE RMOF AT %SDDAT
865 014634 010203                      MOV      R2,R3          ;SETUP EXPECTED RESULTS
866 014636 010337 001140              MOV      R3,%GDDAT
867 014642 042703 160577              BIC      #XNUOF,R3      ;CLEAR UNUSED BITS
868 014646 020337 001142              CMP      R3,%SDDAT      ;CORRECT BITS SET ?
869 014652 001401                      BEQ      90$              ;YES !!
870 014654 104056                      EMT
871 014656                                90$:
872 014656 042702 010000              BIC      #FMT16,R2      ;DON'T SHIFT FORMAT BIT
873 014662 006302                      ASL      R2              ;SHIFT PATTERN AND
874 014664 001347                      BNE      80$              ;TEST NEXT BIT
875 014666                                100$:
876
877

```

\*\*\*\*\*  
 ;\*TEST 16 ERROR REGISTER #2 TRANSFER TEST  
 \*\*\*\*\*

```

TST16:
014666                                SCOPE                      ;SCOPE CALL
014666 000004                      NOP
014670 000240                      MOV      #STACK,SP      ;LOAD THE STACK POINTER
014672 012706 001100              MOV      %BASE,R0       ;R0 = UNIBUS ADDRESS
014676 013700 001276

```



```

014702 013701 001466      MOV    TSTQUE,R1      ;R1 - PCINICP TO DEVICE
014706 012737 000016 001226  MOV    #16,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
878
879 014714 005003      CLR    R3            ;RESET ERROR FLAGS
880 014716 010037 001136  MOV    R0,$BDADR     ;SETUP BAD ADDRESS
881 014722 062737 000042 001136  ADD    #RMR2,$BDADR
882 014730 005037 001140      CLR    $GDDAT        ;SETUP EXPECTED DATA
883 014734 012737 010000 001444  MOV    #FMT16,RMOFO  ;SET 16 BIT FORMAT MODE TO ALLOW
884                                     ;"SE" TO BE SET IN RMR2
885
886                                     ;WRITE ONES IN RMR2. (CLEAR AND CHECK FOR S-A-1 BITS
887 014742 012760 177777 000042  MOV    #-1,RMR2(RO)  ;LOAD RMR2
888
889 014750 004737 055156      JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
890 014754 013760 001444 000032  MOV    RMOFO,RMOF(RO) ;LOAD RMOF
891 014762 016037 000042 001400  MOV    RMR2(RO),RMR2I ;STORE RMR2 IN INPUT BUFFER
892
893                                     ;TEST UNUSED BITS FOR ZERO-FAILURE ON IF
894 014770 013737 001400 001142  MOV    RMR2I,$BDDAT
895 014776 042737 176250 001142  BIC    #^CXNUER2,$BDDAT
896 015004 001403      BEQ    10$           ;BRANCH IF NO ERROR
897 015006 104044      EMT    44
898 015010 052703 000001      BIS    #BIT0,R3      ;SET ERROR FLAG
899
900                                     ;TEST 'DPE', 'IVC', 'LSC' FOR ZERO-FAILURE ON CS, IF
901 015014      10$:
902 015014 013737 001400 001142  MOV    RMR2I,$BDDAT
903 015022 042737 143777 001142  BIC    #^C<DPE!IVC!LSC>,$BDDAT
904 015030 001403      BEQ    20$           ;BRANCH IF NO ERROR
905 015032 104045      EMT    45
906 015034 052703 000001      BIS    #BIT0,R3      ;SET ERROR FLAG
907
908                                     ;TEST 'LBC', 'DPE' FOR ZERO-FAILURE ON DS, IF
909 015040      20$:
910 015040 013737 001400 001142  MOV    RMR2I,$BDDAT
911 015046 042737 175767 001142  BIC    #^C<LBC!DPE>,$BDDAT
912 015054 001403      BEQ    30$           ;BRANCH IF NO ERROR
913 015056 104046      EMT    46
914 015060 052703 000001      BIS    #BIT0,R3      ;SET ERROR FLAG
915
916                                     ;WRITE ONES IN RMR2 THEN WRITE ZEROS AND CHECK FOR S-A-1 BITS.
917 015064      30$:
918 015064 012760 177777 000042  MOV    #-1,RMR2(RO)  ;LOAD RMR2
919 015072 012760 000000 000042  MOV    #0,RMR2(RO)   ;LOAD RMR2
920 015100 016037 000042 001400  MOV    RMR2(RO),RMR2I ;STORE RMR2 IN INPUT BUFFER
921
922                                     ;TEST 'DPE', 'IVC', 'LSC' FOR ZERO-FAILURE ON CS, IF
923 015106 013737 001400 001142  MOV    RMR2I,$BDDAT
924 015114 042737 143777 001142  BIC    #^C<DPE!IVC!LSC>,$BDDAT
925 015122 001403      BEQ    40$           ;BRANCH IF NO ERROR
926 015124 104047      EMT    47
927 015126 052703 000001      BIS    #BIT0,R3      ;SET ERROR FLAG
928
929                                     ;TEST 'LBC', 'DPE' FOR ZERO-FAILURE ON DS, IF
930 015132      40$:
931 015132 013737 001400 001142  MOV    RMR2I,$BDDAT
932 015140 042737 175767 001142  BIC    #^C<LBC!DPE>,$BDDAT
  
```

```

933 015146 001403          BEQ     50$           ;BRANCH IF NO ERROR
934 015150 104050          EMT     50
935 015152 052703 000001  BIS     #BIT0,R3       ;SET ERROR FLAG
936
937          ;WRITE ZEROS IN RMER2 THEN WRITE ONES AND CHECK FOR S-A-O BITS.
938 015156          50$:
939 015156 012760 000000 000042  MOV     #0,RMER2(RO)   ;LOAD RMER2
940 015164 012760 177777 000042  MOV     #-1,RMER2(RO)  ;LOAD RMER2
941 015172 016037 000042 001400  MOV     RMER2(RO),RMER2I ;STORE RMER2 IN INPUT BUFFER
942          ;TEST UNUSED BITS FOR ZERO-FAILURE ON IF
943 015200 013737 001400 001142  MOV     RMER2I,$DDAT
944 015206 042737 176250 001142  BIC     #^CXNUE2,$DDAT
945 015214 001403          BEQ     60$           ;BRANCH IF NO ERROR
946 015216 104044          EMT     44
947 015220 052703 000001  BIS     #BIT0,R3       ;SET ERROR FLAG
948
949          ;TEST USED BITS FOR ONE-FAILURE ON IF
950 015224          60$:
951 015224          MOV     RMER2I,$DDAT
952 015232 042737 001527 001142  BIC     #XNUE2,$DDAT
953 015240 012737 176250 001140  MOV     #^CXNUE2,$GDDAT
954 015246 023737 001140 001142  CMP     $GDDAT,$DDAT
955 015254 001403          BEQ     70$           ;BRANCH IF NO ERROR
956 015256 104051          EMT     51
957 015260 052703 000002  BIS     #BIT1,R3       ;SET ERROR FLAG
958
959          ;IF NO PREVIOUS ERROR, TEST BIT INTERFERENCE WITH SHIFTING ONE BIT.
960 015264          70$:
961 015264          TST     R3           ;ANY ERRORS?
962 015266 001044          BNE     120$          ;YES!!
963 015270 012702 000001  MOV     #1,R2           ;R2=DATA PATTERN
964 015274          80$:
965 015274 004737 055156          JSR     PC,CNTCLR      ;GO CLEAR CONTROLLER
966 015300 013760 001444 000032  MOV     RMOFO,RMOF(RO) ;LOAD RMOF
967 015306 010260 000042          MOV     R2,RMER2(RO)   ;LOAD RMER2
968 015312 016037 000042 001142  MOV     RMER2(RO),$DDAT ;STORE RMER2 AT $DDAT
969 015320 032702 040000          BIT     #SKI,R2        ;IS SKI BEING SET?
970 015324 001003          BNE     90$           ;YES!!
971 015326 042737 040000 001142  BIC     #SKI,$DDAT     ;DONT TEST SKI FOR ZERO
972 015334 032702 000200          BIT     #DVC,R2        ;IS DVC BEING SET??
973 015340 001003          BNE     100$          ;YES!!
974 015342 042737 000200 001142  BIC     #DVC,$DDAT     ;DONT TEST DVC FOR ZERO
975 015350 010237 001140          MOV     R2,$GDDAT
976 015354 042737 001527 001140  BIC     #XNUE2,$GDDAT  ;UNUSED BITS SHOULD BE ZERO
977 015362 023737 001140 001142  CMP     $GDDAT,$DDAT  ;ANY ERRORS??
978 015370 001401          BEQ     110$          ;NO!!
979 015372 104052          EMT     52
980
981 015374 006302          110$:  ASL     R2           ;SHIFT TO NEXT DATA BIT
982 015376 001336          BNE     80$           ;CONTINUE IF NOT DONE
983
984 015400          120$:
985
986          ;*****
          ;*TEST 17 SERIAL NUMBER TEST
          ;*****
    
```

```

015400          TST17:
015400 000004          SCOPE          ;SCOPE CALL
015402 000240          NOP
015404 012706 001100  MOV          #STACK,SP      ;LOAD THE STACK POINTER
015410 013700 001276  MOV          $BASE,R0      ;R0 = UNIBUS ADDRESS
015414 013701 001466  MOV          TSTQUE,R1     ;R1 = POINTER TO DEVICE
015420 012737 000017 001226  MOV          #17,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

987
988 015426 004737 055156  JSR          PC,CNTCLR    ;GO CLEAR CONTROLLER
989 015432 010037 001136  MOV          R0,$BDADR    ;SETUP REGISTER ADDRESS FOR TYPEOUT
990 015436 062737 000030 001136  ADD          #RMSN,$BDADR
991 015444 012702 000031  MOV          #25.,R2     ;READ RMSN 25 TIMES
992
993          ;READ RMSN AND USE THE RESULT AS EXPECTED VALUE
994 015450 016037 000030 001140  MOV          RMSN(R0),$GDDAT ;STORE RMSN AT $GDDAT
995
996          ;READ RMSN AND COMPARE WITH INITIAL VALUE
997 015456
998 015456 016037 000030 001142 10$: MOV          RMSN(R0),$BDDAT ;STORE RMSN AT $BDDAT
999 015464 023737 001140 001142  CMP          $GDDAT,$BDDAT
1000 015472 001401          BEQ          20$          ;BRANCH IF SERIAL NUMBER CONSISTENT
1001 015474 104136          EMT          136
1002
1003          ;DECREMENT COUNT AND CONTINUE IF NOT DONE
1004 015476
1005 015476 005302          20$: DEC          R2
1006 015500 100366          BPL          10$
1007 015502
1008
1009          30$:          ;END OF TEST

```

\*\*\*\*\*  
 :TEST 20 CONTROL BUS PARITY DETECTION TEST  
 \*\*\*\*\*

```

015502          TST20:
015502 000004          SCOPE          ;SCOPE CALL
015504 000240          NOP
015506 012706 001100  MOV          #STACK,SP      ;LOAD THE STACK POINTER
015512 013700 001276  MOV          $BASE,R0      ;R0 = UNIBUS ADDRESS
015516 013701 001466  MOV          TSTQUE,R1     ;R1 = POINTER TO DEVICE
015522 012737 000020 001226  MOV          #20,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

1010
1011          ;SETUP FOR FIRST TEST LOOP (NO ERROR)
1012 015530 005037 001140  CLR          $GDDAT      ;'PAR' SHOULD BE ZERO
1013 015534 111137 001422  MOV          (R1),RMC$20  ;SETUP RMC$2 VALUE
1014 015540 042737 177770 001422  BIC          #*CUNTMSK,RMC$20
1015 015546 012737 000001 001450  MOV          #1,RMHRO    ;INITIALIZE DATA PATTERN
1016 015554 000402          BR          6$          ;SKIP INCREMENT FIRST TIME
1017 015556 006337 001450  5$: ASL          RMHRO    ;SHIFT TO NEXT PATTERN
1018
1019          ;CLEAR AND VERIFY THAT 'PAR' IS RESET
1020 015562
1021 015562 004737 055156  6$: JSR          PC,CNTCLR    ;GO CLEAR CONTROLLER
1022 015566 016037 000014 001142  MOV          RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
1023 015574 016037 000042 001400  MOV          RMER2(R0),RMER2I ;STORE RMER2 IN INPUT BUFFER
1024 015602 042737 177767 001142  BIC          #*CPAR,$BDDAT ;DID 'PAR' RESET?
1025 015610 001415          BEQ          20$          ;YES!!
1026 015612 010037 001136  MOV          R0,$BDADR    ;SETUP REGISTER ADDRESS

```

```
1027 015616 062737 000014 001136 ADD #RMER1, $BDADR
1028 015624 032737 000010 001400 BIT #DPE, RMER2I ;IS 'DPE' SET??
1029 015632 001002 BNE 10$ ;YES!!
1030 015634 104067 EMT 67
1031 015636 000453 BR 50$
1032 015640 104070 10$: EMT 70
015640 104070 BR 50$
1033 015642 000451
1034
1035 ;WRITE TEST PATTERN AND VERIFY 'PAR' STATUS
1036 015644 20$:
1037 015644 013760 001422 000010 MOV RMCS2, RMCS2(RO) ;LOAD RMCS2
1038 015652 013760 001450 000036 MOV RMRO, RMRO(RO) ;LOAD RMRO
1039 015660 016037 000014 001142 MOV RMER1(RO), $BDDAT ;STORE RMER1 AT $BDDAT
1040 015666 042737 177767 001142 BIC #C'PAR, $BDDAT
1041 015674 023737 001140 001142 CMP $GDDAT, $BDDAT ;IS 'PAR' CORRECT??
1042 015702 001410 BEQ 40$ ;YES!!
1043 015704 032737 000020 001422 BIT #PAT, RMCS20 ;SHOULD 'PAR' BE SET?
1044 015712 001002 BNE 30$ ;YES!!
1045 015714 104071 EMT 71
1046 015716 000423 BR 50$
1047 015720 30$:
015720 EMT 72
015722 104072 BR 50$ ;SKIP TO NEXT
1048 015722 000421 ;GO TO NEXT PATTERN
1049
1050 015724 005737 001450 40$: TST RMRO ;IS DATA PATTERN COMPLETE??
1051 015730 001312 BNE 5$ ;NO!!
1052 015732 032737 000020 001422 BIT #PAT, RMCS20 ;IS TEST COMPLETE??
1053 015740 001012 BNE 50$ ;YES!!
1054 ;SETUP FOR SECOND TEST LOOP (ERROR)
1055 015742 052737 000020 001422 BIS #PAT, RMCS20 ;TURN ON BAD PARITY
1056 015750 012737 000010 001140 MOV #PAR, $GDDAT ;EXPECT ERROR
1057 015756 012737 000001 001450 MOV #1, RMRO ;INITIALIZE DATA PATTERN
1058 015764 000676 BR 6$ ;START LOOP
1059
1060 015766 50$: ;END OF TEST
1061
1062 ;*****
;*TEST 21 CONTROL BUS PARITY GENERATION TEST
;*****
015766
015766 000004 TST21: SCOPE ;SCOPE CALL
015770 000240 NOP
015772 012706 001100 MOV #STACK, SP ;LOAD THE STACK POINTER
015776 013700 001276 MOV $BASE, RO ;RO = UNIBUS ADDRESS
016002 013701 001466 MOV TSTQUE, R1 ;R1 = POINTER TO DEVICE
016006 012737 000021 001226 MOV #21, $TSTN ;;SET TEST NUMBER IN APT MAIL BOX
1063
1064 ;SETUP FOR TEST (NO ERROR)
1065 016014 012737 000001 001450 MOV #1, RMRO ;INITIALIZE DATA PATTERN
1066 016022 005037 001140 CLR $GDDAT ;MCPE SHOULD BE ZERO
1067 016026 000402 BR 20$ ;DONT SHIFT FIRST TIME
1068
1069 ;SHIFT DATA PATTERN
1070 016030 10$:
1071 016030 006337 001450 ASL RMRO
```

```

1072 016034          20$:
1073 016034          JSR    PC,CNTCLR      ;GO CLEAR CONTROLLER
1074 016040 004737 055156 000036      MOV    R0,R0,R0      ;LOAD R0
1075
1076          ;TRANSFER DATA TO R0, VERIFY NO 'CMPE' ERRO.
1077 016046 016037 000036 001374      MOI    R0,R0,R0      ;STORE R0 IN INPUT BUFFER
1078 016054 016037 000000 001142      MOV    R0,R0,R0      ;STORE R0 AT SDDAT
1079 016062 042737 157777 001142      BIC    #*CMPE,SDDAT ;WAS BAD PARITY DETECTED??
1080 016070 001402          BEQ    30$           ;NO!!
1081 016072 104073          EMT    73
1082 016074 000403          BR     40$
1083
1084          ;GO TO NEXT PATTERN
1085 016076          30$:
1086 016076 005737 001450          TST    R0,R0        ;DONE ALL PATTERNS??
1087 016102 001352          BNE    10$         ;NO!!
1088 016104          40$:
1089
1090          ;*****
          ;*TEST 22      RMDA,RMDC FAULT TEST
          ;*****
    
```

```

          TST22:
          SCOPE          ;SCOPE CALL
          NOP
          MOV    #STACK,SP ;LOAD THE STACK POINTER
          MOV    #BASE,R0   ;R0 = UNIBUS ADDRESS
          MOV    #TSTQ,R1   ;R1 = POINTER TO DEVICE
          MOV    #22,$TSTN  ;SET TEST NUMBER IN APT MAIL BOX
1091
1092 016132 004737 055156          JSR    PC,CNTCLR      ;GO CLEAR CONTROLLER
1093
1094          ;WRITE ZEROS, THEN ONES IN RMDA,RMDC-READ AND TEST FOR S-A-0
1095 016136 012760 000000 000006      MOV    #0,RMDA(R0)   ;LOAD RMDA
1096 016144 012760 000000 000034      MOV    #0,RMDC(R0)   ;LOAD RMDC
1097 016152 012760 177777 000006      MOV    #-1,RMDA(R0)  ;LOAD RMDA
1098 016160 012760 177777 000034      MOV    #-1,RMDC(R0)  ;LOAD RMDC
1099 016166 016037 000006 001344      MOV    RMDA(R0),RMDAI ;STORE RMDA IN INPUT BUFFER
1100 016174 016037 000034 001372      MOV    RMDC(R0),RMDCI ;STORE RMDC IN INPUT BUFFER
1101 016202 022737 177777 001314      CMP    #-1,RMDAI     ;IS ANY REGISTER ALL ONES??
1102 016210 001410          BEQ    10$           ;YES!!
1103 016212 052737 176000 001372      BIS    #UNUSED,RMDCI ;SET UNUSED BITS
1104 016220 022737 177777 001372      CMP    #-1,RMDCI
1105 016226 001401          BEQ    10$           ;YES!!
1106 016230 104042          EMT    42
1107
1108 016232          10$:
1109
1110          ;*****
          ;*TEST 23      DISK ADDRESS TRANSFER TEST
          ;*****
    
```

```

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



```

1155 016524 012737 000024 001226      MOV      #24,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1156 016532 005003                      CLR      R3              ;RESET ERROR FLAGS
1157 016534 004737 055156              JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
1158 016540 005037 001140              CLR      $GDDAT        ;LOAD EXPECTED
1159 016544 010037 001136              MOV      R0,$BDADR     ;LOAD REG ADDRESS
1160 016550 062737 000034 001136      ADD      #RMDC,$BDADR
1161                                     ;WRITE ONES IN RMDC AND VERIFY THAT UNUSED BITS ARE ZERO
1163 016556 012760 177777 000034      MOV      #-1,RMDC(R0)  ;LOAD RMDC
1164 016564 016037 000034 001142      MOV      RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
1165 016572 042737 001777 001142      BIC      #^XNUDC,$BDDAT ;CLEAR ALL USED BITS
1166 016600 001403                      BEQ      5$             ;BRANCH IF NO ERROR
1167 016602 104134                      EMT      134
1168 016604 052703 ^00001              BIS      #BIT0,R3      ;SET ERROR FLAG
1169
1170                                     ;WRITE ONES IN RMDC, THEN WRITE ZEROS, READ AND CHECK FOR S-A-1 BITS
1171 016610                                     5$:
1172 016610 012760 177777 000034      MOV      #-1,RMDC(R0)  ;LOAD RMDC
1173 016616 012760 000000 000034      MOV      #0,RMDC(R0)   ;LOAD RMDC
1174 016624 016037 000034 001142      MOV      RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
1175 016632 042737 176000 001142      BIC      #XNUDC,$BDDAT ;CLEAR UNUSED BITS
1176 016640 001403                      BEQ      10$            ;BRANCH IF NO ERROR
1177 016642 104037                      EMT      37
1178 016644 052703 000001              BIS      #BIT0,R3      ;SET ERROR FLAG
1179
1180                                     ;WRITE ZEROS, THEN ONES IN RMDC, READ AND CHECK FOR S-A-0 BITS
1181 016650                                     10$:
1182 016650 012760 000000 000034      MOV      #0,RMDC(R0)   ;LOAD RMDC
1183 016656 012760 177777 000034      MOV      #-1,RMDC(R0)  ;LOAD RMDC
1184 016664 016037 000034 001142      MOV      RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
1185 016672 052737 176000 001142      BIS      #XNUDC,$BDDAT ;SET UNUSED BITS
1186 016700 012737 177777 001140      MOV      #-1,$GDDAT    ;LOAD EXPECTED RESULT
1187 016706 023737 001140 001142      CMP      $GDDAT,$BDDAT ;IS RMDC ALL ONES ??
1188 016714 001403                      BEQ      20$            ;YES !!
1189 016716 104040                      EMT      40
1190 016720 052703 000002              BIS      #BIT1,R3      ;SET ERROR FLAG
1191
1192                                     ;OMIT BIT TEST IF ANY ERRORS
1193 016724                                     20$:
1194 016724 005703                      TST      R3
1195 016726 001026                      BNE      60$
1196 016730 012702 000001              MOV      #1,R2          ;R2 = TEST PATTERN
1197
1198                                     ;TEST RMDC WITH SHIFTING ONE BIT
1199 016734                                     30$:
1200 016734 012760 000000 000034      MOV      #0,RMDC(R0)   ;LOAD RMDC
1201 016742 010260 000034              MOV      R2,RMDC(R0)   ;LOAD RMDC
1202 016746 010203                      MOV      R2,R3          ;R3 = EXPECTED RESULT
1203 016750 042703 176000              BIC      #XNUDC,R3     ;CLEAR ANY UNUSED BITS
1204 016754 016037 000034 001142      MOV      RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
1205 016762 023703 001142              CMP      $BDDAT,R3
1206 016766 001404                      BEQ      50$
1207 016770 010337 001140              MOV      R3,$GDDAT
1208 016774 104041                      EMT      41
1209 016776 000402                      BR       60$           ;SKIP TO NEXT
1210
1211

```

```

1211 017000 006302      50$:   ASL      R2           :SHIFT TO NEXT BIT
1212 017002 001354      BNE      30$           :CONTINUE IF R2 NOT ZERO
1213
1214 017004      60$:
1215
1216
:*****
:*TEST 25      ILLEGAL REGISTER TEST
:*****
TST2:
017004      SCOPE           :SCOPE CALL
017004 000004      NOP
017006 000240      MOV      #STACK,SP     :LOAD THE STACK POINTER
017010 012706 001100      MOV      $BASE,R0      :R0 = UNIBUS ADDRESS
017014 013700 001276      MOV      TSTQUE,R1     :R1 = POINTER TO DEVICE
017020 013701 001466      MOV      #25,$TESTN    :SET TEST NUMBER 'N' APT MAIL BOX
017024 012737 000025 001226      CLR      $GDDAT        :'ILR' SHOULD BE ZERO
1217
1218 017032 005037 001140      MOV      #0,R2         :R2=REGISTER INDEX
1219 017036 012702 000000
1220
1221      :CLEAR AND VERIFY THAT 'ILR' STATUS IS ZERO
1222      10$:
1222 017042      JSR      PC,CNTCLR     :GO CLEAR CONTROLLER
1223 017042 004737 055156      MOV      RMER1(R0),$BDDAT :STORE RMER1 AT $BDDAT
1224 017046 016037 000014 001142      BIC      #*CILR,$BDDAT
1225 017054 042737 177775 001142      BEQ      20$           :BRANCH IF ILR IS RESET
1226 017062 001411      CLR      $GDDAT        :SETUP GOOD DATA, REG ADR
1227 017064 005037 001140      MOV      R0,$BDDADR
1228 017070 010037 001136      ADD      #RMER1,$BDDADR
1229 017074 062737 000014 001136      EMT      64
1230 017102 104064      BR       140$
1231 017104 000550
1232
1233      :WRITE THE REGISTER (INDEX=R2) AND TEST ILR STATUS
1234      20$:
1234 017106      MOV      R0,R3         :R3=REG ADDRESS
1235 017106 010003      ADD      R2,R3
1236 017110 060203      MOV      ERRVEC,-(SP)  :US: ERRVEC ON STACK
1237 017112 013746 000004      MOV      ERRVEC+2,-(SP) :PUSH ERRVEC+2 ON STACK
1238 017116 013746 000006      MOV      #40$,ERRVEC   :SETUP FOR BUS TIMEOUT
1239 017122 012737 017164 000004      MOV      #PR6,ERRVEC+2
1240 017130 012737 000300 000006      CLR      R4           :R4=REGISTER VALUE
1241 017136 005001      CMP      #RMCS2,R2
1242 017140 022702 000010      BNE      30$
1243 017144 001001      MOVB     (R1),R4       :SELECT DRIVE IF RMCS2
1244 017146 111104      MOV      R4,(R3)      :WRITE TEST REGISTER
1245 017150 010413      MOV      (SP)+,ERRVEC+2 :POP STACK INTO ERRVEC+2
1246 017152 012637 000006      MOV      (SP)+,ERRVEC  :POP STACK INTO ERRVEC
1247 017156 012637 000004      BR       60$
1248 017162 000416      40$:   MOV      #45$,(SP)   :DUPPY RTI ADDRESS
1249 017164 012716 017172      RTI           :RESTORE PRIORITY
1250 017170 000002
1251 017172      45$:   MOV      (SP)+,ERRVEC+2 :POP STACK INTO ERRVEC+2
1252 017176 012637 000004      MOV      (SP)+,ERRVEC  :POP STACK INTO ERRVEC
1253 017202 020227 000046      CMP      R2,ERRVEC2   :WERE ALL REGISTERS READ??
1254 017206 101003      BHI      50$         :YES!
1255 017210 010337 001136      MOV      R3,$BDDADR
1256 017214 104074      EMT      74

```



```

1257 017216 000503          50$: BR      140$
1258
1259 017220          60$:
1260 017220 016037 0000 4 001142      MOV      RMR1(R0), $BDDAT      ;STORE RMR1 AT $BDDAT
1261 017226 042737 177775 001142      BIC     #CILR, $BDDAT
1262 017234 023737 001140 001142      CMP     $GDDAT, $BDDAT      ;IS 'ILR' OK??
1263 017242 001411          BEQ     80$                  ;YES!!
1264 017244 010337 001174          MOV     R3, $TMP0           ;SAVE ADDRESS
1265 017250 032737 000002 001140      BIT     #ILR, $GDDAT        ;SHOULD 'ILR' BE SET??
1266 017256 001002          BNE     70$                  ;YES!!
1267 017260 104065          EMT     65
1268 017262 000401          BR      80$
1269 017264          70$: EMT     66
1270          ;ADVANCE TO THE NEXT REGISTER ADDRESS
1271 017266          80$:
1272 017266 062702 000002      ADD     #2, R2              ;INCREMENT INDEX
1273 017272 022702 000050      CMP     #50, R2            ;TIME TO TRY RH70 ?
1274 017276 101261          BHI     10$                  ;BRANCH IF NOT
1275 017300 103437          BLO     110$                 ;BRANCH IF ALREADY CHECKED
1280 017302 013746 000004      MOV     ERRVEC, -(SP)       ;PUSH ERRVEC ON STACK
1281 017306 013746 000006      MOV     ERRVEC+2, -(SP)    ;PUSH ERRVEC+2 ON STACK
1282 017312 012737 017410 000004      MOV     #130$, ERRVEC     ;SETUP FOR TIMEOUT
1283 017320 012737 000300 000006      MOV     #PR6, ERRVEC+2
1290 017326 052737 000002 001140      BIS     #ILR, $GDDAT      ;SET ILR
1297 017334 012702 000054      MOV     #54, R2            ;START AT INDEX 54 IF RH70/22REG
1298 017340 012760 001400 000000      MOV     #A17!A16, RMC1(R0) ;SET EXTEND BITS
1299 017346 016037 000050 001406      MOV     50(R0), RMBAEI    ;READ THE EXTENDED BITS
1300 017354 042737 177774 001406      BIC     #177774, RMBAEI   ;CHOP OFF
1301 017362 001002          BNE     90$                  ;BRANCH IF RH70/22-REG
1302 017364 012702 000050      MOV     #50, R2            ;OTHERWISE NOT A RH70 OR RH70 WITH 32 REG
1303 017370 012637 000006      90$: MOV     (SP)+, ERRVEC+2
1304 017374 012637 000004      MOV     (SP)+, ERRVEC
1305 017400 022702 000074      110$: CMP     #74, R2
1306 017404 101410          BLOS   140$                 ;DONE ALL TESTS
1307 017406 000615          BR      10$                  ;YES!!
1308
1309 017410 012716 017416      120$:
1310 017414 000002          MOV     #135$, (SP)       ;DUMMY RTI ADDRESS
1311 017416 012637 000006      RTI
1312 017422 012637 000004      ;RESTORE PRIORITY
1313 017426          135$: MOV     (SP)+, ERRVEC+2
1314          MOV     (SP)+, ERRVEC
1315          140$: ;END OF TEST

```

\*\*\*\*\*  
 ;\*TEST 26 RESET GO BY INIT TEST

```

*****
TST26:
017426          SCOPE          ;SCOPE CALL
017426 000004      NOP
017430 000240      MOV     #STACK, SP      ;LOAD THE STACK POINTER
017432 012706 001100      MOV     $BASE, R0       ;R0 = UNIBUS ADDRESS
017436 013700 001276      MOV     TSTQUE, R1     ;R1 = POINTER TO DEVICE
017442 013701 001466      MOV     #26, $TESTN    ;SET TEST NUMBER IN APT MAIL BOX
017446 012737 000026 001226
1316
1317 017454 004737 055156      JSR     PC, CNTCLR     ;GO CLEAR CONTROLLER

```

```

1318 017460 010037 001136          MOV    R0,$B0ADR      ;SETUP REGISTER ADDRESS FOR MSG
1319
1320                               ;SET GO, INITIALIZE AND VERIFY THAT GO IS RESET
1321 017464 012760 000001 000000    MOV    #GO,RPCS1(R0) ;LOAD RPCS1
1322
1323 017472 004737 055156          JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
1324 017476 016037 000000 001142    MOV    RPCS1(R0),$BDDAT ;STORE RPCS1 AT $BDDAT
1325 017504 042737 177776 001142    BIC    #*CGO,$BDDAT
1326 017512 001403                BEQ    10$           ;BRANCH IF GO IS RESET
1327 017514 005037 001140          CLR    $GDDAT
1328 017520 104137                EMT    137
1329
1330 017522                10$:                ;END OF TEST
1331
1332                               ;*****
;+TEST 27          DIAGNOSTIC MODE TEST
;*****
;+TEST 27:
017522                SCOPE                ;SCOPE CALL
017522 000004                NOP
017524 000240                MOV    #STACK,SP    ;LOAD THE STACK POINTER
017526 012706 001100                MOV    $BASE,R0     ;R0 = UNIBUS ADDRESS
017532 013700 001276                MOV    TSTQUE,R1   ;R1 = POINTER TO DEVICE
017536 013701 001466                MOV    #27,$TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
017542 012737 000027 001226
1333
1334 017550 010037 001136          MOV    R0,$B0ADR    ;SETUP REGISTER ADDRESS
1335 017554 062737 000024 001136    ADD    #RPPR1,$B0ADR
1336 017562 005003                CLR    R3           ;INITIALIZE ERROR FLAGS
1337
1338                               ;INITIALIZE AND VERIFY THAT 'DMD' IS RESET
1339 017564 004737 055156          JSR    PC,CNTCLR     ;GO CLEAR CONTROLLER
1340 017570 016037 000024 001142    MOV    RPPR1(R0),$BDDAT ;STORE RPPR1 AT $BDDAT
1341 017576 042737 177776 001142    BIC    #*CDMD,$BDDAT
1342 017604 001403                BEQ    10$           ;BRANCH IF 'DMD' IS ZERO
1343 017606 005037 001140          CLR    $GDDAT
1344 017612 104075                EMT    75
1345
1346                               ;SET AND RESET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-1
1347 017614                10$:                ;SET AND RESET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-1
1348 017614 012760 000001 000024    MOV    #DMD,RPPR1(R0) ;LOAD RPPR1
1349 017622 012760 000000 000024    MOV    #0,RPPR1(R0)  ;LOAD RPPR1
1350 017630 016037 000024 001142    MOV    RPPR1(R0),$BDDAT ;STORE RPPR1 AT $BDDAT
1351 017636 042737 177776 001142    BIC    #*CDMD,$BDDAT
1352 017644 001405                BEQ    20$           ;BRANCH IF DMD NOT S-A-1
1353 017646 005037 001140          CLR    $GDDAT
1354 017652 104076                EMT    76
1355 017654 052703 0000C1          BIS    #BIT0,R3     ;SET ERROR FLAG
1356
1357                               ;RESET AND SET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-0
1358 017660                20$:                ;RESET AND SET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-0
1359 017660 012760 000000 000024    MOV    #0,RPPR1(R0)  ;LOAD RPPR1
1360 017666 012760 000001 000024    MOV    #DMD,RPPR1(R0) ;LOAD RPPR1
1361 017674 016037 000024 001142    MOV    RPPR1(R0),$BDDAT ;STORE RPPR1 AT $BDDAT
1362 017702 042737 177776 001142    BIC    #*CDMD,$BDDAT
1363 017710 001006                BNE    30$           ;BRANCH IF DMD NOT S-A-0
1364 017712 012737 000001 001140    MOV    #DMD,$GDDAT
  
```

```
1365 017720 104077          EMT 77
1366 017722 052703 000002    BIS  #BIT1,R3      ;SET ERROR FLAG
1367
1368                          ;IF NO PREVIOUS ERROR, TEST FOR BIT INTERFERENCE WITH SHIFTING
1369                          ;ONE BIT PATTERN
1370 017726          30$:
1371 017726 005703          TST  R3              ;ANY ERRORS DETECTED??
1372 017730 001027          BNE  60$              ;YES!!
1373 017732 012702 000002    MOV  #BIT1,R2      ;INITIALIZE DATA PATTERN
1374 017736          40$:
1375 017736 012700 000000 000024  MOV  #0,RMPR1(R0) ;LOAD RMPR1
1376 017744 010260 000024          MOV  R2,RMPR1(R0) ;LOAD RMPR1
1377 017750 016037 000024 001142  MOV  RMPR1(R0),SBDAT ;STORE RMPR1 AT SBDAT
1378 017756 042737 177776 001142  BIC  #^CDMD,SBDAT
1379 017764 001407          BEQ  50$              ;BRANCH IF DMD NOT SET
1380 017766 010237 001174          MOV  R2,STMP0      ;SAVE DATA
1381 017772 010237 001174          MOV  R2,STMP0      ;SAVE DATA
1382 017776 005037 001140          CLR  $GDDAT        ;DMD SHOULD BE ZERO
1383 020002 104100          EMT 100
1384                          ;SHIFT TO NEXT DATA BIT AND CONTINUE TEST IF NOT DONE
1385 020004          50$:
1386 020004 006302          ASL  R2              ;
1387 020006 001353          BNE  40$              ;
1388 020010          60$:
1389                          ;END OF TEST
1390
1391                          ;*****
1392                          ;*TEST 30      MOL TEST
1393                          ;*****
1394 020010          TST30:
1395 020010 000004          SCOPE          ;SCOPE CALL
1396 020012 000240          NOP
1397 020014 012706 001100          MOV  #SIACK,SP      ;LOAD THE STACK POINTER
1398 020020 013700 001276          MOV  $BASE,R0       ;R0 = UNIBUS ADDRESS
1399 020024 013701 001466          MOV  TSTQUE,R1      ;R1 = POINTER TO DEVICE
1400 020030 012737 000030 001226  MOV  #30,STESTN     ;;SET TEST NUMBER IN APT MAIL BOX
1401 020036 004737 055156          JSR  PC,CNTCLR      ;GO CLEAR CONTROLLER
1402 020042 010037 001136          MOV  R0,$BDADR     ;R0=REGISTER ADDRESS
1403 020046 062737 000012 001136  ADD  #RMD5,$BDADR
1404 020054 005003          CLR  R3              ;R3=ERROR FLAG
1405
1406                          ;SET DIAGNOSTIC MODE AND VERIFY THAT 'MOL' IS ZERO
1407 020056 012760 000001 000024  MOV  #DMD,RMPR1(R0) ;LOAD RMPR1
1408 020064 016037 000012 001142  MOV  RMD5(R0),SBDAT ;STORE RMD5 AT SBDAT
1409 020072 042737 167777 001142  BIC  #^MOL,SBDAT
1410 020100 001405          BEQ  10$              ;
1411 020102 005037 001140          CLR  $GDDAT
1412 020106 104101          EMT 101
1413 020110 052703 000001          BIS  #BIT0,R3      ;SET ERROR FLAG
1414
1415                          ;SET MAINTENANCE UNIT READY AND VERIFY THAT 'MOL' IS ONE
1416 020114          10$:
1417 020114 012760 000001 000024  MOV  #DMD,RMPR1(R0) ;LOAD RMPR1
1418 020122 012760 001001 000024  MOV  #DMD!MUR,RMPR1(R0) ;LOAD RMPR1
1419 020130 016037 000012 001142  MOV  RMD5(R0),SBDAT ;STORE RMD5 AT SBDAT
```

```

1411 020136 042737 167777 001142      BIC      #*CMOL,SBDDAT
1412 020144 001006                      BNE      20$
1413 020146 012737 010000 001140      MOV      #MOL,$GDDAT
1414 020154 104102                      EMT      102
1415 020156 052703 000002                      BIS      #BIT1,R3
1416
1417
1418      ;IF NO PREVIOUS ERROR, VERIFY VOLUME VALID IS RESET AND
1419      ;TEST FOR BIT INTERFERENCE
1419 020162 005703      20$:      TST      R3          ;ANY ERROR DETECTED??
1420 020162 001057                      BNE      70$          ;YES!!
1421 020164 016037 000012 001142      MOV      RMD5(R0),SBDDAT ;STORE RMD5 AT SBDDAT
1422 020166 042737 177677 001142      BIC      #*CVV,SBDDAT
1423 020174 001403                      BEQ      25$          ;BRANCH IF VV RESET
1424 020202 005037 001140      CLR      $GEDAT
1425 020204 104135                      EMT      135
1426 020210
1427 020212      25$:      MOV      #1,R2          ;INITIALIZE DATA PATTERN
1428 020212 012702 000001      30$:      MOV      #DMD,RMR1(R0) ;LOAD RMR1
1429 020216 012760 000001 000024      MOV      R2,RMR1(R0)    ;LOAD RMR1
1430 020224 010260 000024                      MOV      RMD5(R0),SBDDAT ;STORE RMD5 AT SBDDAT
1431 020230 016037 000012 001142      BIC      #*CMOL,SBDDAT
1432 020236 042737 167777 001142      CLR      R3          ;SETUP EXPECTED 'MOL'
1433 020244 005003                      CLC
1434 020246 032702 001000      BIT      #MUR,R2
1435 020252 001402                      BEQ      40$
1436 020254 052703 010000      BIS      #MOL,R3          ;'MOL' SHOULD BE ONE
1437 020260 020337 001142      40$:      CMP      R3,SBDDAT      ;IS MOL OK??
1438 020264 001405                      BEQ      50$          ;YES!!
1439 020266 010237 001174      MOV      R2,$TMP0        ;SAVE TEST PATTERN
1440 020272 010337 001140      MOV      R3,$GDDAT
1441 020276 104103                      EMT      105
1442
1443      ;SHIFT TO NEXT PATTERN
1444 020300      50$:      BIC      #DMD,R2          ;DONT SHIFT DMD
1445 020300 042702 000001                      BNE      60$
1446 020304 001002                      MOV      #DMD,R2          ;DONT TRUNCATE TEST
1447 020306 012702 000001      60$:      ASL      R2          ;SHIFT TO NEXT BIT
1448 020312 006302                      BEQ      70$          ;BRANCH IF DONE
1449 020314 001403                      BIS      #DMD,R2          ;KEEP DMD ON
1450 020316 052702 000001      BR      30$          ;CONTINUE
1451 020322 000735
1452
1453 020324      70$:
1454
1455
.....
;*TEST 31      WRITE LOCK TEST
.....
TST31:
020324 000004      SCOPE
020326 000240      NOP          ;SCOPE CALL
020330 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
020334 013700 001276      MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
020340 013701 001466      MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
020344 012737 000031 001226      MOV      #31,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
1456

```

```

1457 020352 005003          CLR      R3          ;R3=ERROR FLAG
1458 020354 010037 001136  MOV      R0,SBDADR  ;SETUP REGISTER ADDRESS
1459 020360 062737 000012 001136  ADD      #RMD5,SBDADH
1460 020366 005037 001140          CLR      $GDDA:    ;WRL SHOULD BE ZERO
1461 020372 004737 055156  JSR      PC,CNTCLR  ;GO CLEAR CONTROLLER
1462
1463          ;SET DIAGNOSTIC MODE AND VERIFY 'WRL' IS ZERO
1464 020376 012760 000001 000024  MOV      #DMD,RMPR1(R0) ;LOAD RMPR1
1465 020404 016037 000012 001142  MOV      RMD5(R0),SBDAT ;STORE RMD5 AT SBDAT
1466 020412 042737 173777 001142  BIC      #*CWRL,SBDAT
1467 020420 001403          BEQ      10$        ;BRANCH IF WRL IS ZERO
1468 020422 104104          EMT      104
1469 020424 052703 000001  BIS      #BIT0,R3    ;SET ERROR FLAG
1470
1471          ;SET MAINTENANCE WRITE PROTECT AND VERIFY 'WRL' IS ONE
1472 020430          10$:
1473 020430 012760 000001 000024  MOV      #DMD,RMPR1(R0) ;LOAD RMPR1
1474 020436 012760 000011 000024  MOV      #DMD!MWP,RMPR1(R0) ;LOAD RMPR1
1475 020444 016037 000012 001142  MOV      RMD5(R0),SBDAT ;STORE RMD5 AT SBDAT
1476 020452 042737 173777 001142  BIC      #*CWRL,SBDAT
1477 020460 001006          BNE      20$        ;BRANCH IF WRL IS NOE
1478 020462 012737 004000 001140  MOV      #WRL,$GDDAT  ;WRL SHOULD BE SET
1479 020470 104105          EMT      105
1480 020472 052703 000002  BIS      #BIT1,R3    ;SET ERROR FLAG
1481
1482          ;IF NO PREVIOUS ERROR, TEST FOR BIT INTERFERENCE ON 'MWP'
1483 020476          20$:
1484 020476 005703          TST      R3          ;ANY OTHER ERROR??
1485 020500 001045          BNE      70$        ;YES!!
1486 020502 012702 000001  MOV      #1,R2       ;INITIALIZE DATA PATTERN
1487
1488          ;TRANSFER DATA TO RMPR1, READ RMD5 AND VERIFY WRL
1489 020506          30$:
1490 020506 012760 000001 000024  MOV      #DMD,RMPR1(R0) ;LOAD RMPR1
1491 020514 010260 000024          MOV      R2,RMPR1(R0)  ;LOAD RMPR1
1492 020520 016037 000012 001142  MOV      RMD5(R0),SBDAT ;STORE RMD5 AT SBDAT
1493 020526 042737 173777 001142  BIC      #*CWRL,SBDAT  ;CLEARUP RECEIVED 'WRL'
1494 020534 005003          CLR      R3          ;GENERATE EXPECTED 'WRL'
1495 020536 032702 000010  BIT      #MWP,R2
1496 020542 001402          BEQ      40$
1497 020544 052703 004000  BIS      #WRL,R3     ;WRL SHOULD BE SET
1498 020550 020337 001142  40$:  CMP      R3,SBDAT    ;IS WRL OK??
1499 020554 001405          BEQ      50$        ;YES!!
1500 020556 010337 001140  MOV      R3,$GDDAT   ;SAVE EXPECTED
1501 020562 010237 001174  MOV      R2,$TAP0    ;SAVE DATA PATTERN
1502 020566 104106          EMT      106
1503
1504          ;ADVANCE TO NEXT DATA BIT
1505 020570          50$:
1506 020570 042702 000001  BIC      #DMD,R2     ;DONT SHIFT DMD
1507 020574 001002          BNE      60$
1508 020576 012702 000001  MOV      #DMD,R2
1509 020602 006302  60$:  ASL      R2          ;DONT TRUNCATE TEST
1510 020604 001403          BEQ      70$        ;SHIFT DATA BIT
1511 020606 052702 000001  BIS      #DMD,R2     ;EXIT IF DONE
1512 020612 000735          BR       30$        ;KEEP DIAGNOSTIC MODE ON
1513          ;CONTINUE TEST
  
```

1514 020614  
 1515  
 1516

70\$: :END OF TEST

\*\*\*\*\*  
 : \*TEST 32 DRIVE FAULT TEST  
 \*\*\*\*\*

TST32:

020614  
 020614 000004  
 020616 000240  
 020620 012706 001100  
 020624 013700 001276  
 020630 013701 001466  
 020634 012737 000032 001226

SCOPE :SCOPE CALL  
 NOP  
 MOV #STACK,SP :LOAD THE STACK POINTER  
 MOV \$BASE,R0 :R0 = UNIBUS ADDRESS  
 MOV TSTQUE,R1 :R1 = POINTER TO DEVICE  
 MOV #32,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

1517  
 1518 020642 004737 055156  
 1519 020646 005003  
 1520 020650 010037 001136  
 1521 020654 062737 000042 001136  
 1522 020662 005037 001140

JSR PC,CNTCLR :GO CLEAR CONTROLLER  
 CLR R3 :INITIALIZE ERROR FLAGS  
 MOV R0,\$BDADR :SETUP REGISTER ADDRESS  
 ADD #RMR2,\$BDADR  
 CLR \$GDDAT :'DVC' AND 'UNS' SHOULD BE ZERO

:SET AND RESET MAINTENANCE DRIVE FAULT, VERIFY THAT 'DVC' IS NOT  
 :STUCK-AT-ONE.

1526 020666 012760 000001 000024  
 1527 020674 012760 000000 000042  
 1528 020702 012760 000000 000014  
 1529 020710 016037 000042 001142  
 1530 020716 042737 177577 001142  
 1531 020724 001406  
 1532 020726 104107  
 1533 020730 052703 000001  
 1534 020734 012737 040000 001140

MOV #DMD,RMR1(R0) :LOAD RMR1  
 MOV #0,RMR2(R0) :LOAD RMR2  
 MOV #0,RMR1(R0) :LOAD RMR1  
 MOV RMR2(R0),\$BDDAT :STORE RMR2 AT \$BDDAT  
 BIC #^CDVC,\$BDDAT :IS 'DVC' RESET??  
 BEQ 10\$ :YES!!  
 EMT 107  
 BIS #BIT0,R3 :SET ERROR FLAG  
 MOV #UNS,\$GDDAT

:VERIFY THAT 'UNS' IS SAME AS 'DVC'

1536  
 1537 020742  
 1538 020742 016037 000014 001142  
 1539 020750 042737 137777 001142  
 1540 020756 023737 001140 001142  
 1541 020764 001414  
 1542 020766 010037 001136  
 1543 020772 062737 000014 001136  
 1544 021000 032737 040000 001140  
 1545 021006 001002  
 1546 021010 104110  
 1547 021012 000401  
 1548 021014  
 1549 021014 104111

10\$: MOV RMR1(R0),\$BDDAT :STORE RMR1 AT \$BDD  
 BIC #^CUNS,\$BDDAT  
 CMP \$GDDAT,\$BDDAT :IS 'UNS' OK??  
 BEQ 30\$ :YES!!  
 MOV R0,\$BDADR :SETUP REGISTER ADDRESS  
 ADD #RMR1,\$BDADR  
 BIT #UNS,\$GDDAT :SHOULD 'UNS' BE ON??  
 BNE 20\$ :YES!!  
 EMT 110  
 BR 30\$

20\$: EMT 111

:RESET AND SET 'MDF', VERIFY THAT 'DVC' IS NOT S-A-O.

1551  
 1552 021016  
 1553 021016 012737 000200 001140  
 1554 021024 012760 000001 000024  
 1555 021032 012760 000101 000024  
 1556 021040 016037 000042 001142  
 1557 021046 042737 177577 001142  
 1558 021054 001012  
 1559 021056 010037 001136  
 1560 021062 062737 000042 001136

30\$: MOV #DVC,\$GDDAT :DVC SHOULD BE ON  
 MOV #DMD,RMR1(R0) :LOAD RMR1  
 MOV #DMD,MDF,RMR1(R0) :LOAD RMR1  
 MOV RMR2(R0),\$BDDAT :STORE RMR2 AT \$BDDAT  
 BIC #^CDVC,\$BDDAT  
 BNE 40\$ :BRANCH IF DVC IS SET  
 MOV R0,\$BDADR :SETUP REGISTER ADDRESS  
 ADD #RMR2,\$BDADR

```

1561 021070 104112      EMT      112
1562 021072 052703 000002    BIS      #BIT1,R3      ;SET ERROR FLAG
1563 021076 005037 001140    CLR      $GDDAT     ;UNS SHOULD BE OFF
1564
1565      ;VERIFY THAT 'UNS' IS SAME AS 'DVC'
1566 021102      40$:
1567 021102 005737 001140    TST      $GDDAT     ;CHANGE DVC TO UNS
1568 021106 001403      BEQ      50$
1569 021110 012737 040000 001140    MOV      #UNS,$GDDAT
1570 021116      50$:
1571 021116 016037 000014 001142    MOV      RMER1(RO),SBDDAT ;STORE RMER1 AT SBDDAT
1572 021124 042737 137777 001142    BIC      #^CUNS,SBDDAT
1573 021132 023737 001140 001142    CMP      $GDDAT,SBDDAT
1574 021140 001414      BEQ      70$        ;BRANCH IF UNS IS OK
1575 021142 010037 001136      MOV      RO,SBADR    ;SETUP REGISTER ADDRESS
1576 021146 062737 000014 001136    ADD      #RMER1,SBADR
1577 021154 032737 040000 001140    BIT      #UNS,$GDDAT ;SHOULD UNS BE ON??
1578 021162 001002      BNE      60$        ;YES!!
1579 021164 104113      EMT      113
1580 021166 000401      BR       70$
1581 021170      60$:
1582 021170 104114      EMT      114
1583
1584      ;IF THERE WERE NO PREVIOUS ERRORS, TEST FOR BIT INTERFERENCE ON
1585      ;'MDF'
1586 021172      70$:
1587 021172 005703      TST      R3
1588 021174 001056      BNE      120$      ;BRANCH IF ANY OTHER ERRORS
1589 021176 012702 000001    MOV      #1,R2     ;INITIALIZE TEST PATTERN
1590 021202      80$:
1591 021202 012760 000001 000024    MOV      #DMD,RMR1(RO) ;LOAD RMR1
1592 021210 012760 000000 000042    MOV      #0,RMR2(RO)  ;LOAD RMR2
1593 021216 010260 000024      MOV      R2,RMR1(RO) ;LOAD RMR1
1594 021222 016037 000042 001142    MOV      #RMER2(RO),SBDDAT ;STORE RMER2 AT SBDDAT
1595 021230 042737 177577 001142    BIC      #^CDVC,SBDDAT ;GET RESULTS
1596 021236 005037 001140      CLR      $GDDAT     ;SETUP EXPECTED
1597 021242 032702 000100      BIT      #MDF,R2    ;WAS MDF SET??
1598 021246 001403      BEQ      90$        ;NO!!
1599 021250 052737 000200 001140    BIS      #DVC,$GDDAT ;YES-DVC SHOULD BE ON
1600 021256 023737 001140 001142    CMP      $GDDAT,SBDDAT
1601 021264 001410      BEQ      100$      ;BRANCH IF DVC IS OK
1602 021266 010037 001136      MOV      RO,SBADR   ;SETUP REGISTER ADDRESS
1603 021272 062737 000042 001136    ADD      #RMER2,SBADR
1604 021300 010237 001174      MOV      R2,STMP0   ;SAVE TEST PATTERN
1605 021304 104115      EMT      115
1606
1607      ;SHIFT TO NEXT BIT POSITION
1608 021306      100$:
1609 021306 042702 000001    BIC      #DMD,R2    ;DONT SHIFT DMD
1610 021312 001002      BNE      110$
1611 021314 012702 000001    MOV      #DMD,R2   ;DONT TRUNCATE TEST
1612 021320 006302      ASL      R2        ;SHIFT
1613 021322 001403      BEQ      120$      ;EXIT IF DONE
1614 021324 052702 000001    BIS      #DMD,R2   ;KEEP DMD ON
1615 021330 000724      BR       80$       ;CONTINUE
1616 021332      120$:
1616 021332      ;END OF TEST

```

1617  
 1618

\*\*\*\*\*  
 :\*TEST 33 SEEK ERROR TEST  
 \*\*\*\*\*

TST33:

021332					SCOPE	:SCOPE CALL
021332	000004				MOV	
021334	000240				MOV	
021336	012706	001100			MOV	#STACK,SP ;LOAD THE STACK POINTER
021342	013700	001276			MOV	\$BASE,R0 ;R0 = UNIBUS ADDRESS
021346	013701	001466			MOV	TSTQUE,R1 ;R1 = POINTER TO DEVICE
021352	012737	000033	001226		MOV	#33,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1619						
1620	021360	004737	055156		JSR	PC,CNTCLR ;GO CLEAR CONTROLLER
1621	021364	000003			CLR	R3 ;CLEAR ERROR FLAGS
1622	021366	010037	001136		MOV	R0,\$BDADR ;SETUP REGISTER ADDRESS
1623	021372	062737	000042	001136	ADD	#RMR2,\$BDADR
1624						
1625						:SET DIAGNOSTIC MODE AND VERIFY THAT 'SKI' CAN BE RESET
1626	021400	012760	000001	000024	MOV	#DMD,RMR1(R0) ;LOAD RMR1
1627	021406	012760	000000	000042	MOV	#0,RMR2(R0) ;LOAD RMR2
1628	021414	016037	000042	001142	MOV	RMR2(R0),\$BDDAT ;STORE RMR2 AT \$BDDAT
1629	021422	042737	137777	001142	BIC	#^CSKI,\$BDDAT
1630	021430	001405			BEQ	10\$ ;BRANCH SKI IS RESET
1631	021432	005037	001140		CLR	\$GDDAT ;SKI SHOULD BE ZERO
1632	021436	104116			EMT	116
1633	021440	052703	000001		BIS	#BIT0,R3 ;SET ERROR FLAG
1634						
1635						:SET MAINTENANCE SEEK ERROR AND VERIFY THAT 'SKI' CAN BE SET
1636	021444				10\$	
1637	021444	012760	000001	000024	MOV	#DMD,RMR1(R0) ;LOAD RMR1
1638	021452	012760	000000	000042	MOV	#0,RMR2(R0) ;LOAD RMR2
1639	021460	012760	000201	000024	MOV	#DMD,#SER,RMR1(R0) ;LOAD RMR1
1640	021466	016037	000042	001142	MOV	RMR2(R0),\$BDDAT ;STORE RMR2 AT \$BDDAT
1641	021474	042737	137777	001142	BIC	#^CSKI,\$BDDAT
1642	021502	001005			BNE	20\$ ;BRANCH IF SKI IS SET
1643	021504	012737	040000	001140	MOV	#SKI,\$GDDAT ;CANT SET SKI
1644	021512	052703	000002		BIS	#BIT1,R3 ;SET ERROR FLAG
1645						
1646						:IF NO PREVIOUS ERROR, CHECK FOR BIT INTERFERENCE SETTING MAINTENANCE
1647						:SEEK ERROR.
1648	021516				20\$:	
1649	021516	005703			TST	R3
1650	021520	001051			BNE	70\$ ;BRANCH IF ANY OTHER ERRORS
1651	021522	012702	000001		MOV	#1,R2 ;INITIALIZE TEST PATTERN
1652	021526				30\$:	
1653	021526	012760	000001	000024	MOV	#DMD,RMR1(R0) ;LOAD RMR1
1654	021534	012760	000000	000042	MOV	#0,RMR2(R0) ;LOAD RMR2
1655	021542	010260	000024		MOV	R2,RMR1(R0) ;LOAD RMR1
1656	021546	016037	000042	001142	MOV	RMR2(R0),\$BDDAT ;STORE RMR2 AT \$BDDAT
1657	021554	042737	137777	001142	BIC	#^CSKI,\$BDDAT ;GET SKI STATUS
1658	021562	005037	001140		CLR	\$GDDAT ;SETUP EXPECTED RESULT
1659	021566	032702	000200		BIT	#SER,R2
1660	021572	001403			BEQ	40\$
1661	021574	052737	040000	001140	BIS	#SKI,\$GDDAT ;SKI SHOULD BE ON
1662	021602	023737	001140	001142	40\$:	CMP \$GDDAT,\$BDDAT
1663	021610	001403			BEQ	50\$ ;BRANCH IF SKI IS OK



```

1664 021612 010237 001174      MOV      R2,SIMPO      ;SAVE TEST PATTERN
1665 021616 '04120      EMT      120
1666
1667      ;ADVANCE TEST PATTERN IN R2
1668 021620      50$:
1669 021620 042702 000001      BIC      #DMD,R2      ;DONT SHIFT DMD BIT
1670 021624 001002      BNE      60$
1671 021626 012702 000001      MOV      #DMD,R2      ;DONT TRUNCATE TEST
1672 021632 006302      60$:      ASL      R2            ;SHIFT 10 NEXT BIT
1673 021634 001403      BEQ      70$          ;EXIT IF DONE
1674 021636 052702 000001      BIS      #DMD,R2      ;KEEP DMD ON
1675 021642 000731      BR       30$          ;CONTINUE TEST
1676
1677 021644      70$:
1678      ;END OF TEST
1679
::*****
:TEST 34      PIP TEST
:*****
TST34:
021644      SCOPE      ;SCOPE CALL
021644 000004      NOP
021646 000240      MOV      #STACK,SP    ;LOAD THE STACK POINTER
021650 012706 001100      MOV      $BASE,R0     ;R0 = UNIBUS ADDRESS
021654 013700 001276      MOV      TSTQUE,R1    ;R1 = POINTER TO DEVICE
021660 013701 001466      MOV      #34,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX
021664 012737 000034 001226      MOV
1680
1681 021672 004737 055156      JSR      PC,CNTCLR    ;GO CLEAR CONTROLLER
1682 021676 005003      CLR      R3          ;RESET ERROR FLAGS
1683 021700 010037 001136      MOV      R0,$BDADR   ;SETUP REGISTER ADDRESS
1684 021704 062737 000012 001136      ADD      #R0,$BDADR
1685
1686      ;SET MAINTENANCE ON CYLINDER 'MOC' AND VERIFY THAT 'PIP' CAN BE RESET.
1687 021712 012760 000001 000024      MOV      #DMD,RMPR1(R0) ;LOAD RMPR1
1688 021720 012760 000401 000024      MOV      #DMD!MOC,RMPR1(R0) ;LOAD RMPR1
1689 021726 016037 000012 001142      MOV      R0,$BDADR ;STORE R0 AT $BDADR
1690 021734 042737 157777 001142      BIC      #!CPIP,$BDADR
1691 021742 001405      BEQ      10$          ;BRANCH IF PIP IS RESET
1692 021744 005037 001140      CLR      $GDDAT
1693 021750 104121      EMT      121
1694 021752 052703 000001      BIS      #BIT0,R3     ;SET ERROR FLAG
1695
1696      ;RESET MAINTENANCE ON CYLINDER AND VERIFY THAT 'PIP' CAN BE SET
1697 021756      10$:
1698 021756 012760 000001 000024      MOV      #DMD,RMPR1(R0) ;LOAD RMPR1
1699 021764 016037 000012 001142      MOV      R0,$BDADR ;STORE R0 AT $BDADR
1700 021772 042737 157777 001142      BIC      #!CPIP,$BDADR
1701 022000 001006      BNE      20$          ;BRANCH IF PIP IS SET
1702 022002 012737 020000 001140      MOV      #PIP,$GDDAT
1703 022010 104122      EMT      122
1704 022012 052703 000002      BIS      #BIT1,R3     ;SET ERROR FLAG
1705
1706      ;IF NO PREVIOUS ERROR, TEST FOR ADJACENT BIT SETTING 'MOC'
1707 022016      20$:
1708 022016 005703      TST      R3
1709 022020 001046      BNE      70$          ;BRANCH IF ANY PREVIOUS ERROR
1710 022022 012702 000001      MOV      #1,R2        ;INITIALIZE TEST PATTERN
  
```

```

1711
1712 :WRITE THE TEST PATTERN, CHECK MOC USING PIP
1713 50$:
1714 022026 012760 000001 000024 MOV #DMD,RPPR1(R0) ;LOAD RPPR1
1715 022034 010260 000024 MOV R2,RPPR1(R0) ;LOAD RPPR1
1716 022040 016037 000012 001142 RND5(R0),SBDDAT ;STORE RND5 AT SBDDAT
1717 022046 042737 157777 001142 BIC #*CPIP,SBDDAT ;GET PIP STATUS
1718 022054 005037 001140 CLR $GDDAT ;SETUP EXPECTED RESULT
1719 022060 032702 000400 BIT #MOC,R2
1720 022064 001003 BNE 40$
1721 022066 052737 020000 001140 BIS #PIP,$GDDAT ;PIP SHOULD BE SET
1722 022074 023737 001140 40$: CMP $GDDAT,$SBDDAT
1723 022102 001403 BEQ 50$ ;BRANCH IF PIP OK
1724 022104 010237 001174 MOV R2,$TMP0 ;SAVE TEST PATTERN
1725 022110 104123 EMT 125
1726
1727 :ADVANCE THE TEST PATTERN
1728 50$:
1729 022112 042702 000001 BIC #DMD,R2 ;DONT SHIFT DMD
1730 022116 001002 BNE 60$
1731 022120 012702 000001 MOV #DMD,R2 ;DONT TRUNCATE TEST
1732 022124 006302 60$: ASL R2 ;SHIFT BIT
1733 022126 001403 BEQ 70$ ;EXIT IF DONE
1734 022130 052702 000001 BIS #DMD,R2 ;KEEP DMD ON
1735 022134 000734 BR 30$ ;CONTINUE TEST
1736
1737 70$: ;END OF TEST
1738
1739 :*****
:TEST 35 EBL TEST
:*****
TST35:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV #BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #35,$TSTN ;;SET TEST NUMBER IN APT MAIL BOX
1740
1741 CLR R3 ;RESET ERROR FLAGS
1742 022164 005003 MOV R0,$BDADR ;SETUP REGISTER ADDRESS
1743 022166 010037 001136 ADD #RPPR1,$BDADR
1744 022172 062737 000024 001136 CLR $GDDAT ;SETUP EXPECTED RESULT
1745
1746 :CLEAR AND VERIFY THAT END OF BLOCK IS RESET
1747 022204 004737 055156 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
1748 022210 016037 000024 001142 MOV RPPR1(R0),SBDDAT ;STORE RPPR1 AT SBDDAT
1749 022216 042737 157777 001142 BIC #*CEBL,SBDDAT
1750 022224 001403 BEQ 10$ ;BRANCH IF EBL IS RESET
1751 022226 104124 EMT 124
1752 022230 052703 000001 BIS #BIT0,R3 ;SET ERROR FLAG
1753
1754 :SET AND RESET DIAGNOSTIC END OF BLOCK, CHECK FOR EBL S-A-1.
1755 10$:
1756 022234 012760 000001 000024 MOV #DMD,RPPR1(R0) ;LOAD RPPR1
1757 022242 012760 020001 000024 MOV #DMD!DEBL,RPPR1(R0) ;LOAD RPPR1
  
```

```

1758 022250 012760 000001 000024 MOV #DMD,RPPR1(R0) .LOAD RPPR1
1759 022256 016037 000024 001142 MOV RPPR1(R0),SDDAT .STORE RPPR1 AT SDDAT
1760 022264 042737 157777 001142 BIC #*CEBL,SDDAT
1761 022272 001403 BEQ 20$ .BRANCH IF EBL IS RESET
1762 022274 104125 EMT 125
1763 022276 052703 000001 BIS #BIT0,R3 .SET ERROR FLAG
1764
1765
1766 022302 012760 000001 000024 20$ RESET AND SET DIAGNOSTIC END OF BLOCK, CHECK FOR EBL S-A-G
1767 022302 012760 000001 000024 MOV #DMD,RPPR1(R0) .LOAD RPPR1
1768 022310 016037 000024 001142 MOV #DMD:DEBL,RPPR1(R0) .LOAD RPPR1
1769 022316 016037 000024 001142 MOV RPPR1(R0),SDDAT .STORE RPPR1 AT SDDAT
1770 022324 042737 157777 001142 BIC #*CEBL,SDDAT
1771 022332 UC1006 BNE 30$ .BRANCH IF EBL IS SET
1772 022334 012737 020000 001140 MOV #EBL,$GDDAT
1773 022342 104126 EMT 126
1774 022344 052703 000002 BIS #BIT1,R3 .SET ERROR FLAG
1775
1776
1777 022350 005703 30$ .IF NO PREVIOUS ERRORS, TEST FOR ADJACENT BIT INTERFERENCE ON 'DEBL'
1778 022350 001042 TST R3
1779 022352 012702 000001 BNE 70$ .BRANCH IF ANY ERROR
1780 022354 012702 000001 MOV #1,R2 .INITIALIZE TEST PATTERN
1781
1782
1783 022360 012760 000001 000024 40$ .WRITE, READ AND VERIFY THE TEST PATTERN IN R2
1784 022360 010260 000024 MOV #DMD,RPPR1(R0) .LOAD RPPR1
1785 022366 016037 000024 001142 MOV R2,RPPR1(R0) .LOAD RPPR1
1786 022372 016037 000024 001142 MOV RPPR1(R0),SDDAT .STORE RPPR1 AT SDDAT
1787 022400 042737 157777 001142 BIC #*CEBL,SDDAT
1788 022406 010203 MOV R2,R3 .GENERATE EXPECTED RESULT
1789 022410 042703 157777 BIC #*CEBL,R3
1790 022414 020337 001142 CMP R3,SDDAT
1791 022420 001405 BEQ 50$ .BRANCH IF EBL IS OK
1792 022422 010337 001140 MOV R3,$GDDAT .SAVE EXPECTED RESULT
1793 022426 010237 001174 MOV R2,STMP0 .SAVE TEST PATTERN
1794 022432 104127 EMT 127
1795
1796
1797 022434 042702 000001 50$ .SHIFT TO NEXT BIT POSITION
1798 022434 001002 BIC #DMD,R2 .DONT SHIFT DMD
1799 022440 012702 000001 BNE 60$
1800 022442 006302 MOV #DMD,R2 .DONT TRUNCATE DMD
1801 022446 001403 60$ ASL R2 .SHIFT TO NEXT BIT
1802 022450 052702 BEQ 70$ .EXIT IF DONE
1803 022452 000740 BIS #DMD,R2 .KEEP DMD ON
1804 022456 000740 BR 40$ .CONTINUE TEST
1805
1806 022460 70$ .END OF TEST
1807
1819
1820

```

```

*****
*TEST 36 LAST SECTOR, LAST TRACK TEST
*TRANSFER TEST PATTERN TO RMDA THEN VERIFY LAST SECTOR 'LS' AND LAST
*SECTOR/TRACK 'LST' FOR EACH TRANSFER. THE TABLE BELOW LISTS THE VALUE
*OF RMDA FOR WHICH LS AND LST ARE SET.
*

```

```

* 18 BIT MODE      16 BIT MODL      16 BIT MODE W/ SSEI SET
*LS = XXX035      XXX036      XXX037
*
*LST= 006435      006436      006437
*
*.....

```

```

022450
022460 000004
022462 00024C
022464 012706 001100
022470 013700 001276
022474 013701 001466
022500 012707 000003 001206
022506 012737 000036 001226
TST36:
SCOPE                                :SCOPE CALL
NOP
MOV #STACK,SP                        :LOAD THE STACK POINTER
MOV SBASE,R0                          :R0 = UNIBUS ADDRESS
MOV TSTQUE,R1                         :R1 = POINTER TO DEVICE
MOV #3,$TIMES                         :DO 3 ITERATIONS
MOV #36,$TESTN                       :SET TEST NUMBER IN APT MAIL BOX

1821 022514 005002
1822 022516 010037 001136
1823 022522 062737 000024 001136
1824 022530 005037 001444
1825 022534 013737 001334 023114
1826 022542 112737 000035 023114
1827 022550
1828 022550 004737 055156 10$
1829 022554 013760 001444 000032
1830 022562 016037 000032 001176
1831 022570 010260 000006
1832 022574 016037 000024 001362
1833 022602 013737 001362 001142
1834 022610 042737 177773 001142
1835 022616 005037 001140
1836 022622 032737 010000 001444
1837 022630 001010
1838 022632 032737 001000 001444
1839 022640 001010
1840 022642 123702 023114
1841 022646 001014
1842 022650 000410
1843 022652 123702 023114 20$
1844 022654 001010
1845 022660 000404
1846 022662 123737 023114 25$
1847 022670 001003
1848 022672 052737 000004 001140 30$
1849 022700 023737 001140 001142 35$
1850 022706 001404
1851 022710 010237 001174
1852 022714 104130
1853 022716 000477
1854 022720 013737 001362 001142 40$
1855 022726 042737 177773 001142
1856 022734 005037 001140
1857 022740 032737 010000 001444
1858 022746 001010
1859 022750 032737 001000 001444
1860 022756 001010

```

```

1864 022760 023702 023114      CMP      80$,R2      ;18 BIT MODE LAST TRACK/SECTOR ?
1865 022764 001013              BNE      65$        ;NO !!
1866 022766 000407              BR       60$
1867 022770 023702 023114      50$:    CMP      80$,R2      ;16 BIT MODE LAST TRACK/SECTOR ?
1868 022774 001007              BNE      65$        ;NO !!
1869 022776 000403              BR       60$
1870 023000 023702 023114      55$:    CMP      80$,R2      ;16 BIT MODE W/ SSEI LAST TRACK/SECTOR ?
1871 023004 001003              BNE      65$        ;NO !!
1872 023006 052737 000002 001140 60$:    BIS      #LST,$GDDAT ;LST SHOULD BE SET
1873 023014 023737 001140 001142 65$:    CMP      $GDDAT,$BDDAT ;IS LST STATUS CORRECT ?
1874 023022 001404              BEQ      70$        ;BR IF YES
1875 023024 010237 001174      MOV      R2,$TMPO   ;SAVE TEST PATTERN
1876 023030 104131              EMT      131
1877 023032 000431              BR       90$        ;SKIP TO NEXT

```

```

;ADVANCE TO NEXT TEST PATTERN, CHANGE TO 16 BIT MODE IF ALL
;18 BIT TESTS DONE.
70$:

```

```

1881 023034 005202              INC      R2          ;INCREMENT PATTERN
1882 023034 001244              BNE      :0$        ;CONTINUE IF NOT DONE
1883 023036 001244              BIT      #FMT16,RMOFO ;DONE 16 BIT TEST ?
1884 023040 032737 010000 001444      BNE      75$        ;YES!!
1885 023046 001007              MOV      #FMT16,RMOFO ;DO 16 BIT FORMAT TEST
1886 023050 012737 010000 001444      MOV      #036,80$   ;STORE LAST SECTOR ADDRESS
1887 023056 112737 000036 023114      MOV      10$
1888 023064 000631              BR
1889
1890 023066 032737 001000 001444 75$:    BIT      #SSEI,RMOFO ;DONE 16 BIT FORMAT W/ SSEI SET ?
1891 023074 001410              BEQ      90$        ;YES !!
1892 023076 052737 001000 001444      BIS      #SSEI,RMOFO ;SET SSEI WITH 16 BIT MODE AND
1893 023104 112737 000037 023114      MOV      #037,80$   ;STORE LAST SECTOR ADDRESS
1894 023112 000616              BR       10$        ;TEST AGAIN

```

```

1895
1896 023114 000000      80$:    .WORD 0          ;HOLDS LAST TRACK/SECTOR ADDRESS
1897

```

```

1898 023116      90$:
1899
1903
;*****
;*TEST 37      RMDA COUNT TEST
;*****

```

```

023116      TST37:
023116 000004              SCOPE              ;SCOPE CALL
023120 000240              NOP
023122 012706 001100      MOV      #STACK,SP ;LOAD THE STACK POINTER
023126 013700 001276      MOV      $BASE,R0  ;R0 = UNIBUS ADDRESS
023132 013701 001466      MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
023136 012737 000003 001206      MOV      #3,$TIMES ;DO 3 ITERATIONS
023144 012737 000037 001226      MOV      #37,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

```

904
1905 023152 010057 001136      MOV      R0,$BDADR ;SETUP REGISTER ADDRESS
1906 023156 062737 000006 001136      ADD      #RMDA,$BDADR
1907 023164 005037 001444      CLR      RMOFO      ;START WITH 18 BIT FORMAT
1908 023170 012737 000035 023622      MOV      #29,1*0$  ;SETUP LAST SECTOR FOR 18 BIT
1909 023176 012737 000001 001140      MOV      #1,$GDDAT ;SETUP FIRST COUNT
1910
1911
1912

```

```

;INCREMENT SECTOR COUNT USING DIAGNOSTIC END OF BLOCK STARTING AT
;SECTOR 0 AND CONTINUING UNTIL TRACK ADDRESS INCREMENTS

```

```

1913 .CLEAR THE MASSBUS
1914 .SET FORMAT
1915 .LOAD SECTOR AND TRACK ADDRESS
1916 .ENABLE DEBUG CLOCK
1917 .SET GO BIT
1918 023204
1919 023204 004737 055156 10$: JSR PC,CNTCLR ;GO CLEAR CONTROLLER
1920 023210 012760 000001 000024 MOV #DMD,RMPR1(RO) ;LOAD RMPR1
1921 023216 012760 000000 000014 MOV #0,RMPER1(RO) ;LOAD RMPER1
1922 023224 012760 000000 000042 MOV #0,RMPER2(RO) ;LOAD RMPER2
1923 023232 012760 000000 000006 MOV #0,RMDA(RO) ;LOAD RMDA
1924 023240 013760 001444 000032 MOV RMOFO,RMOF(RO) ;LOAD RMOF
1925 023246 016037 000032 001174 MOV RMOF(RO),STMPO ;STORE RMOF AT STMPO
1926 023254 012760 040001 000024 MOV #DMD!DBEN,RMPR1(RO) ;LOAD RMPR1
1927 023262 012760 000001 000000 MOV #GO,RMCS1(RO) ;LOAD RMCS1
1928
1929
1930 023270 SET AND RESET EBL TO INCREMENT RMDA THEN VERIFY RMDA.
1931 023270 012760 060001 000024 25$: MOV #DMD!DBEN!DEBL,RMPR1(RO) ;LOAD RMPR1
1932 023276 012760 040001 000024 MOV #DMD!DBEN,RMPR1(RO) ;LOAD RMPR1
1933 023304 016037 000006 001142 MOV RMDA(RO),$BDDAT ;STORE RMDA AT $BDDAT
1934 023312 023737 001142 001140 CMP $BDDAT,$GDDAT
1935 023320 001402 BEQ 30$ ;BRANCH IF RMDA OK
1936 023322 104132 EMT 132
1937 023324 000416 BR 50$ ;OUT OF SYNC-SKIP TO NEXT
1938
1939 :ADVANCE EXPECTED SECTOR COUNT AND CONTINUE IF ONE CYCLE NOT
1940 :COMPLETE
1941 023326 30$:
1942 023326 005237 001140 INC $GDDAT ;INCREMENT EXPECTED SECTOR
1943 023332 123737 001142 023622 CMPEB $BDDAT,110$ ;HAS THE LAST SECTOR JUST COUNTED??
1944 023340 001004 BNE 40$ ;NO!!
1945 023342 105037 001140 CLRB $GDDAT ;YES-NEXT SECTOR SHOULD BE ZERO
1946 023346 105237 001141 INCB $GDDAT+1 ;INCREMENT TRACK ADDRESS
1947 023352 105737 001142 40$: TSTB $BDDAT ;HAS A FULL CYCLE BEEN COUNTED??
1948 023356 001401 BEQ 50$ ;YES-DO NEXT
1949 023360 000743 BR 25$ ;CONTINUE SECTOR TEST
1950
1951 :INCREMENT TRACK COUNT USING DIAGNOSTIC END OF BLOCK. START AT TRACK 0.
1952 :LAST SECTOR AND COUNT ONE COMPLETE TRACK CYCLE.
1953 023362 50$:
1954 023362 013737 023622 001420 MOV 110$,RMDAO ;STARTING TRACK ADDRESS = 0
1955 023370 012737 000400 001140 MOV #TA1,$GDDAT ;FIRST VALUE AFTER INCREMENT
1956
1957 .CLEAR THE MASSBUS
1958 .SET FORMAT
1959 .LOAD LAST SECTOR ADDRESS AND TEST TRACK ADDRESS
1960 .ENABLE DEBUG CLOCK
1961 .SET GO BIT
1962 023376 60$:
1963 023376 004737 055156 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
1964 023402 013760 001444 000032 MOV RMOFO,RMOF(RO) ;LOAD RMOF
1965 023410 013760 001420 000006 MOV RMDAO,RMDA(RO) ;LOAD RMDA
1966 023416 012760 000001 000024 MOV #DMD,RMPR1(RO) ;LOAD RMPR1
1967 023424 012760 040001 000024 MOV #DMD!DBEN,RMPR1(RO) ;LOAD RMPR1
1968 023432 012760 000001 0000 0 MOV #GO,RMCS1(RO) ;LOAD RMCS1
1969

```

```

1970 :CLOCK RMDA USING DIAGNOSTIC END OF BLOCK
1971 023440 012760 060001 000024 MOV #RMD!DBEN!DEBL,RMPR1(RO) ;LOAD RMPR1
1972 023446 012760 040001 000024 MOV #RMD!DBEN,RMPR1(RO) ;LOAD RMPR1
1973
1974 .VERIFY RMDA ACCORDING TO $GDDAT
1975 023454 016037 000006 001142 MOV RMDA(RO),SBDDAT ;STORE RMDA AT SBDDAT
1976 023462 023737 001140 001142 CMP $GDDAT,SBDDAT
1977 023470 001402 BEQ 70$
1978 023472 104133 EMT 133
1979 023474 000453 BR 120$ ;OUT OF SYNC-SKIP TO NEXT
1980
1981 ;SETUP FOR NEXT INCREMENT OF RMDA TRACK ADDRESS
1982 023476 70$: INCB $GDDAT+1 ;ADVANCE EXPECTED TRACK
1983 023476 105237 001141 CMPB SBDDAT+1,LSTRK+1 ;WAS THE LAST TRACK JUST COUNTED??
1984 023502 123737 001143 001335 BNE 80$ ;NO!!
1985 023510 001002 CLR $GDDAT ;YES-NEXT TRACK, SECTOR SHOULD BE ZERO
1986 023512 005037 001140 80$: MOV SBDDAT,RMDAO ;HAS A FULL CYCLE BEEN COUNTED??
1987 023516 013737 001142 001420 BEQ 90$ ;YES!!
1988 023524 001404 MOVB 110$,RMDAO ;INCREMENT FROM LAST SECTOR
1989 023526 113737 023622 001420 BR 60$
1990 023534 000720
1991 023536 90$: BIT #FMT16,RMOFO ;DONE BOTH FORMATS??
1992 023536 032737 010000 001444 BNE 95$ ;YES!!
1993 023544 001007 MOV #FMT16,RMOFO ;SET FORMAT BIT FOR 16
1994 023546 012737 010000 001444 MOV #30,,110$ ;SET LAST SECTOR FOR 16 MODE
1995 023554 012737 000036 023622 BR 100$
1996 023562 000412
1997 023564 95$: BIT #SSE1,RMOFO ;DONE 16 BIT MODE W/ SSE1 SET ?
1998 023564 032737 001000 001444 BNE 120$ ;YES !!
1999 023572 001014 BIS #SSE1,RMOFO ;SET SSE1 W/ 16 BIT MODE AND
2000 023574 052737 001000 001444 MOV #31,,110$ ;SET LAST SECTOR
2001 023602 012737 000037 023622
2002
2003 023610 012737 000001 001140 100$: MOV #1,$GDDAT ;SET FIRST COUNT VALUE
2004 023616 000137 023204 JMP 10$ ;REPEAT TEST
2005
2006 023622 000000 110$: .WORD ;STORAGE FOR LAST SECTOR VALUE
2007
2008 023624 120$:
2009
2010 :*****
: *TEST 40 RMDC COUNT TEST
:*****
TST40:
023624 SCOPE ;SCOPE CALL
023624 000004 NOP
023626 000240 MOV #STACK,SP ;LOAD THE STACK POINTER
023630 012706 MOV $BASE,RO ;RO = UNIBUS ADDRESS
023634 013700 001276 MOV TSTQIE,R1 ;R1 = POINTER TO DEVICE
023640 013701 001466 MOV #3,$TIMES ;DO 3 ITERATIONS
023644 012737 000003 001206 MOV #40,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
023652 012737 000040 001226
2011
2012 ;CLEAR RMDC, SET FORMAT AND SETUP PROGRAM PARAMETERS
2013 023660 010037 001136 MOV RO,$BDADR ;SETUP REGISTER ADDRESS
2014 023664 062737 000034 001136 ADD #RMDC,$BDADR
2015 023672 005037 001477 CLR RMOFO ;START WITH 18 BIT FORMAT
  
```

```

2016 023676 013737 001334 001420      MOV    LSTRK,RMDAO      ;LOAD LAST TRACK AND
2017 023704 112737 000035 001420      MOVB   #29.,RMDAO      ;LAST SECTOR
2018 023712                                     10$:
2019 023712 004737 055156                JSR    PC,CNTCLR       ;GO CLEAR CONTROLLER
2020 023716 012737 000001 001140      MOV    #1,$GDDAT       ;LOAD FIRST INCREMENTAL VALUE
2021 023724 012760 000000 000034      MOV    #0,RMDC(RO)     ;LOAD RMDC
2022
2023                                     :
2024                                     :
2025                                     :
2026                                     :
2027                                     :
2028 023732                                     20$:
2029 023732 004737 055156                JSR    PC,CNTCLR       ;GO CLEAR CONTROLLER
2030 023736 013760 001444 000034      MOV    RMOFO,RMOF(RO)  ;LOAD RMOF
2031 023744 016037 000032 001174      MOV    RMOF(RO),STMP0   ;STORE RMOF AT STMP0
2032 023752 013760 001420 000006      MOV    RMDAO,RMDA(RO)  ;LOAD RMDA
2033 023760 012760 000001 000024      MOV    #DMD,RMPR1(RO)  ;LOAD RMPR1
2034 023766 012760 040001 000024      MOV    #DMD!DBEN,RMPR1(RO) ;LOAD RMPR1
2035 023774 012760 000001 000000      MOV    #GO,RMCS1(RO)   ;LOAD RMCS1
2036
2037                                     :
2038 024002 012760 060001 000024      MOV    #DMD!DBEN!DEBL,RMPR1(RO) ;LOAD RMPR1
2039 024010 012760 040001 000024      MOV    #DMD!DBEN,RMPR1(RO) ;LOAD RMPR1
2040 024016 016037 000034 001142      MOV    RMDC(RO),SBDDAT ;STORE RMDC AT SBDDAT
2041 024024 023737 001140 001142      CMP    $GDDAT,$BDDAT
2042 024032 001402                BEQ    30$              ;BRANCH IF RMDC=RMDC+1
2043 024034 104140                EMT    140
2044 024036 000442                BR     60$              ;OUT OF SYNC-SKIP TO END
2045
2046                                     :
2047 024040                                     30$:
2048 024040 005237 001140                INC    $GDDAT           ;ADVANCE NEXT RESULT
2049 024044 022737 002000 001140      CMP    #1024.,$GDDAT   ;SHOULD NEXT VALUE BE ZERO??
2050 024052 001002                BNE    40$              ;NO!!
2051 024054 005037 001140                CLR    $GDDAT           ;YES-RMDC SHOULD OVERFLOW
2052 024060 005737 001142                TST   $BDDAT           ;IS ONE CYCLE COMPLETE??
2053 024064 001401                BEQ    50$              ;YES!!
2054 024066 000721                BR     20$              ;CONTINUE
2055 024070                                     50$:
2056 024070 032737 010000 00144~      BIT    #FMT16,RMOFO    ;DONE 16 BIT FORMAT MODE ?
2057 024076 001007                BNE    55$              ;YES !!
2058 024100 012737 010000 001444      MOV    #FMT16,RMOFO    ;SET 16 BIT FORMAT AND
2059 024106 112737 000036 001420      MOVB   #30.,RMDAO      ;LAST SECTOR
2060 024114 000676                BR     10$
2061 024116                                     55$:
2062 024116 032737 001000 001444      BIT    #SSEI,RMOFO     ;DONE 16 BIT MODE W/ SSEI SET ?
2063 024124 001007                BNE    60$              ;YES !!
2064 024126 052737 001000 001444      BIS    #SSEI,RMOFO     ;SET SSEI W/ 16 BIT MODE AND
2065 024134 112737 000037 001420      MOVB   #31.,RMDAO      ;LAST SECTOR
2066 024142 000663                BR     10$              ;REPEAT TEST
2067 024144                                     60$:
2068
2069

```

\*\*\*\*\*  
 : TEST 41 LBT TEST  
 \*\*\*\*\*



```

024144          TST41:
024144 00C004          SCOPE          ;SCOPE CALL
024146 000240          NOP
024150 012706 001100  MOV          #STACK,SP      ;LOAD THE STACK POINTER
024154 013700 001276  MOV          $BASE,R0        ;R0 = UNIBUS ADDRESS
024160 013701 001466  MOV          TSTQUE,R1       ;R1 = POINTER TO DEVICE
024164 012737 000003 001206  MOV          #3,$TIMES      ;;DO 3 ITERATIONS
024172 012737 000041 001226  MOV          #4,$STESTN     ;;SET TEST NUMBER IN APT MAIL BOX

2070
2071 024200 010037 001136  MOV          R0,$BDADR      ;SETUP REGISTER ADDRESS
2072 024204 062737 000012 001136  ADD          #RMD5,$BDADR
2073
2074 024212 005037 001444  CLR          RMOFO         ;START WITH 18 BIT MODE
2075 024216 013737 001334 001420  MOV          LSTRK,RMDAO    ;SET LAST TRACK AND
2076 024224 112737 000035 001420  MOVB        #29.,RMDAO    ;LAST SECTOR
2077
2078          : CLEAR THE MASSBUS
2079          : SET FORMAT
2080          : LOAD LAST TRACK AND SECTOR
2081          : LOAD LAST CYLINDER
2082          :
2083          :
2084          :
2084 024232          10$:
2085 024232 004737 055156  JSR          PC,ENTCLR     ;GO CLEAR CONTROLLER
2086 024236 012760 001060 000034  MOV          #560.,RMDC(R0) ;LOAD RMDC
2087 024244 013760 001420 000006  MOV          RMDAO,RMDA(R0) ;LOAD RMDA
2088 024252 013760 001444 000032  MOV          RMOFO,RMOF(R0) ;LOAD RMOF
2089 024260 016037 000032 001174  MOV          RMOF(R0),$STMP0 ;STORE RMOF AT $STMP0
2090 024266 016037 000012 001142  MOV          RMD5(R0),$BDDAT ;STORE RMD5 AT $BDDAT
2091 024274 042737 175777 001142  BIC          #*CLBT,$BDDAT
2092 024302 001403          BEQ          20$         ;BRANCH IF LBT IS RESET
2093 024304 005037 001140  CLR          $GDDAT       ;LBT SHOULD BE ZERO
2094 024310 104141          EMT          141
2095
2096          :
2097          :
2098          :
2099          :
2100          :
2101 024312          20$:
2102 024312 012760 000001 000024  MOV          #RMD,RMPR1(R0) ;LOAD RMPR1
2103 024320 012760 000000 000014  MOV          #0,RMPR1(R0)   ;LOAD RMPR1
2104 024326 012760 000000 000042  MOV          #0,RMPR2(R0)   ;LOAD RMPR2
2105 024334 012760 000001 000000  MOV          #60,RPCS1(R0)  ;LOAD RPCS1
2106 024342 012760 060001 000024  MOV          #RMD!DBEN!DEBL,RMPR1(R0) ;LOAD RMPR1
2107 024350 012760 040001 000024  MOV          #RMD!DBEN,RMPR1(R0) ;LOAD RMPR1
2108 024356 016037 000012 001142  MOV          RMD5(R0),$BDDAT ;STORE RMD5 AT $BDDAT
2109 024364 042737 175777 001142  BIC          #*CLBT,$BDDAT
2110 024372 001005          BNE          30$         ;BRANCH IF LBT IS SET
2111 024374 012737 002000 001140  MOV          #LBT,$GDDAT
2112 024402 104142          EMT          142
2113 024404 000426          BR          40$
2114 024406          30$:
2115 024406 032737 010000 001444  BIT          #FMT16,RMOFO   ;DONE 16 BIT FORMAT
2116 024414 001007          BNE          35$         ;YES !!
2117 024416 012737 010000 001444  MOV          #FMT16,RMOFO   ;SET 16 BIT MODE AND
2118 024424 112737 000036 001420  MOVB        #30.,RMDAO    ;LAST SECTOR
    
```

```

2119 024432 000677          BR      10$          :TEST AGAIN.
2120 024434          35$
2121 024434 032737 001000 001444  BIT      #SSEI,RMOFO  ,DONE 16 BIT MODE W/ SSEI SET ?
2122 024442 001007          BNE      40$          :YES !!
2123 024444 052737 001000 001444  BIS      #SSEI,RMOFO  ,SET SSEI W/ 16 BIT MODE AND
2124 024452 112737 000037 001420  MOVB    #31.,RMDAD   ,LAST SECTOR
2125 024460 000664          BR      10$          :TEST AGAIN.
2126 024462          40$.
2127
2128

```

\*\*\*\*\*  
 :\*TEST 42 COMPOSITE ERROR TEST  
 \*\*\*\*\*

```

024462
024462 000004          TST42:
024464 000240          SCOPE          :SCOPE CALL
024466 012706 001100  MOV      #STACK,SP   ,LOAD THE STACK POINTER
024472 013700 001276  MOV      $BASE,R0    ,R0 = UNIBUS ADDRESS
024476 013701 001466  MOV      TSTQUE,R1   ,R1 = POINTER TO DEVICE
024502 012737 000042 001226  MOV      #42,$TESTN  ,:SET TEST NUMBER IN APT MAIL BOX
2129
2130 024510 010037 001136  MOV      R0,$BDADR   ,SETUP REGISTER ADDRESS
2131 024514 062737 000012 001136  ADD      #RMD5,$BDADR
2132
2133

```

:USING DIAGNOSTIC MODE, CLEAR ALL ERRORS AND VERIFY THAT COMPOSITE  
 :ERROR IS RESET.

```

2134
2135 024522 004737 055156          JSR      PC,CNTCLR   ,GO CLEAR CONTROLLER
2136 024526 012760 000001 000024  MOV      #DMD,RMR1(R0) ,LOAD RMR1
2137 024534 012760 000000 000014  MOV      #0,RMR1(R0)  ,LOAD RMR1
2138 024542 012760 000000 000042  MOV      #0,RMR2(R0)  ,LOAD RMR2
2139 024550 016037 000012 001142  MOV      RMD5(R0),$BDDAT ,STORE RMD5 AT $BDDAT
2140 024556 042737 137777 001142  BIC      #*CERR,$BDDAT
2141 024564 001403          BEQ      10$          :BRANCH IF ERR IS RESET
2142 024566 005037 001140  CLR      $GDDAT
2143 024572 104143          EMT      143
2144 024574 012737 040000 001140 10$: MOV      #ERR,$GDDAT
2145

```

:SET BOTH ERROR REGISTERS AND VERIFY THAT COMPOSITE ERROR IS SET

```

2146
2147 024602 012760 177777 000014  MOV      #-1,RMR1(R0) ,LOAD RMR1
2148 024610 012760 177777 000042  MOV      #-1,RMR2(R0) ,LOAD RMR2
2149 024616 016037 000012 001142  MOV      RMD5(R0),$BDDAT ,STORE RMD5 AT $BDDAT
2150 024624 042737 137777 001142  BIC      #*CERR,$BDDAT
2151 024632 001001          BNE      20$          :BRANCH IF ERR IS SET
2152 024634 104144          EMT      144
2153

```

:VERIFY THAT COMPOSITE ERROR SETS FOR EACH BIT OF RMR1

```

2154
2155 024636          20$:
2156 024636 012702 000001  MOV      #1,R2          :INITIALIZE TEST PATTERN
2157

```

:WRITE THE TEST PATTERN AND VERIFY THAT ERR IS SET

```

2158
2159 024642          30$:
2160 024642 004737 055156          JSR      PC,CNTCLR   ,GO CLEAR CONTROLLER
2161 024646 012760 000001 000024  MOV      #DMD,RMR1(R0) ,LOAD RMR1
2162 024654 012760 000000 000014  MOV      #0,RMR1(R0)  ,LOAD RMR1
2163 024662 012760 000000 000042  MOV      #0,RMR2(R0)  ,LOAD RMR2
2164 024670 010260 000014          MOV      R2,RMR1(R0) ,LOAD RMR1
2165 024674 016037 000012 001142  MOV      RMD5(R0),$BDDAT ,STORE RMD5 AT $BDDAT

```

```

2166 024702 042737 137777 001142      BIC      #^CERR,$BDDAT
2167 024710 001005                      BNF      40$      ;BRANCH IF COMPOSITE ERROR SET
2168 024712 010237 001174      MOV      R2,$STMP0      ;SAVE RMR1 TEST PATTERN
2169 024716 005037 001176      CLR      $TMP1          ;SAVE RMR2 TEST PATTERN
2170 024722 104145      EMT      145
2171
2172      ;ADVANCE THE TEST PATTERN FOR RMR1
2173 024724      40$:
2174 024724 006302      ASL      R2
2175 024726 001345      BNE      30$      ;CONTINUE IF TEST NOT DONE
2176
2177      ;VERIFY THAT COMPOSITE ERROR SETS FOR EACH BIT OF RMR2
2178 024730      50$:
2179 024730 012702 000001      MOV      #1,R2          ;INITIALIZE TEST PATTERN
2180 024734 012737 010000 001444      MOV      #FMT16,RMOF0   ;SET 16 BIT FORMAT
2181
2182      ;WRITE THE TEST PATTERN AND VERIFY THAT ERR IS SET
2183 024742      60$:
2184 024742 004737 055156      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
2185 024746 012760 000001 000024      MOV      #DMD,RMR1(RO)  ;LOAD RMR1
2186 024754 012760 000000 000014      MOV      #0,RMR1(RO)    ;LOAD RMR1
2187 024762 012760 000000 000042      MOV      #0,RMR2(RO)    ;LOAD RMR2
2188 024770 010260 000042      MOV      R2,RMR2(RO)    ;LOAD RMR2
2189 024774 013760 001444 000032      MOV      RMOF0,RMOF(RO) ;LOAD RMOF
2190 025002 016037 000012 001142      MOV      RMOF($RO),$BDDAT ;STORE RMOF AT $BDDAT
2191 025010 042737 137777 001142      BIC      #^CERR,$BDDAT
2192 025016 012737 040000 001140      MOV      #ERR,$GDDAT    ;SETUP EXPECTED VALUE FOR COMP ERROR
2193 025024 032702 001527      BIT      #XMR2,R2
2194 025030 001402      BEQ      65$      ;BRANCH IF TEST BIT IS A USED BIT
2195 025032 005037 001140      CLR      $GDDAT        ;TEST BIT IS NOT USED - ERR SHOULD BE 0
2196 025036 023737 001140 001142 65$      CMP      $GDDAT,$BDDAT
2197 025044 001405      BEQ      70$      ;BRANCH IF COMP ERROR IS OK
2198 025046 005037 001174      CLR      $STMP0        ;SAVE RMR1 TEST PATTERN
2199 025052 010237 001176      MOV      R2,$TMP1      ;SAVE RMR2 TEST PATTERN
2200 025056 104145      EMT      145
2201
2202      ;ADVANCE THE TEST PATTERN FOR RMR2
2203 025060      70$:
2204 025060 006302      ASL      R2
2205 025062 001327      BNE      60$      ;CONTINUE IF TEST NOT DONE
2206 025064      80$:
2207
2208      ;*****
      ;*TEST 43      WRITE GO TEST
      ;*****
2209 025064      TST43:
2210 025064 000004      SCOPE                      ;SCOPE CALL
2211 025066 000240      NOP
2212 025070 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
2213 025074 013700 001276      MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
2214 025100 013701 001466      MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
2215 025104 012737 000043 001226      MOV      #43,$TESTN     ;;SET TEST NUMBER IN API MAIL BOX
2216
2217 025112 010037 001136      MOV      R0,$BADDR      ;COPY RMCS1 ADDRESS
2218 025116 005002      CLR      R2              ;INITIALIZE FUNCTION CODE
2219

```

```

2213 ;CLEAR THE MASSBUS, SET DIAGNOSTIC MODE AND ENABLE DEBUG CLOCK
2214 10$:
2215 025120 004737 055156 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
2216 025124 012760 000001 000024 MOV #DMD,RMR1(R0) ;LOAD RMR1
2217 025132 012760 041001 000024 MOV #DMD!DBEN!MUR,RMR1(R0) ;LOAD RMR1
2218 025140 012760 000000 000014 MOV #0,RMR1(R0) ;LOAD RMR1
2219 025146 012760 000000 000042 MOV #0,RMR2(R0) ;LOAD RMR2
2220
2221 ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1, VERIFY GO IS SET
2222 025154 010203 MOV R2,R3 ;SETUP FUNCTION CODE
2223 025156 052703 000001 BIS #GO,R3
2224 025162 010360 000000 MOV R3,RMCS1(R0) ;LOAD RMCS1
2225 025166 016037 000000 001142 MOV RMCS1(R0),SBDDAT ;STORE RMCS1 AT SBDDAT
2226 025174 032737 000001 001142 BIT #GO,SBDDAT
2227 025202 001007 BNE 20$ ;BRANCH IF GO IS SET
2228
2229 ;REPORT THE ERROR-CANT SET GO WITH THIS FUNCTION CODE
2230 025204 042737 177700 001142 BIC #*CFNCMSK,SBDDAT
2231 025212 010337 001140 MOV R3,$GDDAT ;SAVE FUNCTION CODE
2232 025216 104146 EMT 146
2233 025220 000405 BR 30$
2234
2235 ;ADVANCE R2 TO THE NEXT FUNCTION CODE
2236 20$:
2237 025222 062702 000002 ADD #2,R2
2238 025226 022702 000076 CMP #1LF76,R2
2239 025232 103332 BHI 10$
2240
2241 30$ ;END OF TEST
2242
2243 ;*****
;TEST 44 BRANCH MULTIPLEXOR TEST
;*****
;*****
;*****
2244 025234 000004 SCOPE ;SCOPE CALL
2245 025234 000240 NOP
2246 025240 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
2247 025244 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
2248 025250 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
2249 025254 012737 000044 001226 MOV #44,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
2250
2251 MOV R0,$BDADR ;COPY REGISTER ADDRESS
2252 ADD #RMR2,$BDADR
2253 MOV #100$,R2 ;INITIALIZE TABLE POINTER
2254
2255 ;CLEAR THE MASSBUS AND SET DEBUG CLOCK ENABLE
2256 10$:
2257 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
2258 025300 004737 055156 MOV #DMD,RMR1(R0) ;LOAD RMR1
2259 025304 012760 000001 000024 MOV #DMD!DBEN!MUR,RMR1(R0) ;LOAD RMR1
2260 025312 012760 041001 000024 MOV #0,RMR1(R0) ;LOAD RMR1
2261 025320 012760 000000 000014 MOV #0,RMR2(R0) ;LOAD RMR2
2262 025324 012760 000000 000042 MOV #0,RMR2(R0) ;LOAD RMR2
2263
2264 ;THE TEST BIT SHOULD BE ONE BECAUSE THE ADDRESS IS ALL ONES WHEN
2265 ;THE COMMAND SEQUENCER IS INITIALIZED.
2266 025334 016037 060040 001142 MOV RMR2(R0),SBDDAT ;STORE RMR2 AT SBDDAT
    
```

```

2260 025342 032737 010000 001142 BIT #1ST,$BDDAT
2261 025350 001010 BNE 15$ :BRANCH IF TEST BIT IS ON
2262 025352 042737 167777 001142 BIC #^C1ST,$BDDAT :SETUP FOR ERROR TYPE
2263 025360 012737 010000 001140 MOV #TST,$GDDAT
2264 025366 104147 EMT 147
2265 025370 000452 BR 40$ :SKIP REST OF TEST
    
```

:GET THE FUNCTION CODE FROM THE TABLE AND TRANSFER IT TO THE DEVICE.  
 :THEN STEP THE COMMAND SEQUENCER ACCORDING TO THE TABLE.

```

2269 025372 15$:
2270 025372 111203 MOV# (R2),R3
2271 025374 052703 000001 BIS #GO,R3
2272 025400 042703 177700 BIC #^CFNCSK,R3 :R3=FUNCTION CODE, GO BIT
2273 025404 010360 000000 MOV R3,RMCS1(R0) :LOAD RMCS1
2274 025410 010337 001174 MOV R3,$TMP0 :SAVE R3 FOR ERROR MSG
2275
2276 025414 116203 000001 MOV# 1(R2),R3 :GET CLOCK COUNT IN R3
2277 025420 042703 177400 BIC #^C377,R3
    
```

20\$:  
 :LOAD RMPR1  
 :LOAD RMPR1  
 :DECREMENT CLOCK COUNT  
 :ISSUE CLOCKS TILL ZERO

```

2279 025424 012760 141001 000024 MOV #DMD!DBEN!MUR!DBCK,RMPR1(R0)
2280 025432 012760 041001 000024 MOV JMD!DBEN!MUR,RMPR1(R0)
2281 025440 005303 DEC R3
2282 025442 001370 BNE 20$
    
```

:GET THE TEST BIT AND COMPARE IT WITH THE TABLE ENTRY

```

2283
2284
2285 025444 016037 000040 001142 MOV RMPR2(R0),SBDDAT :STORE RMPR2 AT SBDDAT
2286 025452 042737 167777 001142 BIC #^CTST,$BDDAT
2287 025460 016237 000002 001140 MOV 2(R2),$GDDAT
2288 025466 023737 001140 001142 CMP $GDDAT,$BDDAT
2289 025474 001402 BEQ 30$ :BRANCH IF TEST BIT OK
2290 025476 104150 EMT 150
2291 025500 000406 BR 40$ :SKIP REST OF TEST
    
```

:MOVE THE TABLE POINTER AND CONTINUE IF NEXT ENTRY POSITIVE

```

2292
2293
2294 025502 30$:
2295 025502 062702 000004 ADD #4,R2
2296 025506 105762 000001 TSTB 1(R2)
2297 025512 100401 BMI 40$ :BRANCH IF DONE TEST
2298 025514 000671 BR 10$ :REPEAT TEST
2299 025516 000436 BR 200$ :JUMP OVER TABLE
    
```

TABLE OF FUNCTION CODES, CLOCK COUNTS, AND TEST BITS

```

2300
2301
2302 025520 100$:
2303 025520 000 .BYTE NOP :MUX ADDRESS=DATA COMMAND
2304 025521 001 .BYTE 1
2305 025522 000000 .WORD 0 :TEST BIT=0
2306
2307 025524 000 .BYTE NOP :MUX ADDRESS=UNIT READY
2308 025525 002 .BYTE 2
2309 025526 000000 .WORD 0 :TEST BIT=0
2310
2311 025530 010 .BYTE DRVCL :MUX ADDRESS=F4
2312 025531 001 .BYTE 1
2313 025532 010000 .WORD TST :TEST BIT=1
2314
2315 025534 050 .BYTE JCD :MUX ADDRESS=F4
2316 025535 001 .BYTE 1
    
```

2317	025536	000000	.WORD	0	.TEST BIT=0
2318					
2319	025540	012	.BYTE	MLEASE	:MUX ADDRESS=F4
2320	025541	001	.BYTE	1	
2321	025542	010000	.WORD	TST	:TEST BIT=1
2322					
2323	025544	052	.BYTE	WCH	:MUX ADDRESS=F4
2324	025545	001	.BYTE	1	
2325	025546	000000	.WORD	0	.TEST BIT=0
2326					
2327	025550	020	.BYTE	RIP	:MUX ADDRESS=F4
2328	025551	001	.BYTE	1	
2329	025552	010000	.WORD	TST	:TEST BIT=1
2330					
2331	025554	060	.BYTE	WD	:MUX ADDRESS=F4
2332	025555	001	.BYTE	1	
2333	025556	000000	.WORD	0	:TEST BIT=0
2334					
2335	025560	022	.BYTE	PAKACK	:MUX ADDRESS=F4
2336	025561	001	.BYTE	1	
2337	025562	010000	.WORD	TST	:TEST BIT=1
2338					
2339	025564	062	.BYTE	WH	:MUX ADDRESS=F4
2340	025565	001	.BYTE	1	
2341	025566	000000	.WORD	0	:TEST BIT=0
2342					
2343	025570	030	.BYTE	SEARCH	:MUX ADDRESS=F4
2344	025571	001	.BYTE	1	
2345	025572	010000	.WORD	TST	:TEST BIT=1
2346					
2347	025574	070	.BYTE	RD	:MUX ADDRESS=F4
2348	025575	001	.BYTE	1	
2349	025576	000000	.WORD	0	:TEST BIT=0
2350					
2351	025600	032	.BYTE	ILF32	:MUX ADDRESS=F4
2352	025601	001	.BYTE	1	
2353	025602	010000	.WORD	TST	:TEST BIT=1
2354					
2355	025604	072	.BYTE	RH	:MUX ADDRESS=F4
2356	025605	001	.BYTE	1	
2357	025606	000000	.WORD	0	:TEST BIT=0
2358					
2359	025610	000	.BYTE		:END OF TABLE
2360	025611	377	.BYTE	-1	
2361	025612	000000	.WORD		
2362	025614				
2363					
2364					

2008: :END OF TEST

\*\*\*\*\*  
 :\*TEST 45 SET/RESET GO TEST

\*\*\*\*\*  
 TST45:

025614	000004		SCOPE		:SCOPE CALL
025616	000240		NOP		
025620	012706	001100	MOV	#STACK, SP	:LOAD THE STACK POINTER
025624	013700	001276	MOV	SBASE, R0	:R0 = UNIBUS ADDRESS
025630	013701	001466	MOV	TSTQUE, R1	:R1 = POINTER TO DEVICE

```

2365 025634 012737 000045 001226      MOV    #45,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
2366 025642 012702 026256              MOV    #200$,R2        ;INITIALIZE FUNCTION CODE POINTER
2367
2368 ;CLEAR, THEN SET DIAGNOSTIC MODE, CLEAR COMPOSITE ERROR, SET MEDIUM
2369 ;ON LINE AND ENABLE DEBUG CLOCK
2370 10$:
2371 025646 004737 055156              JSR    PC,$HNTCLR      ;GO CLEAR CONTROLLER
2372 025652 012760 000001 000024      MOV    #DM0,$RMR1(RO) ;LOAD RMR1
2373 025660 012760 041001 000024      MOV    #DM0,$MR!DBEN,$RMR1(RO) ;LOAD RMR1
2374 025666 012760 000000 000014      MOV    #0,$RMR1(RO)   ;LOAD RMR1
2375 025674 012760 000000 000042      MOV    #0,$RMR2(RO)   ;LOAD RMR2
2376
2377 ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1 AND VERIFY GO IS SET
2378 025702 111203              MOV    (R2),R3        ;GET FUNCTION CODE
2379 025704 042703 177701              BIC    #*CILF76,R3    ;CLEAR UNUSED BITS
2380 025710 052703 000001              BIS    #GO,R3         ;SET GO
2381 025714 010360 000000              MOV    R3,$RMCS1(RO)  ;LOAD RMCS1
2382 025720 016037 000000 001142      MOV    $RMCS1(RO),$SDDAT ;STORE RMCS1 AT $SDDAT
2383 025726 032737 000001 001142      BIT    #GO,$SDDAT
2384 025734 001011              BNE    20$            ;BRANCH IF GO IS SET
2385 025736 042737 177700 001142      BIC    #*CFNCMSK,$SDDAT
2386 025744 010337 001140              MOV    R3,$SDDAT     ;SAVE EXPECTED RESULT
2387 025750 010037 001136              MOV    R0,$SDDADR    ;COPY REGISTER ADDRESS
2388 025754 104151              EMT    151
2389 025756 000536              BR     100$
2390
2391 ;GET READY STATUS AND VERIFY THAT IT IS THE COMPLEMENT OF GO
2392 20$:
2393 025760 005037 001140              CLR    $SDDAT         ;EXPECT DRY TO BE OFF
2394 025764 032737 000001 001142      BIT    #GO,$SDDAT    ;WAS GO SET??
2395 025772 001003              BNE    30$           ;YES!!
2396 025774 012737 000200 001140      MOV    #DRY,$SDDAT   ;GO WAS NOT SET, DRY SHOULD BE
2397 026002 016037 000012 001142      MOV    $RMS(RO),$SDDAT ;STORE RMS AT $SDDAT
2398 026010 042737 177577 001142      BIC    #*CDRY,$SDDAT
2399 026016 023737 001140 001142      CMP    $SDDAT,$SDDAT
2400 026024 001406              BEQ    40$           ;BRANCH IF DRY IS OK
2401 026026 010037 001136              MOV    R0,$SDDADR    ;COPY REGISTER ADDRESS
2402 026032 062737 000012 001136      ADD    #RMS,$SDDADR
2403 026040 104152              EMT    152
2404
2405 ;STEP THE DEBUG CLOCK AND VERIFY THAT GO REMAINS SET
2406 40$:
2407 026042 116204 000001              MOV    1(R2),R4       ;GET NUMBER OF CLOCK CYCLES
2408 026046 042704 177400              BIC    #*C377,R4
2409 026052
2410 026052 012760 141001 000024      MOV    #DM0,$MR!DBEN!DBCK,$RMR1(RO) ;LOAD RMR1
2411 026060 012760 041001 000024      MOV    #DM0,$MR!DBEN,$RMR1(RO) ;LOAD RMR1
2412 026066 016037 000000 001142      MOV    $RMCS1(RO),$SDDAT ;STORE RMCS1 AT $SDDAT
2413 026074 042737 177700 001142      BIC    #*CFNCMSK,$SDDAT ;CLEAR UNUSED BITS
2414 026102 010037 001136              MOV    R0,$SDDADR    ;SETUP REGISTER ADDRESS
2415 026106 010337 001140              MOV    R3,$SDDAT     ;SAVE EXPECTED RESULT
2416
2417 ;DECREMENT CLOCK COUNT AND EXIT LOOP IF ZERO
2418 026112 005304              DEC    R4
2419 026114 001406              BEQ    60$
  
```

2421	026116	032737	000001	001142	BIC	#GO,\$GDDAT	: IS GO STILL SET??
2422	026124	001352			BNE	50\$	: YES
2424	026126	104153			EMT	153	
2423	026130	000451			BR	100\$	: OUT OF SYNC=SKIP TO NEXT
2424							
2425							
2426	026132						
2427	026132	032737	000001	001142	BIC	#GO,\$BDDAT	: IS GO RESET??
2428	026140	001405			BEG	70\$	: YES
2429	026142	042737	000001	001140	BIC	#GO,\$GDDAT	: SETUP EXPECTED RESULT
2430	026150	104154			EMT	154	
2431	026152	000440			BR	100\$	
2432	026154						
2433	026154	012737	000200	001140	MOV	#DRY,\$GDDAT	: EXPECT DRIVE READY TO BE SET
2434	026162	032737	000001	001142	BIC	#GO,\$BDDAT	: DID GO RESET??
2435	026170	001402			BEG	80\$	: YES!!
2436	026172	005037	001140		CLR	\$GDDAT	: GO IS SET-
2437	026176						
2438	026204	042737	000012	001142	MOV	RMD5(R0),\$BDDAT	: STORE RMD5 AT \$BDDAT
2439	026212	010037	001136		BIC	#CDRY,\$BDDAT	
2440	026216	062737	000012	001136	MOV	R0,\$BDDADR	: COPY REGISTER ADDRESS
2441	026224	023737	001140	001142	ADD	#RMD5,\$BDDADR	
2442	026232	001401			JMP	\$GDDAT,\$BDDAT	: IS DRIVE READY OK??
2443	026234	104152			BEG	90\$	: YES!!
2444					EMT	152	
2445							
2446	026236						
2447	026236	062702	000002		ADD	#2,R2	: MOVE TABLE POINTER
2448	026242	105762	000001		TSTB	1(R2)	: END OF TABLE??
2449	026246	100402			BMI	100\$	: YES!!
2450	026250	000137	025646		JMP	10\$	: TEST THIS FUNCTION CODE
2451	026254	000423			BR	300\$	: GO TO NEXT TEST
2452							
2453							
2454	026256						
2455	026256	002			.BYTE	ILF02	: ILLEGAL FUNCTION CODE #2
2456	026257	001			.BYTE	1	
2457							
2458	026260	004			.BYTE	SEEK	: SEEK COMMAND
2459	026261	001			.BYTE	1	
2460							
2461	026262	006			.BYTE	RECAL	: RECALIBRATE COMMAND
2462	026263	001			.BYTE	1	
2463							
2464	026264	014			.BYTE	OFFSET	: OFFSET COMMAND
2465	026265	001			.BYTE	1	
2466							
2467	026266	016			.BYTE	RTC	: RETURN TO CENTER LINE COMMAND
2468	026267	001			.BYTE	1	
2469							
2470	026270	024			.BYTE	ILF24	: ILLEGAL FUNCTION CODE #24
2471	026271	001			.BYTE	1	
2472							
2473	026272	026			.BYTE	ILF26	: ILLEGAL FUNCTION CODE #26
2474	026273	001			.BYTE	1	
2475							



2476	026274	034	.BYTE	ILF34	:ILLEGAL FUNCTION CODE #34
2477	026275	001	.BYTE	1	
2478					
2479	026276	036	.BYTE	ILF36	:ILLEGAL FUNCTION CODE #36
2480	026277	001	.BYTE	1	
2481					
2482	026300	042	.BYTE	ILF42	:ILLEGAL FUNCTION CODE #42
2483	026301	001	.BYTE	1	
2484					
2485	026302	044	.BYTE	ILF44	:ILLEGAL FUNCTION CODE #44
2486	026303	001	.BYTE	1	
2487					
2488	026304	046	.BYTE	ILF46	:ILLEGAL FUNCTION CODE #46
2489	026305	001	.BYTE	1	
2490					
2491	026306	054	.BYTE	ILF54	:ILLEGAL FUNCTION CODE #54
2492	026307	001	.BYTE	1	
2493					
2494	026310	056	.BYTE	ILF56	:ILLEGAL FUNCTION CODE #56
2495	026311	001	.BYTE	1	
2496					
2497	026312	064	.BYTE	ILF64	:ILLEGAL FUNCTION CODE #64
2498	026313	001	.BYTE	1	
2499					
2500	026314	066	.BYTE	ILF66	:ILLEGAL FUNCTION CODE #66
2501	026315	001	.BYTE	1	
2502					
2503	026316	074	.BYTE	ILF74	:ILLEGAL FUNCTION CODE #74
2504	026317	001	.BYTE	1	
2505					
2506	026320	076	.BYTE	ILF76	:ILLEGAL FUNCTION CODE #76
2507	026321	001	.BYTE	1	
2508					
2509	026322	000	.BYTE		:END OF TABLE
2510	026323	377	.BYTE	-1	
2511					
2512	026324				300\$: :END OF TEST
2513					
2514					

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

\*\*\*\*\*  
 TST44:  
 \*\*\*\*\*

026324					
026324	000004		SCOPE		:SCOPE CALL
026326	000240		NOP		
026330	012706	001100	MOV	#STACK,SP	:LOAD THE STACK POINTER
026334	013700	001276	MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
026340	013701	001466	MOV	TSTQUE,R1	:R1 = POINTER TO DEVICE
026344	012737	000046 001226	MOV	#46,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
2515					
2516	026352	012702 026610	MOV	#100\$,R2	:INITIALIZE TABLE POINTER
2517	026356	010037 001136	MOV	R0,\$BDADR	:COPY RMCS1 ADDRESS
2518					
2519					
2520	026362				:CLEAR MASSBUS, THEN SET MEDIUM ON LINE AND ENABLE DEBUG CLOCK
2521	026362	004737 055156	JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
2522	026366	012760 000001 000024	MOV	#DMD,RMMR1(R0)	:LOAD RMMR1

```

2523 026374 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
2524 026402 012760 000000 000014      MOV      #0,RMER1(RO) ;LOAD RMER1
2525 026410 012760 000000 000042      MOV      #0,RMER2(RO) ;LOAD RMER2
2526
2527      ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1, VERIFY GO IS SET
2528 026416 111203      MOV      (R2),R3 ;GET FUNCTION CODE FROM
2529 026420 042703 177701      BIC      #^C1LF76,R3 ;TABLE AND SET GO
2530 026424 052703 000001      BIS      #GO,R3
2531 026430 010337 001140      MOV      R3,$GDDAT ;SAVE FUNCTION CODE FOR MSG
2532 026434 C.0360 000000      MOV      R3,RMCS1(RO) ;LOAD RMCS1
2533 026440 016037 000000 00:142      MOV      RMCS1(RO),$BDDAT ;STORE RMCS1 AT $BDDAT
2534 026446 032737 000001 001142      BIT      #GO,$BDDAT
2535 026454 001005      BNE      20$ ;BRANCH IF GO IS SET
2536 026456 042737 177700 001142      BIC      #^CFNCMSK,$BDDAT
2537 026464 104151      EMT      151
2538 026466 000447      BR       60$ ;OUT OF SYNC-SKIP
2539
2540      ;GET THE NUMBFR OF CLOCK CYCLES FROM THE TABLE, SAVE EXPECTED STATUS
2541 026470      20$:
2542 026470 116204 000001      MOV      1(R2),R4 ;R4=CLOCK COUNT
2543 026474 042704 177400      BIC      #^C377,R4
2544
2545      ;STEP THE DEBUG CLOCK AND VERIFY GO STATUS ON UNTIL CLOCK COUNT EXPIRES.
2546 026500      30$:
2547 026500 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMPR1(RO) ;LOAD RMPR1
2548 026506 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
2549 026514 016037 000000 001142      MOV      RMCS1(RO),$BDDAT ;STORE RMCS1 AT $BDDAT
2550 026522 042737 177700 001142      BIC      #^CFNCMSK,$BDDAT
2551 026530 005304      DEC      R4
2552 026532 001406      BEQ      40$ ;BRANCH IF GO SHOULD BE OFF
2553 026534 032737 000001 001142      BIT      #GO,$BDDAT
2554 026542 001356      BNE      30$ ;CONTINUE IF GO IS ON
2555 026544 104153      EMT      153
2556 026546 000417      BR       60$ ;OUT OF SYNC-SKIP
2557
2558      ;VERIFY THAT GO RESET AT END1
2559 026550      40$:
2560 026550 032737 000001 001142      BIT      #GO,$BDDAT ;DID GO RESET??
2561 026556 001405      BEQ      50$ ;YES!!
2562 026560 042737 000001 001140      BIC      #GO,$GDDAT
2563 026566 104154      EMT      154
2564 026570 000406      BR       60$
2565
2566      ;GET THE NEXT FUNCTION CODE FROM THE TABLE
2567 026572      50$:
2568 026572 062702 000002      ADD      #2,R2
2569 026576 105762 000001      TSTB    1(R2)
2570 026602 100401      BMI      60$ ;BRANCH IF END OF TABLE
2571 026604 000666      BR       10$ ;TEST THIS FUNCTION CODE
2572 026606 000404      BR       200$ ;JUMP OVER TABLE
2573
2574      ;TABLE OF FUNCTION CODES AND CLOCK COUNTS USED DURING TEST
2575 026610      100$:
2576 026610 012      .BYTE   RELEASE ;RELEASE COMMAND
2577 026611 002      .BYTE   2
2578
2579 026612 030      .BYTE   SEARCH ;SEARCH COMMAND
  
```

2580 026613 002  
2581  
2582 026614 032  
2583 026615 002  
2584  
2585 026616 000  
2586 026617 377  
2587 026620  
2588  
2589

.BYTE 2  
.BYTE 1LF32 ;ILLEGAL FUNCTION #32  
.BYTE 2  
.BYTE ;END OF TABLE  
.BYTE -1 ;END OF TEST

200\$:

\*\*\*\*\*  
\*TEST 47 SET PULSE TEST  
\*\*\*\*\*

\*\*\*\*\*  
\*TEST 47:  
\*\*\*\*\*

026620  
026620 00C004  
026622 000240  
026624 012706 001100  
026630 013700 001276  
026634 013701 001466  
026640 012737 000047 001226  
2590  
2591 026646 010037 001136  
2592 026652 062737 000024 001136  
2593 026660 012702 027116  
2594  
2595  
2596 026664  
2597 026664 004737 055156  
2598 026670 012760 000001 000024  
2599 026676 012760 041001 000024  
2600 026704 012760 000000 000014  
2601 026712 012760 000000 000042  
2602  
2603  
2604 026720 016037 000024 001142  
2605 026726 042737 17677 001142  
2606 026734 001404  
2607 026736 005037 001140  
2608 026742 104155  
2609 026744 000463  
2610  
2611  
2612 026746  
2613 026746 111203  
2614 026750 052703 000001  
2615 026754 042703 177700  
2616 026760 010360 000900  
2617 026764 010337 001174  
2618  
2619  
2620 026770 116203 000001  
2621 026774 042703 177400  
2622  
2623  
2624 027000 016204 000002  
2625  
2626

SCOPE ;SCOPE CALL  
NOP  
MOV #STACK,SP ;LOAD THE STACK POINTER  
MOV \$BASE,R0 ;R0 = UNIBUS ADDRESS  
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE  
MOV #47,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX  
MOV R0,\$BDADR ;COPY REG ADDRESS FOR MSG  
ADD #RMR1,\$BDADR  
MOV #100\$,R2 ;INITIALIZE TABLE POINTER  
;CLEAR THE MASS BUS, ENABLE DEBUG CLOCK, AND RESET ERROR REGISTERS  
10\$:  
JSR PC,CNTCLR ;GO CLEAR CONTROLLER  
MOV #DMD,RMR1(R0) ;LOAD RMR1  
MOV #DMD!DBEN!MUR,RMR1(R0) ;LOAD RMR1  
MOV #0,RMR1(R0) ;LOAD RMR1  
MOV #0,RMR2(R0) ;LOAD RMR2  
;VERIFY THAT CONTINUE, "CONT" IS RESET AFTER CLEAR  
MOV RMR1(R0),\$BDDAT ;STORE RMR1 AT \$BDDAT  
BIC #CCONT,\$BDDAT  
BEQ 20\$ ;BRANCH IF CONT WAS CLEARED  
CLR \$GDDAT ;FOR ERROR MSG  
EMT 155  
BR 70\$  
;GET THE FUNCTION CODE FROM THE TABLE AND TRANSFER IT TO RMC1  
20\$:  
MOVB (R2),R3  
BIS #GO,R3  
BIC #CFNCMSK,R3 ;R3=FUNCTION CODE AND GO  
MOV R3,RMC1(R0) ;LOAD RMC1  
MOV R3,\$TMP0 ;SAVE FUNCTION CODE FOR MSG  
;GET THE CLOCK COUNT FROM THE TABLE  
MOVB 1(R2),R3  
BIC #C377,R3  
;GET THE BIT STREAM FOR CONTINUE FROM THE TABLE  
MOV 2(R2),R4  
;STEP THE COMMAND SEQUENCER AND VERIFY CONTINUE STATUS

```

147 SET PULSE TEST
2627 027004
2628 027004 012760 141001 000024 30S: MOV #DMD:DBEN:PIR:DECK,RPIR1(RG) ;LOAD RPIR1
2629 027012 012760 041001 000024 MOV #DMD:DBEN:PIR,RPIR1(R0) ;LOAD RPIR1
2630 027020 016037 000024 001142 MOV RPIR1(R0),SBDDAT ;STORE RPIR1 AT SBDDAT
2631 027026 042737 177677 001142 BIC #CCONT,SBDDAT
2632 027034 005037 001140 CLR $GDDAT ;GENERATE EXPECTED CONTINUE
2633 027040 032704 000001 BIT #BIT0,R4
2634 027044 001403 BEQ 40S
2635 027046 012737 000100 001140 MOV #CONT,$GDDAT
2636 027054 023737 001140 001142 40S: CMP $GDDAT,SBDDAT
2637 027062 001402 BEQ 50S ;BRANCH IF CONTINUE IS OK
2638 027064 104156 EMT 156
2639 027066 000412 BR 70S ;SKIP
2640
2641 ;DECREMENT CLOCK COUNT AND SHIFT BIT STREAM
2642 027070 50S:
2643 027070 005303 DEC R3
2644 027072 001402 BEQ 60S ;BRANCH IF CLOCK COUNT EXPIRED
2645 027074 006204 ASR R4 ;SHIFT TO NEXT CONTINUE BIT
2646 027076 000742 BR 30S ;TEST NEXT CLOCK CYCLE
2647
2648 ;ADVANCE TABLE POINTER-EXIT IF DONE
2649 027100 60S:
2650 027100 062702 000004 ADD #4,R2
2651 027104 105762 000001 TSTB 1(R2)
2652 027110 100401 BMI 70S ;EXIT IF CLOCK COUNT NEGATIVE
2653 027112 000654 BR 10S ;CONTINUE TEST
2654 027114 000442 BR 200S ;JUMP OVER TABLE
2655
2656 ;TABLE OF FUNCTION CODE^ CLOCK COUNTS AND CONTINUE BITS FOR TEST
2657 027116 100S:
2658 027116 000 .BYTE NOP ;NOP COMMAND
2659 027117 004 .BYTE 4 ;4 CLOCKS
2660 027120 000000 .WORD ^B0000 ;CONTINUE=0000
2661
2662 027122 002 .BYTE ILF02 ;ILLEGAL FUNCTION 2
2663 027123 002 .BYTE 2
2664 027124 000000 .WORD ^B00
2665
2666 027126 004 .BYTE SEEK ;SEEK COMMAND
2667 027127 002 .BYTE 2
2668 027130 000000 .WORD ^B00
2669
2670 027132 006 .BYTE RECAL ;RECALIBRATE COMMAND
2671 027133 002 .BYTE 2
2672 027134 000000 .WORD ^B00
2673
2674 027136 010 .BYTE DRVCLR ;DRIVE CLEAR COMMAND
2675 027137 002 .BYTE 2
2676 027140 000001 .WORD ^B01
2677
2678 027142 012 .BYTE RELEASE ;RELEASE COMMAND
2679 027143 003 .BYTE 3
2680 027144 000000 .WORD ^B000
2681
2682 027146 014 .BYTE OFFSET ;OFFSET COMMAND
2683 027147 002 .BYTE 2

```

```

2684 027150 000000 .WORD ^B00
2685
2686 027152 016 .BYTE RTC ;RETURN TO CENTER COMMAND
2687 027153 002 .BYTE 2
2688 027154 000000 .WORD ^B00
2689
2690 027156 020 .BYTE RIP ;READ IN PRESET COMMAND
2691 027157 004 .BYTE 4
2692 027160 000016 .WORD ^B1110
2693
2694 027162 022 .BYTE PAKACK ;PACK ACKNOWLEDGE
2695 027163 004 .BYTE 4
2696 027164 000016 .WORD ^B1110
2697
2698 027166 024 .BYTE ILF24 ;ILLEGAL FUNCTION 24
2699 027167 022 .BYTE 2
2700 027170 000000 .WORD ^B00
2701
2702 027172 026 .BYTE ILF26 ;ILLEGAL FUNCTION 26
2703 027173 002 .BYTE 2
2704 027174 000000 .WORD ^B00
2705
2706 027176 030 .BYTE SEARCH ;SEARCH COMMAND
2707 027177 003 .BYTE 3
2708 027200 000000 .WORD ^B000
2709
2710 027202 032 .BYTE ILF32 ;ILLEGAL FUNCTION 32
2711 027203 003 .BYTE 3
2712 027204 000000 .WORD ^B000
2713
2714 027206 034 .BYTE ILF34 ;ILLEGAL FUNCTION 34
2715 027207 002 .BYTE 2
2716 027210 000000 .WORD ^B00
2717
2718 027212 036 .BYTE ILF36 ;ILLEGAL FUNCTION 36
2719 027213 002 .BYTE 2
2720 027214 000000 .WORD ^B00
2721
2722 027216 000 .BYTE ;END OF TABLE
2723 027217 377 .BYTE -1
2724 027220 000000 .WORD
2725
2726 027222 2008: ;END OF TEST
2727
2728

```

```

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

```

```

027222
027222 000004 TST50: SCOPE ;SCOPE CALL
027224 000240 MOP
027226 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
027232 013700 001273 MOV $BASE,R0 ;R0 = UNIQUE ADDRESS
027236 013701 001766 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
027242 012737 000150 001226 MOV #50,$TESIN ;;SET TEST NUMBER IN APT MAIL BOX
2729
2730 027250 010037 01136 MOV R0,$BDADR ;SETUP REG ADDRESS

```

2731 027254 062737 000042 001136  
 2732 027262 005002  
 2733  
 2734  
 2735 027264  
 2736 027264 004737 055156  
 2737 027270 012760 000001 000024  
 2738 027276 012760 041001 000024  
 2739 027304 012760 000000 000014  
 2740 027312 012760 000000 000042  
 2741 027320 016037 000042 001142  
 2742 027326 042737 16777 001142  
 2743 027334 001404  
 2744 027336 005037 001140  
 2745 027342 104157  
 2746 027344 000444  
 2747  
 2748  
 2749  
 2750 027346  
 2751 027346 010203  
 2752 027350 052703 000001  
 2753 027354 010360 000000  
 2754 027360 012760 141001 000024  
 2755 027366 012760 041001 000024  
 2756 027374 016037 000042 001142  
 2757 027402 042737 167777 001142  
 2758 027410 016237 063640 001140  
 2759 027416 042737 167777 001140  
 2760 027424 023737 001140 001142  
 2761 027432 001403  
 2762 027434 010237 001174  
 2763 027440 104160  
 2764  
 2765  
 2766 027442  
 2767 027442 062702 000002  
 2768 027446 022702 000076  
 2769 027452 103401  
 2770 027454 000703  
 2771  
 2772 027456  
 2773  
 2774

```

ADD #RMER2, $BDADR
CLR R2 ;R2=FUNCTION CODE

:INITIALIZE AND VERIFY THAT IVC STATUS IS ZERO.
10$:
JSR PC, CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD, RMMR1(RO) ;LOAD RMMR1
MOV #DMD!MUR!DBEN, RMMR1(RO) ;LOAD RMMR1
MOV #0, RMER1(RO) ;LOAD RMER1
MOV #0, RMER2(RO) ;LOAD RMER2
MOV RMER2(RO), $BDDAT ;STORE RMER2 AT $BDDAT
BIC #*CIVC, $BDDAT
BEQ 20$ ;BRANCH IF IVC IS ZERO
CLR $GDDAT
EMT 157
BR 40$ ;SKIP REST OF TEST

:LOAD THE FUNCTION CODE WITH GO BIT, STEP THE COMMAND SEQUENCER OFF
:ADDRESS 0 AND VERIFY IVC STATUS.
20$:
MOV R2, R3 ;SETUP FUNCTION CODE
BIS #GO, R3
MOV R3, RMCS1(RO) ;LOAD RMCS1
MOV #DMD!MUR!DBEN!DBCK, RMMR1(RO) ;LOAD RMMR1
MOV #DMD!MUR!DBEN, RMMR1(RO) ;LOAD RMMR1
MOV RMER2(RO), $BDDAT ;STORE RMER2 AT $BDDAT
BIC #*CIVC, $BDDAT ;SET ACTUAL STATUS
MOV INCDTB(R2), $GDDAT ;SETUP EXPECTED STATUS FROM
BIC #*CIVC, $GDDAT ;FUNCTION CODE TABLE
CMP $GDDAT, $BDDAT
BEQ 30$ ;BRANCH IF IVC IS OK
MOV R2, STMP0 ;SAVE FUNCTION CODE FOR MSG
EMT 160

:ADVANCE FUNCTION CODE AND REPEAT TEST IF NOT DONE
30$:
ADD #2, R2
CMP #1LF76, R2
BLO 40$ ;BRANCH IF DONE TEST
BR 10$

40$:
;END OF TEST

```

027456  
 027456 000004  
 027460 000240  
 027462 012706 001100  
 027466 013700 001276  
 027472 013701 001466  
 027476 012737 000051 001226  
 2775  
 2776 027504 010037 001136  
 2777 027510 062737 000042 001136

```

*****
: *TEST 51 SET LSC TEST
*****
TST51:
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 #51, STSTN ;;SET TEST NUMBER IN APT MAIL BOX

MOV RO, $BDADR
ADD #RMER2, $BDADR

```

```

2778
2779
2780 027516 004737 055156
2781 027522 012760 000001 000024
2782 027530 012760 040001 000024
2783 027536 012760 000000 000014
2784 027544 012760 000000 000042
2785 027552 016037 000042 001142
2786 027560 042737 173777 001142
2787 027566 001403
2788 027570 005037 001140
2789 027574 104161
2790
2791
2792 027576
2793 027604 012760 000001 000000
2794 027612 004777 151720
2795 027616 005737 001534
2796 027622 001375
2797 027624 004777 151710
2798
2799 027630 012760 000001 000024
2800 027636 016037 000042 001142
2801 027644 042737 173777 001142
2802 027652 001004
2803 027654 012737 004000 001140
2804 027662 104162
2805
2806 027664
2807
2808
;INITIALIZE AND VERIFY THAT LOSS OF SYSTEM CLOCK, 'LSC', IS RESET
JSR PC,CNTCLR ;GO CLEAR CONTROLLER
MOV #DMD,RMR1(RO) ;LOAD RMR1
MOV #DMD!DBEN,RMR1(RO) ;LOAD RMR1
MOV #0,RMR1(RO) ;LOAD RMR1
MOV #0,RMR2(RO) ;LOAD RMR2
MOV RMR2(RO),SDDAT ;STORE RMR2 AT SDDAT
BIC #^CLSC,SDDAT
BEQ 10$ ;BRANCH IF LSC IS ZERO
CLR $GDDAT
EIT 161

;WITH DEBUG CLOCK ENABLED, SET GO AND WAIT FOR ONE SHOT TO SET
10$:
MOV #GO,RMCS1(RO) ;LOAD RMCS1
MOV #1,WATCH ;SET WATCHDOG TIMER VALUE
JSR PC,@CLOCK ;START THE CLOCK
20$:
TST WATCH
BNE 20$ ;WAIT FOR WATCH ZERO
JSR PC,@STOPCL ;STOP THE CLOCK

;ONE SHOT SHOULD BE SET-DISABLE DIAGNOSTIC CLOCK AND LSC SHOULD SET.
MOV #DMD,RMR1(RO) ;LOAD RMR1
MOV RMR2(RO),SDDAT ;STORE RMR2 AT SDDAT
BIC #^CLSC,SDDAT
BNE 30$ ;BRANCH IF LSC SET
MOV #LSC,$GDDAT
EIT 162

30$: ;END OF TEST

;*****
;*TEST 52 DECODE TEST
;*****
TST52:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #52,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

2809
2810 027712 005037 001426
2811 027716
2812 027716 004737 030276
2813
2814
2815 027722 013760 001426 000014
2816 027730 012760 000023 000000
2817 027736 012703 000003
2818 027742
2819 027742 012760 141001 000024
2819 027750 012760 041001 000024
2820 027756 005303
2821 027760 001370
5$:
CLR RMR10 ;NO ERROR FIRST TEST
JSR PC,100$ ;INITIALIZE

;EXECUTE A PACK ACKNOWLEDGE AND CHECK VOLUME VALID
MOV RMR10,RMR1(RO) ;LOAD RMR1
MOV #PACACK!GO,RMCS1(RO) ;LOAD RMCS1
MOV #3,R3
10$:
MOV #DMD!MUR!DBEN!DBCK,RMR1(RO) ;LOAD RMR1
MOV #DAJ!MUR!DBEN,RMR1(RO) ;LOAD RMR1
DEC R3
BNE 10$ ;ISSUE NEXT CLOCK IF COUNT NOT 0
    
```

```

2822 027752 016037 000012 001142 MOV RMD5(R0),SBDAT ;STORE RMD5 AT SBDAT
2823 027770 042737 177677 001142 BIC #*CVV,SBDAT
2824 027776 001414 BEQ 20$ ;BRANCH IF VV IS ZERO
2825 030000 005737 001426 TST RMR10
2826 030004 001527 BEQ 70$ ;BRANCH IF VV SHOULD BE SET
2827 030006 005037 001140 CLR $GDDAT ;SETUP ERROR MESSAGE
2828 030012 010037 001136 MOV R0,SBDADR
2829 030016 062737 000012 001136 ADD #RMD5,SBDADR
2830 030024 104163 EMT 163
2831 030026 000522 BR 80$ ;SKIP
2832 030030 20$:
2833 030030 004737 030276 JSR PC,100$ ;INITIALIZE AND SET DIAGNOSTIC MODE
2834
2835 ;EXECUTE A READ IN PRESET AND CHECK VOLUME VALID
2836 030034 013760 001426 000014 MOV RMR10,RMR1(R0) ;LOAD RMR1
2837 030032 012760 000021 000000 MOV #RIP!GO,RMCS1(R0) ;LOAD RMCS1
2838 030050 012703 000003 MOV #3,R3 ;R3=CLOCK COUNT
2839 030054 30$:
2840 030054 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMR1(R0) ;LOAD RMR1
2841 030062 012760 041001 000024 MOV #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
2842 030070 005303 DEC R3
2843 030072 001370 BNE 30$ ;ISSUE NEXT CLOCK IF COUNT NOT ZERO
2844 030074 016037 000012 001142 MOV RMD5(R0),SBDAT ;STORE RMD5 AT SBDAT
2845 030102 042737 177677 001142 BIC #*CVV,SBDAT
2846 030110 001414 BEQ 40$ ;BRANCH IF VOLUME VALID NOT SET
2847 030112 005737 001426 TST RMR10
2848 030116 001462 BEQ 70$ ;BRANCH IF VOLUME VALID SHOULD BE SET
2849 030120 005037 001140 CLR $GDDAT ;SETUP ERROR MESSAGE
2850 030124 010037 001136 MOV R0,SBDADR
2851 030130 062737 000012 001136 ADD #RMD5,SBDADR
2852 030136 104163 EMT 163
2853 030140 000455 BR 80$ ;SKIP
2854 030142 40$:
2855 004737 030276 JSR PC,100$ ;INITIALIZE AND SET DIAGNOSTIC MODE
2856
2857 ;EXECUTE A WRITE CHECK DATA AND CHECK OCCUPIED
2858 030146 013760 001426 000014 MOV RMR10,RMR1(R0) ;LOAD RMR1
2859 030154 012760 000051 000000 MOV #WCD!GO,RMCS1(R0) ;LOAD RMCS1
2860 030162 012703 000002 MOV #2,R3 ;R3=CLOCK COUNT
2861 030166 50$:
2862 030166 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMR1(R0) ;LOAD RMR1
2863 030174 012760 041001 000024 MOV #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
2864 030202 005303 DEC R3
2865 030204 001370 BNE 50$ ;ISSUE NEXT CLOCK IF COUNT NOT ZERO
2866 030206 016037 000024 001142 MOV RMR1(R0),SBDAT ;STORE RMR1 AT SBDAT
2867 030214 042737 077777 001142 BIC #*CCC,SBDAT
2868 030222 001414 BEQ 60$ ;BRANCH IF OCCUPIED IS RESET
2869 030224 005737 001426 TST RMR10
2870 030230 001415 BEQ 70$ ;BRANCH IF OCCUPIED SHOULD BE SET
2871 030232 005037 001140 CLR $GDDAT ;SETUP ERROR MESSAGE
2872 030236 010037 001136 MOV R0,SBDADR
2873 030242 062737 000024 001136 ADD #RMR1,SBDADR
2874 030250 104164 EMT 164
2875 030252 000410 BR 80$
2876 030254 60$:
;VOLUME VALID AND OCCUPIED DID NOT SET-SEE IF COMP ERROR WAS ACTIVE

```



```

2877 030254 005737 001426      TST      RMER10
2878 030260 001005              BNE      80$      ;BRANCH IF COMP ERROR WAS SET
2879
2880                          ;COULD NOT SET VV OR OCCUPIED-SUSPECT DECODE FLOP NOT SETTING
2881 030262 104165      EMT      165
2882
2883                          ;REPEAT TEST WITH COMPOSITE ERROR ACTIVE-VERIFY THAT DECODE FLOP
2884                          ;DOES NOT SET, AS INDICATED BY VOLUME VALID AND OCCUPIED.
2885 030264
2886 030264 012737 040000 001426 7CS:      MOV      #UNSAFE,RMER10 ;USE UNSAFE TO SET COMP ERROR
2887 030272 000511      BR      5$
2888
2889 030274 000510 80$:      BR      200$      ;END OF TEST
2890
2891                          ;*****
2892                          ;SUBROUTINE USED DURING TEST
2893                          ;*****
2894
2895                          ;USING DIAGNOSTIC MODE, RESET VOLUME VALID AND COMPOSITE ERROR.
2896                          ;VERIFY THAT VV, ERR, AND OCC ARE ZERO.
2897 030276
2898 030276 004737 055156      100$:      JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
2899 030302 012760 000001 000024      MOV      #DMD,RMR1(RO) ;LOAD RMR1
2900 030310 012760 041001 000024      MOV      #DMD!RMR1!DBEN,RMR1(RO) ;LOAD RMR1
2901 030316 012760 000000 000014      MOV      #0,RMR1(RO) ;LOAD RMR1
2902 030324 012760 000000 000042      MOV      #0,RMR2(RO) ;LOAD RMR2
2903 030332 005037 001140      CLR      $GDDAT ;SETUP FOR ERROR MSG
2904 030336 010037 001136      MOV      RO,$BDADR
2905 030342 062737 000012 001136      ADD      #RMD5,$BDADR
2906 030350 016037 000012 001142      MOV      RMD5(RO),$BDAT ;STORE RMD5 AT $BDAT
2907 030356 042737 137777 001142      BIC      #^CERR,$BDAT
2908 030364 001402      BEQ      110$      ;BRANCH IF COMP ERROR ZERO
2909 030366 104143      EMT      143
2910 030370 000447      BR      140$
2911 030372
2912 030372 016037 000012 001142 110$:      MOV      RMD5(RO),$BDAT ;STORE RMD5 AT $BDAT
2913 030400 042737 177677 001142      BIC      #^CVV,$BDAT
2914 030406 001402      BEQ      120$      ;BRANCH IF VOLUME VALID ZERO
2915 030410 104135      EMT      135
2916 030412 000436      BR      140$      ;SKIP TEST
2917 030414
2918 030414 016037 000024 001142 120$:      MOV      RMR1(RO),$BDAT ;STORE RMR1 AT $BDAT
2919 030422 042737 077777 001142      BIC      #^COC,$BDAT
2920 030430 001407      BEQ      130$      ;BRANCH IF OCCUPIED ZERO
2921 030432 010037 001136      MOV      RO,$BDADR ;SETUP ERROR MESSAGE
2922 030436 062737 000024 001136      ADD      #RMR1,$BDADR
2923 030444 104166      EMT      166
2924 030446 000420      BR      140$      ;SKIP TEST
2925
2926                          ;TO VERIFY THAT THE DECODE FLOP IS RESET, LOAD AN ILLEGAL FUNCTION
2927                          ;IN RMCS1 AND VERIFY THAT ILF DOES NOT SFT.
2928 030450
2929 030450 012760 000024 000000 130$:      MOV      #ILF24,RMCS1(RO) ;LOAD RMCS1
2930 030456 016037 000014 001142      MOV      RMER1(RO),$BDAT ;STORE RMER1 AT $BDAT
2931 030464 042737 177776 001142      BIC      #^CILF,$BDAT
2932 030472 001410      BEQ      150$      ;BRANCH IF ILF IS ZERO
2933 030474 010037 001136      MOV      #0,$BDADR ;SETUP ERROR MESSAGE
    
```

```

2932 030500 062737 000014 001136      ADD    #RMER1,$BDADR
2933 030506 104167                      EMT    167
2934 030510 012716 030516      140$:  MOV    #200$,(SP)      ;DCNI GO BACK TO TEST
2935                                ;
2936 030514 000207      150$:  RTS    PC              ;RETURN TO TEST OR EXIT TEST
2937                                ;
2938 030516      200$:
2939
2940
;*****
;*TEST 53      SET/RESET VOLUME VALID TEST
;*****
TST53:
030516
030516 000004                      SCOPE
030520 000240                      NOP      ;SCOPE CALL
030522 012706 001100      MOV    #STACK,SP      ;LOAD THE STACK POINTER
030526 013700 001276      MOV    $BASE,R0       ;R0 = UNIBUS ADDRESS
030532 013701 001466      MOV    TSTQUE,R1      ;R1 = POINTER TO DEVICE
030536 012737 000053 001226      MOV    #53,$TSTN      ;;SET TEST NUMBER IN APT MAIL BOX
2941
2942 030544 010037 001136      MOV    R0,$BDADR      ;SETUP REGISTER ADDRESS
2943 030550 062737 000012 001136      ADD    #RMD$,$BDADR
2944 030556 012702 030762      MOV    #100$,R2       ;R2=TABLE POINTER
2945
;INITIALIZE AND USE DIAGNOSTIC MODE TO RESET VOLUME VALID
10$:
2947 030562
2948 030562 004737 055156      JSR    PC,CNTCLR      ;GO CLEAR CONTROLLER
2949 030566 012760 000001 000024      MOV    #DMD,$RPR1(R0) ;LOAD RPR1
2950 030574 012760 041001 000024      MOV    #DMD!MUR!DBEN,$RPR1(R0) ;LOAD RPR1
2951 030602 012760 000000 000014      MOV    #0,$RMER1(R0)  ;LOAD RMER1
2952 030610 012760 000000 000042      MOV    #0,$RMER2(R0)  ;LOAD RMER2
2953 030616 016037 000012 001142      MOV    RMD$($R0),$BDDAT ;STORE RMD$ AT $BDDAT
2954 030624 042737 177677 001142      BIC    #*CVV,$BDDAT
2955 030632 001403                      BEQ    20$             ;BRANCH IF VOLUME VALID ZERO
2956 030634 005037 001140      CLR    $GDDAT
2957 030640 104135                      EMT    135
2958
;EXECUTE THE FUNCTION CODE IN THE TABLE
20$:
2960 030642
2961 030642 111203                      MOVB   (R2),R3         ;GET FUNCTION CODE
2962 030644 042703 177701                      BIC    #*C1LF76,R3
2963 030650 052703 0000G1                      BIS    #60,R3
2964 030654 010360 000000      MOV    R3,$RMC$1(R0)  ;LOAD RMC$1
2965 030660 116204 000001      MOVB   1(R2),R4        ;GET CLOCK COUNT
2966 030664 042704 177400                      BIC    #*C377,R4
2967 030670
30$:
2968 030670 012760 141001 000024      MOV    #DMD!DBEN!MUR!DBCK,$RPR1(R0) ;LOAD RPR1
2969 030676 012760 041001 000024      MOV    #DMD!DBEN!MUR,$RPR1(R0) ;LOAD RPR1
2970 030704 005304                      DEC    R4
2971 030706 001370                      BNE    30$            ;ISSUE COCKS TIL R4 ZERO
2972 030710 016037 000012 001142      MOV    RMD$($R0),$BDDAT ;STORE RMD$ AT $BDDAT
2973 030716 042737 177677 001142      BIC    #*CVV,$BDDAT
2974 030724 001007                      BNE    40$            ;BRANCH IF VOLUME VALID SET
2975 030726 010337 001174      MOV    R3,$STMPD      ;SAVE FUNCTION CODE FOR MSG
2976 030732 012737 000100 001140      MOV    #VV,$GDDAT
2977 030740 104170                      EMT    170
2977 030742 000406                      BR     50$

```

```

2978
2979
2980 030744
2981 030744 062702 000002
2982 030750 105762 000001
2983 030754 100401
2984 030756 000701
2985 030760 000403
2986
2987
2988 030762
2989 030762 020
2990 030763 003
2991
2992 030764 022
2993 030765 003
2994
2995 030766 000
2996 030767 377
2997
2998 030770
2999
3000

030770
030770 000004
030772 000240
030774 012706 001100
031000 013700 001276
031004 013701 001466
031010 012737 000054 001226

3001
3002 031016 005002
3003 031020
3004 031020 004737 054732
031024 000402
031026 104000
3005 031030 000460
3006 031032 012704 000002
3007
3008
3009 031036 012760 041001 000024
3010 031044 010203
3011 031046 052703 000001
3012 031052 010360 000000
3013 031056
031056 012760 141001 000024
3014 031064 012760 041001 000024
3015 031072 005304
3016 031074 001370
3017 031076 016037 000014 001142
3018 031104 042737 177776 001142
3019 031112 016237 063640 001140
3020 031120 042737 177776 001140
3021 031126 023737 001140 001142

```

```

ADVANCE THE TABLE POINTER, EXIT IF DONE
40$:
ADD #2,R2
TSTB 1(R2)
BNI 50$ ;EXIT IF COUNT IS NEGATIVE
BR 10$
50$: BR 200$ ;JUMP OVER TABLE

;TABLE OF FUNCTION CODES AND CLOCK COUNTS
100$:
.BYTE RIP ;READ IN PRESET COMMAND
.BYTE 3
.BYTE PAKACK ;PACK ACKNOWLEDGE COMMAND
.BYTE 3
.BYTE ;END OF TABLE
.BYTE -1
200$: ;END OF TEST

;*****
;*TEST 54 ILLEGAL FUNCTION TEST
;*****
;*****
;*****
TST54:
SCOPE ;SCOPE CALL
NDP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV SBASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #54,$TSTH ;;SET TEST NUMBER IN APT MAIL BOX

10$: CLR R2 ;INITIALIZE FUNCTION CODE VALUE
JSR PC,SETVV ;GO SET VOLUME VALID
BR 20$ ;BRANCH TO 20$ IF NO ERROR
EMT
BR 50$ ;SKIP TEST IF ERROR
20$: MOV #2,R4 ;R4=CLOCK COUNT

;EXECUTE THE TEST FUNCTION CODE AND VERIFY ILF
MOV #DMD!MUR!DBEN,RMPR1(R0) ;LOAD RMPR1
MOV R2,R3 ;SETUP FUNCTION CODE IN R3
BIS #60,R3
MOV R3,RMCS1(R0) ;LOAD RMCS1

30$: MOV #DMD!MUR!DBEN!DBCK,RMPR1(R0) ;LOAD RMPR1
MOV #DMD!MUR!DBEN,RMPR1(R0) ;LOAD RMPR1
DEC R4
BNE 30$
MOV RMER1(R0),SBDDAT ;STORE RMER1 AT SBDDAT
BIC #*CILF,$GDDAT ;SETUP ACTUAL ILF STATUS
MOV FNCDTB(R2),$GDDAT ;GET EXPECTED ILF STATUS
BIC #*CILF,$GDDAT
CMP $GDDAT,$SBDDAT

```

```

3022 031134 001410          BEQ    40$          ;BRANCH IF ILF IS OK
3023 031136 010037 001136    MOV    R0,$BDDADR ;SETUP FOR ERROR MSG
3024 031142 062737 000014 001136    ADD    #RMR1,$BDDADR
3025 031150 010237 001174    MOV    R2,$TMP0
3026 031154 104171          EMT    171
3027
3028          ;ADVANCE TO THE NEXT FUNCTION CODE AND REPEAT TEST
3029 031156          40$:
3030 031156 062702 000002          ADD    #2,R2
3031 031162 022702 000076          CMP    #ILF76,R2
3032 031166 103401          BLO   50$
3033 031170 000713          BR    10$
3034 031172          50$:          ;END OF TEST
3035
3036          ;*****
          ;*TEST 55      OCCUPIED TEST
          ;*****
          TST55:
          SCOPE          ;SCOPE CALL
          NOP
          MOV    #STACK,SP ;LOAD THE STACK POINTER
          MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
          MOV    TSTQUE,R1 ;R1 = POINTER TO DEVICE
          MOV    #55,$TSTN ;:SET TEST NUMBER IN APT MAIL BOX
3037
3038 031220 005002          CLR    R2          ;INITIALIZE FUNCTION CODE
3039
3040          ;GET THE DEVICE READY
3041 031222          10$:
3042 031222 004737 054732          JSR   PC,SETVV    ;GO SET VOLUME VALID
          BR    20$    ;BRANCH TO 20$ IF NO ERROR
          EMT
          BR    50$
3043 031232 000464
3044
3045          ;ENABLE DEBUG CLOCK AND LOAD THE FUNCTION CODE
3046 031234          20$:
3047 031234 012760 041001 000024          MOV    #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
          MOV    R2,R3    ;ASSEMBLE FUNCTION CODE AND
          BIS    #60,R3    ;GO BIT IN R3
          MOV    R3,RMCS1(R0) ;LOAD RMCS1
          MOV    #2,R4    ;R4=CLOCK COUNT
3048 031242 010203
3049 031244 052703 000001
3050 031250 010360 0000C0
3051 031254 012704 000002
3052
3053          ;STEP THE DEBUG CLOCK UNTIL SET PULSE IS ACTIVE
3054 031260          30$:
3055 031260 012760 141001 000024          MOV    #DMD!MUR!DBEN!DBCK,RMR1(R0) ;LOAD RMR1
          MOV    #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
          DEC    R4
          BNE   30$    ;ISSUE NEXT CLOCK TIL R4 ZERO
3056 031266 012760 041001 000024
3057 031274 005304
3058 031276 001370
3059
3060          ;VERIFY OCCUPIED STATUS
3061 031300 016037 000024 001142          MOV    RMR1(R0),$BDDAT ;STORE RMR1 AT $BDDAT
3062 031306 042737 077777 001142          BIC   #*COCC,$BDDAT
3063 031314 005037 001140          CLR   $GDDAT ;GENERATE OCC FROM AOE
3064 031320 032762 001000 063640          BIT   #AOE,FNCD1(R2)
3065 031326 001403          BEQ   35$
3066 031330 012737 100000 001140          MOV   #OCC,$GDDAT
  
```

```

3067 031336 023737 001140 001142 358 CMP $GDDAT,$SPDADR
3068 031344 001411 BEC 40$ ; BRANCH IF OCC IS ON
3069 031346 010237 001174 MOV R2,$TMPG ; SAVE FUNCTION CODE
3070 031352 010037 001136 MOV R0,$BDADR ; SETUP REGISTER ADDRESS
3071 031356 062737 000024 001136 ADD #RMPR1,$BDADR
3072 031364 104173 EMT '73
3073 031366 000406 BR 50$

```

ADVANCE TO NEXT FUNCTIONCODE, EXIT IF DONE

```

3074
3075
3076 031370
3077 031370 062702 000002 ADD #2,R2
3078 031374 072702 000076 CMP #1L'76,R2
3079 031400 103401 BLO 50$ ; EXIT IF DONE
3080 031402 000707 BR 10$

```

50\$: ; END OF TEST

\*\*\*\*\*  
: TEST 56 READ IN PRESET TEST

```

031404
031404 0J0004
031406 000240
031410 012706 001100
031414 013700 001276
031420 013701 001466
031424 012737 000056 001226

```

```

TST56:
SCOPE ; SCOPE CALL
NOP
MOV #STACK,SP ; LOAD THE STACK POINTER
MOV $BASE,R0 ; R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ; R1 = POINTER TO DEVICE
MOV #56,$TESTN ; SET TEST NUMBER IN APT MAIL BOX

```

```

3085
3086
3087 031432 004737 055156 JSR PC,CNTCLR ; GC CLEAR CONTROLLER
3088 031436 012760 000001 000024 MOV #DMD,RMPR1(R0) ; LOAD RMPR1
3089 031444 012760 041001 000024 MOV #DMD:ML'DBEN,RMPR1(R0) ; LOAD RMPR1
3090 031452 012760 000000 000014 MOV #0,RMER1(R0) ; LOAD RMER1
3091 031460 012760 000000 000042 MOV #0,RMER2(R0) ; LOAD RMER2

```

```

; CLEAR AND ENABLE DEBUG CLOCK - LEAVE VOLUME VALID RESET
; LOAD ALL ONES IN RMDA, RMDC AND RMOF
MOV #-1,RMDA(R0) ; LOAD RMDA
MOV #-1,RMDC(R0) ; LOAD RMDC
MOV #-1,RMOF(R0) ; LOAD RMOF
MOV #-1,RMOF(R0) ; LOAD RMOF AGAIN TO SET SSEI

```

```

3092
3093
3094 031466 012760 177777 000006
3095 031474 012760 177777 000034
3096 031502 012760 177777 000032
3097 031510 012760 177777 000032
3098
3099 031516 012760 000021 000030 MOV #RIP!GO,RMCS1(R0) ; LOAD RMCS1
3100 031524 012702 000003 MOV #3,R2 ; R2=CLOCK COUNT

```

```

; LOAD READ IN PRESET COMMAND AND STEP THE CLOCK TILL SET PULSE
10$:
MOV #DMD:MUR'DBEN'DBCK,RMPR1(R0) ; LOAD RMPR1
MOV #DMD:MUR'DBEN,RMPR1(R0) ; LOAD RMPR1
DEC R2
BNE 10$ ; ISSUE 3 CLOCKS

```

```

3101 031530
3102 031530 012760 141001 000024
3103 031536 012760 041001 000024
3104 031544 005302
3105 031546 001370
3106
3107
3108 031550 016002 000006
3109 031554 005702
3110 031556 001413
3111 031560 016002 000034
3112 031564 042702 176000

```

```

; SEE IF RMDA OR RMDC OR RMOF IS ZERO
MOV RMDA(R0),R2 ; STORE RMDA AT R2
TST R2
BEQ 20$ ; BRANCH IF RMDA IS ZERO
MOV RMDC(R0),R2 ; STORE RMDC AT R2
BIC #XNUDC,R2 ; CLEAR UNUSED BITS

```

```

3113 031570 001406          BEQ      20$          ;BRANCH IF RMDC IS ZERO
3114 031572 016002 000032  MOV      RMOF(RO),R2    ;STORE RMOF AT R2
3115 031576 042702 160577  BIC      #XNUOF,R2      ;CLEAR UNUSED BITS
3116 031602 001401          BEQ      20$          ;BRANCH IF RMOF IS ZERO
3117
3118                          ;READ IN PRESET COMMAND DIDNT CLEAR ANY OF THE 3 REGISTERS
3119 031604 104174          EMT      174
3120
3121 031606          20$:          ;END OF TEST
3122
3123  ::*****

```

\*TEST 57 RIP/RMOF TEST

```

031606
031606 000004          TST57:  SCOPE          ;SCOPE CALL
031610 000240          NOP
031612 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
031616 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
031622 013701 001466  MOV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
031626 012737 000057 001226  MOV      #57,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
3124
3125 031634 010037 001136  MOV      R0,$BDADR      ;SETUP REGISTER ADDRESS AND
3126 031640 062737 000032 001136  ADD      #RMOF,$BDADR
3127 031646 005037 001140  CLR      $GDDAT        ;EXPECTED RMOF
3128
3129                          ;INITIALIZE AND SET BITS IN RMOF
3130 031552 004737 055156  JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
3131 031655 012760 000001 000024  MOV      #DMD,RMR1(R0)  ;LOAD RMR1
3132 031664 012760 041001 000024  MOV      #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
3133 031672 012760 000000 000014  MOV      #0,RMR2(R0)   ;LOAD RMR2
3134 031700 012760 000000 000042  MOV      #0,RMR2(R0)   ;LOAD RMR2
3135 031706 012760 177777 000032  MOV      #-1,RMOF(RO)  ;LOAD RMOF
031717 012760 177777 000032  MOV      #-1,RMOF(RO)  ;LOAD RMOF AGAIN TO SET SSET
3136
3137                          ;EXECUTE A READ IN PRESET IN DIAGNOSTIC MODE TILL SET PULSE
3138 031722 012760 000021 000000  MOV      #RIP!GO,RMCS1(RO) ;LOAD RMCS1
3139 031730 012702 000003  MOV      #3,R2          ;R2=CLOCK COUNT
3140 031734          10$:          MOV      #DMD!MUR!DBEN!DBCK,RMR1(R0) ;LOAD RMR1
031734 012760 141001 000024  MOV      #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
3141 031742 012760 041001 000024  MOV      #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
3142 031750 005302  DEC      R2
3143 031752 001370  BNE     10$            ;ISSUE 3 CLOCKS
3144
3145                          ;VERIFY THAT RMOF IS ZERO
3146 031754 016037 000032 001142  MOV      RMOF(RO),$BDAT ;STORE RMOF AT $BDAT
3147 031762 042737 160577 001142  BIC      #XNUOF,$BDAT
3148 031770 001401  BEQ      20$          ;BRANCH IF RMOF IS ZERO
3149 031772 104175  EMT      175
3150
3151 031774          20$:          ;END OF TEST
3152
3153  ::*****

```

\*TEST 60 RMDA/RMDC/RIP TEST

TST60:

031774

```

031774 000004          SCOPE          ;SCOPE CALL
031776 000240          NOP
03200G 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
032004 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
032010 013701 001466  MCV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
032014 012737 000060 001226  MOV      #60,$TESTN     ;;SFT TEST NUMBER IN APT MAIL BOX

3154
3155 032022 005037 001140  CLR      $GDDAT
3156
3157 ;CLEAR, ENABLE DEBUG CLOCK, THEN PRESEI RMDA AND RMDC
3158 032026 004737 055156  JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
3159 032032 012760 000001 000024  MOV      #DMD,RMPR1(R0) ;LOAD RMPR1
3160 032040 012760 041001 000024  MOV      #DMD!MUR!DBEN,RMPR1(R0) ;LOAD RMPR1
3161 032046 012760 000000 000014  MOV      #0,RMER1(R0)   ;LOAD RMER1
3162 032054 012760 000000 000014  MOV      #0,RMER1(R0)   ;LOAD RMER1
3163 032062 012760 177777 000006  MOV      #-1,RMDA(R0)   ;LOAD RMDA
3164 032070 012760 177777 000034  MOV      #-1,RMDC(R0)   ;LOAD RMDC
3165
3166 ;EXECUTE READ IN PRESEI TILL SET PULSE
3167 032076 012760 000021 000000  MOV      #RIP!GO,RMCS1(R0) ;LOAD RMCS1
3168 032104 012702 000003  MOV      #3,R2
3169 032110 10$:
3170 032110 012760 141001 000024  MOV      #DMD!MUR!DBEN!DBCK,RMPR1(R0) ;LOAD RMPR1
3171 032116 012760 041001 000024  MOV      #DMD!MUR!DBEN,RMPR1(R0) ;LOAD RMPR1
3172 032126 001370  DEC      R2
3173  BNE     10$           ;ISSUE 3 CLOCKS
3174
3174 ;VERIFY RMDA IS ZERO
3175 032130 016037 000006 001142  MOV      RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
3176 032136 005737 001142  TST     $BDDAT
3177 032142 001406  BEQ     20$           ;BRANCH IF RMDA RESET
3178 032144 010037 001136  MOV      R0,$BDADR
3179 032150 062737 000006 001136  ADD     #RMDA,$BDADR
3180 032156 104176  EMT     176
3181
3182 ;VERIFY RMDC IS ZERO
3183 032160 20$:
3184 032160 016037 000034 001142  MOV      RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
3185 032166 042737 176000 001142  BIC     #XNUDC,$BDDAT
3186 032174 001406  BEQ     30$           ;BRANCH IF RMDC RESET
3187 032176 010037 001136  MOV      R0,$BDADR
3188 032202 062737 000034 001136  ADD     #RMDC,$BDADR
3189 032210 104302  EMT     302
3190
3191 032212 30$:
3192  ;END OF TEST
3193

;*****
;*TEST 61      OFFSET COMMAND TEST
;*****

TST61:
032212 000004          SCOPE          ;SCOPE CALL
032214 000240          NOP
032216 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
032222 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
032226 013701 001466  MOV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
032232 012737 000061 001226  MOV      #61,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
  
```

```

3194
3195 032240 010037 001136          MOV    R0,$BDADR
3196 032244 062737 000012 001136    ADD    #RMD5,$BDADR
3197 032252 012737 000001 001140    MOV    #OM,$GDDAT
3198
3199 032260 004737 054732          JSR    PC,SETVV          ;GO SET VOLUME VALID
      032264 000402          BR     10$              ;BRANCH TO 10$ IF NO ERROR
      032266 104000          EMT
3200 032270 000433          BR     40$
3201 052272          10$:
      032272 012760 000000 000034    MOV    #0,RMDC(R0)      ;LOAD RMDC
3202
3203          ;ENABLE DEBUG CLOCK AND EXECUTE OFFSET COMMAND
3204 032300 012760 041001 000024    MOV    #DMD!MUR!DBEN,RMPR1(R0) ;LOAD RMPR1
3205 032306 012760 000015 000000    MOV    #OFFSET!GO,RMCS1(R0)    ;LOAD RMCS1
3206 032314 012702 000002          MOV    #2,R2            ;R2=CLOCK COUNT
3207 032320          20$:
      032320 012760 141001 000024    MOV    #DMD!MUR!DBEN!DBCK,RMPR1(R0) ;LOAD RMPR1
3208 032326 012760 041001 000024    MOV    #DMD!MUR!DBEN,RMPR1(R0) ;LOAD RMPR1
3209 032334 005302          DEL    R2
3210 032336 001370          BNE    20$              ;ISSUE 2 CLOCKS
3211
3212          ;VERIFY THAT OFFSET MODE IS SET
3213 032340 016037 000012 001142    MOV    RMD5(R0),$BDDAT ;STORE RMD5 AT $BDDAT
3214 032346 042737 177776 001142    BIC    #^COM,$BDDAT
3215 032354 001001          BNE    40$              ;BRANCH IF OM IS SET
3216 032356 104200          EMT    200
3217 032360          40$:
      ;END OF TEST
3218
3219          ;*****
      ;*TEST 62          RETURN TO CENTER TEST
      ;*****

      TST62:
      SCOPE          ;SCOPE CALL
      NOP
      MOV    #STACK,SP ;LOAD THE STACK POINTER
      MOV    $BASE,R0  ;R0 = UNIBUS ADDRESS
      MOV    TSTQUE,R1 ;R1 = POINTER TO DEVICE
      MOV    #62,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
3220
3221 032406 010037 001136          MOV    R0,$BDADR
3222 032412 062737 000012 001136    ADD    #RMD5,$BDADR
3223
3224 032420 004737 054732          JSR    PC,SETVV          ;GO SET VOLUME VALID
      032424 000402          BR     10$              ;BRANCH TO 10$ IF NO ERROR
      032426 104000          EMT
3225 032430 000465          BR     60$
3226
3227          ;SET OFFSET DIRECTION AND OFFSET MODE
3228 032432          10$:
3229 032432 012760 000200 000032    MOV    #OFD,RMOF(R0)
3230
3231 032440 004737 055054          JSR    PC,SETOM          ;GO SET OFFSET MODE
      032444 000401          BR     20$              ;BRANCH TO 20$ IF NO ERROR
      032446 104000          EMT
3232
  
```



```

3233                                     ENABLE DEBUG CLOCK AND EXECUTE RETURN TO CENTER COMMAND
3234 032450                               20$: JSR PC,SETVV ;GO SET VOLUME VALID
3235 032450 004737 054732                BR 30$ ;BRANCH TO .0$ IF NO ERROR
    032454 000402                        EMT
    032456 104000                        BR 60$
3236 032460 000451
3237 032462                               30$: MOV #DMD,MUR'DBEN,RMMR1(RO) ;LOAD RMMR1
3238 032462 012760 041001 000024        MOV #RTC!GO,PMCS1(RO) ;LOAD RMCS1
3239 032470 012760 000017 000000        MOV #2,R2
3240 032476 012702 000002                               40$: MOV #DMD!MUR'DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
3241 032502 012760 141001 000024        MOV #DMD!MUR'DBEN,RMMR1(RO) ;LOAD RMMR1
3242 032510 012760 041001 000024        DEC R2
3243 032511 005302                        BNE 40$ ;ISSUE 2 CLOCKS
3244 032520 001370
3245
3246                                     ;VERIFY THAT OFFSET MODE IS RESET
3247 032522 016037 000012 001142        MOV RMD5(RO),SBDDAT ;STORE RMD5 AT SBDDAT
3248 032530 042737 17777E 001142        BIC #*COM,SBDDAT
3249 032536 001403                        BEQ 50$ ;BRANCH IF OFFSET MODE RESET
3250 032540 005037 001140                CLR $GDDAT
3251 032544 104201                        EMT 201
3252
3253                                     ;VERIFY THAT OFFSET DIRECTION IS RESET
3254 032546                               50$: MOV RMOF(RO),SBDDAT ;STORE RMOF AT SBDDAT
3255 032546 016037 000032 001142        BIC #*CFD,SBDDAT
3256 032554 042737 177577 001142        BEQ 60$ ;BRANCH IF OFD IS RESET
3257 032562 001410
3258 032564 005037 001140                CLR $GDDAT
3259 032570 010037 001136                MOV RO,SBADR
3260 032574 062737 000032 001136        ADD #RMOF,SBADR
3261 032602 104202                        EMT 202
3262 032604                               60$: ;END OF TEST
3263
3264                                     ;*****
                                     ;*TEST 63 RMD5 CLEAR OFFSET TEST
                                     ;*****
032604                                     ;TEST63:
032604 000004                               SCOPE ;SCOPE CALL
032606 000240                               MOV NO-
032610 012706 001100                       MOV #STACK,SP ;LOAD THE STACK POINTER
032614 013700 001276                       MOV $BASE,RO ;RO = UNIBUS ADDRESS
032620 013701 001466                       MOV !STQUE,R1 ;R1 = POINTER TO DFVICE
032624 012737 000063 001226               MOV #53,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
3265
3266 032632 010037 001136                       MOV RO,SBADR
3267 032636 062737 000012 001136         ADD #RMD5,SBADR
3268
3269 032644 004737 054732                JSR PC,SETVV ;GO SET VOLUME VALID
    032650 000402                        BR 10$ ;BRANCH TO 10$ IF NO ERROR
    032652 104000                        EMT
    032654 000421                        BR 40$ ;SKIP REST OF TEST
3270 032654 000421
3271 032656                               10$: JSR PC,SETOM ;GO SET OFFSET MODE
3272 032656 004737 055054                BR 20$ ;BRANCH TO 20$ IF NO ERROR
    032662 000401                        EMT
    032664 104000
    
```

```

3273
3274 WRITE THE DESIRED CYLINDER REGISTER AND VERIFY THAT OFFSET IS ZERO
3275 032666 012760 000000 000034 20$: MOV #0,RMD(RO) ;LOAD RMDC
3276 032666 016037 000012 001142 MOV RMD(RO),SBDDAT ;STORE RMD(RO) AT SBDDAT
3277 032674 042737 177776 001142 BIC #*COM,SBDDAT
3278 032702 001403 BEQ 40$
3279 032710 005037 001140 CLR $GDDAT
3280 032712 104203 EMT 203
3281 032716
3282
3283 032720 40$: ;END OF TEST
3284
3285

```

\*\*\*\*\*  
\*TEST 64 EBL CLEAR OFFSET TEST  
\*\*\*\*\*

```

032720 TS164: SCOPE ;SCOPE CALL
032722 NOP
032724 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
032730 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
032734 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
032740 012737 000064 001226 MOV #64,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
3286
3287 032746 013737 001334 001420 MOV LSTRK,RMDAO ;SET LAST TRACK AND
3288 032754 112737 000036 001420 MOV #30,RMDAO ;LAST SECTOR
3289 032762 010037 001136 MOV R0,$BADDR ;SETUP REGISTER FOR ERROR MSG
3290 032766 062737 000012 001136 ADD #RMD(RO),$BADDR
3291
3292 032774 004737 054732 JSR PC,SETVV ;GO SET VOLUME VALID
033000 BR 10$ ;BRANCH TO 10$ IF NO ERROR
033002 EMT 100000
3293 033004 000440 BR 30$ ;SKIP REST OF TEST IF ERROR
3294 033006
3295 033006 004737 055054 10$: JSR PC,SETOM ;GO SET OFFSET MODE
033012 BR 20$ ;BRANCH TO 20$ IF NO ERROR
033014 EMT 104000
3296 033016
3297 033016 012760 010000 000032 20$: MOV #FMT16,RMF(RO)
3298 033024 013760 001420 000006 MOV RMDAO,RMDA(RO) ;LOAD RMDA
3299
3300 ;FORCE END OF BLOCK AND VERIFY THAT OFFSET MODE IS CLEARED
3301 033032 012760 041001 000024 MOV #RMD(RO),RMDA(RO) ;LOAD RMDA
3302 033040 012760 000001 000000 MOV #GO,RMCSI(RO) ;LOAD RMCSI
3303 033046 012760 061001 000024 MOV #RMD(RO),RMDA(RO) ;LOAD RMDA
3304 033054 012760 041001 000024 MOV #RMD(RO),RMDA(RO) ;LOAD RMDA
3305 033062 016037 000012 001142 MOV RMD(RO),SBDDAT ;STORE RMD(RO) AT SBDDAT
3306 033070 042737 177776 001142 BIC #*COM,SBDDAT
3307 033076 001403 BEQ 30$ ;BRANCH IF OFFSET IS ZERO
3308 033100 005037 001140 CLR $GDDAT
3309 033104 104204 EMT 204
3310 30$: ;END OF TEST
3311
3312

```

\*\*\*\*\*  
\*TEST 65 RUN AND GO TEST  
\*\*\*\*\*

```

033106          TST65:
033106 000004          SCOPE          ;SCOPE CALL
033110 000240          NOP
033112 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
033116 013700 001276  MOV      $BASE,R0        ;R0 = UNIBUS ADDRESS
033122 013701 001466  MOV      TSTQUE,R1       ;R1 = POINTER TO DEVICE
033126 012737 000065 001226  MOV      #65,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX

3313
3314 033134 005037 001140          CLR      $GDDAT          ;INITIALIZE EXPECTED RESULT
3315 033140 005002          CLR      R2              ;INITIALIZE FUNCTION CODE
3316 033142 010037 001136  MOV      R0,$BDADR
3317 033146 062737 000024 001136  ADD      #RMR1,$BDADR

3318
3319          ;CLEAR THE MASSBUS AND ENABLE DEBUG CLOCK
3320 033154          10$:
3321 033154 004737 055156          JSR      PC,CNTCLR      ;GO CLEAR CONTROLLER
3322 033160 012760 000001 000024  MOV      #DMD,RMR1(R0)  ;LOAD RMR1
3323 033166 012760 040001 000024  MOV      #DMD!DBEN,RMR1(R0) ;LOAD RMR1
3324 033174 012760 000000 000014  MOV      #0,RMR1(R0)   ;LOAD RMR1
3325 033202 012760 000000 000042  MOV      #0,RMR2(R0)   ;LOAD RMR2
3326
3327          ;LOAD THE FUNCTION CODE AND VERIFY RUN AND GO FLOP
3328 033210 010203          MOV      R2,R3          ;ASSEMBLE FUNCTION CODE AND GO
3329 033212 052703 000001          BIS      #GO,R3
3330 033216 012737 000200 001534  MOV      #200,WATCH     ;SET WATCHDOG TIMER VALUE
033224 004777 146306          JSR      PC,@CLOCK     ;START THE CLOCK
3331 033230 010360 000000          MOV      R3,RMCS1(R0)  ;LOAD RMCS1
3332
3333          15$:
033234 016037 000024 001142  MOV      RMR1(R0),$BDAT ;STORE RMR1 AT $BDAT
033242 042737 137777 001142  BIC      #*CRG,$BDAT
3334 033250 023737 001140 001142  CMP      $GDDAT,$BDAT
3335 033256 001411          BEQ      20$           ;BRANCH IF RUN AND GO FLOP OK
3336 033260 005737 001534          TST     WATCH         ;TAKE ANOTHER SAMPLE IF CLOCK NOT ZFRO
3337 033264 001363          BNE     15$
3338 033266 004777 146246          JSR      PC,@STOPCL   ;STOP THE CLOCK
3339 033272 010237 001174          MOV      R2,$TMP0     ;SAVE FUNCTION CODE FOR MSG
3340 033276 104205          EMT     20$
3341 033300 000416          BR      40$           ;SKIP REST OF
3342
3343          20$:
033302 004777 146232          JSR      PC,@STOPCL   ;STOP THE CLOCK
3344
3345          ;ADVANCE TO NEXT FUNCTION CODE - EXIT IF DONE
3346 033306 062702 000002          ADD     #2,R2
3347 033312 022702 000076          CMP     #1LF76,R2
3348 033316 103407          BLO     40$           ;EXIT IF DONE
3349 033320 020227 000050          CMP     R2,#WCD       ;CHANGE EXPECTED RESULT IF
3350 033324 103403          BLO     30$           ;DATA COMMAND
3351 033326 012737 040000 001140  MOV      #RG,$GDDAT
3352 033334 000707          BR      10$          ;REPEAT TEST
3353
3354          40$:
033336          ;END OF TEST
3355
3356          ;*****
          ;*TEST 66      SET IAE TEST
          ;*****

033336          TST66:
    
```

```

033336 000004          SCOPE          ;SCOPE CALL
033340 000240          NOP
033342 012706 001100  MOV      #STACK,SP      ;LOAD THE STACK POINTER
033346 011700 001276  MOV      $BASE,R0       ;R0 = UNIBUS ADDRESS
033352 013701 001466  MOV      TSTQUE,R1      ;R1 = POINTER TO DEVICE
033356 012737 000066 001226  MOV      #66,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX

3357
3358 033364 012702 033532          MOV      #100$,R2      ;R2-TABLE POINTER
3359 033370          10$:
3360 033370 004737 054732          JSR      PC,SETVV      ;GO SET VOLUME VALID
033374 000402          BR       20$          ;BRANCH TO 20$ IF NO ERROR
033376 104000          EMT
3361 033400 000453          BR       50$          ;SKIP REST OF TEST
3362
3363          ;LOAD INVALID TRACK, SECTOR AND CYLINDER ADDRESS AND SET FORMAT TO 18
3364 033402          20$:
3365 033402 012760 177777 000006  MOV      #-1,RMDA(R0)   ;LOAD RMDA
3366 033410 012760 177777 000034  MOV      #-1,RMDC(R0)   ;LOAD RMDC
3367 033416 012760 000000 000032  MOV      #0,RMOF(R0)    ;LOAD RMOF
3368
3369          ;ENABLE DEBUG CLOCK AND LOAD FUNCTION CODE IN RMCSI WITH GO ON
3370 033424 012760 041001 000024  MOV      #DMD!MJR!DBEN,RMPR1(R0) ;LOAD RMPR1
3371 033432 111203          MOV      (R2),R3
3372 033434 042703 177701          BIC      #~CILF76,R3
3373 033440 052703 000001          BIS      #GO,P3
3374 033444 010360 000000          MOV      R3,RMCSI(R0)   ;LOAD RMCSI
3375 033450 116204 000001          MOV      1(R2),R4      ;GET CLOCK COUNT
3376 033454 042704 177400          BIC      #~C377,R4
3377
3378          ;CLOCK THE COMMAND SEQUENCES
3379 033460          30$:
3380 033460 012760 141001 000024  MOV      #DMD!MJR!DBEN!DBCK,RMPR1(R0) ;LOAD RMPR1
3381 033466 012760 041001 000024  MOV      #DMD!MJR!DBEN,RMPR1(R0) ;LOAD RMPR1
3382 033474 005304          DEC      R4
3383 033476 001370          BNE     30$
3384
3385          ;SEE IF IAE HAS SET
3386 033500 016004 000014          MOV      RMPR1(R0),R4   ;STORE RMPR1 AT R4
3387 033504 042704 175777          BIC      #~CIAE,R4
3388 033510 001007          BNE     50$          ;BRANCH IF IAE SET
3389
3390          ;IAE DID NOT SET - TRY ANOTHER FUNCTION CODE
3391 033512 062702 000002          ADD      #2,R2
3392 033516 105742 000001          TSTB    1(R2)
3393 033522 100401          BMI     40$          ;BRANCH IF ALL CODES TRIED
3394 033524 000721          BR      10$
3395
3396          ;CANNOT SET IAE WITH ANY COMBINATION OF ADDRESS AND FUNCTION CODE
3397 033526          40$:
3398 033526 104206          EMT     206
3399
3400 033530 000411          50$: BR      200$      ;JUMP OVER TABLE
3401
3402          ;TABLE OF FUNCTION CODES AND CLOCK COUNTS FOR TEST
3403 033532          100$:
3404 033532          .BYTE   SEARCH        ;SEARCH COMMAND
3405 033533          .BYTE   2
    
```

```

3406
3407 033534 004 .BYTE SEEK .SEEK COMMAND
3408 033535 002 .BYTE 2
3409
3410 033536 062 .BYTE WH ;WRITE HEADER COMMAND
3411 033537 002 .BYTE 2
3412
3413 033540 052 .BYTE WCH ;WRITE CHECK HEADER COMMAND
3414 033541 002 .BYTE 2
3415
3416 033542 072 .BYTE RH ;READ HEADER COMMAND
3417 033543 002 .BYTE 2
3418
3419 033544 060 .BYTE WD ;WRITE DATA COMMAND
3420 033545 002 .BYTE 2
3421
3422 033546 050 .BYTE WCD ;WRITE CHECK DATA COMMAND
3423 033547 002 .BYTE 2
3424
3425 033550 070 .BYTE RD ;READ DATA COMMAND
3426 033551 002 .BYTE 2
3427
3428 033552 000 .BYTE ;END OF TABLE
3429 033553 377 .BYTE -1
3430
3431 033554 200$ ;END OF TEST
3432
3433

```

```

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

```

```

033554 TST67:
033554 000004 ;SCOPE CALL
033556 000240
033560 01 06 001100 MOV #STACK, SP ;LOAD THE STACK POINTER
033564 013700 001276 MOV SBASE, R0 ;R0 = UNIBUS ADDRESS
033570 013701 001466 MOV TSTQUE, R1 ;R1 = POINTER TO DEVICE
033574 012737 000067 001226 MOV #67, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
3434
3435 03 02 005002 CLR R2 ;INITIALIZE FUNCTION CODE
3436 033604
3437 033604 004737 054732 JSR PC, SETVV ;GO SET VOLUME VALID
033610 000462 BR 20$ ;BRANCH TO 20$ IF NO ERROR
033612 104000 EMT
3438 033614 000472 BR 50$
3439
3440 ;LOAD INVALID TRACK, SECTOR AND CYLINDER ADDRESS AND SET FORMAT
3441 ;TO 18 BIT MODE
3442 033616 20$:
3443 033616 012760 177777 000006 MOV #-1, RMDA(R0) ;LOAD RMDA
3444 033624 012760 177777 000034 MOV #-1, RMDC(R0) ;LOAD RMDC
3445 033632 012760 000000 000032 MOV #0, RMOF(R0) ;LOAD RMOF
3446
3447 ;ENABLE DEBUG CLOCK AND LOAD FUNCTION CODE IN RMC51 WITH GO ON
3448 033640 012760 041001 000024 MOV #DMD!MUR!DBEN, RMPR1(R0) ;LOAD RMPR1
3449 033646 010203 MOV R2, R3 ;ASSEMBLE CODE AND GO
3450 033650 052703 000001 BIS #GO, R3

```

```
3451 033654 010360 000000      MOV     R3,RMCS1(RO)    ;LOAD RMCS1
3452
3453                          ;CLOCK THE COMMAND SEQUENCER TO SET PULSE
3454 033660 012704 000002      MOV     #2,R4
3455 033664 012760 141001 000024 30$:   MOV     #DMC:RUR:DBEN:DBCK,RMPR1(RO) ;LOAD RMPR1
033664 012760 041001 000024      MOV     #DMC:RUR:DBEN,RMPR1(RO) ;LOAD RMPR1
3456 033672 012760 041001 000024      DEC     R4
3457 033700 005304              BNE     30$
3458 033702 001370
3459
3460                          ;VERIFY IAE ACCORDING TO FUNCTION CODE TABLE
3461 033704 016037 000014 001142      MOV     RMR1(RO),%CDDA1 ;STORE RMR1 AT %CDDA1
3462 033712 042737 175777 001142      BIC     #^CIAE,%CDDAT
3463 033720 016237 063640 001140      MOV     FNCDTB(R2),%CDDA1 ;ASSEMBLE EXPECTED IAE
3464 033726 042737 175777 001140      BIC     #^CIAE,%CDDAT
3465 033734 023737 001140 001142      CMP     %CDDAT,%CDDAT
3466 033742 001411              BEQ     40$              ;BRANCH IF IAE OK
3467 033744 010037 001136              MOV     R0,%CDDADR      ;SET UP ERROR MSG
3468 033750 062737 000014 001136      ADD     #RMR1,%CDDADR
3469 033756 010237 001174              MOV     R2,%TMP0
3470 033762 104207              EMT     207
3471 033764 000406              BR      50$              ;SKIP REST OF TEST
3472
3473                          ;ADVANCE TO NEXT FUNCTION CODE - EXIT IF DONE
3474 033766 062702 000002 40$:   ADD     #2,R2
3475 033766 023702 000076      CMP     ILF76,R2
3476 033772 103401              BLO     50$
3477 033776 000701              BR      10$
3478 034000
3479 034002 50$:   ;END OF TEST
3480
3481                          ;*****
;*TEST 70          INVALID TRACK/SECTOR TEST
                          ;*****
3482                          TST70:
034002 000004              SCOPE          ;SCOPE CALL
034004 000240              NOP
034006 012706 001100      MOV     #STACK,SP      ;LOAD THE STACK POINTER
034012 013700 001276      MOV     %BASE,R0       ;R0 = UNIBUS ADDRESS
034016 013701 001466      MOV     TSTQUE,R1      ;R1 = POINTER TO DEVICE
034022 012737 000070 001226      MOV     #70,%STESTN    ;;SET TEST NUMBER IN APT MAIL BOX
3483 034030 013737 001334 001420      MOV     LSTRK,RMDAO    ;LOAD LAST TRACK ADDRESS
3484 034036 105237 001421      INCB   RMDAO+1        ;SETUP FIRST INVALID ADDRESS
3485 034042
3486 034042 004737 054732 10$:   JSR     PC,SETVV       ;GO SET VOLUME VALID
034046 000402              BR      20$           ;BRANCH TO 20$ IF NO ERROR
034050 104000              EMT
3487 034052 000477              BR      100$          ;SKIP REST OF TEST
3488
3489                          ;CLEAR DESIRED CYLINDER, LOAD INVALID ADDRESS AND SET
3490                          ;18 BIT FORMAT
3491 034054 012760 000000 000034 20$:   MOV     #0,RMDC(RO)    ;LOAD RMDC
3492 034054 013760 001420 000006      MOV     RMDAO,RMDA(RO) ;LOAD RMDA
3493 034062 012760 000000 000034      MOV     #0,RMPDF(RO)  ;LOAD RMPDF
```

```

3495
3496
3497 034076 012760 041001 000024 .EXECUTE A SEARCH COMMAND TO WHERE 'SET PULSE' IS ACTIVE
3498 034104 012760 000031 000000      MOV #DMD!MUR!DBEN,RMR1(RO) ;LOAD RMR1
3499 034112 012703 000002      MOV #SEARCH!GO,RMC1(RO) ;LOAD RMC1
3500 034116      MOV #2,R3
3501 034116 012760 141001 000024 30$: MOV #DMD!MUR!DBEN!DBCK,RMR1(RO) ;LOAD RMR1
3502 034124 012760 041001 000024      MOV #DMD!MUR!DBEN,RMR1(RO) ;LOAD RMR1
3503 034132 005303      DEC R3
3504 034134 001370      BNE 30$ ;ISSUE 2 CLOCKS
3505
3506
3507 034136 016037 000014 001142 ;VERIFY IAE IS SET
3508 034144 042737 175777 001142      MOV RMR1(RO),SDDAT ;STORE RMR1 AT SDDAT
3509 034152 001015      BIC #C!AE,SDDAT
3510 034154 012737 002000 001140      BNE 40$ ;BRANCH IF IAE IS ON
3511 034162 010037 001136      MOV #IAE,$GDDAT ;SETUP ERROR MESSAGE
3512 034166 062737 000014 001136      MOV RO,SDDADR
3513 034174 013737 001420 001174      ADD #RMR1,SDDADR
3514 034202 104210      MOV RMDAO,$TMPD
3515 034204 000422      EMT 210
3516      BR 100$
3517
3518 034206      ;ADVANCE TO NEXT ENTRY IN TABLE - EXIT IF DONE
3519 034206 105737 001421 40$: TSTB RMDAO+1 ;TESTING INVALID SECTORS ?
3520 034212 001411      BEQ 50$ ;YES !!
3521 034214 105237 001421      INCB RMDAO+1 ;INCREMENT TRACK ADDRESS
3522 034220 123727 001421 000200      CMPB RMDAO+1,#128. ;DONE ?
3523 034226 101705      BLOS 10$ ;NO, TEST NEXT TRACK ADDRESS
3524 034230 012737 000035 001420      MOV #29.,RMDAO ;LOAD LAST SECTOR ADDRESS AND
3525 ;TRACK 0.
3526 034236 005237 001420 50$: INC RMDAO ;INCREMENT SECTOR ADDRESS
3527 034242 123727 001420 000200      CMPB RMDAO,#128. ;DONE ?
3528 034250 101674      BLOS 10$ ;NO, TEST NEXT SECTOR ADDRESS
3529
3530 034252 100$:
3531
3532
3533
3534
3535
3536
3537
3538
3539
034252
034254 000004
034256 000200
034256 012706 001100
034262 013700 001276
034266 013701 001466
034272 012737 000071 001226
3533
3534 034300 012737 001061 001446
3535 034306
3536 034306 004737 054732 10$: JSR PC,SETVV ;GO SET VOLUME VALID
034312 000402      BR 20$ ;BRANCH TO 20$ IF NO ERROR
034314 104000      EMT
3537 034316 000460      BR 50$ ;SKIP IF ERROR
3538 034320
3539 034320 013760 001446 000034 20$: MOV RMDCO,RMDC(RO) ;LOAD RMDC

```

\*\*\*\*\*  
\*TEST 71 INVALID CYLINDER TEST  
\*\*\*\*\*

\*\*\*\*\*  
TST71:  
\*\*\*\*\*

```

SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV SBASE,RO ;RC = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #71,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

```

10$: MOV #561.,RMDCO ;SET FIRST INVALID CYLINDER
JSR PC,SETVV ;GO SET VOLUME VALID
BR 20$ ;BRANCH TO 20$ IF NO ERROR
EMT
BR 50$ ;SKIP IF ERROR

```

```

20$: MOV RMDCO,RMDC(RO) ;LOAD RMDC

```

```

3540 034326 012760 000000 000006      MOV      #0,RMDA(RO)      ;LOAD RMDA
3541
3542                                     ;ENABLE DEBUG (LOCK AND EXECUTE SEARCH COMMAND
3543 034334 012760 041001 000024      MOV      #DMD,MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
3544 034342 012760 000031 000000      MOV      #SEARCH!GO,RMCS1(RO)    ;LOAD RMCS1
3545 034350 012703 000002                                     MOV      #2,R3
3546 034354                                     30$:
3547 034362 012760 141001 000024      MO:      #DMD!MUR!DBEN!DBCK,RMPR1(RO) ;LOAD RMPR1
3548 034370 005303 041901 000024      MOV      #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
3549 034372 001370                                     DEC      R3
3550                                     BNE     30$              ;ISSUE 2 CLOCKS
3551                                     ;VERIFY IAE IS SET
3552 034374 016037 000014 001142      MOV      RMR1(RO),SDDAT          ;STORE RMR1 AT SDDAT
3553 034402 042737 175777 001142      BIC     #'CIAE,SDDAT
3554 034410 001015 002000 001140      BNE     40$                  ;BRANCH IF IAE IS SET
3555 034412 012737 002000 001140      MOV      #IAE,SGDDAT            ;SETUP ERROR MESSAGE
3556 034420 010037 001136                                     MOV      RO,SDDADR
3557 034424 062737 000014 001136      ADD     #RMR1,SDDADR
3558 034432 013737 001446 001174      MOV      RMDCO,$TMP0
3559 034440 104211                                     EMT     211
3560 034442 000406                                     BR      50$
3561
3562                                     ;ADVANCE CYLINDER ADDRESS
3563 034444                                     40$:
3564 034444 005237 001446 002000      INC     RMDCO                ;INCREMENT CYLINDER ADDRESS
3565 034450 023727 001446 002000      CMP     RMDCO,#1024.         ;DONE ?
3566 034456 03713                                     BLO    10$                  ;NO, TEST AGAIN
3567 034460                                     50$:
3568
3569
::*****
:TEST 72          SET AOE TEST
::*****
TST72:
034460                                     SCOPE                          ;SCOPE CALL
034460 000004      NOP
034462 000240      MOV      #STACK,SP          ;LOAD THE STACK POINTER
034464 012706 001100      MOV      $BASE,RO           ;RO = UNIBUS ADDRESS
034470 013700 001276      MOV      TSTQUE,R1          ;R1 = POINTER TO DEVICE
034474 013701 001466      MOV      #3,$TIME          ;;DO 3 ITERATIONS
034500 012737 000003 001206      MOV      #72,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX
034506 012737 000072 001226
3570
3571 034514 005037 001444      CLR     RMOFO                ;18 BIT FORMAT MODE
3572 034520 013737 001334 001420      MOV     LSTRK,RMDAO          ;SET LAST TRACK AND
3573 034526 112737 000035 001420      MOV     #29.,RMDAO          ;LAST SECTOR
3574
3575                                     ;ENABLE DEBUG CLOCK AND LOAD LAST SECTOR ADDRESS, MEMORY ADDRESS AND
3576                                     ;WORD COUNT, THEN LOAD WRITE DATA COMMAND WITH GO SET
3577 034534                                     10$:
3578 034534 004737 054732      JSR     PC,SETVV            ;GO SET VOLUME VALID
3579 034540 000402                                     BR      15$                ;BRANCH TO 15$ IF NO ERRGR
3580 034544 104000      EMT
3581 034546 000517      BR     40$                  ;SKIP TEST IF ERROR
3582 034546 012760 041001 000024      MOV     #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
3582 034544 013760 001444 000032      MOV     RMOFO,RMOF(RO)      ;LOAD RMOF

```



```
3583 034562 016037 000032 001174      MOV      RMOF(RO),STMP0      ;STORE RMOF AT STMP0
3584 034570 012760 001060 000034      MOV      #560.,RMD0C(RO)    ;LOAD RMD0C
3585 034576 013760 001420 000006      MOV      RMDA0,RMDA(RO)     ;LOAD RMDA
3586 034604 012760 177000 000002      MOV      #-512.,RPMC(RO)    ;LOAD RPMC
3587 034612 012760 104536 000004      MOV      #BUFFER,RMBA(RO)   ;LOAD RMBA
3588 034620 012760 000061 000000      MOV      #WD!GO,RMCS1(RO)   ;LOAD RMCS1
3589 034626 012702 000014                MOV      #14,R2              ;R2 = CLOCK COUNT
3590                                ;CLOCK COUNT 2-05 3-04 14-60...
3591                                ;CLOCK THE COMMAND SEQUENCER TO GENERATE SET PULSE
```

```
3592 034632                20$:
3593 034632 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMPR1(RO) ;LOAD RMPR1
3594 034640 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
3595 034646 005302                DEC      R2
3596 034650 001370                BNE     20$                  ;ISSUE 2 CLOCKS
```

```
3597                                ;FORCE EBL TO GET TO OVERFLOW ADDRESS
3598
3599 034652 012760 061001 000024      MOV      #DMD!MUR!DBEN!DEBL,RMPR1(RO) ;LOAD RMPR1
3600 034660 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
```

```
3601                                ;VERIFY THAT ADDRESS OVERFLOW ERROR IS SET
3602
3603 034666 016037 000014 001142      MOV      RMER1(RO),SBDDAT    ;STORE RMER1 AT SBDDAT
3604 034674 042737 176777 001142      BIC     #^CAOE,SBDDAT
3605 034702 001012                BNE     30$                  ;BRANCH IF AOE IS SET
3606 034704 010037 001136                MOV     RO,SBADR             ;SETUP ERROR MESSAGE
3607 034710 062737 000014 001136      ADD     #RMER1,SBADR
3608 034716 012737 001000 001140      MOV     #AOE,$GDDAT
3609 034724 104212                EMT     212
3610 034726 000426                BR      40$
```

```
3611 034730                30$:
3612 034730 032737 010000 001444      BIT     #FMT16,RMOFO         ;DONE 16 BIT FORMAT TEST ?
3613 034736 001007                BNE     35$                  ;YES !!
3614 034740 012737 010000 001444      MOV     #FMT16,RMOFO        ;SET 16 BIT FORMAT MODE AND
3615 034746 112737 000036 001420      MOVVB  #30.,RMDA0           ;LAST SECTOR
3616 034754 030667                BR      10$
```

```
3617                                35$:
3618 034756 032737 001000 001444      BIT     #SSEI,RMOFO         ;DONE 16 BIT TEST W/ SSEI SET ?
3619 034764 001007                BNE     40$                  ;YES !!
3620 034766 052737 001000 001444      BIS     #SSEI,RMOFO        ;SET SSEI W/ 16 BIT MODE AND
3621 034774 112737 000037 001420      MOVVB  #31.,RMDA0           ;LAST SECTOR
3622 035002 000654                BR      10$
```

```
3623                                40$:
3624
3625                                ;*****
3626                                ;*TEST 73      SET RMR TEST
3627                                ;*****
```

```
035004                                TST73:
035004 000004                                SCOPE                                ;SCOPE CALL
035006 000240                                NOP
035010 012706 001100      MOV     #STACK,SP           ;LOAD THE STACK POINTER
035014 013700 001276      MOV     $BASE,R0            ;R0 = UNIBUS ADDRESS
035020 013701 001466      MOV     TSTQUE,R1          ;R1 = POINTER TO DEVICE
035024 012737 000073 001226      MOV     #73,$TSTN          ;:SET TEST NUMBER IN APT MAIL BOX
3626
3627 035032 010037 001136      MOV     RO,SBADR           ;SETUP REGISTER ADDRESS FOR MSG
3628 035036 062737 000014 001136      ADD     #RMER1,SBADR
3629 035044 012702 035212      MOV     #100$,R2           ;INITIALIZE TABLE POINTER
```

```

3630
3631
3632 035050
3633 035050 00473 055156
3634 035054 012760 000001 000024
3635 035062 012760 041001 000024
3636 035070 012760 000000 000012
3637 035076 012760 000000 000042
3638
3639
3640 035104 012760 000001 000000
3641 035112 011203
3642 035114 060003
3643 035116 012713 041001
3644
3645
3646 035122 016037 000014 001142
3647 035130 042737 17773 001142
3648 035136 016237 000002 001140
3649 035144 023737 001140 001142
3650 035152 001411
3651 035154 011237 001174
3652 035160 032762 000004 000002
3653 035166 001002
3654 035170 104213
3655 035172 000401
3656 035174 104214
3657
3658
3659 035176
3660 035176 062702 000004
3661 035202 005712
3662 035204 100401
3663 035206 000720
3664 035210 000410
3665
3666
3667 035212
3668 035212 000016
3669 035214 000000
3670
3671 035216 000024
3672 035220 000000
3673
3674 035222 000006
3675 035224 000004
3676
3677 035226 177777
3678 035230 000000
3679
3680 035232
3681
3682

```

```

AND ENABLE DEF...
MOV #0,RMR1(RO) .LOAD RMR1
MOV #0,RMR2(RO) .LOAD RMR2
MOV #0,RMR1(RO) .LOAD RMR1
MOV #0,RMR2(RO) .LOAD RMR2
SET GO THEN WRITE THE REGISTER SPECIFIED BY THE TABLE
MOV #GO,RMC1(RO) .LOAD RMC1
MOV (R2),R3 .GENERATE REGISTER ADDRESS
ADD R0,R3
MOV #DMD'MUR'DBEN,(R3) .WRITE THE REGISTER
VERIFY RMR ACCORDING TO TABLE
MOV RMR1(RO),SBDDAT .STORE RMR1 AT SBDDAT
BIC #*(RMR,SBDDAT
MOV 2(R2),SGDDAT .GET EXPECTED RESULT FROM TABLE
CMP SGDDAT,SBDDAT
BEQ 30$ .BRANCH IF RMR IS OK
MOV (R2),STMPD .SAVE TEST REGISTER
BIT #RMR,2(R2)
BNE 20$ .BRANCH IF ERROR SHOULD BE ONE
EMT 213
BR 30$
20$: EMT 214
ADVANCE TABLE POINTER, EXIT IF DONE
30$: ADD #4,R2
TST (R2)
BMI 40$ .EXIT IF ENTRY NEGATIVE
BR 10$
40$: BR 200$ .JUMP OVER TABLE
TABLE OF REGISTER ADDRESSES AND RMR VALUES
100$: .WORD RMA$ .ATTENTION SUMMARY REG
.WORD 0 .RMR = 0
.WORD RMR1 .MAINTENANCE REG
.WORD 0 .RMR = 0
.WORD RMDA .DISK ADDRESS REG
.WORD RMR .RMR = 1
.WORD -1 .END OF TABLE
.WORD 0
200$: .END OF TEST
*****
**EST 74 PGM STATUS CHECK
*****

```

```

035232          574.
C35232 000004          SCOPE          :SCOPE CALL
035234 000240          NOP
035236 012706 001100  MOV          #STACK,SP      ;LOAD THE STACK POINTER
035242 013700 001276  MOV          $BASE,R0       ;R0 = UNIBUS ADDRESS
035246 013701 001466  MOV          TSTQUE,R1      ;R1 = POINTER TO DEVICE
035252 012737 000074 001226  MOV          #74,$TESTN     ;;SET TEST NUMBER IN APT MAIL BO
3683
3684          ;CLEAR AND READ DRIVE TYPE AND DRIVE STATUS
3685 J35260 004737 055156  JSR          PC,CNTCLR     ;GO CLEAR CONTROLLER
3686 J35264 016037 000026 001174  MOV          RMDT(R0),$TMPO ;STORE RMDT AT $TMPO
3687 J35272 016037 000012 001142  MOV          RMD5(R0),$DDAT ;STORE RMD5 AT $DDAT
3688
3689          ;COMIT TEST IF DRQ IS ON - ELSE VERIFY THAT PGM IS OFF
3690 C35300 032737 004000 001174  BIT          #DRQ,$TMPO
3691 035306 001014          BNE          10$          ;BRANCH IF DRQ IS ON
3692 035310 042737 176777 001142  BIC          #^CPGM,$DDAT
3693 035316 001410          BEQ          10$          ;BRANCH IF PGM IS OFF
3694 035320 005037 001140          CLR          $GDDAT
3695 035324 010037 001136  MOV          R0,$DADR
3696 035330 062737 000012 001134  ADD          #RMD5,$GADR
3697 035336 104215          EMT          215
3698 035340          10$:          ;END OF TEST
3699
5700          ;*****
          ;*TEST 75          DVA/DPR STATUS CHECK
          ;*****

```

```

035340          TST75.
035340 000004          SCOPE          :SCOPE CALL
035342 000240          NOP
035344 012706 001100  MOV          #STACK,SP      ;LOAD THE STACK POINTER
035350 013 00 001276  MOV          $BASE,R0       ;R0 = UNIBUS ADDRESS
035354 013/01 001466  MOV          TSTQUE,R1      ;R1 = POINTER TO DEVICE
035360 012737 000075 001226  MOV          #75,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
5701
5702          ;CLEAR AND VERIFY THAT DVA IS SET
5703 035366 004737 055156  JSR          PC,CNTCLR     ;GO CLEAR CONTROLLER
5704 035372 016037 000000 001142  MOV          RMC51(R0),$DDAT ;STORE RMC51 AT $DDAT
5705 035400 042737 173777 001142  BIC          #^CDVA,$DDAT
5706 035406 001006          BNE          10$          ;BRANCH IF DVA IS ON
5707 035410 012737 004000 001140  MOV          #DVA,$GDDAT
5708 035416 010037 001136  MOV          R0,$DADR
5709 035422 104216          EMT          216
5710
5711          ;VERIFY THAT DPR IS SET
5712          10$:
5713 035424 016037 000012 001142  MOV          RMD5(R0),$DDAT ;STORE RMD5 AT $DDAT
5714 035432 042737 177377 001142  BIC          #^CDPR,$DDAT
5715 035440 001011          BNE          20$
5716 035442 012737 000400 001140  MOV          #DPR,$GDDAT
5717 035450 010037 001136  MOV          R0,$DADR
5718 035454 062737 000012 001136  ADD          #RMD5,$DADR
5719 035462 104217          EMT          217
5720
5721 035464          20$:          ;END OF TEST
5722

```

3723

\*\*\*\*\*  
 \*TEST 76 PORT REQUEST TEST, PART 1  
 \*\*\*\*\*

TST76:

	035464					SCOPE		:SCOPE CALL
	035464	0000C'				NOP		
	035466	000240						
	035470	012706	001100			MOV	#STACK,SP	:LOAD THE STACK POINTER
	035474	013700	001276			MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	035500	013701	001466			MOV	TSTQUE,R1	:R1 = POINTER TO DEVICE
	035504	012737	000076	001226		MOV	#76,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
3724								
3725	035512	004737	054732			JSR	PC,SETVV	:GO SET VOLUME VALID
	035516	000402				BR	10\$	:BRANCH TO 10\$ IF NO ERROR
	035520	104000				EMT		
3726	035522	000434				BR	20\$	
3727								
3728								EXECUTE A RELEASE TO RESET REQUEST FLOP
3729	035524							10\$:
3730	035524	012760	000013	000000		MOV	#RELEASE!GO,RMCS1(R0)	:LOAD RMCS1
3731								
3732								:READ RMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET
3733	035532	016002	000040			MOV	RMR2(R0),R2	:STORE RMR2 AT R2
3734	035536	042702	037777			BIC	#C<RQA!RQB>,R2	
3735	035542	001024				BNE	20\$	
3736								
3737								:READ RMCS1 TO SET REQUEST FLOP
3738	035544	016002	000000			MOV	RMCS1(R0),R2	:STORE RMCS1 AT R2
3739								
3740								:VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET
3741	035550	016005	000012			MOV	RMS(R0),R5	:STORE RMS AT R5
3742	035554	032705	001000			BIT	#PGM,R5	:SEE IF PGM IS SET
3743	035560	001415				BEQ	20\$	:DONT TEST REQUEST IF PGM IS ZERO
3744	035562	016037	000040	001142		MOV	RMR2(R0),\$BDDAT	:STORE RMR2 AT \$BDDAT
3745	035570	042737	037777	001142		BIC	#C<RQA!RQB>,\$BDDAT	
3746	035576	001006				BNE	20\$	:BRANCH IF REQUEST IS SET
3747	035600	010037	001136			MOV	R0,\$BDDADR	
3748	035604	062737	000040	001136		ADD	#RMR2,\$BDDADR	
3749	035612	104220				EMT	220	
3750	035614							20\$:
3751								
3752								

\*\*\*\*\*  
 \*TEST 77 PORT REQUEST TEST, PART 2  
 \*\*\*\*\*

TST77:

	035614					SCOPE		:SCOPE CALL
	035614	000004				NOP		
	035616	000240						
	035620	012706	001100			MOV	#STACK,SP	:LOAD THE STACK POINTER
	035624	013700	001276			MOV	\$BASE,R0	:R0 = UNIBUS ADDRESS
	035630	013701	001466			MOV	TSTQUE,R1	:R1 = POINTER TO DEVICE
	035634	012737	000077	001226		MOV	#77,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX
3753								
3754	035642	004737	054732			JSR	PC,SETVV	:GO SET VOLUME VALID
	035646	000402				BR	10\$	:BRANCH TO 10\$ IF NO ERROR
	035650	104000				EMT		
3755	035652	000435				BR	20\$	

```
3756  
3757  
3758 035654 012760 000013 000000 :EXECUTE A RELEASE TO RESET REQUEST FLOP  
10$: MOV #RELEASE!GO,RMC51(R0) ;LOAD RM  
3760  
3761 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT  
3762 035662 016002 000040 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT  
3763 035666 042702 037777 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT  
3764 035672 001025 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT  
3765  
3766 :WRITE THE ATTENTION SUMMARY REGISTER TO SET REQUEST FLOP  
3767 035674 001025 000016 :WRITE THE ATTENTION SUMMARY REGISTER TO SET REQUEST FLOP  
3768  
3769 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3770 035700 016005 000012 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3771 035706 032705 001000 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3772 035712 001415 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3773 035714 016037 000040 001142 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3774 035722 042737 037777 001142 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3775 035730 001006 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3776 035732 010037 001136 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3777 035736 062737 000040 001136 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3778 035744 104221 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3779 035746 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3780  
3781
```

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

```
035746  
035746 000004  
035750 000240  
035752 012706 001100  
035756 013700 001276  
035762 013701 001466  
035766 012737 000100 001226  
5782  
3783 035774 004737 054732 :SCOPE CALL  
036000 000402 :LOAD THE STACK POINTER  
036002 104000 :R0 = LWIBUS ADDRESS  
3784 036004 000435 :R1 = POINTER TO DEVICE  
3785 :SET TEST NUMBER IN APT MAIL BLK  
3786 :EXECUTE A RELEASE COMMAND TO RESET REQUEST FLOP  
3787 036006 :EXECUTE A RELEASE COMMAND TO RESET REQUEST FLOP  
3788 036006 012760 000013 000000 :EXECUTE A RELEASE COMMAND TO RESET REQUEST FLOP  
3789  
3790 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET  
3791 036014 016002 000040 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET  
3792 036020 042702 037777 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET  
3793 036024 001025 :READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET  
3794  
3795 :WRITE RMDA TO SET REQUEST FLOP  
3796 036026 012760 000000 000006 :WRITE RMDA TO SET REQUEST FLOP  
3797  
3798 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3799 036034 016005 000012 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
3800 036040 032705 001000 :VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET
```

```
10$: MOV #RELEASE!GO,RMC51(R0) ;LOAD RMC51  
:READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET  
MOV RMMR2(R0),R2 ;STORE RMMR2 AT R2  
BIC #C<RQA!RQB>,R2  
BNE 20$  
:WRITE RMDA TO SET REQUEST FLOP  
MOV #0,RMDA(R0) ;LOAD RMDA  
:VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET  
MOV RMD5(R0),R5 ;STORE RMD5 AT R5  
BIT #PGM,R5 ;SEE IF PGM IS SET
```

3801	036044	001415				BEQ	20\$	:DONT TEST REQUEST IF PGM IS ZERO
3802	036046	016037	000040	001142		MOV	RMPR2(RO),SBDDAT	:STORE RMPR2 AT SBDDAT
3803	036054	042737	037777	001142		BIC	#C<RQA!ROB>,SBDDAT	
3804	036062	001006				BNE	20\$	
3805	036064	010037	001136			MOV	RC,SBDDADR	
3806	036070	062737	000040	001136		ADD	#RMPR2,SBDDADR	
3807	036076	104222				EMT	222	

3808  
 3809 036100  
 3810  
 3811

20\$:  
 :\*\*\*\*\*  
 :\*TEST 101 RELEASE TEST  
 :\*\*\*\*\*

	036100							TST101:
	036100	000004					SCOPE	:SCOPE CALL
	036102	000240					NOB	
	036104	012706	001100				MOV	#STACK,SP :LOAD THE STACK POINTER
	036110	013700	001276				MOV	SBASE,RO :RO = UNIBUS ADDRESS
	036114	013701	001466				MOV	TSTQUE,R1 :R1 = POINTER TO DEVICE
	036120	012737	000101	001226			MOV	#101,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

3812  
 3813  
 3814  
 3815 036126 004737 054732

:REQUEST FLOP SHOULD SET WHEN WRITTING RMCS1  
 JSR PC,SETVV :GO SET VOLUME VALID  
 BR 10\$ :BRANCH TO 10\$ IF NO ERROR  
 EMT  
 BR 40\$

3816	036136	000454						
3817								
3818								:EXECUTE A RELEASE COMMAND
3819	036140						10\$:	
3820	036140	012760	041001	000024			MOV	#DMD!DBEN!MUR,RMPR1(RO) :LOAD RMPR1
3821	036146	012760	000013	000000			MOV	#RELEASE!GO,RMCS1(RO) :LOAD RMCS1
3822	036154	012702	000002				MOV	#2,R2
3823	036160						20\$:	

3824	036160	012760	141001	000024			MOV	#DMD!DBEN!MUR!DBCK,RMPR1(RO) :LOAD RMPR1
3825	036166	012760	041001	000024			MOV	#DMD!DBEN!MUR,RMPR1(RO) :LOAD RMPR1
3826	036174	005302					DEC	R2
3827	036176	001370					BNE	20\$ :ISSUE 2 CLOCKS

3828								:VERIFY REQUEST FLOPS ARE RESET
3829	036200	016037	000026	001174			MOV	RMDT(RO),STMP0 :STORE RMDT AT STMP0
3830	036206	016037	000040	001142			MOV	RMPR2(RO),SBDDAT :STORE RMPR2 AT SBDDAT
3831	036214	042737	037777	001142			BIC	#C<RQA!ROB>,SBDDAT
3832	036222	001422					BEQ	40\$ :BRANCH IF REQUESTS ARE RESET
3833	036224	032737	004000	001174			BIT	#DRQ,STMP0
3834	036232	001410					SEQ	30\$ :BRANCH IF SINGLE PORT DEVICE
3835	036234	032737	100000	001142			BIT	#RQA,SBDDAT
3836	036242	001412					BEQ	40\$ :BRANCH IF RQA IS RESET
3837	036244	032737	040000	001142			BIT	#ROB,SBDDAT
3838	036252	001406					BEQ	40\$ :BRANCH IF ROB IS RESET

3839  
 3840  
 3841  
 3842 036254  
 3843 036254 010037 001136  
 3844 036260 062737 000040 001136

:DRQ IS ZERO AND A REQUEST FLOP IS ON, OR, DRQ IS ONE AND  
 :BOTH REQUEST FLOPS ARE ON  
 30\$:  
 MOV RO,SBDDADR  
 ADD #RMPR2,SBDDADR

3845 036266 104223  
3846 036270  
3847  
3848

408: EMT 223 :END OF TEST

\*\*\*\*\*  
: \*TEST 102 WRITE ATA TEST

\*\*\*\*\*  
TST102:  
SCOPE :SCOPE CALL  
NOP  
MOV #STACK,SP :LOAD THE STACK POINTER  
MOV \$BASE,R0 :R0 = UNIBUS ADDRESS  
MOV TSTQUE,R1 :R1 = POINTER TO DEVICE  
MOV #102,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

036270  
036270 000004  
036272 000240  
036274 012706 001100  
036300 013700 001276  
036304 013701 001466  
036310 012737 000102 001226

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

3849  
3850  
3851 036316 004737 055156  
3852 036322 012760 000001 000024  
3853 036330 012760 000000 000014  
3854 036336 012760 000000 000042  
3855 036344 012760 001001 000024  
3856  
3857

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

:WRITE THE ATTENTION SUMMARY REGISTER

3858 036352 111102  
3859 036354 042702 177770  
3860 036360 116203 0637 0  
3861 036364 042703 177400  
3862 036370 010360 000016  
3863  
3864

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

:READ RMD5 AND VERIFY THAT ATA IS RESET

3865 036374 016037 000012 001142  
3866 036402 042737 077777 001142  
3867 036410 001411  
3868 036412 010037 001136  
3869 036416 062737 000012 001136  
3870 036424 005037 001140  
3871 036430 034224  
3872 036432 070417  
3873  
3874

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

:READ RMAS AND VERIFY ATA IS RESET

3875 036434  
3876 036434 016037 000016 001142  
3877 036442 005103  
3878 036444 040337 001142  
3879 036450 001410  
3880 036452 005037 001140  
3881 036456 010037 001136  
3882 036462 062737 000016 001136  
3883 036470 104225  
3884 036472  
3885  
3886

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

20\$: :END OF TEST

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

\*\*\*\*\*  
TST103: SCOPE :SCOPE CALL

036472  
036472 000004

```

036474 000240      NOP
036476 012706 001100      MOV #STACK,SP      ;LOAD THE STACK POINTER
036502 013700 001276      MOV $BASE,R0       ;R0 = UNIBUS ADDRESS
036506 013701 001466      MOV TSTQUE,R1      ;R1 = POINTER TO DEVICE
036512 012737 000103 001226      MOV #103,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

3887
3888 036520 004737 055156      JSR PC,CNTCLR      ;GO CLEAR CONTROLLER
3889 036524 012760 000001 000024      MOV #DMD,RMR1(RO) ;LOAD RMR1
3890 036532 012760 000000 000014      MOV #0,RMER1(RO)  ;LOAD RMER1
3891 036540 012760 000000 000042      MOV #0,RMER2(RO)  ;LOAD RMER2
3892 036546 012760 041001 000024      MOV #DMD!MUR!DBEN,RMR1(RO);LOAD RMR1
3893
3894 ;WITH DEBUG CLOCK ENABLED, SET GO AND VERIFY THAT ATA IS RESET
3895 036554 012760 000001 000000      MOV #GO,RMCS1(RO) ;LOAD RMCS1
3896 036562 016037 000012 001142      MOV RMD5(RO),$BDDAT ;STORE RMD5 AT $BDDAT
3897 036570 042737 077777 001142      BIC #^CATA,$BDDAT
3898 036576 001410      BEQ 10$           ;BRANCH IF ATA IS RESET
3899 036600 005037 001140      CLR $GDDAT
3900 036604 010037 001136      MOV R0,$BDADR
3901 036610 062737 000012 001136      ADD #RMD5,$BDADR
3902 036616 104226      EMT 226
3903
3904 036620      10$: ;END OF TEST
3905
3906 ;*****
;*TEST 104 UNIT READY ATA TEST
;*****

TST104:
036620 000004      SCOPE ;SCOPE CALL
036622 000240      NOP
036624 012706 001100      MOV #STACK,SP      ;LOAD THE STACK POINTER
036630 013700 001276      MOV $BASE,R0       ;R0 = UNIBUS ADDRESS
036634 013701 001466      MOV TSTQUE,R1      ;R1 = POINTER TO DEVICE
036640 012737 000104 001226      MOV #104,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX

3907
3908 ;SET DIAGNOSTIC MODE AND CLEAR ATA
3909 036646 004737 055156      JSR PC,CNTCLR      ;GO CLEAR CONTROLLER
3910 036652 012760 000001 000024      MOV #DMD,RMR1(RO) ;LOAD RMR1
3911 036660 012760 000000 000014      MOV #0,RMER1(RO)  ;LOAD RMER1
3912 036666 012760 000000 000042      MOV #0,RMER2(RO)  ;LOAD RMER2
3913 036674 012760 177777 000016      MOV #-1,RMAS(RO)  ;LOAD RMAS
3914
3915 ;SET UNIT READY AND VERIFY ATA IS SET
3916 036702 012760 001001 000024      MOV #DMD!MUR,RMR1(RO) ;LOAD RMR1
3917 036710 016037 000012 001142      MOV RMD5(RO),$BDDAT ;STORE RMD5 AT $BDDAT
3918 036716 042737 077777 001142      BIC #^CATA,$BDDAT
3919 036724 001011      BNE 10$           ;BRANCH IF ATA IS SET
3920 036726 010037 001136      MOV R0,$BDADR
3921 036732 062737 000012 001136      ADD #RMD5,$BDADR
3922 036740 012737 100000 001140      MOV #ATA,$GDDAT
3923 036746 104227      EMT 227
3924
3925 ;CLEAR ATA, RESET UNIT READY, AND VERIFY ATA IS SET
3926 036750      10$:
3927 036750 012760 177777 000016      MOV #-1,RMAS(RO)  ;LOAD RMAS
3928 036756 012760 000001 000024      MOV #DMD,RMR1(RO) ;LOAD RMR1
    
```



```

3929 036764 016037 000012 001142      MOV      RMD5(RO),SDDAT ;STORE RMD5 AT SDDAT
3930 036772 042737 077777 001142      BIC      #^CATA,SDDAT
3931 037000 001011                BNE      20$
3932 037002 010037 001136                MOV      RO,$BDADR
3933 037006 062737 000012 001136      ADD      #RMD5,$BDADR
3934 037014 012737 100000 001140      MOV      #ATA,$GDDAT
3935 037022 104230                EMT      230
  
```

3937 037024 20\$: ;END OF TEST

```

3938
3939
:*****
:*TEST 105      ERROR ATA TEST
:*****
  
```

```

037024
037024 000004                SCOPE
037026 000240                NOP      ;SCOPE CALL
037030 012706 001100      MOV      #STACK,SP ;LOAD THE STACK POINTER
037034 013700 001276      MOV      $BASE,RO ;RO = UNIBUS ADDRESS
037040 013701 001466      MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
037044 012737 000105 001226  MOV      #105,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
  
```

```

3940
3941
3942 037052 004737 055156      JSR      PC,CNTCLR ;GO CLEAR CONTROLLER
3943 037056 012760 000001 000024  MOV      #DMD,RMR1(RO) ;LOAD RMR1
3944 037064 012760 000000 000014  MOV      #0,RMR1(RO) ;LOAD RMR1
3945 037072 012760 000000 000042  MOV      #0,RMR2(RO) ;LOAD RMR2
3946 037100 012760 177777 000016  MOV      #-1,RMAS(RO) ;LOAD RMAS
  
```

```

3947
3948
3949 037106 012760 177777 000014  MOV      #-1,RMR1(RO) ;LOAD RMR1
3950 037114 016037 000012 001142  MOV      RMD5(RO),SDDAT ;STORE RMD5 AT SDDAT
3951 037122 042737 077777 001142  BIC      #^CATA,SDDAT
3952 037130 001011                BNE      10$
3953 037132 012737 100000 001140  MOV      #ATA,$GDDAT
3954 037140 010037 001136                MOV      RO,$BDADR
3955 037144 062737 000012 001136  ADD      #RMD5,$BDADR
3956 037152 104231                EMT      231
  
```

3958 037154 10\$: ;END OF TEST

```

3959
3960
:*****
:*TEST 106      REGISTER TRANSFER ATA TEST
:*****
  
```

```

037154
037154 000004                SCOPE
037156 000240                NOP      ;SCOPE CALL
037160 012706 001100      MOV      #STACK,SP ;LOAD THE STACK POINTER
037164 013700 001276      MOV      $BASE,RO ;RO = UNIBUS ADDRESS
037170 013701 001466      MOV      TSTQUE,R1 ;R1 = POINTER TO DEVICE
037174 012737 000106 001226  MOV      #106,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
  
```

```

3961
3962 037202 012702 037330      MOV      #100$,R2 ;INITIALIZE TABLE ADDRESS
  
```

```

3963
3964
3965 037206      ;FORCE COMPOSITE ERROR AND RESET ATA
5$:
  
```

```

3966 037206 004737 055156 JSR PC,ENTCLR ;GO CLEAR CONTROLLER
3967 037212 012760 000001 000024 MOV #DMD,RMR1(RO) ;LOAD RMR1
3968 037220 012760 177777 000014 MOV #-1,RMR1(RO) ;LOAD RMR1
3969 037226 012760 177777 000016 MOV #-1,RMS(RO) ;LOAD RMS
3970
3971 ;WRITE THE TEST REGISTER AND VERIFY ATA
3972 037234 011203 MOV (R2),R3 ;GENERATE REGISTER ADDRESS
3973 037236 060003 ADD R0,R3
3974 037240 005013 CLR (R3)
3975 037242 016037 000012 001142 MOV RMS(RO),SDDAT ;STORE RMS AT SDDAT
3976 037250 042737 077777 001142 BIC #^ATA,SDDAT
3977 037256 016237 000002 001140 MOV 2(R2),SDDAT
3978 037264 023737 001140 001142 CMP SDDAT,SDDAT
3979 037272 001410 BEQ 10$ ;BRANCH IF ATA IS OK
3980 037274 010337 001174 MOV R3,$TMP0
3981 037300 010037 001136 MOV R0,$DADR
3982 037304 062737 000012 001136 ADD #RMS,$DADR
3983 037312 104232 EMT 232
3984
3985 ;MOVE TABLE POINTER - EXIT IF DONE
3986 037314 10$:
3987 037314 062702 000004 ADD #4,R2
3988 037320 005712 TST (R2)
3989 037322 100401 BMI 20$ ;EXIT IF ENTRY MINUS
3990 037324 000730 BR 5$
3991 037326 000410 BR 200$ ;JUMP OVER TABLE
3992
3993 ;TABLE OF REGISTER ADDRESSES AND ATA BITS
3994 037330 100$:
3995 037330 000016 .WORD RMS
3996 037332 000000 .WORD
3997
3998 037334 000006 .WORD RMDA
3999 037336 100000 .WORD ATA
4000
4001 037340 000000 .WORD RMS1
4002 037342 000000 .WORD
4003
4004 037344 177777 .WORD -1 ;END OF TABLE
4005 037346 000000 .WORD
4006
4007 037350 200$: ;END OF TEST
4008
4009 ;*****
; *TEST 107 P SET ATA TEST
;*****
037350 TST107:
037350 000004 SCOPE ;SCOPE CALL
037352 000240 NOP
037354 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
037360 013700 001276 MOV SBASE,R0 ;R0 = UNIBUS ADDRESS
037364 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
037370 012737 000107 00 226 MOV #107,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
4010
4011 037376 012702 037540 10$: MOV #100$,R2 ;INITIALIZE TABLE POINTER
4012 037402

```

```

4013 037402 J04737 054732 JSR PC,SETVV ;GO SET VOLUME VALID
      037406 000402 BR 20$ ;BRANCH TO 20$ IF NO ERROR
      037410 104000 EMT
4014 037412 000451 BR 50$
4015
4016 ;EXECUTE THE COMMAND FROM THE TABLE
4017 037414 20$:
4018 037414 012760 041001 000024 MOV #DMD:MLR:DBEN,RMPR1(R0) ;LOAD RMPR1
4019 037422 011203 MOV (R2),R3 ;GET FUNCTION CODE
4020 037424 052703 000001 B:5 #GO,R3
4021 037430 010360 000000 MOV P3,RMCS1(R0) ;LOAD RMCS1
4022 037434 012703 000003 MOV #3,R3
4023 037440 30$:
4024 037440 012760 141001 000024 MOV #DMD:MUR:DBEN:DBCK,RMPR1(R0) ;LOAD RMPR1
4025 037446 012760 041001 000024 MOV #DMD:MUR:DBEN,RMPR1(R0) ;LOAD RMPR1
4026 037454 005303 DEC R3
4027 037456 001370 BNE 30$
4028
4029 ;VERIFY THAT ATA IS SET
4030 037460 016037 000012 001142 MOV RMD5(R0),%BDDAT ;STORE RMD5 AT %BDDAT
4031 037466 042737 077777 001142 BIC #^CATA,%BDDAT
4032 037474 001013 BNE 40$
4033 037476 010037 001136 MOV R0,%BDADR
4034 037502 062737 000012 001136 ADD #RMD5,%BDADR
4035 037510 012737 000000 001140 MOV #ATA,%GDDAT
4036 037516 011237 001174 MOV (R2),%TMPO
4037 037522 104233 EMT 233
4038
4039 ;ADVANCE TABLE POINTER-EXIT IF DONE
4040 037524 40$:
4041 037524 062702 000002 ADD #2,R2
4042 037530 005712 TST (R2)
4043 037532 100401 BMI 50$
4044 037534 000722 BR 10$
4045 037536 000403 50$: BR 200$ ;JUMP OVER TABLE
4046
4047 ;TABLE OF FUNCTION CODES
4048 037540 100$:
4049 037540 000014 .WORD OFFSET
4050 037542 000016 .WORD RTC
4051 037544 177777 .WORD -1 ;END OF TABLE
4052 037546 200$:
4053
4054 ;*****
; *TEST 110 SET WLE TEST
;*****
037546 TST110:
037546 000004 SCOPE ;SCOPE CALL
037550 000240 NOP
037552 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
037556 013700 001276 MOV %BASE,R0 ;R0 = UNIBUS ADDRESS
037562 013701 001466 MOV %TSTQUE,R1 ;R1 = POINTER TO DEVICE
037566 012737 000110 001126 MOV #110,%TESTN ;;SET TEST NUMBER IN APT MAIL BOX
4055
4056 037574 012702 037770 10$ MOV #100$,R2 ;INITIALIZE TABLE POINTER
4057 037600

```

```

4058 037600 004737 054732      JSR      PC,SETVV      ;GO SET VOLUME VALID
      037604 000402      BR        20$         ;BRANCH TO 20$ IF NO ERROR
      037606 104000      EMT
4059 037610 000466      BR        50$
4060
4061      ;ENABLE DEBUG CLOCK AND SET WRITE PROTECT ACCORDING TO TABLE
4062 037612 20$:
4063 037612 016203 000002      MOV      2(R2),R3      ;GET WRITE PROTECT FROM TABLE
4064 037616 052703 041001      BIS      #DMD'PUR'DBEN,R3 ;SET OTHER MAINT BITS
4065 037622 010360 000024      MOV      R3,RMPR1(RO) ;LOAD RMPR1
4066
4067      ;LOAD AND EXECUTE THE COMMAND FROM THE TABLE
4068 037626 011204      MOV      (R2),R4      ;GET FUNCTION CODE FROM TABLE
4069 037630 052704 000001      BIS      #GO,R4      ;SET GO
4070 037634 010460 000000      MOV      R4,RMCS1(RO) ;LOAD RMCS1
4071 037640 012705 000002      MOV      #2,R5      ;R5=CLOCK COUNT
4072 037644 052703 100000 30$:      BIS      #DBCK,R3      ;SET CLOCK
4073 037650 010360 000024      MOV      R3,RMPR1(RO) ;LOAD RMPR1
4074 037654 042703 100000      BIC      #DBCK,R3      ;RESET LOCK
4075 037660 010360 000024      MOV      R3,RMPR1(RO) ;LOAD RMPR1
4076 037664 005305      DEC      R5
4077 037666 001366      BNE      30$         ;ISSUE 2 CLOCKS
4078
4079      ;VERIFY THAT WRITE LOCK ERROR IS ACCORDING TO TABLE
4080 037670 016037 000014 001142      MOV      RMPR1(RO),%BDDAT ;STORE RMPR1 AT %BDDAT
4081 037676 042737 173777 001142      BIC      #^CWLE,%BDDAT
4082 037704 026237 000004 001142      CMP      4(R2),%BDDAT
4083 037712 001417      BEQ      40$         ;BRANCH IF WLE IS OK
4084 037714 016237 000004 001140      MOV      4(R2),%GDDAT ;SAVE DATA FOR ERROR MSG
4085 037722 010037 001136      MOV      RO,%BDADR
4086 037726 062737 000014 001136      ADD      #RMPR1,%BDADR
4087 037734 011237 001174      MOV      (R2),%STMP0 ;%STMP0=FUNCTION CODE
4088 037740 016237 000002 001176      MOV      2(R2),%STMP1 ;%STMP1=WRITE PROTECT
4089 037746 104234      EMT
4090 037750 000406      BR        50$
4091
4092      ;ADVANCE TABLE POINTER TO NEXT FUNCTION CODE-EXIT IF DONE
4093 037752 40$:
4094 037752 062702 000006      ADD      #6,R2
4095 037756 005762 000002      TST      2(R2)
4096 037762 100401      BMI      50$
4097 037764 000705      BR        10$        ;REPEAT TEST
4098
4099 037766 000416 50$:      BR        200$       ;JUMP OVER TABLE
4100
4101      ;TABLE OF FUNCTION CODES, WRITE PROTECT AND WRITE LOCK ERRORS
4102 037770 100$:
4103 037770 000060      .WORD   WD           ;WRITE DATA COMMAND
4104 037772 000010      .WORD   MWP         ;WRITE PROTECT ON
4105 037774 004000      .WORD   WLE         ;WRITE LOCK ERROR ONE
4106
4107 037776 000060      .WORD   WD           ;WRITE DATA COMMAND
4108 040000 000000      .WORD   MWP         ;MWP OFF
4109 040002 000000      .WORD   WLE         ;WLE OFF
4110
4111 040004 000070      .WORD   RD           ;READ DATA COMMAND
4112 040006 000010      .WORD   MWP         ;MWP ON
    
```

```

4113 040010 000000 .WORD .WLE OFF
4114 .WORD RIP .READ IN PRESET COMMAND
4115 040012 000020 .WORD MWP ;MWP ON
4116 040014 000010 .WORD ;WLE OFF
4117 040016 000000 .WORD
4118 .WORD .END OF TABLE
4119 040020 000000 .WORD -1
4120 040022 177777 .WORD
4121 200$: ;END OF TEST
4122 040024 ;*****
4123 ;*TEST 111 EXCEPTION TEST
4124 ;*****

040024 ;*****
040024 000004 SCOPE ;SCOPE CALL
040026 000240 NOP
040030 012706 001100 MOV #STACK.SP ;LOAD THE STACK POINTER
040033 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
040040 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
040044 012737 000111 001226 MOV #111,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

4125 ;WITH OCCUPIED SET, EACH OF THE FOLLOWING ERRORS SHOULD CAUSE AN
4126 ;EXCEPTION:
4127 :
4128 : PAR, IF RUN AND GO IS SET
4129 : RMR, IF RUN AND GO IS SET
4130 : ILR, IF RUN AND GO IS SET
4131 :
4132 : DCK
4133 : HCE
4134 : HCRC
4135 : FER
4136 : OPI
4137 :
4138 040052 012702 040324 MOV #100$,R2 ;INITIALIZE TABLE POINTER
4139
4140 ;INITIALIZE AND VERIFY THAT EXCEPTION IS RESET
4141 10$:
4142 040056 004737 055156 JSR PC,ENTCLR ;GO CLEAR CONTROLLER
4143 040062 016037 000024 001142 MOV RMR1(R0),$BDDAT ;STORE RMR1 AT $BDDAT
4144 040070 042737 167777 001142 BIC #*CREX,$BDDAT
4145 040076 001410 BEQ 20$ ;BRANCH IF EXCEPTION IS 0
4146 040100 010037 001136 MOV R0,$BDADR
4147 040104 062737 000024 001136 ADD #RMR1,$BDADR
4148 040112 005037 001140 CLR $GDDAT
4149 040116 104235 EMT 235
4150 040120 20$:
4151 040120 004737 054732 JSR PC,SETVV ;GO SET VOLUME VALID
4152 040124 000402 BR 30$ ;BRANCH TO 30$ IF NO ERROR
4153 040126 104000 EMT
4154 040130 000474 BR 60$
4155 ;EXECUTE A WRITE DATA COMMAND TO SET OCCUPIED, RUN AND GO
4156 30$:
4157 040132 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
040140 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
  
```

ZRMBAU RMBU DSKLS PT1 MACRO V04.00 14-JAN-82 16:33:00 PAGE 13-87  
1111 EXCEPTION TEST

```

4158 040146 012760 041001 000024      MOV      #DMD:MUR:DBEN,RMMR1(R0) ;LOAD RMMR1
4159 040154 012760 000061 000000      MOV      #WD:GO,RMCS1(R0)      ;LOAD RMCS1
4160 040162 012703 000002                MOV      #2,R3
4161 040166                40$:    MOV      #DMD:MUR:DBEN:DBCK,RMMR1(R0) ;LOAD RMMR1
      040166 012760 141001 000024      MOV      #DMD:MUR:DBEN,RMMR1(R0) ;LOAD RMMR1
4162 040174 012760 041001 000024      DEC      R3
4163 040202 005303                BNE     40$                      ;ISSUE 2 CLOCKS
4164 040204 001370
4165
4166                ;LOAD ERROR REGISTER WITH ENTRY FROM TABLE AND VERIFY THAT EXCEPTION IS SET
4167 040206 011260 000014      MOV      (R2),RMR1(R0)      ;LOAD RMR1
4168 040212 012737 000200 001534      MOV      #200,WATCH        ;SET WATCHDOG TIMER VALUE
      040220 004777 141312                JSR      PC,@CLOCK          ;START THE CLOCK
4169 040224                45$:    MOV      RMMR1(R0),SBDAT      ;STORE RMMR1 AT SBDAT
      040224 016037 000024 001142      BIC     #^CREX,SBDAT
4170 040232 042737 167777 001142      BNE     50$
4171 040240 001021                TST     WATCH                ;HAS CLOCK EXPIRED ??
4172 040242 005737 001534      BNE     45$                  ;NO - TAKE ANOTHER SAMPLE
4173 040246 001366                JSR     PC,@STOPCL          ;STOP THE CLOCK
4174 040250 004777 141264      MOV     #REX,$GDDAT
4175 040254 012737 010000 001140      MOV     R0,$SDADR
4176 040262 010037 001136      ADD     #RMMR1,$SDADR
4177 040266 062737 000024 001136      MOV     (R2),$TMP0
4178 040274 011237 001174      EMT     236
4179 040300 104236      BR      60$
4180 040302 000407
4181
4182                ;ADVANCE TABLE POINTER-EXIT IF DONE
4183 040304                50$:    JSR     PC,@STOPCL          ;STOP THE CLOCK
4184 040304 004777 141230      ADD     #2,R2
4185 040310 062702 000002      TST     (R2)
4186 040314 005712      BEQ     60$
4187 040316 001401      BR      10$
4188 040320 000656      BR      10$
4189 040322 000402                60$:    BR      200$              ;JUMP OVER TABLE
4190
4191                ;PRESENTLY, THE TABLE HAS ONLY ONE ENTRY, THE TEST USING RMR. THE
4192                ;TABLE SHOULD BE EXPANDED TO TEST ALL THE CONDITIONS LISTED ABOVE IF
4193                ;A HARDWARE CHANGE IS MADE SUCH THAT IT IS POSSIBLE TO WRITE ERROR
4194                ;REGISTER 1 WITH GO SET.
4195 040324                100$:   .WORD   RMR                  ;RMR IS CAUSED BY HARDWARE
4196 040324 000004      .WORD
4197
4198 040326 000000                200$:   .WORD
4199 040330      .WORD                  ;END OF TEST
4200
4201                ;*****
                ;*TEST 112      RECALIBRATE TEST
                ;*****
TST112:
040330                SCOPE                  ;SCOPE CALL
040330 000004      NOP
040332 000240      MOV     #STACK,SP          ;LOAD THE STACK POINTER
040334 012706 001100      MOV     $BASE,R0           ;R0 = UNIBUS ADDRESS
040340 013700 001276      MOV     TSTQUE,R1         ;R1 = POINTER TO DEVICE
040344 013701 001466      MOV
040350 012737 000112 001226      MOV     #112,$TSTN        ;:SET TEST NUMBER IN APT MAIL BOX

```

```

4206 040356 040737 054732
4207 040362 000403
4208 040364 104000
4209 040366 000137 041664
4210 040372
4211 040372 012760 041001 000024
4212 040400 012760 000000 000014
4213 040406 012760 000000 000042
4214 040414 012760 000007 000000
4215
4216
4217 040422 012702 000002
4218 040426
4219 040426 012760 141001 000024
4220 040434 012760 041001 000024
4221 040442 005302
4222 040444 001370
4223
4224 040446 012760 040001 000024
4225 040454 012702 000002
4226 040460
4227 040466 012760 140001 000024
4228 040466 012760 040001 000024
4229 040474 005302
4230 040476 001370
4231
4232 040500 016037 000014 001142
4233 040506 042737 157777 001142
4234 040514 001011
4235 040516 012737 020000 001140
4236 040524 010037 001136
4237 040530 062737 000014 001136
4238 040536 104241
4239
4240 040540 012737 040546 001124
4241
4242
4243 040546
4244 040546 004737 054732
4245 040552 000403
4246 040554 104000
4247 040554 000137 041664
4248 040562
4249 040562 012760 041001 000024
4250 040570 012760 000000 000014
4251 040576 012760 000000 000042
4252 040604 012760 000007 000000

```

VERIFY THAT OPI SETS IF UNIT READY DROPS AFTER DECODE

```

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

```

ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND

```

05$:
MOV #DMD:MUR'DBEN,RMR1(R0) ;LOAD RMR1
MOV #0,RMR1(R0) ;LOAD RMR1
MOV #0,RMR2(R0) ;LOAD RMR2
MOV #RECAL!GO,RMCS1(R0) ;LOAD RMCS1

```

STEP COMMAND SEQUENCER TO RECAL COM (2 CLOCKS)

```

MOV #2,R2
15$:
MOV #DMD:MUR'DBEN'DBCK,RMR1(R0) ;LOAD RMR1
MOV #DMD:MUR'DBEN,RMR1(R0) ;LOAD RMR1
DEC R2
BNE 20$

```

DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)

```

MOV #CMD!DBEN,RMR1(R0) ;LOAD RMR1
MOV #2,R2
30$:
MOV #DMD!DBFN!DBCK,RMR1(R0) ;LOAD RMR1
MOV #DMD!DBEN,RMR1(R0) ;LOAD RMR1
DEC R2
BNE 30$

```

VERIFY THAT OPI IS SET

```

MOV RMR1(R0),%BDDAT ;STORE RMR1 AT %BDDAT
BIC #%OPI,%BDDAT
BNE 40$
MOV #OPI,%GDDAT
MOV R0,%BDADR
ADD #RMR1,%BDADR
EMT 241

```

40\$: MOV #50\$,%LPERR ;CHANGE LOOP ON ERROR ADDRESS

VERIFY THAT RECALIBRATE ABORTS DURING EXECUTION

```

50$:
JSR PC,SETVV ;GO SET VOLUME VALID
BR 60$ ;BRANCH TO 60$ IF NO ERROR
EMT
JMP 330$

```

ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND

```

60$:
MOV #DMD:MUR'DBEN,RMR1(R0) ;LOAD RMR1
MOV #0,RMR1(R0) ;LOAD RMR1
MOV #0,RMR2(R0) ;LOAD RMR2
MOV #RECAL!GO,RMCS1(R0) ;LOAD RMCS1

```

```

4253
4254
4255 040612 012702 000003 ;STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
4256 040616 70$: MOV #3,R2
4257 040616 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
4258 040624 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4259 040632 005302 DEC R2
4260 040634 001370 BNE 70$
4261
4262 ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
4263 040636 012760 041101 0002- MOV #DMD!MUR!DBEN!MDF,RMMR1(RO) ;LOAD RMMR1
4264
4265 ;STEP 2 CLOCKS AND VERIFY GC IS RESET
4266 040644 012702 000002 MOV #2,R2
4267 040650 80$: MOV #DMD!MUR!DBEN!MDF!DBCK,RMMR1(RO) ;LOAD RMMR1
4268 040656 012760 141101 000024 MOV #DMD!MUR!DBEN!MDF,RMMR1(RO) ;LOAD RMMR1
4269 040664 005302 DEC R2
4270 040666 001370 BNE 80$
4271 040670 016037 000000 001142 MOV RMCS1(RO),%BDDAT ;STORE RMCS1 AT %BDDAT
4272 040676 042737 177776 001142 BIC %CGO,%BDDAT
4273 040704 001405 BEQ 90$
4274 040706 010037 001136 MOV RO,%BDADR
4275 040712 005037 001140 CLR %GDDAT
4276 040716 104242 EMT 242
4277
4278 040720 012737 040726 001124 90$: MOV #100$,%LPERR ;CHANGE LOOP ON ERROR ADDRESS
4279
4280 ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESN. CLEAR
4281 040726 100$:
4282 040726 004737 054732 ISR PC,SETVV ;GO SET VOLUME VALID
4283 040732 000403 BR 110$ ;BRANCH TO 110$ IF NO ERROR
4284 040734 104000 EMT
4285 040736 000137 041664 JMP 330$
4286
4287 ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
4288 110$: MOV #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4289 040742 012760 041001 000024 MOV #0,RMER1(RO) ;LOAD RMER1
4290 040750 012760 000000 000014 MOV #0,RMER2(RO) ;LOAD RMER2
4291 040756 012760 000000 000042 MOV #RECAL!GC RMCS1(RO) ;LOAD RMCS1
4292 040764 012760 000007 000000
4293
4294 ;STEP THE COMMAND SEQUENCER
4295 040772 012702 000017 MOV #15,,R2
4296 120$: MOV #DMD!MUR!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
4297 040776 012760 141001 000024 MOV #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4298 041004 012760 041001 000024 DEC R2
4299 041012 005302 BNE 120$
4300
4301 ;VERIFY THAT OPI IS SET
4302 041016 016037 000014 001142 MOV RMER1(RO),%BDDAT ;STORE RMER1 AT %BDDAT
4303 041024 042737 157777 001142 BIC %COPI,%BDDAT
4304 041032 001011 BNE 130$
4305 041034 010037 001136 MOV RO,%BDADR
4306 041040 062737 000014 001136 ADD #RMER1,%BDADR
4307 041046 012737 020000 001140 MOV #OPI,%GDDAT

```



```

4307 041054 104243          EMT      2:3
4308
4309 041056 012737 041064 001124 130$:  MOV      #150$, $LPERR      ;CHANGE LOOP ON ERROR ADDRESS
4310
4311          ;VERIFY ATA SETS IF DRIVE COMPLETES RECALIBRATE (ON CYLINDER SETS)
4312 041064          150$:
4313 041064 004737 054732      JSR      PC, SETVV      ;GO SET VOLUME VALID
      041070 000403      BR       160$          ;BRANCH TO 160$ IF NO ERROR
      041072 104000      EMT
4314 041074 000137 041664      JMP      330$
4315
4316          ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
4317 041100          160$:
4318 041100 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC, RMMR1(R0) ;LOAD RMMR1
4319 041106 012760 000000 000014      MOV      #0, RMR1(R0) ;LOAD RMR1
4320 041114 012760 000000 000042      MOV      #0, RMR2(R0) ;LOAD RMR2
4321 041122 012760 000007 000000      MOV      #RECAL!GO RMCS1(R0) ;LOAD RMCS1
4322
4323          ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4324 041130 012702 000015      MOV      #13, R2
4325 041134          170$:
      041134 012760 141401 000024      MOV      #DMD!MUR!DBEN!DBCK!MOC, RMMR1(R0) ;LOAD RMMR1
4326 041142 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC, RMMR1(R0) ;LOAD RMMR1
4327 041150 005302      DEC      R2
4328 041152 001370      BNE     170$
4329
4330          ;DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER
4331 041154 012760 041401 000024      MOV      #DMD!MUR!DBEN, RMMR1(R0) ;LOAD RMMR1
4332 041162 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC, RMMR1(R0) ;LOAD RMMR1
4333
4334          ;STEP COMMAND SEQUENCER TO SET ATTENTION (3 CLOCKS)
4335 041170 012702 000003      MOV      #3, R2
4336 041174          180$:
4337 041174 012760 141401 000024      MOV      #DMD!MUR!DBEN!MOC!DBCK, RMMR1(R0) ;LOAD RMMR1
4338 041202 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC, RMMR1(R0) ;LOAD RMMR1
4339 041210 005302      DEC      R2
4340 041212 001370      BNE     180$
4341
4342          ;VERIFY ATA IS SET
4343 041214 016037 000012 001142      MOV      RMD5(R0), $GDDAT ;STORE RMD5 AT $GDDAT
4344 041222 042737 077777 001142      BIC     #CATA, $BDDAT
4345 041230 001011      BNE     190$
4346 041232 012737 100000 001140      MOV      #ATA, $GDDAT
4347 041240 010037 001136      MOV      R0, $BDADR
4348 041244 052737 000012 001136      ADD     #RMD5, $BDADR
4349 041252 104244      EMT      244
4350
4351 041254 012737 041262 001124 190$:  MOV      #200$, $LPERR      ;CHANGE LOOP ON ERROR ADDRESS
4352
4353          ;VERIFY THAT RECALIBRATE ABORTS AFTER EXECUTION DURING WAIT LOOP
4354 041262          200$:
4355 041262 004737 054732      JSR      PC, SETVV      ;GO SET VOLUME VALID
      041266 000402      BR       210$          ;BRANCH TO 210$ IF NO ERROR
      041270 104000      EMT
4356 041272 000574      BR       330$
4357
4358          ;ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND

```

```

4359 041274 210$.
4360 041274 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMPR1(RO) ;LOAD RMPR1
4361 041302 012760 000000 000014 MOV #0,RMER1(RO) ;LOAD RMER1
4362 041310 012760 000000 000042 MOV #0,RMER2(RO) ;LOAD RMER2
4363 041316 012760 000007 000000 MOV #RECAL!GO,RMCS1(RO) ;LOAD RMCS1
4364
4365 ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4366 041324 012702 000015 MOV #13,R2
4367 041330 220$:
4368 041330 012760 141401 000027 MOV #DMD!MUR!DBEN!MOC!DBCK,RMPR1(RO) ;LOAD RMPR1
4369 041336 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMPR1(RO) ;LOAD RMPR1
4370 041344 005302 DEC R2
4371 041346 001370 BNF 220$
4372
4373 ;DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
4374 041350 012760 041001 000024 MOV #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
4375
4376 ;STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
4377 041356 012702 000007 MOV #7,R2
4378 041362 230$:
4379 041362 012760 141001 000024 MOV #DMD!MUR!DBEN!DBCK,RMPR1(RO) ;LOAD RMPR1
4380 041370 012760 041001 000024 MOV #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
4381 041376 005302 DEC R2
4382 041400 001370 BNF 230$
4383
4384 ;VERIFY THAT GO IS STILL SET
4385 041402 016937 000000 001142 MOV RMCS1(RO),SBDDAT ;STORE RMCS1 AT SBDDAT
4386 041410 042737 177776 001142 BIC #*CGO,SBDDAT
4387 041416 001006 BNE 240$
4388 041420 012737 000001 001140 MOV #GO,$GDDAT
4389 041426 010037 001136 MOV RO,$BDADR
4390 041432 104245 EMT 240$
4391
4392 ;SET SEEK INCOMPLETE ERROR
4393 041434 012760 041201 000024 240$:
4394 MOV #DMD!MUR!DBEN!MSER,RMPR1(RO) ;LOAD RMPR1
4395
4396 ;STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
4397 041442 012702 000003 MOV #3,R2
4398 041446 012760 141201 000024 250$:
4399 041454 012760 041201 000024 MOV #DMD!MUR!DBEN!MSER!DBCK,RMPR1(RO) ;LOAD RMPR1
4400 041464 001370 MOV #DMD!MUR!DBEN!MSER,RMPR1(RO) ;LOAD RMPR1
4401 041466 016937 000000 001142 DEC R2
4402 041474 042737 177776 001142 BNE 250$
4403 041502 001405 MOV RMCS1(RO),SBDDAT ;STORE RMCS1 AT SBDDAT
4404 041504 010037 001136 BIC #*CGO,SBDDAT
4405 041510 005037 001140 BEQ 260$
4406 041514 104246 MOV RO,$BDADR
4407 CLR $GDDAT
4408 041516 012737 041524 001124 260$: MOV #300$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4409
4410 ;VERIFY THE TAG BUS DURING RECALIBRATE
4411 041524 004737 054732 300$:
4412 041524 000402 JSR PC,SETVV ;GO SFT VOLUME VALID
BR 310$ ;BRANCH TO 310$ IF NO ERROR
    
```

041532 104000  
4413 041534 000453

ZMT  
BR 330\$

:ENABLE DEMO LOCK AND LOAD RECALIBRATE COMMAND

4414  
4415  
4416 041536  
4417 041536 012760 041401 000024  
4418 041544 012760 000000 000014  
4419 041552 012760 000000 000022  
4420 041560 012760 000007 000000  
4421 041566 012702 041666

310\$:  
MOV #DMD MUR'DBEN'MOC,RMMR1(RO) ;LOAD RMMR1  
MOV #C,RMER1(RO) ;LOAD RMER1  
MOV #C,RMER2(RO) ;LOAD RMER2  
MOV #RECAL!GO,RMCS1(RO) ;LOAD RMCS1  
MOV #400\$,R2 ;INITIALIZE TABLE POINTER

4422  
4423  
4424 041572  
041572 016037 000040 001142  
4425 041600 042737 150000 001142  
4426 041606 021237 001142  
4427 041612 001411  
4428 041614 011237 001140  
4429 041620 010037 001136  
4430 041624 062737 000040 001136  
4431 041632 104247  
4432 041634 000413

:VERIFY TAG BUS ACCORDING TO TABLE

315\$:  
MOV RMMR2(RO), \$BDDAT ;STORE RMMR2 AT \$BDDAT  
BIC #RQA!RQB!TST,\$BDDAT  
CMP (R2), \$BDDAT  
BEQ 320\$  
MOV (R2), \$GDDAT  
MOV R0, \$BDADR  
ADD #RMMR2, \$BDADR  
EIT 247  
BR 330\$

4433  
4434  
4435 041636  
4436 041636 062702 000002  
4437 041642 005712  
4438 041644 100407  
4439  
4440

:ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE

320\$:  
ADD #2,R2  
TST (R2)  
BMI 330\$ ;EXIT IF ENTRY NEGATIVE

4441 041646 012760 151001 000024  
4442 041654 012760 C'1401 000024  
4443 041662 000743  
4444 041664 000416  
4445  
4446

:STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION

MOV #DMD!DBEN!MUR!MC!DBCK,RMMR1(RO) ;LOAD RMMR1  
MOV #DMD!DBEN!MUR!MOC,RMMR1(RO) ;LOAD RMMR1  
BR 315\$  
330\$: BR 500\$ ;JUMP OVER TABLE

4447 041666  
4448 041666 001777  
4449 041670 001777  
4450 041672 006100  
4451 041674 006100  
4452 041676 026100  
4453 041700 026100  
4454 041702 026100  
4455 041704 026100  
4456 041706 026100  
4457 041710 026100  
4458 041712 006100  
4459 041714 006100  
4460 041716 001777  
4461

:TABLE OF TAG BUS CONTROL AND BIT VALUES

400\$:  
.WORD 1777 ;BUS BITS AT HIGH IMPEDANCE STATE  
.WORD 1777  
.WORD CC!CH!BB06 ;CONTROL BITS ENABLED, BIT 6 ON  
.WORD CC!CH!BB06  
.WORD TAG!CC!CH!BB06 ;TAG COMES ON  
.WORD TAG!CC!CH!BB06  
.WORD TAG!CC!CH!BB06  
.WORD TAG!CC!CH!BB06  
.WORD TAG!CC!CH!BB06  
.WORD TAG!CC!CH!BB06  
.WORD TAG!CC!CH!BB06 ;TAG GOES OFF  
.WORD CC!CH!BB06  
.WORD CC!CH!BB06 ;CONTROL BITS DISABLED  
.WORD 1777  
.WORD -1 ;END OF TABLE

4462 041720 177777  
4463  
4464 041722  
4465  
4466

500\$: ;END OF TEST

\*\*\*\*\*  
:TEST 113 SEEK TEST

```

041722
041722 000004
041724 000240
041726 012706 001100
041732 013700 001276
041736 013701 001466
041742 012737 000113 001226
4467
4468
4469
4470
4471 041750 004737 054732
041754 000403
041756 104000
4472 041760 000137 043400
4473
4474
4475
4476 041764
4477 041764 012760 041001 000024
4478 041772 012760 000000 000006
4479 042000 012760 000000 000034
4480 042006 012760 000000 000014
4481 042014 012760 000000 000022
4482 042022 012760 000005 000000
4483
4484
4485 042030 012702 000002
4486 042034
042034 012760 141001 000024
4487 042042 012760 041001 000024
4488 042050 005302
4489 042052 001370
4490
4491
4492 042054 012760 040001 000024
4493 042062 012702 000002
4494 042066
042066 012760 140001 000024
4495 042074 012760 040001 000024
4496 042102 005302
4497 042104 001370
4498
4499
4500 042106 016037 000014 001142
4501 042114 042737 157777 001142
4502 042122 001011
4503 042124 012737 020000 001140
4504 042132 010037 001136
4505 042136 062737 000014 001136
4506 042144 104250
4507
4508 042146 012737 042154 001124
4509
4510

```

.....  
 TST113:  
 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 #113,\$TESTN ;;SET TEST NUMBER IN AP MAIL BOX

:VERIFY THAT OPI SETS IF UNIT READY DROPS DURING COMMAND EXECUTION  
 JSR PC,SETVV ;GO SET VOLUME VALID  
 BR 10\$ ;BRANCH TO 10\$ IF NO ERROR  
 EMT  
 JMP 330\$

:ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR  
 ADDRESS, AND LOAD SEEK COMMAND  
 10\$:  
 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1  
 MOV #0,RMDA(R0) ;LOAD RMDA  
 MOV #0,RMDC(R0) ;LOAD RMDC  
 MOV #0,RMER1(R0) ;LOAD RMER1  
 MOV #0,RMER2(R0) ;LOAD RMER2  
 MOV #SEEK!GO,RMCS.(R0) ;LOAD RMCS1

:STEP COMMAND SEQUENCER TO SEEK COM (2 CLOCKS)  
 20\$:  
 MOV #2,R2  
 MOV #DMD!MUR!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1  
 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1  
 DEC R2  
 BNE 20\$

:DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)  
 30\$:  
 MOV #DMD!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1  
 MOV #DMD!DBEN,RMMR1(R0) ;LOAD RMMR1  
 DEC R2  
 BNE 30\$

:VERIFY THAT OPI IS SET  
 MOV RMER1(R0),\$BDDAT ;STORE RMER1 AT \$BDDAT  
 BIC #COP1,\$BDDAT  
 BNE 40\$  
 MOV #OPI,\$GDDAT  
 MOV R0,\$BDADR  
 ADD #RMER1,\$BDADR  
 EMT 250

40\$:  
 MOV #50,\$LPERR ;CHANGE LOOP ON ERROR ADDRESS

:VERIFY THAT SEEK ABORTS DURING EXECUTION

```

4511 042154
4512 042154 004737 054732
      042160 000403
      042162 104000
4513 042164 000137 043400
4514
4515
4516
4517 042170
4518 042170 012760 041001 000024
4519 042176 012760 000000 000006
4520 042204 012760 000000 000034
4521 042212 012760 000000 000014
4522 042220 012760 000000 000042
4523 042226 012760 000005 000000
4524
4525
4526 042234 012702 000003
4527 042240
4528 042240 012760 141001 000024
4529 042246 012760 041001 000024
4530 042254 005302
4531 042256 001370
4532
4533
4534 042260 012760 041101 000024
4535
4536
4537 042266 012702 000002
4538 042272
      042272 012760 141101 000024
4539 042300 012760 041101 000024
4540 042306 005302
4541 042310 001370
4542 042312 016037 000000 001142
4543 042320 042737 177776 001142
4544 042326 001405
4545 042330 010037 001136
4546 042334 005037 001130
4547 042340 104251
4548
4549 042342 012737 042350 001124
4550
4551
4552 042350
4553 042350 004737 054732
      042354 000403
      042356 104000
4554 042360 000137 043400
4555
4556
4557
4558 042364
4559 042364 012760 041001 000024
4560 042372 012760 000000 000006
4561 042400 012760 000000 000034
4562 042406 012760 000000 000014

50S:
      JSR PC,SETVV ;GO SET VOLUME VALID
      BR 60S ;BRANCH TO 60S IF NO ERROR
      EMT
      JMP 330S

;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
;ADDRESS, AND LOAD SEEK COMMAND
60S:
      MOV #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
      MOV #0,RMDA(RO) ;LOAD RMDA
      MOV #0,RMDC(RO) ;LOAD RMDC
      MOV #0,RMER1(RO) ;LOAD RMER1
      MOV #0,RMER2(RO) ;LOAD RMER2
      MOV #SEEK!GO,RMCS1(RO) ;LOAD RMCS1

;STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
      MOV #3,R2
70S:
      MOV #DMD!MUR!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
      MOV #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
      DEC R2
      BNE 70S

;SET DRIVE FAULT TO CAUSE ABORT CONDITION
      MOV #DMD!MUR!DBEN!MDF,RMMR1(RO) ;LOAD RMMR1

;STEP 2 CLOCKS AND VERIFY GD IS RESET
      MOV #2,R2
80S:
      MOV #DMD!MUR!DBEN!MDF!DBCK,RMMR1(RO) ;LOAD RMMR1
      MOV #DMD!MUR!DBEN!MDF,RMMR1(RO) ;LOAD RMMR1
      DEC R2
      BNE 80S
      MOV RMCS1(R),%SDDAT ;STORE RMCS1 AT %SDDAT
      BIC #^GO,%SDDAT
      BEQ 90S
      MOV R0,%SDADR
      CLR %SDDAT
      EMT 251

90S:
      MOV #100S,%SLPERR ;CHANGE LOOP ON ERROR ADDRESS

;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
100S:
      JSR PC,SETVV ;GO SET VOLUME VALID
      BR 110S ;BRANCH TO 110S IF NO ERROR
      EMT
      JMP 330S

;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
;ADDRESS, AND LOAD SEEK COMMAND
110S:
      MOV #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
      MOV #0,RMDA(RO) ;LOAD RMDA
      MOV #0,RMDC(RO) ;LOAD RMDC
      MOV #0,RMER1(RO) ;LOAD RMER1
  
```

1113 SEEK TEST

```

4563 042414 012760 000000 000042      MOV    #0,RMER2(RO)      ;LOAD RMER2
4564 042422 012760 000005 000000      MOV    #SEEK GO,RMCS1(PO) ;LOAD RMCS1
4565
4566                                     ;STEP THE COMMAND SEQUENCER
4567 042430 012702 000017      MOV    #15.,R2
4568 042434
4569 042434 012760 141001 000024      MOV    #DMD!MUR!DHEN!DBCK,RMMR1(RO) ;LOAD RMMR1
4570 042442 012760 041001 000024      MOV    #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4571 042450 005302
4572 042452 001370      DEC    R2
4573                                     BNE    120$
4574
4575 042454 016037 000014 001142      ;VERIFY THAT OPI IS SET
4576 042462 042737 157777 001142      MOV    RMER1(RO),%SDDAT ;STORE RMER1 AT %SDDAT
4577 042470 001011      BIC    #^COP1,%SDDAT
4578 042472 010037 001136      BNE    130$
4579 042476 062737 000014 001136      MOV    RO,%SDDADR
4580 042504 012737 000000 001140      ADD    #RMER1,%SDDADR
4581 042512 104252      MOV    #OPI,%SDDAT
4582                                     EMT    252
4583 042514 012737 042522 001124      130$: MOV    #150$,%L%RR ;CHANGE LOOP ON ERROR ADDRESS
4584
4585                                     ;VERIFY ATA SETS IF DRIVE COMPLETES SEEK (ON CYLINDER SETS)
4586 042522
4587 042522 004737 054732      150$: JSR    PC,SETVV ;GO SET VOLUME VALID
4588 042526 000403      BR     160$ ;BRANCH TO 160$ IF NO ERROR
4589 042530 104000      EMT
4590 042532 000137 043400      JMP    330$
4591
4592                                     ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4593                                     ;ADDRESS, AND LOAD SEEK COMMAND
4594 042536
4595 042536 012760 041401 000024      160$: MOV    #DMD!MUR!DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
4596 042544 012760 000000 000006      MOV    #0,RMDA(RO) ;LOAD RMDA
4597 042552 012760 000000 000034      MOV    #0,RMDC(RO) ;LOAD RMDC
4598 042560 012760 000000 000014      MOV    #0,RMER1(RO) ;LOAD RMER1
4599 042566 012760 000000 000042      MOV    #0,RMER2(RO) ;LOAD RMER2
4600 042574 012760 000005 000000      MOV    #SEEK GO,RMCS1(RO) ;LOAD RMCS1
4601
4602                                     ;STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4603 042602 012702 000015      MOV    #13.,R2
4604 042606
4605 042606 012760 141401 000024      170$: MOV    #DMD!MUR!DBEN!DBCK!MOC,RMMR1(RO) ;LOAD RMMR1
4606 042614 012760 041401 000024      MOV    #DMD!MUR!DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
4607 042622 005302      DEC    R2
4608 042624 001370      BNE    170$
4609
4610                                     ;DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER
4611 042626 012760 041001 000024      MOV    #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4612 042634 012760 041401 000024      MOV    #DMD!MUR!DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
4613
4614                                     ;STEP COMMAND SEQUENCER TO SET ATTENTION (3 CLOCKS)
4615 042642 012702 000003      MOV    #3.,R2
4616 042646
4617 042646 012760 141401 000024      180$: MOV    #DMD!MUR!DBEN!MOC!DBCK,RMMR1(RO) ;LOAD RMMR1
4618 042654 012760 041401 000024      MOV    #DMD!MUR!DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
4619 042662 005302      DEC    R2

```

```

4617 042664 001370          BNE      180$
4618
4619          .VERIFY ATA IS SET
4620 042666 016037 000012 001142      MOV      RMS1(RO), $BDDAT ;STORE RMS1 AT $BDDAT
4621 042674 042737 077777 001142      BIC      #^ATA, $BDDAT
4622 042702 001011          BNE      190$
4623 042704 012737 100000 001140      MOV      #ATA, $GDDAT
4624 042712 010037 001136          MOV      RO, $BDADR
4625 042716 062737 000012 001136      ADD      #RMS1, $BDADR
4626 042724 104253          EMT      253
4627
4628 042726 012737 042734 001124 190$:  MOV      #200$, $LPERR ;CHANGE LOOP ON ERROR ADDRESS
4629
4630          .VERIFY THAT SEEK ABORTS AFTER EXECUTION DURING WAIT LOOP
4631 042734          200$:
4632 042734 004737 054732          JSR      PC, SETVV ;GO SET VOLUME VALID
4633 042740 000403          BR      210$ ;BRANCH TO 210$ IF NO ERROR
4634 042742 104000          EMT
4635 042744 000137 043400          JMP      330$
4636
4637          .ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4638          .ADDRESS AND LOAD SEEK COMMAND
4639 042750          210$:
4640 042750 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC, RMPR1(RO) ;LOAD RMPR1
4641 042756 012760 000000 000006      MOV      #0, RMDA(RO) ;LOAD RMDA
4642 042764 012760 000000 000034      MOV      #0, RMDC(RO) ;LOAD RMDC
4643 042772 012760 000000 000014      MOV      #0, RMR1(RO) ;LOAD RMR1
4644 043000 012760 000000 000042      MOV      #0, RMR2(RO) ;LOAD RMR2
4645 043006 012760 000005 000000      MOV      #SEEK!GO, RMCS1(RO) ;LOAD RMCS1
4646
4647          .STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
4648 043014 012702 000015          MOV      #13, R2
4649 043020          220$:
4650 043020 012760 141401 000024      MOV      #DMD!MUR!DBEN!MOC!DBCK, RMPR1(RO) ;LOAD RMPR1
4651 043026 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC, RMPR1(RO) ;LOAD RMPR1
4652 043034 005302          DEC      R2
4653 043036 001370          BNE      220$
4654
4655          .DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
4656 043040 012760 041001 000024      MOV      #DMD!MUR!DBEN, RMPR1(RO) ;LOAD RMPR1
4657
4658          .STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
4659 043046 012702 000007          MOV      #7, R2
4660 043052          230$:
4661 043052 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK, RMPR1(RO) ;LOAD RMPR1
4662 043060 012760 041001 000024      MOV      #DMD!MUR!DBEN, RMPR1(RO) ;LOAD RMPR1
4663 043066 005302          DEC      R2
4664 043070 001370          BNE      230$
4665
4666          .VERIFY THAT GO IS STILL SET
4667 043072 016037 000000 001142      MOV      RMCS1(RO), $BDDAT ;STORE RMCS1 AT $BDDAT
4668 043100 042737 177776 001142      BIC      #CGO, $BDDAT
4669 043106 001006          BNE      40$
4670 043110 012737 000001 001140      MOV      #GO, $GDDAT
4671 043116 010037 001136          MOV      RO, $BDADR
4672 043122 104254          EMT      254
4673
    
```

```

4672                                     :SET SEEK INCOMPLETE ERROR
4673 043124 012760 041201 000024 240$: MOV #DMD!MUR!DBEN!MSER,RMPR1(RO) ;LOAD RMPR1
4674 043124 012760 041201 000024
4674                                     :STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
4675 MGV #3,R2
4676 043132 012702 000003 250$: MOV #DMD!MUR!DBEN!MSER!DBCK,RMPR1(RO) ;LOAD RMPR1
4677 043136 012760 141201 000024 MOV #DMD!MUR!DBEN!MSER,RMPR1(RO) ;LOAD RMPR1
4678 043144 012760 041201 000024 DEC R2
4679 043152 005302 BNE 250$
4680 043154 001370 MOV RMCS1(RO),SDDAT ;STORE RMCS1 AT SDDAT
4681 043156 016037 000000 001142 BIC #*CGO,SDDAT
4682 043164 042737 17776 001142 BEQ 260$
4683 043172 001405 MOV R0,SBDADR
4684 043174 010037 001136 CLR $GDDAT
4685 043200 005037 001140 EMT 255
4686 043204 104255
4687
4688 043206 012737 043220 001124 260$: MOV #300$,SLPERR ;CHANGE LOOP ON ERROR ADDRESS
4689 043214 012703 000001 MOV #1,R3 ;INITIALIZE CYLINDER ADDRESS
4690
4691                                     :VERIFY THE TAG BUS DURING SEEK
4692 043220 300$: JSR PC,SETVV ;GO SET VOLUME VALID
4693 043220 004737 054732 BR 310$ ;BRANCH TO 310$ IF NO ERROR
4694 043224 000402
4695 043226 104000
4696 043230 000460
4697
4698                                     :ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4699 043232 310$: MOV #DMD!MUR!DBEN!MOC,RMPR1(RO) ;LOAD RMPR1
4700 043232 012760 041401 000024 MOV #0,RMDA(RO) ;LOAD RMDA
4701 043240 012760 000000 000006 MOV R3,RMDC(RO) ;LOAD RMDC
4702 043246 010360 000034 MOV #0,RMER1(RO) ;LOAD RMER1
4703 043252 012760 000000 000014 MOV #0,RMER2(RO) ;LOAD RMER2
4704 043260 012760 000005 000042 MOV #SEEK!GO,RMCS1(RO) ;LOAD RMCS1
4705 043274 012702 043414 MOV #400$,R2 ;INITIALIZE TABLE POINTER
4706
4707                                     :VERIFY TAG BUS ACCORDING TO TABLE AND CYLINDER IN R3
4708 043300 315$: MOV RMPR2(RO),SDDAT ;STORE RMPR2 AT SDDAT
4709 043300 016037 000040 001142 BIC #RQA!RQB!TST,SDDAT
4710 043306 042737 150000 001142 MOV (R2),$GDDAT
4711 043314 011237 001140 BIS R3,$GDDAT ;OR CYLINDER ADDRESS IN
4712 043320 050337 001140 001142 CMP $GDDAT,SDDAT ;COMPARE EXPECTED AND RECEIVED
4713 043324 023737 001140 001142 BEQ 320$ ;BRANCH IF TAG BUS OK
4714 043334 010037 001136 MOV R0,SBDADR
4715 043340 062737 000040 001136 ADD #RMPR2,SBDADR
4716 043346 104256 EMT 256
4717 043350 000413 BR 330$
4718
4719                                     :ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
4720 043352 320$: ADD #2,R2
4721 043352 062702 000002 TST (R2)
4722 043356 005712 BMI 330$ ;EXIT IF ENTRY NEGATIVE
4723 043360 100407
  
```



```

4724
4725
4726 043362 012760 151001 000024 ;STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
4727 043370 012760 041401 000024 MOV #DMD!DBEN!MUR!MOV!DBCK,RMMR1(RO) ;LOAD RMMR1
4728 043376 000740 BR 315$ ;LOAD RMMR1
4729
4730 ;REPEAT TAG BUS TEST FOR EACH PRIME CYLINDER, I.E., 1,2,4,....
4731 043400 330$: ASL R3 ;SHIFT TO NEXT CYLINDER
4732 043400 006303 CMP R3,#1024.
4733 043402 020327 002000 BHIS 340$ ;EXIT IF WAS DONE
4734 043406 10300$ BR 300$ ;TEST NEXT CYLINDER
4735 043410 000703 BR 500$ ;JUMP OVER TABLE
4736 043412 000416
4737
4738 ;TABLE OF TAG BUS CONTROL AND BIT VALUES
4739 043414 400$: .WORD 1777 ;BUS BITS AT HIGH IMPEDANCE STATE
4740 043414 001777 .WORD 1777
4741 043416 001777 .WORD CC ;CONTROL BITS ENABLED, BIT 6 ON
4742 043420 004000 .WORD CC
4743 043422 004000 .WORD TAG!CC ;TAG COMES ON
4744 043424 024000 .WORD TAG!CC
4745 043426 024000 .WORD TAG!CC
4746 043430 024000 .WORD TAG!CC
4747 043432 024000 .WORD TAG!CC
4748 043434 024000 .WORD TAG!CC
4749 043436 024000 .WORD TAG!CC
4750 043440 004000 .WORD CC ;TAG GOES OFF
4751 043442 004000 .WORD CC
4752 043444 001777 .WORD 1777 ;CONTROL BITS DISABLED
4753
4754 043446 177777 .WORD -1 ;END OF TABLE
4755
4756 043450 500$: ;END OF TEST
4757
4758
;*****
;*TEST 114 SEARCH TEST
;*****
043450
043450 000004 TST114: SCOPE ;SCOPE CALL
043452 000240 NOP
043454 012706 001100 MOV #STACK,SP ;LOAD THE STACK POINTER
043460 013700 001276 MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
043464 013701 001466 MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
043470 012737 000114 001226 MOV #114,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
4759
4760
4761 ;VERIFY THAT OPI SETS IF UNIT READY DROPS DURING COMMAND EXECUTION
4762
4763 043476 004737 054732 JSR PC,SETVV GO SET VOLUME VALID
043502 000403 BR 10$ ;BRANCH TO 10$ IF NO ERROR
043504 104300 EMT
4764 043506 000137 045544 JMP 330$
4765
4766 ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4767 ;ADDRESS, AND LOAD SEARCH COMMAND
4768 043512 10$:
  
```

```

4769 043512 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4770 043520 012760 000000 000006      MOV      #0,RMDA(RO) ;LOAD RMDA
4771 043526 012760 000000 000034      MOV      #0,RMDC(RO) ;LOAD RMDC
4772 043534 012760 000000 000014      MOVB    #0,RMER1(RO) ;LOAD RMER1
4773 043542 012760 000000 000042      MOV      #0,RMER2(RO) ;LOAD RMER2
4774 043550 012760 000031 000000      MOV      #SEARCH!GO,RMCS1(RO) ;LOAD RMCS1
4775
4776                                     ;STEP COMMAND SEQUENCER TO SEARCH COM (2 CLOCKS)
4777 043556 012702 000002      MOV      #2,R2
4778 043562
4779 043562 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
4780 043570 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4781 043576 005302      DEC      R2
4782 043600 001370      BNE     20$
4783
4784 043602 012760 040001 000024      MOV      #DMD!DBEN,RMMR1(RO) ;LOAD RMMR1
4785 043610 012702 000002      MOV      #2,R2
4786 043614
4787 043614 012760 140001 000024      MOV      #DMD!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
4788 043622 012760 040001 000024      MOV      #DMD!DBEN,RMMR1(RO) ;LOAD RMMR1
4789 043630 005302      DEC      R2
4790 043632 001370      BNE     30$
4791
4792 043634 016037 000014 001142      MOV      RMER1(RO),SBDDAT ;STORE RMER1 AT SBDDAT
4793 043642 042737 157777 001142      BIC     #*COPI,SBDDAT
4794 043650 001011      BNE     40$
4795 043652 012737 020000 001140      MOV      #OPI,$GDDAT
4796 043660 010037 001136      MOV      RO,SBADR
4797 043664 062737 000014 001136      ADD     #RMER1,$BADR
4798 043672 104257      EMT     257
4799
4800 043674 012737 043702 001124      40$:    MOV      #50$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
4801
4802                                     ;VERIFY THAT SEARCH ABORTS DURING EXECUTION
4803 043702
4804 043702 004737 054752      50$:    JSR     PC,SETVV ;GO SET VOLUME VALID
4805 043706 000403      BR      60$ ;BRANCH TO 60$ IF NO ERROR
4806 043710 104000      EMT
4807 043712 000137 045544      JMP     330$
4808
4809                                     ;FNABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4810                                     ;ADDRESS, AND LOAD SEARCH COMMAND
4811 043716
4812 043716 012760 041001 000024      60$:    MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
4813 043724 012760 000000 000006      MOV      #0,RMDA(RO) ;LOAD RMDA
4814 043732 012760 000000 000034      MOV      #0,RMDC(RO) ;LOAD RMDC
4815 043740 012760 000000 000014      MOV      #0,RMER1(RO) ;LOAD RMER1
4816 043746 012760 000000 000042      MOV      #0,RMER2(RO) ;LOAD RMER2
4817 043754 012760 000031 000000      MOV      #SEARCH!GO,RMCS1(RO) ;LOAD RMCS1
4818
4819                                     ;STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
4820 043762 012702 000003      MOV      #3,R2
4821 043766
4822 043766 012760 141001 000024      70$:    MOV      #DMD!MUR!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
4823 043774 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1

```

```

4822 044002 005302          DEC    R2
4823 044004 001370          BNE    70$
4824
4825
4826 044006 012760 041101 000024 ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
      MOV    #DMD!MUR!DBEN!MDF,RMPR1(RO) ;LOAD RMPR1
4827
4828
4829 044014 012702 000002 ;STEP 2 CLOCKS AND VERIFY GO IS RESET
      MOV    #2,R2
4830 044020
      80$:
      044020 012760 141101 000024 MOV    #DMD!MUR!DBEN!MDF!DBCK,RMPR1(RO) ;LOAD RMPR1
      044026 012760 041101 000024 MOV    #DMD!MUR!DBEN!MDF,RMPR1(RO) ;LOAD RMPR1
4831 044026 012760 041101 000024
4832 044034 005302          DEC    R2
4833 044036 001370          BNE    80$
4834 044040 016037 000000 001142 MOV    RMCS1(RO),SBDDAT ;STORE RMCS1 AT SBDDAT
4835 044046 042737 177776 001142 BIC    #^CGO,SBDDAT
4836 044054 001405          BEQ    90$
4837 044056 010037 001136 MOV    RO,SBADR
4838 044062 005037 001140 CLR    $GDDAT
4839 044066 104260          EMT    260
4840
4841
4842 ;(THE OTHER TWO ABORT TESTS IN THE COMMAND SEQUENCER ARE TESTED
4843 ;DURING DATA COMMAND TESTS)
4844 044070 012737 044076 001124 90$: MOV    #100$,SLPERR ;CHANGE LOOP ON ERROR ADDRESS
4845
4846 ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
4847 044076
      100$:
4848 044076 004737 054732 JSR    PC,SETVV ;GO SET VOLUME VALID
      044102 000403 BR     110$ ;BRANCH TO 110$ IF NO ERROR
      044104 104000 EMT
4849 044106 000137 045544 JMP    330$
4850
4851 ;ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
4852 ;ADDRESS, AND LOAD SEARCH COMMAND
4853 044112
      110$:
4854 044112 012760 041001 000024 MOV    #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
4855 044120 012760 000000 000006 MOV    #0,RMDA(RO) ;LOAD RMDA
4856 044126 012760 000000 000034 MOV    #0,RMDC(RO) ;LOAD RMDC
4857 044134 012760 000000 000014 MOV    #0,RMER1(RO) ;LOAD RMER1
4858 044142 012760 000000 000042 MOV    #0,RMER2(RO) ;LOAD RMER2
4859 044150 012760 000031 000000 MOV    #SEARCH!GO,RMCS1(RO) ;LOAD RMCS1
4860
4861 ;STEP THE COMMAND SEQUENCER
4862 044156 012702 000023 MOV    #19.,R2
4863 044162
      120$:
4864 044162 012760 141001 000024 MOV    #DMD!MUR!DBEN!DBCK,RMPR1(RO) ;LOAD RMPR1
4865 044170 012760 041001 000024 MOV    #DMD!MUR!DBEN,RMPR1(RO) ;LOAD RMPR1
4866 044176 005302          DEC    R2
4867 044200 001370          BNE    120$
4868
4869 ;VERIFY THAT OPI IS SET
4870 044202 010037 000014 001142 MOV    RMER1(RO),SBDDAT ;STORE RMER1 AT SBDDAT
4871 044210 042737 157777 001142 BIC    #^COPI,SBDDAT
4872 044216 001011          BNE    130$
4873 044220 010037 001136 MOV    RO,SBADR
4874 044224 042737 000014 001136 ADD    #RMER1,SBADR
4875 044232 012737 020000 001140 MOV    #OPI,$GDDAT
  
```

```

4876 044242 012737 044250 001124 1308: MOV #1508,R2 ;CHANGE LOOP ON ERROR ADDRESS
4877
4878 044242 012737 044250 001124 1308: MOV #1508,R2 ;CHANGE LOOP ON ERROR ADDRESS
4879
4880
4881
4882 044250
4883 044250 004737 054732 JSR PC,SET ;GO SET VOLUME VALID
      044254 000403 BR 1608 ;BRANCH TO 1608 IF NO ERROR
      044256 104000 CMT
4884 044260 000137 045544 JMP 3308
4885
4886
4887
4888 044264
4889 044264 012760 041401 00002
4890 044272 012760 000000 000006 MOV #0,RMDA(R0) ;LOAD RMDA
4891 044300 012760 000000 000034 MOV #0,RMDC(R0) ;LOAD RMDC
4892 044306 012760 000000 000014 MOV #0,RMER1(R0) ;LOAD RMER1
4893 044314 012760 000000 000042 MOV #0,RMER2(R0) ;LOAD RMER2
4894 044322 012760 000031 000000 MOV #SEARCH!GO,RPCS1(R0) ;LOAD RPCS1
4895
4896
4897 044330 012702 000021
4898 044334 012760 141401 000024
      044334 012760 141401 000024 MOV #DMD!MUR!DBEN!DBCK!MOC,RMPR1(R0) ;LOAD RMPR1
4899 044342 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMPR1(R0) ;LOAD RMPR1
4900 044350 005302 DEC R2
4901 044352 001370 BNE 1708
4902
4903
4904
4905 044354 012760 041005 000024
      044354 012760 041005 000024 MOV #DMD!MUR!DBEN!MI,RMPR1(R0) ;LOAD RMPR1
4906
4907
4908 044362 012760 051401 000024
      044362 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMPR1(R0) ;LOAD RMPR1
4909
4910
4911 044370 012702 000002
4912 044374 012760 151401 000024
      044374 012760 151401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN!DBCK,RMPR1(R0) ;LOAD RMPR1
4913 044402 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMPR1(R0) ;LOAD RMPR1
4914 044402 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMPR1(R0) ;LOAD RMPR1
4915 044410 005302 DEC R2
4916 044412 001370 BNE 1808
4917
4918
4919
4920 044414 012760 051403 000024
4921 044422 012760 051443 000024
4922 044430 012760 051473 000024
      044414 012760 051403 000024 MOV #DMD!MUR!MOC!DBEN!MSEN!MSC,RMPR1(R0) ;LOAD RMPR1
      044422 012760 051443 000024 MOV #DMD!MUR!MOC!DBEN!MSEN!MSC!MS,RMPR1(R0) ;LOAD RMPR1
      044430 012760 051473 000024 MOV #DMD!MUR!MOC!DBEN!MSEN!MSC,RMPR1(R0) ;LOAD RMPR1
4923
4924
4925 044436 012702 000003
4926 044442 012760 151401 000024
      044442 012760 151401 000024 MOV #DMD!MUR!MOC!DBEN!MSEN!DBCK,RMPR1(R0) ;LOAD RMPR1
4927 044450 012760 051401 000024 MOV #DMD!MUR!MOC!DBEN!MSEN,RMPR1(R0) ;LOAD RMPR1
4928 044456 005302 DEC R2

```

4929 044460 001370  
 4930  
 4931  
 4932 044462 016037 000012 001142  
 4933 044470 042737 077777 001142  
 4934 044476 001011  
 4935 044500 012737 100000 001140  
 4936 044506 010037 001136  
 4937 044512 062737 000017 001136  
 4938 044520 104262  
 4939  
 4940 044522 012737 044530 001124  
 4941  
 4942  
 4943 044530  
 4944 044530 004737 054732  
 044534 000703  
 044536 104000  
 4945 044540 000137 045544  
 4946  
 4947  
 4948  
 4949 044544  
 4950 044544 012740 041401 000024  
 4951 044552 012760 000000 000006  
 4952 044560 012760 000000 000034  
 4953 044566 012760 000000 000014  
 4954 044574 012760 000000 000042  
 4955 044602 012760 000031 000000  
 4956  
 4957  
 4958 044610 012702 000021  
 4959 044614  
 4960 044614 012760 141401 000024  
 4961 044622 012760 041401 000024  
 4962 044630 005302  
 4963 044632 001370  
 4964  
 4965  
 4966 044634 012760 041001 000024  
 4967  
 4968  
 4969 044642 012702 000007  
 4970 044646  
 4971 044646 012760 141001 000024  
 4972 044654 012760 041001 000024  
 4973 044662 005302  
 4974 044664 001370  
 4975  
 4976  
 4977 044666 016037 000000 001142  
 4978 044674 042737 177776 001142  
 4979 044702 001006  
 4980 044704 012737 000001 001140  
 4981 044712 010037 001136  
 4982 044716 104263  
 4983

BNE 185\$  
 :VERIFY ATA IS SET  
 MOV RMDS(RO),SBDDAT :STORE RMDS AT SBDDAT  
 BIC #\*CATA,SBDDAT  
 BNE 190\$  
 MOV #ATA,\$GUDAT  
 MOV RO,\$SDADR  
 ADD #RMDS,\$SDADR  
 EMT 262  
 190\$: MOV #200\$,SLPERR :CHANGE LOOP ON ERROR ADDRESS  
 :VERIFY THAT SEARCH ABORTS AFTER EXECUTION DURING SEARCH SEEK LOOP  
 200\$: JSR PC,SETVV :GO SET VOLUME VALID  
 BR 210\$ :BRANCH TO 210\$ IF NO ERROR  
 EMT  
 JMP 330\$  
 :ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR  
 :ADDRESS AND LOAD SEARCH COMMAND  
 210\$: MOV #DMD!MUR!DBEN!MDC,RMPR1(RO) :LOAD RMPR1  
 MOV #0,RMDA(RO) :LOAD RMDA  
 MOV #0,RMDC(RO) :LOAD RMDC  
 MOV #0,RMER1(RO) :LOAD RMER1  
 MOV #0,RMER2(RO) :LOAD RMER2  
 MOV #SEARCH!GO,RMCS1(RO) :LOAD RMCS1  
 :STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (17 CLOCKS)  
 MOV #17,,R2  
 220\$: MOV #DMD!MUR!DBEN!MDC!DBCK,RMPR1(RO) :LOAD RMPR1  
 MOV #DMD!MUR!DBEN!MDC,RMPR1(RO) :LOAD RMPR1  
 DEC R2  
 BNE 230\$  
 :DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0  
 MOV #DMD!MUR!DBEN,RMPR1(RO) :LOAD RMPR1  
 :STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)  
 MOV #7,R2  
 230\$: MOV #DMD!MUR!DBEN!DBCK,RMPR1(RO) :LOAD RMPR1  
 MOV #DMD!MUR!DBEN,RMPR1(RO) :LOAD RMPR1  
 DEC R2  
 BNE 230\$  
 :VERIFY THAT GO IS STILL SET  
 MOV RMCS1(RO),SBDDAT :STORE RMCS1 AT SBDDAT  
 BIC #\*GO,SBDDAT  
 BNE 240\$  
 MOV #GO,\$GDDAT  
 MOV RO,\$SDADR  
 EMT 263

```

4984 :SET SEEK INCOMPLETE ERROR
4985 044720 012760 041201 000024 240$ MOV #DMD!MUR!DBEN!MSEN,RMPR1(RO) ;LOAD RMPR1
044720
4986 :STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
4987 MOV #5,R2
4988 044726 012702 000003 250$
4989 044732 012760 141201 000024 MOV #DMD!MUR!DBEN!MSEN,DBCK,RMPR1(RO) ;LOAD RMPR1
044732 012760 041201 000024 MOV #DMD!MUR!DBEN!MSEN,RMPR1(RO) ;LOAD RMPR1
4990 044740 005302 DEC R2
4991 044746 001370 BNE 250$
4992 044750 016037 000000 001142 MCV RMCS1(RO),SBDAT ;STORE RMCS1 AT SBDAT
4993 044752 02737 177776 001142 BIC #CGO,SBDAT
4994 044760 00405 260$ BEQ 260$
4995 044766 010037 001136 MOV RO,SBDADR
4996 044770 005037 001140 CLR SBDAT
4997 044774 104264 EMT 254
4998 045000
4999
5000 045002 012737 045010 001124 260$ MOV #265$,SI PERR ;CHANGE LOOP ON ERROR ADDRESS
5001
5002 :VERIFY THAT SEARCH ABORTS DURING SECTOR COMPARE LOOP
5003 045010 004737 054732 265$: JSR #C,SE VM ;GO SET VOLUME VALID
5004 045014 000403 BR 270$ ;BRANCH TO 270$ IF NO ERROR
045016 104000 EMT
5005 045020 000137 045544 JMP 330$
5006
5007 :ENABLE DEBU CLOCK, LOAD CYLINDER, TRACK AND SECTOR
5008 :ADDRESS, AND LOAD SEARCH COMMAND
5009 045024 012760 041401 000024 270$: MOV #DMD!MUR!DBEN!MOC,RMPR1(RO) ;LOAD RMPR1
5010 045024 012760 000000 000006 MOV #0,RMDA(RO) ;LOAD RMDA
5011 045040 012760 000000 000034 MOV #0,RMDC(RO) ;LOAD RMDC
5012 045046 012760 000000 000014 MOV #0,RMER1(RO) ;LOAD RMER1
5013 045054 012760 000000 000047 MOV #0,RMER2(RO) ;LOAD RMER2
5014 045062 012760 000031 000000 MOV #SEARCH!GO,RMCS1(RO) ;LOAD RMCS1
5015
5016
5017 :STEP COMMAND SEQUENCER TO FIRST DATA TEST (17 CLOCKS)
5018 045070 012702 000021 275$: MOV #17,R2
5019 045074 012760 141401 000024 MOV #DMD!MUR!DBEN!DBCK!MOC,RMPR1(RO) ;LOAD RMPR1
045074 012760 041401 000024 MOV #DMD!MUR!DBEN!MOC,RMPR1(RO) ;LOAD RMPR1
5020 045102 005302 DEC R2
5021 045110 001370 BNE 275$
5022
5023
5024 :DROP ON CYLINDER TO RESET LATCH, AND RAISE INDEX PULSE
5025 :TO SET FORMAT CHANGE FLOP
5026 045114 012760 041005 000024 MOV #DMD!MUR!DBEN!MI,RMPR1(RO) ;LOAD RMPR1
5027
5028 :RAISE ON CYLINDER AND INHIBIT SEARCH TIMEOUT
5029 045122 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMPR1(RO) ;LOAD RMPR1
5030
5031 :STEP COMMAND SEQUENCER TO SEARCH ENABLE (2 CLOCKS)
5032 045130 012702 000002 280$: MOV #2,R2
5033 045134 012760 151401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,DBCK,RMPR1(RO) ;LOAD RMPR1
5034 045134 012760 051401 000024 MOV #DMD!MUR!DBEN!MOC!MSEN,RMPR1(RO) ;LOAD RMPR1
5035

```

```

5036 0 5150 005302
5037 045152 001370
5038
5039
5040 045154 012702 000
5041 045160
5042 045160 012037 000024 001142
5043 045166 042737 173777 001142
5044 045174 001411
5045 045176 012760 151401 000024
5046 045204 012760 051401 000024
5047 045212 005302
5048 045214 001361
5049 045216 000411
5050 045220 012737 004000 001140
5051 045226 010037 001136
5052 045232 062737 000024 001136
5053 045240 104265
5054
5055
5056 045242 012760 051501 000024
5057
5058
5059 045250 012702 000002
5060 045254 012760 051501 000024
5061 045262 012760 051501 000024
5062 045270 005302
5063 045272 001370
5064 045274 016037 000000 001142
5065 045302 042737 177776 001142
5066 045310 001406
5067 045312 012737 000000 001140
5068 045320 010037 001136
5069 045324 104260
5070
5071 045326 012737 045340 001124
5072 045334 012703 000001
5073
5074
5075 045340
5076 045340 004737 054732
5077 045344 000402
5078 045346 104000
5079 045350 000475
5080
5081 045352
5082 045352 012760 041401 000024
5083 045360 012760 000000 000006
5084 045366 010360 000034
5085 045372 012760 000000 000014
5086 045400 012760 000000 000042
5087 045406 012760 000031 000000
5088

```

```

BNE
VERIFY THAT SEARCH IS ON DURING SECTOR COMPARE LOOP
MOV R2 = CLOCK COUNT
MOV RMMR1(RO), $BDDAT ; STORE RMMR1 AT $BDDAT
BIC #^CESRC, $BDDAT
REQ 252$ ; BRANCH IF SEARCH NOT ENABLED
MOV #DMD!MUR!MOC!DBEN!MSEN!DBCK, RMMR1(RO) ; LOAD RMMR1
MOV #DMD!MUR!MOC!DBEN!MSEN!, RMMR1(RO) ; LOAD RMMR1
DEC R2
BNE 261$
BR 283$
MOV #ESRC, $GDADR
MOV RO, $BDADR
ADD #RMMR1, $BDADR
EMT 265

; SET DRIVE FAULT TO CAUSE ABORT CONDITION
283$:
MOV #DMD!PJR!DBEN!MOC!MSEN!MDF, RMMR1(RO) ; LOAD RMMR1

; STEP 2 CLOCKS AND VERIFY GO IS RESET
MOV #2, R2 ; R2 = CLOCK COUNT
285$:
MOV #DMD!MUR!DBEN!MOC!MSEN!MDF!DBCK, RMMR1(RO) ; LOAD RMMR1
MOV #DMD!MUR!DBEN!MOC!MSEN!MDF, RMMR1(RO) ; LOAD RMMR1
DEC R2
BNE 285$
MOV RMCS1(RO), $BDDAT ; STORE RMCS1 AT $BDDAT
BIC #^CGO, $BDDAT
REQ 290$
MOV #0, $GDADR
MOV RO, $BDADR
EMT 260

290$:
MOV #300$, $LPERR ; CHANGE LOOP ON ERROR ADDRESS
MOV #1, R3 ; INITIALIZE CYLINDER ADDRESS

; VERIFY THE TAG BUS DURING SEARCH
300$:
JSR PC, SETVV ; GO SET VOLUME VALID
BR 310$ ; BRANCH TO 310$ IF NO ERROR
EMT 330$
BR 330$

; ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
; ADDRESS AND LOAD SEARCH COMMAND
310$:
MOV #DMD!MUR!DBEN!MOC, RMMR1(RO) ; LOAD RMMR1
MOV #0, RMDA(RO) ; LOAD RMDA
MOV R3, RMDC(RO) ; LOAD RMDC
MOV #0, RMER1(RO) ; LOAD RMER1
MOV #0, RMER2(RO) ; LOAD RMER2
MOV #SEARCH!GO, RMCS1(RO) ; LOAD RMCS1

```

```

5089
5090
5091
5092
5093
5094 045414 012702 000011
5095
5096
5097 045420
5098 045420 012760 141401 000024
5099 045426 012760 041401 000024
5100 045434 005302
5101 045436 001370
5102 045440 012702 045602
5103
5104
5105
5106 045444
5107 045444 016037 000040 001142
5108 045452 042737 150000 001142
5109 045460 011237 001140
5110 045466 050337 001140
5111 045470 023737 001140 001142
5112 045476 001407
5113 045500 010037 001136
5114 045504 062737 000040 001136
5115 045512 104266
5116 045514 000420
5117
5118 045516
5119 045516 062702 000002
5120 045522 005712
5121 045524 100407
5122
5123
5124 045526 012760 141401 000024
5125 045534 012760 041401 000024
5126 045542 000740
5127
5128
5129 045544
5130 045544 006303
5131 045546 020327 002000
5132 045552 103001
5133 045554 000671
5134 045556 000424
5135
5136
5137 045560
5138 045560 001777
5139 045562 001777
5140 045564 001777
5141 045566 001777
5142 045570 001777
5143 045572 004000
5144 045574 004000

:*****
:      MOV      #400$,R2      :INITIALIZE TABLE POINTER
:
:      :HARDWARE ECO CHANGE TO THE PLA OF THE
:      :CS BOARD.
:      :CLOCK THE SEQUENCER THRU THE FIRST
:      :9. COMMAND SEQUENCES TO ALLOW THE PROGRAM
:      :TO RUN WITH OR WITHOUT THE ECO.
312$:
:      MOV      #DMD!DBEN!MUR!MOC!DBCK,RMMR1(RO)      :LOAD RMMR
:      MOV      #DMD!DBEN!MUR!MOC,RMMR1(RO)      :LOAD RMMR1
:      DEC      R2      :DONE 9. CLOCKS ?
:      BNE     312$      :NO !
:      MOV      #450$,R2      :INITIALIZE NEW TABLE POINTER
:*****
:VERIFY TAG BUS ACCORDING TO TABLE AND CYLINDER IN R3
315$:
:      MOV      RMMR2(RO),SBDDAT      :STORE RMMR2 AT SBDDAT
:      BIC     #RQA!RQB!TST,SBDDAT
:      MOV     (R2),SGDDAT
:      BIS     R3,SGDDAT      :OR CYLINDER ADDRESS IN
:      CMP     $GDDAT,SBDDAT      :COMPARE EXPECTED AND RECEIVED
:      BEQ     320$      :BRANCH IF TAG BUS OK
:      MOV     R0,$BDDADR
:      ADD     #RMMR2,$BDDADR
:      EMT     266
:      BR     340$

:ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
320$:
:      ADD     #2,R2
:      TST     (R2)
:      EMT     330$      :EXIT IF ENTRY NEGATIVE

:STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
:      MOV     #DMD!DBEN!MUR!MOC!DBCK,RMMR1(RO)      :LOAD RMMR
:      MOV     #DMD!DBEN!MUR!MOC,RMMR1(RO)      :LOAD RMMR1
:      BR     315$

:REPEAT TAG BUS TEST FOR EACH PRIME CYLINDER, I.E. 1,2,4,...
330$:
:      ASL     R3      :SHIFT TO NEXT CYLINDER
:      CMP     R3,#1024
:      BHS     340$      :EXIT IF WAS DONE
:      BR     300$      :TEST NEXT CYLINDER
340$:
:      BR     500$      :JUMP OVER TABLE

:TABLE OF TAG BUS CONTROL AND BIT VALUES
400$:
:      .WORD   1777      :BUS BITS AT HIGH IMPEDANCE STATE
:      .WORD   1777
:      .WORD   1777
:      .WORD   1777
:      .WORD   1777
:      .WORD   CC      :CONTROL BITS ENABLED, BIT 0 ON
:      .WORD   CC
  
```



```

5145 045576 024000 .WORD TAG!CC ;TAG COMES ON
5146 045600 024000 .WORD TAG!CC
5147 045602 450$: .WORD TAG!CC ;START TABLE HERE FOR HARDWARE ECO CHANGE
5148 045602 024000 .WORD TAG!CC
5149 045604 024000 .WORD TAG!CC
5150 045606 024000 .WORD TAG!CC
5151 045610 024000 .WORD TAG!CC
5152 045612 177777 .WORD -1 ;END TABLE HERE FOR HARDWARE ECO CHANGE
5153
5154 .WORD CC ;TAG GOES OFF
5155 045614 004000 .WORD CC
5156 045616 001777 .WORD 1777 ;CONTROL BITS DISABLED
5157 045620 001777 .WORD 1777
5158 045622 001777 .WORD 1777
5159 045624 001777 .WORD 1777
5160
5161 045626 177777 .WORD -1 ;END OF TABLE
5162
5163 045630 500$: ;END OF TEST
5164
5165

```

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

\*\*\*\*\*  
TST115:  
\*\*\*\*\*

```

045630 000004 .SCOPE ;SCOPE CALL
045632 000240 .NOP
045634 012706 001100 .MOV #STACK,SP ;LOAD THE STACK POINTER
045640 013700 001276 .MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
045644 013701 001466 .MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
045650 012737 000115 001226 .MOV #115,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
5166
5167 045656 004737 054732 .JSR PL,SETVV ;GO SET VOLUME VALID
045662 000402 .BR 10$ ;BRANCH TO 10$ IF NO ERROR
045664 104000 .EMT
5168 045666 000550 .BR 90$
5169
5170 ;ENABLE DEBUG CLOCK AND LOAD SEARCH COMMAND
5171 10$:
5172 045670 .MOV #DMD!MUR!DBEN!MOC,RMR1(R0) ;LOAD RMR1
5173 045676 012760 000000 000014 .MOV #0,RMR1(R0) ;LOAD RMR1
5174 045704 012760 000000 000042 .MOV #0,RMR2(R0) ;LOAD RMR2
5175 045712 012760 000000 000034 .MOV #0,RMDC(R0) ;LOAD RMDC
5176 045720 012760 000000 000006 .MOV #0,RMDA(R0) ;LOAD RMDA
5177 045726 012760 000031 000000 .MOV #SEARCH!GO,RMCS1(R0) ;LOAD RMCS1
5178
5179 ;EXECUTE SEARCH TO TEST FOR ON LATCH RESET (17 CLOCKS)
5180 045734 012702 000021 .MOV #17.,R2
5181 20$:
5182 045740 012760 141401 000024 .MOV #DMD!MUR!DBEN!MOC!DBCK,RMR1(R0) ;LOAD RMR1
5183 045746 012760 041401 000024 .MOV #DMD!MUR!DBEN!MOC,RMR1(R0) ;LOAD RMR1
5184 045754 005302 .DEC R2
5185 045756 001370 .BNE 20$
5186
5187 045760 012760 041001 000024 .MOV #DMD!MUR!DBEN,RMR1(R0) ;LOAD RMR1
5188 045766 012760 041401 000024 .MOV #DMD!MUR!DBEN!MOC,RMR1(R0) ;LOAD RMR1

```

```

5189
5190
5191 045774 012702 000002
5192 046000 012760 151401 000024
5193 046006 012760 051401 000024
5194 046014 005302
5195 046016 00:370
5196
5197
5198
5199 046020 012702 000005
5200 046024 012760 151401 000024
5201 046032 012760 051401 000024
5202 046040 016037 000024 001142
5203 046046 012737 173777 001142
5204 046054 001403
5205 046056 005302
5206 046060 001361
5207 046062 000412
5208 046064 012737 004000 001140
5209 046072 010027 001136
5210 046076 062737 000024 001136
5211 046104 104265
5212 046106 000440
5213
5214
5215 046110
5216 046110 012760 041401 000024
5217 046116 012737 070000 001534
5218 046124 004777 133406
5219 046130 016037 000014 001142
5220 046136 042737 157777 001142
5221 046144 010107
5222 046146 005737 001534
5223 046152 001366
5224 046154 004777 133360
5225 046160 012737 020000 001140
5226 046166 010037 001136
5227 046172 062737 000014 001136
5228 046200 104267
5229 046202 000402
5230 046204 004777 133330
5231 046210
5232
5233

:STEP COMMAND SEQUENCER TO SECTOR COMPARE LOOP (? CLOCKS)
30$: MOV #2,R2
MOV #DMD!DBEN!MUR!MOC!DBCK!MSEN,RMMR1(RO) ;LOAD RMMR1
MOV #DMD!DBEN!MUR!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
DEC R2
BNE 30$

:THE COMMAND SEQUENCER IS NOW IN THE SECTOR COMPARE LOOP. STEP
:THROUGH THE LOOP AND VERIFY SEARCH IS ENABLED.
40$: MOV #5,R2
MOV #DMD!DBEN!MUR!MOC!DBCK!MSEN,RMMR1(RO) ;LOAD RMMR1
MOV #DMD!DBEN!MUR!MOC!MSEN,RMMR1(RO) ;LOAD RMMR1
MOV RMMR1(RO),SBDAT ;STORE RMMR1 AT SBDAT
BIC #ACESRL,SBDAT
BEQ 50$ ;BRANCH IF SEARCH NOT ENABLED
DEC R2
BNE 40$
BR 60$
50$: MOV #ESRC,SGDAT ;SETUP ERROR MSG
MOV RO,SBDADR
ADD #RMMR1,SBDADR
EMT 265
BR 90$

:DROP 'MSEN' TO ENABLE SEARCH TIMEOUT AND WAIT FOR OPI TO SET.
60$: MOV #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
MOV #70000,WATCH ;SET WATCHDOG TIMER VALUE
JSR PC,@CLOCK ;START THE CLOCK
70$: MOV RMMR1(RO),SBDAT ;STORE RMMR1 AT SBDAT
BIC #COPI,SBDAT
BNE 80$
TST WATCH
BNE 70$
JSR PC,@STOPCL ;STOP THE CLOCK
MOV #OPI,SGDAT ;SETUP ERROR MSG
MOV RO,SBDADR
ADD #RMMR1,SBDADR
EMT 267
BR 90$
80$: JSR PC,@STOPCL ;STOP THE CLOCK
90$: ;END OF TEST

*****
:TEST 116 DATA COMMAND TESTS (1)
*****
TST116:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
  
```

```

046220 013700 001276      MOV    $BASE,R0      ;R0 = UNIBUS ADDRESS
046224 013701 001466      MOV    1$TQUE,R1     ;R1 = POINTER TO DEVICE
046230 012737 000116 001226  MOV    4116,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX

5234
5235
5236
5237 046236 004737 054732      JSR    PC,SETVV      ;GO SET VOLUME VALID
046242 000402              BR     10$          ;BRANCH TO 10$ IF NO ERROR
046244 104000
5238 046246 000471              BR     40$

5239
5240
5241 046250              ;ENABLE DEBUG CLOCK AND LOAD READ COMMAND
5242 046250 012760 041401 000024 10$: MOV    #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5243 046256 012760 000000 000014  MOV    #0,RMER1(R0) ;LOAD RMER1
5244 046264 012760 000000 000042  MOV    #0,RMER2(R0) ;LOAD RMER2
5245 046272 012760 000000 000006  MOV    #0,RMDA(R0) ;LOAD RMDA
5246 046300 012760 000000 000034  MOV    #0,RMDC(R0) ;LOAD RMDC
5247 046306 012760 000007 000000  MOV    #RE!GO,RMCS1(R0) ;LOAD RMCS1
5248
5249
5250 046314 012702 000003      ;STEP COMMAND SEQUENCER TO UNIT READY TEST (3 CLOCKS)
5251 046320              MOV    #3,R2
5252 046320 012760 141401 000024 30$: MOV    #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5253 046326 012760 041401 000024  MOV    #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5254 046334 005302              DEC    R2
5255 046336 001370              LNE   20$
5256
5257 046340 012760 040401 000024 ;DROP UNIT READY
5258              MOV    #DMD!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5259
5260 046346 012702 000002      ;STEP SEQUENCER AND VERIFY OPI SETS (2 CLOCKS)
5261 046352              MOV    #2,R2
5262 046352 012760 140401 000024 30$: MOV    #DMD!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
5263 046366 005302              MOV    #DMD!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5264 046370 001370              DEC    R2
5265 046372 016037 000014 001142  BNE   30$
5266 046400 042737 157777 001142  MOV    RMER1(R0),$BDDAT ;STORE RMER1 AT $BDDAT
5267 046406 001011              BIC   #^COPI,$BDDAT
5268 046410 012737 020000 001140  BNE   40$
5269 046416 016037 001136  MOV    #OPI,$GDDAT
5270 046422 062737 000014 001136  MOV    R0,$BDADR
5271 046430 104270              ADD   #RMER1,$BDADR
5272              EMT  2:0
5273 046432 012737 046440 001124 40$: MOV    #50$,$LPERR ;CHANGE LOOP ON ERROR TEST
5274
5275
5276 046440              ;VERIFY DATA COMMAND ABORTS AT LOCATION 129
5277 046440 004737 054732      50$: JSR    PC,SETVV      ;GO SET VOLUME VALID
046444 000402              BR     60$          ;BRANCH TO 60$ IF NO ERROR
046446 104000
5278 046450 000576              EMT
5279              BR     150$
5280
5281 046452              ;ENABLE DEBUG CLOCK AND LOAD READ COMMAND
60$:

```

```

5282 046452 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMPR1(R0)      ;LOAD RMPR1
5283 046460 012760 000000 000014      MOV      #0,RMER1(R0)                      ;LOAD RMER1
5284 046466 012760 000000 000042      MOV      #0,RMER2(R0)                      ;LOAD RMER2
5285 046474 012760 000000 000006      MOV      #0,RMDA(R0)                       ;LOAD RMDA
5286 046502 012760 000000 000034      MOV      #0,RMDC(R0)                       ;LOAD RMDC
5287 046510 012760 000071 000000      MOV      #RD!GO,RMCS1(R0)                  ;LOAD RMCS1
5288
5289                                     ;STEP COMMAND SEQUENCER TO ABORT TEST AT LOCATION 129 (4 CLOCKS)
5290 046516 012702 000004      MOV      #4,R2
5291 046522                                     70$:
5292 046522 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMPR1(R0) ;LOAD RMPR1
5293 046530 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMPR1(R0)      ;LOAD RMPR1
5294 046536 005302                                     DEC      R2
5295 046540 001370                                     BNE     70$
5296
5297 046542 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMPR1(R0) ;LOAD RMPR1
5298
5299                                     ;SET DEVICE FAULT TO CAUSE ABORT CONDITION
5300 046550 012702 000001      MOV      #1,R2
5301 046554                                     80$:
5302 046554 012760 141501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF!DBCK,RMPR1(R0) ;LOAD RMPR1
5303 046562 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMPR1(R0) ;LOAD RMPR1
5304 046570 005302                                     DEC      R2
5305 046572 001370                                     BNE     80$
5306
5307                                     ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5308                                     ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5309 046574 012702 000020      MOV      #16,R2                          ;MAXIMUM NUMBER OF BIT CLOCKS
5310 046600                                     85$:
5311 046600 012760 045501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF!MCLK,RMPR1(R0) ;LOAD RMPR1
5312 046606 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMPR1(R0) ;LOAD RMPR1
5313 046614 016037 000024 001142      MOV      RMPR1(R0),SBDDAT                  ;STORE RMPR1 AT SBDDAT
5314 046622 042737 157777 001142      BIC     #^CEBL,SBDDAT
5315 046630 001014      BNE     90$                               ;BRANCH IF EBL IS SET
5316 046632 005302      DEC     R2
5317 046634 001361      BNE     85$                               ;CONTINUE BIT CLOCKS IF COUNT NOT 0
5318 046636 012737 020000 001140      MOV     #EBL,$GDDAT
5319 046644 010037 001136      MOV     R0,$BDADR
5320 046650 062737 000024 001136      ADD     #RMPR1,$BDADR
5321 046656 104271      EMT    271
5322 046660 000472      BR     150$
5323
5324 046602                                     ;STEP THE SEQUENCER THROUGH ITS TEST FOR EBL (2 CLOCKS)
5325 046662 012702 000002      MOV     #2,R2
5326 046666                                     90$:
5327 046666 012760 141501 000024      MOV     #DMD!MUR!MOC!DBEN!MDF!DBCK,RMPR1(R0) ;LOAD RMPR1
5328 046674 012760 041501 000024      MOV     #DMD!MUR!MOC!DBEN!MDF,RMPR1(R0) ;LOAD RMPR1
5329 046702 005302      DEC     R2
5330 046704 001370      BNE     100$
5331
5332                                     ;ABORT EBL SHOULD NOW BE INACTIVE - FORCE BIT CLOCK USING THE
5333                                     ;MAINTENANCE REGISTER TO RESET EBL (16 BIT CLOCKS)
5334 046706 012702 000020      MOV     #16,R2                          ;MAXIMUM NUMBER OF BIT CLOCKS
5335 046712                                     110$:

```

```

5336 046712 012760 045501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(RO) ;LOAD RMMR1
5337 046720 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMMR1(RO) ;LOAD RMMR1
5338 046726 005302                      DEC      R2
5339 046730 001370                      BNE     110$ ;ISSUE 16 BIT CLOCKS THEN TEST
5340
5341 046732                      :VERIFY EBL IS NOW RESET
5342 046732 016037 000024 001142      MOV      RMMR1(RO),S3DDAT ;STORE RMMR1 AT S3DDAT
5343 046740 042737 157777 001142      BIC     #^CEBL,S3DDAT
5344 046746 001411                      BEQ     130$ ;BRANCH IF EBL IS RESET
5345 046750 005037 001140      CLR     $GDDAT
5346 046754 010037 001136      MOV     RO,$BDADR
5347 046760 062737 000024 001136      ADD     #RMMR1,$BDADR
5348 046766 104273                      EMT     273
5349 046770 000426                      BR      150$
5350
5351                      :VERIFY GO RESETS WITHIN 4 CLOCK CYCLES
5352 046772                      130$:
5353 046772 012702 000004      MOV     #4,R2
5354 046776                      140$:
5355 047004 012760 141501 000024      MOV     #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(RO) ;LOAD RMMR1
5356 047012 005302 041501 000024      MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(RO) ;LOAD RMMR1
5357 047014 001370                      DEC     R2
5358 047016 016037 000000 001142      BNE     140$
5359 047024 042737 177776 001142      MOV     RMCS1(RO),S3DDAT ;STORE RMCS1 AT S3DDAT
5360 047032 001405                      BIC     #^CGO,S3DDAT
5361 047034 005037 001140      BEQ     150$
5362 047040 010037 001136      CLR     $GDDAT
5363 047044 104274                      MOV     RO,$BDADR
5364                      EMT     274
5365 047046 012737 047054 001124      150$: MOV     #200$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
5366
5367                      :VERIFY SEQUENCER BRANCHES TO SEEK WHEN RUN AND GO FLOP SETS
5368 047054                      200$:
5369 047054 004737 054732      JSR     PC,SETVV ;GO SET VOLUME VALID
5370 047060 062402                      BR      210$ ;BRANCH TO 210$ IF NO ERROR
5371 047062 104000                      EMT
5372 047064 000512                      BR      250$
5373
5373 047066                      :ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5374 047066 012760 041401 000024      210$: MOV     #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5375 047074 012760 000000 000014      MOV     #0,RMER1(RO) ;LOAD RMER1
5376 047102 012760 000000 000042      MOV     #0,RMER2(RO) ;LOAD RMER2
5377 047110 012760 000000 000006      MOV     #0,RMDA(RO) ;LOAD RMDA
5378 047116 012760 000000 000034      MOV     #0,RMDC(RO) ;LOAD RMDC
5379 047124 012760 000071 000000      MOV     #RD!GO,RMCS1(RO) ;LOAD RMCS1
5380
5381                      :MOVE SEQUENCER TO TEST FOR RUN AND GO AT LOCATION 130 (5 CLOCKS)
5382 047132 012702 000005      220$: MOV     #5,R2
5383 047136
5384 047136 012760 141401 000024      MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
5385 047144 012760 041401 000024      MOV     #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5386 047152 005302                      DEC     R2
5387 047154 001370                      BNE     220$

```

```

5388 ;VERIFY RUN AND GO IS SET
5389 047156 016037 000024 001142 MOV RMMR1(RO),SBDDAT ;STORE RMMR1 AT SBDDAT
5390 047164 042737 137777 001142 BIC #^CRG,SBDDAT
5391 047172 001012 BNE 230$
5392 047174 012737 040000 001140 MOV #RG,$GDDAT
5393 047202 010037 001136 MOV RO,$BDADR
5394 047206 062737 000024 001136 ADD #RMMR1,$BDADR
5395 047214 104275 EMT 275
5396 047216 000435 BR 250$
5397
5398 ;VERIFY THAT CYLINDER TAG COMES UP IN ONE CLOCK CYCLE
5399 047220 230$:
5400 047220 012702 000001 MOV #1,R2
5401 047224 240$:
5402 047232 012760 141401 000024 MOV #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
5403 047240 005302 041401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5404 047242 001370 DEC R2
5405 047244 016037 000040 001142 BNE 240$
5406 MOV RMMR2(RO),SBDDAT ;STORE RMMR2 AT SBDDAT
5407
5408 ;*****
5409 BIC #RQA!RQB!TAG,SBDDAT
5410 ;THE FOLLOWING CODE WAS ADDED
5411 ;TO ALLOW THE PROGRAM TO RUN WITH
5412 ;OR WITHOUT THE ECO TO THE CS BOARD.
5413 047252 042737 162000 001142 BIC #RQA!RQB!TAG!CH,SBDDAT ;HARDWARE ECO CHANGE CAUSES CC AND
5414 ;CH TO SET AT THE SAME TIME
5415 ;*****
5416
5417 047260 012737 004000 001140 MOV #CC,$GDDAT
5418 047266 023737 001140 001142 CMP $GDDAT,SBDDAT
5419 047274 001406 BEQ 250$
5420 047276 010037 001136 MOV RO,$BDADR
5421 047302 062737 000040 001136 ADD #RMMR2,$BDADR
5422 047310 104276 EMT 276
5423
5424 047312 012737 047324 001124 250$: MOV #260$,SLPERR ;CHANGE LOOP ON ERROR ADDRESS
5425
5426 ;VERIFY DATA COMMAND ABORTS AT COMMAND SEQUENCER LOCATIONS 144, 145
5427 047320 012702 000144 MOV #144,R2 ;INITIALIZE TEST LOCATION
5428 047324 260$:
5429 047324 004737 054732 JSR PC,SETVV ;GO SET VOLUME VALID
5430 047330 000402 BR 270$ ;BRANCH TO 270$ IF NO ERROR
5431 047332 104000 EMT
5432 047334 000553 BR 320$
5433
5434 ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5435 270$:
5436 047336 MOV #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5437 047344 012760 000030 000014 MOV #0,RMER1(RO) ;LOAD RMER1
5438 047352 012760 000000 000042 MOV #0,RMER2(RO) ;LOAD RMER2
5439 047360 012760 000000 000006 MOV #0,RMDA(RO) ;LOAD RMDA
5440 047366 012760 000000 000034 MOV #0,RMDC(RO) ;LOAD RMDC
5441 047374 012760 000071 000000 MOV #RD!GO,RMCS1(RO) ;LOAD RMCS1
5442
5443 ;WAIT FOR RUN AND GO TO SET
  
```

```

5442 047402 012737 000310 001534      MOV      #200, WATCH      ;SET WATCHDOG TIMER VALUE
      047410 004777 132122      JSR      PC,@CLOCK      ;START THE CLOCK
5443 047414      280$:      MOV      RMMR1(RO), $BDDAT      ;STORE RMMR1 AT $BDDAT
      047414 016037 000024 001142      BIC      #^CRG, $BDDAT
5444 047422 042737 137777 001142      BNE
5445 047430 001017      290$
5446 047432 005737 001534      TST      WATCH
5447 047436 001366      BNE      280$
5448 047440 004777 132074      JSR      PC,@STOPCL      ;STOP THE CLOCK
5449 047444 012737 040000 001140      MOV      #RG, $GDDAT
5450 047452 010037 001136      MOV      RO, $BDADR
5451 047456 062737 000024 001136      ADD      #RMMR1, $BDADR
5452 047464 104275      EMT
5453 047466 000476      BR       320$
5454 047470      290$:      JSR      PC,@STOPCL      ;STOP THE CLOCK
      047470 004777 132044
5455      ;MOVE COMMAND SEQUENCER TO ABORT TEST (LOCATION 144 OR 145)
5456      MOV      #6, R3      ;SETUP CLOCK COUNT
5457 047474 012703 000006      CMP      #144, R2
5458 047500 022702 000144      BEQ      300$
5459 047504 001402      MOV      #7, R3
5460 047506 012703 000007      300$:      MOV      #DMD!MUR!MOC!DBEN!DBCK, RMMR1(RO)      ;LOAD RMMR1
5461 047512 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN, RMMR1(RO)      ;LOAD RMMR1
      047512 012760 041401 000024
5462 047520 012760 041401 000024      DEC      R3
5463 047526 005303      BNE      300$
5464 047530 001370
5465
5466      ;SET DRIVE FAULT TO FORCE ABORT CONDITION
5467 047532 012760 041501 009024      MOV      #DMD!MUR!MOC!DBEN!MDF, RMMR1(RO) ;LOAD RMMR1
5468
5469      ;CLOCK SEQUENCER THROUGH ITS TEST FOR ABORT (1 CLOCK)
5470 047540 012703 000001      MOV      #1, R3
5471 047544      305$:      MOV      #DMD!MUR!MOC!DBEN!MDF!DBCK, RMMR1(RO) ;LOAD RMMR1
      047544 012760 141501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF, RMMR1(RO) ;LOAD RMMR1
5472 047552 012760 041501 000024      DEC      R3
5473 047560 005303      BNE      305$      ;ISSUE 2 CLOCKS
5474 047562 001370
5475
5476      ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5477      ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
      MOV      #16., R2      ;MAXIMUM NUMBER OF BIT CLOCKS
5478 047564 012702 000020      306$:      MOV      #DMD!MUR!MOC!DBEN!MDF!MCLK, RMMR1(RO) ;LOAD RMMR1
      047570 012760 045501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF, RMMR1(RO) ;LOAD RMMR1
5480 047576 012760 041501 000024      MOV      RMMR1(RO), $BDDAT      ;STORE RMMR1 AT $BDDAT
5481 047604 016037 000024 001142      BIC      #^CEBL, $BDDAT
5482 047612 042737 157777 001142      BNE      310$      ;BRANCH IF EBL IS SET
5483 047620 001013
5484
5485 047622 005302      DEC      R2
5486 047624 001361      BNE      306$      ;CONTINUE BIT CLOCKS IF COUNT NOT 0
5487 047626 012737 020000 001140      MOV      #EBL, $GDDAT
5488 047634 010037 001136      MOV      RO, $BDADR
5489 047640 062737 000024 001136      ADD      #RMMR1, $BDADR
5490 047646 104271      EMT
5491 047650 022702 000144      310$:      CMP      #144, R2
5492 047654 001003      BNE      320$
  
```

```
5493 047656 012702 000145      MOV      #145,R2
5494 047662 000620      BR       260$
5495
5496 047664 012737 047674 001124 320$:  MOV      #330$, $LPERR      ;CHANGE LOOP ON ERROR ADDRESS
5497
5498      ;VERIFY HEAD TAG DURING DATA COMMAND
5499 047672 005002      CLR      R2      ;INITIALIZE TRACK ADDRESS = 0
5500 047674      330$:
5501 047674 004737 054732      JSR      PC,SETVV      ;GO SET VOLUME VALID
      047700 000402      BR       340$      ;BRANCH TO 340$ IF NO ERROR
      047702 104000      EMT
5502 047704 000561      BR       400$
5503
5504      ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5505 047706      340$:
5506 047706 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5507 047714 012760 000000 000014      MOV      #0,RMER1(RO) ;LOAD RMER1
5508 047722 012760 000000 000042      MOV      #0,RMER2(RO) ;LOAD RMER2
5509 047730 010260 000006      MOV      R2,RMDA(RO) ;LOAD RMDA
5510 047734 012760 000000 000034      MOV      #0,RMDC(RO) ;LOAD RMDC
5511 047742 012760 000071 000000      MOV      #RD!GO,RMCS1(RO) ;LOAD RMCS1
5512
5513      ;WAIT FOR RUN AND GO TO SET
5514 047750 012737 000310 001534      MOV      #200, WATCH ;SET WATCHDOG TIMER VALUE
      047756 004777 131554      JSR      PC,@CLOCK ;START THE CLOCK
5515 047762      350$:
      047762 016037 000024 001142      MOV      RMMR1(RO), $BDDAT ;STORE RMMR1 AT $BDDAT
5516 047770 042737 137777 001142      BIC      #^CRG,$BDDAT
5517 047776 001017      BNE      360$
5518 050000 005737 001534      TST      WATCH
5519 050004 001366      BNE      350$
5520 050006 004777 131526      JSR      PC,@STOPCL ;STOP THE CLOCK
5521 050012 012737 040000 001140      MOV      #RG,$GDDAT
5522 050020 010037 001136      MOV      RO,$BDADR
5523 050024 062737 000024 001136      ADD      #RMMR1,$BDADR
5524 050032 104275      EMT
5525 050034 000505      BR       400$
5526 050036 004777 131476      360$:  JSR      PC,@STOPCL ;STOP THE CLOCK
5527
5528      ;STEP COMMAND SEQUENCER TO HEAD SEQUENCE, LOCATION 156 (17 CLOCKS)
5529 050042 012703 000021      MOV      #17.,R3
5530 050046      370$:
      050046 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
5531 050054 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5532 050062 005303      DEC      R3
5533 050064 001370      BNE      370$
5534
5535      ;DROP AND RAISE ON CYLINDER TO RESET ON LATCH
5536 050066 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
5537 050074 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5538
5539      ;*****
5540      ;      MOV      #450$,R3      ;INITIALIZE TABLE POINTER
5541
5542      ;      ;HARDWARE ECO CHANGE TO THE PLA ON THE
5543      ;      ;CS BOARD.
```



```

5544 050102 012703 000004      MOV      #4,R3          ;CLOCK THE SEQUENCER THRU THE FIRST
5545                               ;4 COMMAND SEQUENCES TO ALLOW THE PROGRAM
5546                               ;TO RUN WITH OR WITHOUT THE ECO.
5547 050106                               372$:
5548 050106 012760 141401 000024      MOV      #DMD!DBEN!MUR!MOC!DBCK,RMMR1(RO) ;LOAD RMMR1
5549 050114 012760 041401 000024      MOV      #DMD!DBEN!MUR!MOC,RMMR1(RO) ;LOAD RMMR1
5550 050122 005303                               DEC      R3          ;DONE 4 CLOCKS ?
5551 050124 001370                               BNE     372$        ;NO !!
5552 050126 012703 050262      MOV      #475$,R3     ;INITIALIZE NEW TABLE POINTER
5553                               ;:*****
5554                               ;:VERIFY TAG BUS ACCORDING TO TABLE AND TRACK ADDRESS IN R2
5555                               375$:
5556 050132 016037 000040 001142      MOV      RMMR2(RO),SBDDAT ;STORE RMMR2 AT SBDDAT
5557 050132 016037 000040 001142      BIC     #RQA!RQB!TST,SBDDAT
5558 050140 042737 150000 001142      MOV      (R3),SGDDAT
5559 050146 011337 001140      MOV      R2,R4          ;GENERATE EXPECTED TAG BUS
5560 050152 010204                               SWAB    R4
5561 050154 000304                               BJS     R4,$GDDAT
5562 050156 050437 001140      ;COMPARE EXPECTED AND RECEIVED TAG BUS DATA
5563                               CMP     $GDDAT,$BDDAT
5564                               BNE     390$
5565 050162 023737 001140 001142      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
5566 050170 001021                               MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5567 050172 012760 141401 000024
5568 050200 012760 041401 000024
5569
5570                               ;ADVANCE TO NEXT TABLE ENTRY
5571 050206 062703 000002      ADD     #2,R3
5572 050212 005713                               TST     (R3)
5573 050214 100401                               BMI     380$
5574 050216 000745                               BR      375$
5575
5576                               ;SHIFT TO NEXT TRACK ADDRESS-EXIT LOOP IF DONE
5577 050220                               380$:
5578 050220 062702 000400      ADD     #TA1,R2          ;ADVANCE TRACK ADDRESS
5579 050224 020237 001334      CMP     R2,LSTRK        ;DONE ALL TRACKS ?
5580 050230 101007                               BHI     400$          ;YES, EXIT
5581 050232 000620                               BR      330$
5582
5583                               ;ERROR ON TAG BUS DURING HEAD SEQUENCE
5584 050234                               390$:
5585 050234 010037 001136      MOV     R0,$BDADR
5586 050240 062737 000040 001136      ADD     #RMMR2,$BDADR
5587 050246 104276                               EMT     276
5588
5589 050250 000440      400$: BR      500$          ;JUMP OVER TABLE
5590
5591                               ;TABLE OF TAG BUS DURING HEAD SEQUENCE
5592 050252                               450$:
5593 050252 002000      .WORD  CH
5594 050254 002000      .WORD  CH
5595 050256 022000      .WORD  CH!TAG
5596 050260 022000      .WORD  CH!TAG
5597 050262                               475$:
5598 050262 022000      .WORD  CH!TAG          ;START TABLE HERE FOR HARDWARE ECO CHANGE
5599 050264 022000      .WORD  CH!TAG
5600 050266 022000      .WORD  CH!TAG
  
```

```

5601 050270 022000 .WORD CH!TAG
5602 050272 177777 .WORD -1 ;END TABLE HERE FOR HARDWARE ECO CHANGE
5603
5604 ; .WORD CH
5605 050274 002000 .WORD CH
5606 050276 001777 .WORD 1777
5607 050300 001777 .WORD 1777
5608 050302 001777 .WORD 1777
5609 050304 001777 .WORD 1777
5610 050306 001777 .WORD 1777
5611 050310 001777 .WORD 1777
5612 050312 001777 .WORD 1777
5613 050314 001777 .WORD 1777
5614 050316 001777 .WORD 1777
5615 050320 001777 .WORD 1777
5616 050322 001777 .WORD 1777
5617 050324 001777 .WORD 1777
5618 050326 001777 .WORD 1777
5619 050330 001777 .WORD 1777
5620 050332 001777 .WORD 1777
5621 050334 001777 .WORD 1777
5622 050336 001777 .WORD 1777
5623 050340 001777 .WORD 1777
5624 050342 001777 .WORD 1777
5625 050344 001777 .WORD 1777
5626 050346 001777 .WORD 1777
5627
5628 050350 177777 .WORD -1 ;END OF TABLE
5629
5630 050352 500$: ;END OF TEST
5631
5632

```

\*\*\*\*\*  
 :\*TEST 117 DATA COMMAND TESTS (2)  
 \*\*\*\*\*

```

050352
050352 000004
050354 000240
050356 012706 001100
050362 013700 001276
050366 013701 001466
050372 012737 000117 001226
5633
5634
5635
5636 050400 004737 054732 JSR PC,SETVV ;GO SET VOLUME VALID
050404 000402 BR 10$ ;BRANCH TO 10$ IF NO ERROR
050406 104000 EMT
5637 050410 000514 BR 60$
5638
5639 ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND DURING CYLINDER SEQUENCE
10$:
5640 050412
5641 050412 012760 041401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5642 050420 012760 000000 000014 MOV #0,RMER1(RO) ;LOAD RMER1
5643 050426 012760 000000 000042 MOV #0,RMER2(RO) ;LOAD RMER2
5644 050434 012760 000000 000006 MOV #0,RMDA(RO) ;LOAD RMDA
5645 050442 012760 000000 000034 MOV #0,RMDC(RO) ;LOAD RMDC

```

```

5646 050450 012760 000071 000000      MOV      #RD!GO,RMCS1(R0)          ;LOAD RMCS1
5647
5648
5649 050456 012737 000310 001534      ;WAIT FOR RUN AND GO TO SET
                                MOV      #200.,WATCH          ;SET WATCHDOG TIMER VALUE
                                JSR      PC,@CLOCK             ;START THE CLOCK
5650 050470 016037 000024 001142      20$:      MOV      RMMR1(R0),SBDDAT          ;STORE RMMR1 AT SBDDAT
                                BIC      #^CRG,SBDDAT
                                BNE      30$
                                TST      WATCH
                                BNE      20$
                                JSR      PC,@STOPCL           ;STOP THE CLOCK
5651 050476 042737 137777 001142      MOV      #RG,$GDDAT
5652 050504 001017 000000 001140      MOV      R0,$BDADR
5653 050506 005737 001534      ADD      #RMMR1,$BDADR
5654 050512 001366 000024 001136      EMT      275
5655 050514 004777 131020 001136      BR       60$
5656 050520 012737 040000 001140      30$:      JSR      PC,@STOPCL           ;STOP THE CLOCK
5657 050526 010037 001136
5658 050532 062737 000024 001136
5659 050540 104275
5660 050542 000437
5661 050544 004777 130770
5662
5663      ;STEP COMMAND SEQUENCER AND VERIFY OPI SETS (19 CLOCKS)
5664 050550 012702 000023      MOV      #19.,R2
5665 050554 012760 141401 000024      40$:      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)      ;LOAD RMMR1
                                MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0)      ;LOAD RMMR1
                                DEC      R2
                                BNE      40$
5666 050562 012760 041401 000024      MOV      RMER1(R0),SBDDAT          ;STORE RMER1 AT SBDDAT
5667 050570 005302 001370 001142      BIC      #^COPI,SBDDAT
5668 050572 001370 000014 001142      BNE      50$                      ;BRANCH IF OPI SET
5669 050574 016037 000014 001140      MOV      #OPE,$GDDAT
5670 050602 042737 157777 001140      MOV      R0,$BDADR
5671 050610 001011 020000 001140      ADD      #RMER1,$BDADR
5672 050612 012737 020000 001140      EMT      277
5673 050620 010037 001136
5674 050624 062737 000014 001136      50$:      MOV      #60$,$LPERR          ;CHANGE LOOP ON ERROR ADDRESS
5675 050632 104277
5676 050634 012737 050642 001124
5677
5678      ;VERIFY DATA COMMAND ABORTS DURING SEEK WAIT LOOP
5679 050642 004737 054732      60$:      JSR      PC,SETVV          ;GO SET VOLUME VALID
5680 050642 000402 054732      BR       70$                      ;BRANCH TO 70$ IF NO ERROR
5681 050650 104000
5682 050652 000567
5683
5684 050654 012760 041401 000024      ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5685 050654 012760 000000 000014      70$:      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0)      ;LOAD RMMR1
5686 050662 012760 000000 000014      MOV      #0,RMER1(R0)          ;LOAD RMER1
5687 050670 012760 000000 000042      MOV      #0,RMER2(R0)          ;LOAD RMER2
5688 050676 012760 000000 000034      MOV      #0,RMDC(R0)          ;LOAD RMDC
5689 050704 012760 000000 000006      MOV      #0,RMDA(R0)          ;LOAD RMDA
5690 050712 012760 000071 000000      MOV      #RD!GO,RMCS1(R0)      ;LOAD RMCS1
5691
5692      ;WAIT FOR RUN & GO TO SET
5693 050720 012737 000310 001534      MOV      #200.,WATCH          ;SET WATCHDOG TIMER VALUE
                                JSR      PC,@CLOCK             ;START THE CLOCK
5694 050732 016037 000024 001142      80$:      MOV      RMMR1(R0),SBDDAT          ;STORE RMMR1 AT SBDDAT
  
```

```

5695 050740 042737 137777 001142      BIC      #^CRG,$BDDAT
5696 050746 001017                      BNE      90$
5697 050750 005737 001534                      TST      WATCH
5698 050754 001366                      BNE      80$
5699 050756 004777 130556                      JSR      PC,@STOPCL      ;STOP THE CLOCK
5700 050762 012737 040000 001140      MOV      #RG,$GDDAT
5701 050770 010037 001136                      MOV      R0,$BDADR
5702 050774 062737 000024 001136      ADD      #RMMR1,$BDADR
5703 051002 104275                      EMT      275
5704 051004 000512                      BR       140$
5705 051006                      90$:
5706 051006 004777 130526                      JSR      PC,@STOPCL      ;STOP THE CLOCK
5707                      ;STEP COMMAND SEQUENCER TO ON LATCH TEST AT LOCATION 156 (17 CLOCKS)
5708 051012 012702 000021                      MOV      #17.,R2
5709 051016                      100$:
5710 051016 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)      ;LOAD RMMR1
5711 051024 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0)      ;LOAD RMMR1
5712 051032 005302                      DEC      R2
5713 051034 001370                      BNE      100$
5714                      ;DROP ON CYLINDER TO RESET LATCH
5715 051036 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5716                      ;MOVE COMMAND SEQUENCER TO SEEK WAIT LOOP (31 CLOCKS)
5717                      110$:
5718 051044 012702 000037                      MOV      #31.,R2
5719 051050                      110$:
5720 051050 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0)      ;LOAD RMMR1
5721 051056 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5722 051064 005302                      DEC      R2
5723 051066 001370                      BNE      110$
5724                      ;STEP THROUGH SEEK WAIT LOOP (6 CLOCKS) 2 TIMES
5725 051070 012702 000006                      MOV      #6.,R2
5726 051074                      120$:
5727 051074 012760 141001 000024      MOV      #DMD!MUR!DBEN!DBCK,RMMR1(R0)      ;LOAD RMMR1
5728 051102 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5729 051110 005302                      DEC      R2
5730 051112 001370                      BNE      120$
5731                      ;SET SEEK INCOMPLETE ERROR TO CAUSE ABORT
5732 051114 012760 041201 000024      MOV      #DMD!MUR!DBEN!MSER,RMMR1(R0)      ;LOAD RMMR1
5733                      ;CLOCK THE SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
5734                      130$:
5735 051122 012702 000002                      MOV      #2.,R2
5736 051126                      130$:
5737 051126 012760 141201 000024      MOV      #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0)      ;LOAD RMMR1
5738 051134 012760 041201 000024      MOV      #DMD!MUR!DBEN!MSER,RMMR1(R0) ;LOAD RMMR1
5739 051142 005302                      DEC      R2
5740 051144 001370                      BNE      130$
5741                      ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTERS TO
5742                      ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5743 051146 012702 000020                      MOV      #16.,R2      ;MAXIMUM NUMBER OF BIT CLOCKS
5744 051152                      135$:
5745 051152 012760 045501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0)      ;LOAD RMMR1
5745 051160 012760 041501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
  
```

```

5746 051166 016037 000024 001142      MOV      RMMR1(R0),SBDDAT      ;STORE RMMR1 AT SBDDAT
5747 051174 042737 157777 001142      BIC      #^CEBL,SBDDAT
5748 051202 001013                    BNE      140$
5749
5750 051204 005302                    DEC      R2
5751 051206 001361                    BNE      135$                  ;CONTINUE BIT CLOCKS IF COUNT NOT 0
5752 051210 012737 020000 001140      MOV      #EBL,$GDDAT
5753 051216 010037 001136      MOV      R0,$BDADR
5754 051222 062737 000024 001136      ADD      #RMMR1,$BDADR
5755 051230 104300                    EMT      300
5756 051232 012737 051240 001124 140$:  MOV      #150$,$LPERR          ;CHANGE LOOP ON ERROR ADDRESS
5757
5758                                ;VERIFY DATA COMMAND ABORTS DURING OFFSET IF ON CYLINDER LATCH
5759                                ;DOESNT RESET
5760 051240                    150$:
5761 051240 004737 054732      JSR      PC,SETVV              ;GO SET VOLUME VALID
5762 051244 000402                    BR      160$                  ;BRANCH TO 160$ IF NO ERROR
5763 051246 104000                    EMT
5764 051250 000536                    BR      220$
5765
5766                                ;LOAD TRACK, SECTOR AND CYLINDER ADDRESSES
5767 051252                    160$:
5768 051252 012760 000000 000006      MOV      #0,RMDA(R0)          ;LOAD RMDA
5769 051260 012760 000000 000034      MOV      #0,RMDC(R0)          ;LOAD RMDC
5770
5771 051266 004737 055054      JSR      PC,SETOM              ;GO SET OFFSET MODE
5772 051272 000402                    BR      170$                  ;BRANCH TO 170$ IF NO ERROR
5773 051274 104000                    EMT
5774 051276 000523                    BR      220$
5775
5776                                ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5777 051300                    170$:
5778 051300 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5779 051306 012760 000000 000014      MOV      #0,RMER1(R0)         ;LOAD RMER1
5780 051314 012760 000000 000042      MOV      #0,RMER2(R0)         ;LOAD RMER2
5781 051322 012760 000071 000000      MOV      #RD!GO,RMCS1(R0)     ;LOAD RMCS1
5782
5783                                ;WAIT FOR RUN AND GO TO SET
5784 051330 012737 000310 001534      MOV      #200$,WATCH          ;SET WATCHDOG TIMER VALUE
5785 051336 004777 130174                    JSR      PC,@CLOCK            ;START THE CLOCK
5786
5787 051342                    180$:
5788 051342 016037 000024 001142      MOV      RMMR1(R0),SBDDAT      ;STORE RMMR1 AT SBDDAT
5789 051350 042737 137777 001142      BIC      #^CRG,SBDDAT
5790 051356 001017                    BNE      190$
5791 051360 005737 001534                    TST      WATCH
5792 051364 001366                    BNE      180$
5793 051366 004777 130146                    JSR      PC,@STOPCL          ;STOP THE CLOCK
5794 051372 012737 040000 001140      MOV      #RG,$GDDAT
5795 051400 010037 001140      MOV      R0,$GDDAT
5796 051404 062737 000024 001136      ADD      #RMMR1,$BDADR
5797 051412 104275                    EMT      275
5798 051414 000454                    BR      220$
5799
5800                                190$:
5801 051416 004777 130116                    JSR      PC,@STOPCL          ;STOP THE CLOCK
5802
5803                                ;STEP COMMAND SEQUENCER TO ON LATCH TEST AT LOCATION 156 (17 CLOCKS)
5804 051422 012702 000021                    MOV      #17$,R2
  
```

```

5796 051426 200$:
      051426 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
5797 051434 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5798 051442 005302      DEC      R2
5799 051444 001370      BNE     200$
5800
5801      ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
5802      ;AT LOCATION 166
5803 051446 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
5804 051454 012760 041401 000024      MOV      #DMD!MUR!MOC,RMMR1(RO) ;LOAD RMMR1
5805
5806      ;MOVE SEQUENCER TO SET OPI AND EBL (39 CLOCKS)
5807 051462 012702 000047      MOV      #39.,R2
5808 051466 210$:
      051466 012760 141401 000024      MOV      #DMD!MUR!DBEN!MOC!DBCK,RMMR1(RO) ;LOAD RMMR1
5809 051474 012760 041401 000024      MOV      #DMD!MUR!DBEN!MOC,RMMR1(RO) ;LOAD RMMR1
5810 051502 005302      DEC      R2
5811 051504 001370      BNE     210$
5812
5813      ;VERIFY OPI IS SET
5814 051506 016037 000014 001142      MOV      RMER1(RO),SBDDAT ;STORE RMER1 AT SBDDAT
5815 051514 042737 157777 001142      BIC     #*COPI,SBDDAT
5816 051522 0C1011      BNE     220$
5817 051524 012737 020000 001140      MOV      #OPI,$GDDAT
5818 051532 010037 001136      MOV      RO,$BDADR
5819 051536 062737 000014 001136      ADD     #RMER1,$BDADR
5820 051544 104277      EMT     277
5821 051546 012737 051554 001124 220$: MOV      #230$,$LPERR ;CHANGE LOOP ON ERROR ADDRESS
5822
5823      ;VERIFY DATA COMMAND ABORTS DURING OFFSET WAIT LOOP
5824
5825 051554 230$:
      051554 004737 054732      JSR     PC,SETVV ;GO SET VOLUME VALID
      051560 000403      BR     240$ ;BRANCH TO 240$ IF NO ERROR
      051562 104000      EMT
5826 051564 000137 052174      JMP     310$
5827
5828      ;LOAD SECTOR,TRACK AND CYLINDER ADDRESS
5829 051570 240$:
5830 051570 012760 000000 000006      MOV      #0,RMDA(RO) ;LOAD RMDA
5831 051576 012760 000000 0C0034      MOV      #0,RMDC(RO) ;LOAD RMDC
5832
5833 051604 004737 055054      JSR     PC,SETOM ;GO SET OFFSET MODE
      051610 000402      BR     245$ ;BRANCH TO 245$ IF NO ERROR
      051612 104000      EMT
5834 051614 000567      BR     310$
5835
5836      ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5837 051616 245$:
      051616 012760 041401 0C0024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
5838 051624 012760 000000 000014      MOV      #0,RMER1(RO) ;LOAD RMER1
5839 051632 012760 000000 000042      MOV      #0,RMER2(RO) ;LOAD RMER2
5840 051640 012760 000071 000000      MOV      #RD!GO,RMCS1(RO) ;LOAD RMCS1
5841
5842      ;WAIT FOR RUN AND GO TO SET
5843 051646 012737 000310 001534      MOV      #200.,WATCH ;SET WATCHDOG TIMER VALUE
      051654 004777 127656      JSR     PC,@CLOCK ;START THE CLOCK
  
```

5844	051660				250\$:				
	051660	016037	000024	001142		MOV	RMMR1(RO),SBDDAT		;STORE RMMR1 AT SBDDAT
5845	051666	042737	137777	001142		BIC	#^CRG,SBDDAT		
5846	051674	001017				BNE	260\$		
5847	051676	005737	001534			TST	WATCH		
5848	051702	001366				BNE	250\$		
5849	051704	004777	127630			JSR	PC,@STOPCL		;STOP THE CLOCK
5850	051710	012737	040000	001140		MOV	#RG,\$GDDAT		
5851	051716	010037	001136			MOV	RO,\$BDADR		
5852	051722	062737	000024	001136		ADD	#RMMR1,\$BDADR		
5853	051730	104275				EMT	275		
5854	051732	000520				BR	310\$		
5855	051734				260\$:				
	051734	004777	127600			JSR	PC,@STOPCL		;STOP THE CLOCK
5856									
5857									
5858	051740	012702	000021						;STEP SEQUENCER TO LOCATION 156 (17 CLOCKS)
5859	051744					MOV	#17.,R2		
	051744	012760	141401	000024	270\$:				
5860	051752	012760	041401	000024		MOV	#DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO)		;LOAD RMMR1
5861	051760	005302				MOV	#DMD!MUR!MOC!DBEN,RMMR1(RO)		;LOAD RMMR1
5862	051762	001370				DEC	R2		
5863						BNE	270\$		
5864									;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
5865									;AT LOCATION 166
5866	051764	012760	041001	000024		MOV	#DMD!MUR!DBEN,RMMR1(RO)		;LOAD RMMR1
5867	051772	012760	041401	000024		MOV	#DMD!MUR!MOC!DBEN,RMMR1(RO)		;LOAD RMMR1
5868									
5869									;MOVE SEQUENCER TO LOCATION 174 (37 CLOCKS)
5870	052000	012702	000045			MOV	#37.,R2		
5871	052004				280\$:				
	052004	012760	141401	000024		MOV	#DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO)		;LOAD RMMR1
5872	052012	012760	041401	000024		MOV	#DMD!MUR!MOC!DBEN,RMMR1(RO)		;LOAD RMMR1
5873	052020	005302				DEC	R2		
5874	052022	001370				BNE	280\$		
5875									
5876									;DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER RESET
5877	052024	012760	041001	000024		MOV	#DMD!MUR!DBEN,RMMR1(RO)		;LOAD RMMR1
5878									
5879									;STEP SEQUENCER THROUGH OFFSET WAIT LOOP TWICE (7 CLOCKS)
5880	052032	012702	000007			MOV	#7.,R2		
5881	052036				290\$:				
	052036	012760	141001	000024		MOV	#DMD!MUR!DBEN!DBCK,RMMR1(RO)		;LOAD RMMR1
5882	052044	012760	041001	000024		MOV	#DMD!MUR!DBEN,RMMR1(RO)		;LOAD RMMR1
5883	052052	005302				DEC	R2		
5884	052054	001370				BNE	290\$		
5885									
5886									;SET DRIVE FAULT TO CAUSE ABORT CONDITION
5887	052056	012760	041101	000024		MOV	#DMD!MUR!DBEN!MDF,RMMR1(RO)		;LOAD RMMR1
5888									
5889									;STEP SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
5890	052064	012702	000002			MOV	#2,R2		
5891	052070				300\$:				
	052070	012760	141101	000024		MOV	#DMD!MUR!DBEN!MDF!DBCK,RMMR1(RO)		;LOAD RMMR1
5892	052076	012760	041101	000024		MOV	#DMD!MUR!DBEN!MDF,RMMR1(RO)		;LOAD RMMR1
5893	052104	005302				DEC	R2		
5894	052106	001370				BNE	300\$		

```

5895
5896      :ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5897      :FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5898 052110 012702 000020      MOV      #16.,R2      ;MAXIMUM NUMBER OF BIT CLOCKS
5899 052114      305$:
      MOV      #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(RO)      ;LOAD RMMR1
5900 052122 012760 045501 000024      MOV      #DMD!MUR!MOC!DBEN!MDF,RMMR1(RO) ;LOAD RMMR1
5901 052130 016037 000024 001142      MOV      RMMR1(RO),SBDDAT      ;STORE RMMR1 AT SBDDAT
5902 052136 042737 157777 001142      BIC      #^CEBL,SBDDAT
5903 052144 001013      BNE      310$
5904
5905 052146 005302      DEC      R2
5906 052150 001361      BNE      305$      ;CONTINUE BIT CLOCKS IF COUNT NOT 0
5907 052152 012737 020000 001140      MOV      #EBL,$GDDAT
5908 052160 010037 001136      MOV      RO,$BDADR
5909 052164 062737 000024 001136      ADD      #RMMR1,$BDADR
5910 052172 104271      EMT      271
5911
5912 052174 012737 052202 001124 310$: MOV      #320$,SLPERR      ;CHANGE LOOP ON ERROR ADDRESS
5913
5914      ;VERIFY THAT DATA COMMAND ABORTS DURING SECTOR WAIT LOOP AT LOCATION 179
5915
5916 052202      320$:
      052202 004737 054732      JSR      PC,SETVV      ;GO SET VOLUME VALID
      052206 000403      BR       330$      ;BRANCH TO 330$ IF NO ERROR
      052210 104000      EMT
5917 052212 000137 052640      JMP      420$
5918
5919      ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
5920 052216      330$:
5921 052216 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO)      ;LOAD RMMR1
5922 052224 012760 000000 000014      MOV      #0,RMER1(RO)      ;LOAD RMER1
5923 052232 012760 000000 000042      MOV      #0,RMER2(RO)      ;LOAD RMER2
5924 052240 012760 000000 000006      MOV      #0,RMDA(RO)      ;LOAD RMDA
5925 052246 012760 000000 000034      MOV      #0,RMDC(RO)      ;LOAD RMDC
5926 052254 012760 000071 000000      MOV      #RD!GO,RMCS1(RO)      ;LOAD RMCS1
5927
5928      ;WAIT FOR RUN AND GO TO SET
5929 052262 012737 000310 001534      MOV      #200.,WATCH      ;SET WATCHDOG TIMER VALUE
      052270 004777 127242      JSR      PC,@CLOCK      ;START THE CLOCK
5930 052274      340$:
      052274 016037 000024 001142      MOV      RMMR1(RO),SBDDAT      ;STORE RMMR1 AT SBDDAT
5931 052302 042737 137777 001142      BIC      #^CRG,SBDDAT
5932 052310 001017      BNE      350$
5933 052312 005737 001534      TST      WATCH
5934 052316 001366      BNE      340$
5935 052320 004777 127214      JSR      PC,@STOPCL      ;STOP THE CLOCK
5936 052324 012737 040000 001140      MOV      #RG,$GDDAT
5937 052332 010037 001136      MOV      RO,$BDADR
5938 052336 062737 000024 001136      ADD      #RMMR1,$BDADR
5939 052344 104275      EMT      275
5940 052346 000534      BR       420$
5941 052350      350$:
      052350 004777 127164      JSR      PC,@STOPCL      ;STOP THE CLOCK
5942
5943      ;STEP SEQUENCER TO LOCATION 156 (17 CLOCKS)
5944 052354 012702 000021      MOV      #17.,R2
  
```



```

5945 052360 012760 141401 000024 360$: MOV #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
      052360 012760 041401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5946 052366 012760 041401 000024 DEC R2
5947 052374 005302 BNE 360$
5948 052376 001370
5949
5950 ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
5951 ;AT LOCATION 166
5952 052400 012760 041001 000024 MOV #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
5953
5954 ;MOVE SEQUENCER TO SECTOR WAIT (34 CLOCKS)
5955 052406 012702 000042 MOV #34.,R2
5956 052412 370$: MOV #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
      052412 012760 141401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5957 052420 012760 041401 000024 DEC R2
5958 052426 005302 BNE 370$
5959 052430 001370
5960
5961 ;STEP THROUGH SECTOR WAIT LOOP TWICE AND VERIFY SEARCH IS ENABLED
5962 ;DURING THE LOOP (6 CLOCKS)
5963 052432 012702 000006 MOV #6.,R2
5964 052436 380$: MOV #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
      052436 012760 141401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
5965 052444 012760 041401 000024 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
5966 052452 016037 000024 001142 BIC #^CESRC,SBDDAT
5967 052460 042737 173777 001142 BEQ 390$
5968 052466 001403 DEC R2
5969 052470 005302 BNE 380$
5970 052472 001361 BR 400$
5971 052474 000412 390$: MOV #ESRC,$GDDAT
      052476 012737 004000 001140 MOV RO,$BDADR
5972 052476 012737 004000 001140 ADD #RMMR1,$BDADR
5973 052504 010037 001136 EMT 301
5974 052510 062737 000024 001136 BR 420$
5975 052516 104301
5976 052520 000447
5977
5978 ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
5979 052522 400$: MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
      052522 012760 041501 000024
5980
5981 ;STEP SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
5982 052530 012702 000002 MOV #2.,R2
5983 052534 410$: MOV #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0) ;LOAD RMMR1
      052534 012760 141501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5984 052542 012760 041501 000024 DEC R2
5985 052550 005302 BNE 410$
5986 052552 001370
5987
5988 ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
5989 ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
5990 052554 012702 000020 MOV #16.,R2 ;MAXIMUM NUMBER OF BIT CLOCKS
5991 052560 415$: MOV #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0) ;LOAD RMMR1
      052560 012760 045501 000024 MOV #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
5992 052566 012760 041501 000024
5993
5994 ;VERIFY EBL IS SET
5995 052574 016037 000024 001142 MOV RMMR1(R0),SBDDAT ;STORE RMMR1 AT SBDDAT
  
```

```

5996 052602 042737 157777 001142      BIC      #^CEBL,$BDDAT
5997 052610 001013                      BNE      420$
5998
5999 052612 005302                      DEC      R2
6000 052614 001361                      BNE      415$ ;CONTINUE BIT CLOCKS IF COUNT NOT 0
6001 052616 012737 020000 001140      MOV      #EBL,$GDDAT
6002 052624 010037 001136      MOV      R0,$BDADR
6003 052630 062737 000024 001136      ADD      #RMMR1,$BDADR
6004 052636 104271                      EMT      271
6005
6006 052640                      420$: ;END OF TEST
6007
6008

```

\*\*\*\*\*  
 :\*TEST 120 DATA COMMAND TESTS (3)  
 \*\*\*\*\*

```

052640
052640 000004
052642 000240
052644 012706 001100
052650 013700 001276
052654 013701 001466
052660 012737 000120 001226
TST120:
SCOPE ;SCOPE CALL
NOP
MOV #STACK,SP ;LOAD THE STACK POINTER
MOV $BASE,R0 ;R0 = UNIBUS ADDRESS
MOV TSTQUE,R1 ;R1 = POINTER TO DEVICE
MOV #120,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

```

```

6009
6010 ;VERIFY THE TAG BUS DURING DATA COMMAND
6011 ;FIRST PART USES OFFSET FORWARD
6012
6013

```

```

;LOAD TEST PARAMETERS IN REGISTER OUTPUT BUFFER
6014 052666 012737 001060 001446      MOV      #560.,RMDCO ;LAST CYLINDER
6015 052674 013737 001334 001420      MOV      LSTRK,RMDAO ;SET LAST TRACK AND
6016 052702 112737 000035 001420      MOV      #29.,RMDAO ;LAST SECTOR
6017 052710 012737 000200 001444      MOV      #OFD,RMOFO ;FORWARD OFFSET
6018 052716 012737 000071 001412      MOV      #RD!GO,RMCS10 ;READ DATA
6019 052724 012737 177400 001414      MOV      #-256.,RMWCO ;WORD COUNT
6020 052732 012737 104536 001416      MOV      #BUFFER,RMBAO ;BUFFER ADDRESS
6021

```

```

6022 ;EXECUTE COMMAND AND VERIFY TAG BUS USING SUBROUTINE
6023 052740 004737 052762      JSR      PC,10$
6024

```

```

;SECOND PART USES OFFSET REVERSE
6025 ;LOAD TEST PARAMETERS IN REGISTER OUTPUT BUFFER
6026 ;REVERSE OFFSET
6027 052744 112737 000000 001444      MOV      #0,RMOFO
6028

```

```

;EXECUTE COMMAND AND VERIFY TAG BUS USING SUBROUTINE
6029
6030 052752 004737 052762      JSR      PC,10$
6031 052756 000137 054046      JMP      300$
6032

```

\*\*\*\*\*  
 :SUBROUTINE USED DURING TEST  
 \*\*\*\*\*

```

6033
6034
6035
6036
6037 052762
6038 052762 004737 054732      JSR      PC,SETVV ;GO SET VOLUME VALID
        052766 000403      BR       20$ ;BRANCH TO 20$ IF NO ERROR
        052770 104000      EMT
6039 052772 000137 053664      JMP      160$
6040

```

```

6041 ;LOAD TRACK, SECTOR AND CYLINDER ADDRESS, LOAD OFFSET
6042 052776 20$:
6043 052776 013760 001420 000006 MOV RMDAO,RMDA(RO) ;LOAD RMDA
6044 053004 013760 001446 000034 MOV RMDCO,RMDC(RO) ;LOAD RMDC
6045 053012 013760 001444 000032 MOV RMOFO,RMOF(RO) ;LOAD RMOF
6046
6047 053020 004737 055054 JSR PC,SETOM ;GO SET OFFSET MODE
053024 000403 BR 30$ ;BRANCH TO 30$ IF NO ERROR
053026 104000 EMT
6048 053030 000137 053664 JMP 160$

;LOAD BUFFER ADDRESS AND WORD COUNT
6049
6050 30$:
6051 053034 MOV RMWCO,RMWC(RO) ;LOAD RMWC
6052 053034 013760 001414 000002 MOV RMBAO,RMBA(RO) ;LOAD RMBA
6053 053042 013760 001416 000004
6054
6055 ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
6056 053050 012760 041401 000024 MOV #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
6057 053056 012760 000000 000014 MOV #0,RMER1(RO) ;LOAD RMER1
6058 053064 012760 000000 000042 MOV #0,RMER2(RO) ;LOAD RMER2
6059 053072 013760 001412 000000 MOV RMCS10,RMCS1(RO) ;LOAD RMCS1
6060
6061 ;WAIT FOR RUN AND GO TO SET
6062 053100 012737 000310 001534 MOV #200,WATCH ;SET WATCHDOG TIMER VALUE
053106 004777 126424 JSR PC,@CLOCK ;START THE CLOCK
6063 40$:
053112 MOV RMMR1(RO),SBDDAT ;STORE RMMR1 AT SBDDAT
6064 053120 016037 000024 001142 BIC #^CRG,SBDDAT
053126 001020 BNE 50$
6065 053130 005737 001534 TST WATCH
6066 053134 001366 BNE 40$
6067 053136 004777 126376 JSR PC,@STOPCL ;STOP THE CLOCK
6068 053142 012737 040000 001140 MOV #RG,$GDDAT
6069 053150 010037 001136 MOV RO,$BDADR
6070 053154 062737 000024 001136 ADD #RMMR1,$BDADR
6071 053162 104275 EMT
6072 053164 000137 053664 JMP 160$
6073 50$:
6074 053170 JSR PC,@STOPCL ;STOP THE CLOCK
053174 004777 126344 MOV #200$,R4 ;R4 = TABLE POINTER
6075 053174 012704 053666
6076
6077 ;STEP SEQUENCER TO HEAD SEQUENCE AT LOCATION 156 (17 CLOCKS)
6078 053200 012705 000021 MOV #17.,R5 ;R5 = CLOCK COUNT
6079 60$:
053204 016037 000040 001142 MOV RMMR2(RO),SBDDAT ;STORE RMMR2 AT SBDDAT
6080
6081 ;:*****
6082 053212 013737 001142 001174 MOV SBDDAT,$TMP0 ;IF CC AND CH ARE SET AT THE SAME TIME
6083 ;:THE ECO TO THE CS BOARD IS IMPLEMENTED.
6084
6085 053220 042737 171777 001174 BIC #^C<CC!CH>,$TMP0 ;SAVE CC AND CH BITS
6086 053226 022737 006000 001174 CMP #CC!CH,$TMP0 ;ARE CC AND CH SET ?
6087 053234 001414 BEQ 65$ ;YES, BRANCH TO NEW LOCATION
6088 ;:*****
6089
6090 053236 042737 150000 001142 BIC #RQA!RQB!TST,$BDDAT
6091 053244 012437 001140 MOV (R4)+,$GDDAT
  
```

```

6092 053250 053737 001446 001140      BIS      RMDCO,$GDDAT      ;OR CYLINDER ADDRESS
6093 053256 023737 001140 001142      CMP      $GDDAT,$BDDAT
6094 053264 001011                      BNE      70$             ;BRANCH IF TAG BUS WRONG
6095 053266                      65$:
6096 053266 012760 141401 000024      MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0) ;LOAD RMMR1
6097 053274 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
6098 053302 005305                      DEC      R5
6099 053304 001337                      BNE      60$
6100 053306 000407                      BR       80$
6101 053310 010037 001136 001136      70$:  MOV      R0,$BDADR
6102 053314 052737 000040 001136      ADD      #RMMR2,$BDADR
6103 053322 104276                      EMT      276
6104 053324 000557                      BR       160$
6105
6106                      ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST AT
6107                      ;LOCATION 166
6108 053326                      80$:
6109 053326 012760 041001 000024      MOV      #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
6110 053334 012760 041401 000024      MOV      #DMD!MUR!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
6111
6112                      ;STEP SEQUENCER TO END OF OFFSET AT LOCATION 174 (37 CLOCKS)
6113 053342 012705 000045                      MOV      #37.,R5 ;RELOAD CLOCK COUNT
6114 053346 016037 000040 001142      90$:  MOV      RMMR2(R0),$BDDAT ;STORE RMMR2 AT $BDDAT
6115
6116                      ;;*****
6117 053354 013737 001142 001174      MOV      $BDDAT,$TMPO ;IF CC AND CH ARE SET AT SAME TIME
6118                      ;THE ECO TO THE CS BOARD IS IMPLEMENTED.
6119
6120 053362 042737 171777 001174      BIC      #^C<<CC!CH>,$TMPO ;SAVE CC AND CH BITS
6121 053370 022737 006000 001174      CMP      #CC!CH,$TMPO ;ARE CC AND CH SET ?
6122 053376 001003                      BNE      92$ ;NO !!
6123 053400 162704 000002                      SUB      #2,R4 ;ADJUST THE TABLE ADDRESS
6124 053404 000441                      BR       115$ ;TO OTHER LOCATION
6125 053406                      92$:
6126                      ;;*****
6127
6128 053406 042737 150000 001142      BIC      #RQA!RQB!TST,$BDDAT
6129 053414 011437 001140                      MOV      (R4),$GDDAT
6130 053420 032714 002000                      BIT      #CH,(R4)
6131 053424 001425                      BEQ      110$ ;BRANCH IF CONTROL/HEADER NOT ON
6132 053426 032714 004000                      BIT      #CC,(R4)
6133 053432 001416                      BEQ      100$ ;BRANCH IF HEADER TAG
6134
6135                      ;CONTROL TAG SHOULD BE ON-SETUP EXPECTED OFFSET
6136 053434 052737 000010 001140      BIS      #BB03,$GDDAT ;ASSUME OFD IS NOT SET
6137 053442 032737 000200 001444      BIT      #OFD,RMOFO
6138 053450 001406                      BEQ      95$
6139 053452 042737 000010 001140      BIC      #BB03,$GDDAT ;RESET BUS BIT 3 - DIRECTION IS REV
6140 053460 052737 000004 001140      BIS      #BB02,$GDDAT
6141 053466 000404                      BR       110$
6142
6143                      ;HEADER TAG SHOULD BE ON - SETUP EXPECTED TRACK ADDRESS
6144 053470                      100$:
6145 053470 113703 001421                      MOV      RMDAO+1,R3 ;GET TRACK
6146 053474 050337 001140                      BIS      R3,$GDDAT
6147

```

```

6148                                     ;COMPARE EXPECTED AND RECEIVED TAG BUS DATA
6149 053500                                     110$:
6150 053500 023737 001140 001142          CMP      $GDDAT,$BDDAT
6151 053506 001013                                     BNE      120$
6152 053510                                     115$:
6153 053510 012760 141401 000024          MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
6154 053516 012760 141401 000024          MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
6155 053524 062704 000002          ADD      #2,R4 ;MOVE TABLE POINTER
6156 053530 005305          DEC      R5 ;DECREMENT CLOCK COUNT
6157 053532 001305          BNE      90$
6158 053534 000407          BR       130$
6159 053536 010037 001136 001136 120$: MOV      RO,$BDDADR
6160 053542 062737 000040          ADD      #RMMR2,$BDDADR
6161 053550 104276          EMT      276
6162 053552 000444          BR       160$
6163
6164                                     ;DROP ON CYLINDER TO RESET LATCH, RAISE ON CYLINDER TO PASS TEST AT
6165                                     ;SEQUENCER LOCATION 175
6166 053554                                     130$:
6167 053554 012760 041001 000024          MOV      #DMD!MUR!DBEN,RMMR1(RO) ;LOAD RMMR1
6168 053562 012760 041401 000024          MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
6169
6170                                     ;STEP SEQUENCER TO SECTOR WAIT LOOP (8 CLOCKS)
6171 053570 012705 000010          MOV      #8.,R5
6172 053574 016037 000040 001142 140$: MOV      RMMR2(RO),$BDDAT ;STORE RMMR2 AT $BDDAT
6173
6174                                     ;:*****
6175                                     ; BIC      #RQA!RQB!TST,$BDDAT
6176                                     ;
6177                                     ;THE FOLLOWING CODE WAS ADDED
6178                                     ;TO ALLOW THE PROGRAM TO RUN WITH
6179                                     ;OR WITHOUT ECO TO THE CS BOARD.
6180 053602 042737 171777 001142          BIC      #^C<CC!CH>,$BDDAT ;SAVE CC AND CH BITS
6181 053610 012737 006000 001140          MOV      #CC!CH,$GDDAT ;GET EXPECTED DATA
6182                                     ;:*****
6183
6184 053616 023737 001140 001142          CMP      $GDDAT,$BDDAT ;GOOD DATA SAME AS LAST CMP
6185 053624 001011          BNE      150$ ;BRANCH IF ERROR
6186 053626 012760 141401 000024          MOV      #DMD!MUR!MOC!DBEN!DBCK,RMMR1(RO) ;LOAD RMMR1
6187 053634 012760 041401 000024          MOV      #DMD!MUR!MOC!DBEN,RMMR1(RO) ;LOAD RMMR1
6188 053642 005305          DEC      R5
6189 053644 001353          BNE      140$
6190 053646 000406          BR       160$
6191 053650 010037 001136 001136 150$: MOV      RO,$BDDADR
6192 053654 062737 000040          ADD      #RMMR2,$BDDADR
6193 053662 104276          EMT      276
6194
6195 053664 000207          160$: RTS      PC
6196
6197                                     ;TABLE OF TAG BUS CONTROL AND DATA VALUES
6198 053666          200$:
6199 053666 001777          .WORD    1777 ;LOCATION 0
6200 053670 001777          .WORD    1777 ;LOCATION 25
6201 053672 001777          .WORD    1777 ;LOCATION 26
6202 053674 001777          .WORD    1777 ;LOCATION 128
6203 053676 001777          .WORD    1777 ;LOCATION 129

```

6204	053700	001777	.WORD	1777	:LOCATION	130
6205	053702	004000	.WORD	CC	:LOCATION	144
6206	053704	004000	.WORD	CC	:LOCATION	145
6207	053706	024000	.WORD	CC!TAG	:LOCATION	146
6208	053710	024000	.WORD	CC!TAG	:LOCATION	147
6209	053712	024000	.WORD	CC!TAG	:LOCATION	148
6210	053714	024000	.WORD	CC!TAG	:LOCATION	149
6211	053716	024000	.WORD	CC!TAG	:LOCATION	150
6212	053720	024000	.WORD	CC!TAG	:LOCATION	151
6213	053722	004000	.WORD	CC	:LOCATION	152
6214	053724	004000	.WORD	CC	:LOCATION	153
6215	053726	0C1777	.WORD	1777	:LOCATION	154
6216	053730	002000	.WORD	CH	:LOCATION	156
6217	053732	002000	.WORD	CH	:LOCATION	157
6218	053734	022000	.WORD	CH!TAG	:LOCATION	158
6219	053736	022000	.WORD	CH!TAG	:LOCATION	159
6220	053740	022000	.WORD	CH!TAG	:LOCATION	160
6221	053742	022000	.WORD	CH!TAG	:LOCATION	161
6222	053744	022000	.WORD	CH!TAG	:LOCATION	162
6223	053746	022000	.WORD	CH!TAG	:LOCATION	163
6224	053750	002000	.WORD	CH	:LOCATION	164
6225	053752	002000	.WORD	CH	:LOCATION	165
6226	053754	001777	.WORD	1777	:LOCATION	232
6227	053756	001777	.WORD	1777	:LOCATION	233
6228	053760	001777	.WORD	1777	:LOCATION	234
6229	053762	001777	.WORD	1777	:LOCATION	235
6230	053764	001777	.WORD	1777	:LOCATION	236
6231	053766	001777	.WORD	1777	:LOCATION	237
6232	053770	001777	.WORD	1777	:LOCATION	238
6233	053772	001777	.WORD	1777	:LOCATION	239
6234	053774	001777	.WORD	1777	:LOCATION	240
6235	053776	001777	.WORD	1777	:LOCATION	241
6236	054000	001777	.WORD	1777	:LOCATION	242
6237	054002	001777	.WORD	1777	:LOCATION	243
6238	054004	001777	.WORD	1777	:LOCATION	244
6239	054006	001777	.WORD	1777	:LOCATION	245
6240	054010	001777	.WORD	1777	:LOCATION	246
6241	054012	001777	.WORD	1777	:LOCATION	247
6242	054014	001777	.WORD	1777	:LOCATION	248
6243	054016	001777	.WORD	1777	:LOCATION	249
6244	054020	001777	.WORD	1777	:LOCATION	250
6245	054022	001777	.WORD	1777	:LOCATION	251
6246	054024	001777	.WORD	1777	:LOCATION	252
6247	054026	001777	.WORD	1777	:LOCATION	166
6248	054030	006000	.WORD	CC!CH	:LOCATION	169
6249	054032	006000	.WORD	CC!CH	:LOCATION	170
6250	054034	026000	.WORD	CC!CH!TAG	:LOCATION	171
6251	054036	026000	.WORD	CC!CH!TAG	:LOCATION	172
6252	054040	026000	.WORD	CC!CH!TAG	:LOCATION	173
6253	054042	026000	.WORD	CC!CH!TAG	:LOCATION	174
6254	054044	026000	.WORD	CC!CH!TAG	:LOCATION	175,ETC
6255						
6256	054046					

300\$:

:END OF TEST

```

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 054046 000004          $EOSP:  SCOPE
10 054050 000240          NOP
11 054052 013700 001466  MOV      TSTQUE,RO          ;GET POINTER TO TSTQUE
12 054056 062700 000002  ADD      #2,RO          ;ADJUST POINTER TO NEXT DEVICE
13 054062 010037 001466  MOV      RO,TSTQUE      ;SAVE POINTER TO TSTQUE
14 054066 005710          TST      (RO)          ;ANY MORE DEVICES FOR TEST ?
15 054070 001402          BEQ      1$          ;BR IF NO
16 054072 000137 007006  JMP      READY          ;YES, JUMP TO 'READY' ROUTINE
17 054076 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

054104          $EOP:
054104 000240          NOP
054106 005037 001116  CLR      $TSTNM          ;;ZERO THE TEST NUMBER
054112 005037 001206  CLR      $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
054116 005237 001230  INC      $PASS          ;;INCREMENT THE PASS NUMBER
054122 042737 100000 001230  BIC      #100000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
054130 005327          DEC      (PC)+          ;;LOOP?
054132 000001          $EOPCT: .WORD 1
054134 003066          BGT      $DOAGN          ;;YES
054136 012737          MOV      (PC)+,a(PC)+  ;;RESTORE COUNTER
054140 000001          $ENDCT: .WORD 1
054142 054132          $EOPCT
054144 104401 054152  TYPE      ,65$          ;;TYPE ASCIZ STRING
054150 000407          BR      64$          ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <12><15>/END PASS #/
64$:
054170          MOV      $PASS,-(SP)          ;;SAVE $PASS FOR TYPEOUT
054170 013746 001230          ;;TYPE PASS NUMBER
054174 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
054176 005737 001126  TST      $ERTTL          ;;SEE IF ANY ERRORS THIS PASS
054202 001431          BEQ      $GT42P          ;;BR IF NO ERRORS TO REPORT
054204 104401 054212  TYPE      ,67$          ;;TYPE ASCIZ STRING
054210 000421          BR      66$          ;;GET OVER THE ASCIZ
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:
054254          MOV      $ERTTL,-(SP)          ;;SAVE $ERTTL FOR TYPEOUT
054254 013746 001126          ;;TOTAL NUMBER OF ERRORS
054260 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
054262 005037 001126  CLR      $ERTTL          ;;CLEAR ERROR TOTAL
    
```

054266	104401	001217	\$GT42P:	TYPE	,SRLF	::TYPE CARRIAGE RETURN, LINE FEED
054272	013700	000042	\$GET42:	MOV	@42,R0	::GET MONITOR ADDRESS
054276	001405			BEQ	\$DOAGN	::BRANCH IF 1. MONITOR
054300	000005			RESET		::CLEAR THE 1 .LD
054302	004710		\$ENDAD:	JSR	PC,(R0)	::GO TO MONITOR
054304	000240			NOP		::SAVE ROOM
054306	000240			NOP		::FOR
054310	000240			NOP		::ACT11
054312			\$DOAGN:			
054312	000137			JMP	@(PC)+	::RETURN
054314	007006		\$RTNAD:	.WORD	READY	
054316	377	377	\$ENULL:	.BYTE	-1,-1,0	::NULL CHARACTER STRING
		000		.EVEN		



```

1          .SBTTL  CLOCK SUBROUTINES
2
3          ;ROUTINE TO SIZE FOR CLOCKS (KW11-L OR KW11-P)
4
5 054322 000240          SIZCLK: NOP
6 054324 013746 000004  MOV      ERRVEC,-(SP)      ;;PUSH ERRVEC ON STACK
7 054330 013746 000006  MOV      ERRVEC+2,-(SP)    ;;PUSH ERRVEC+2 ON STACK
8 054334 012737 054420 000004  MOV      #10$,ERRVEC      ;LOAD 04 TRAP VECTORS
9 054342 012737 000300 000006  MOV      #PR6,ERRVEC+2
10
11          ;SEE IF A KW11-P CLOCK IS PRESENT - GO TO 10$ IF NOT PRESENT
12 054350 005777 125136          TST      @SLPCSR          ;TEST FOR P CLOCK
13 054354 012737 054562 001536  MOV      #PCLOCK,CLOCK    ;LOAD SUBROUTINE ADDRESS
14 054362 012737 054704 001540  MOV      #PSTOP,STOPCL    ;LOAD STOP ADDRESS
15 054370 012777 054650 125120  MOV      #PCOUNT,@SLPVEC  ;LOAD P CLOCK INTERRUPT VECTOR
16 054376 012777 000300 125114  MOV      #PR6,@SLPVEC+2
17 054404 013777 001526 125112  MOV      $LLVEC+2,$LLVEC  ;CLEAR L CLOCK INTERRUPT VECTOR
18 054412 005077 125110          CLR      @LLVEC+2
19 054416 000454          BR      30$
20 054420 012716 054426 10$:  MOV      #15$,(SP)      ;DUMMY RTI ADDRESS
21 054424 000002          RTI      ;RESTORE PRIORITY
22
23          ;NO P CLOCK-SEE IF L CLOCK IS PRESENT-GO TO 20$ IF NOT PRESET
24 054426          15$:
25 054426 012737 054504 000004  MOV      #20$,ERRVEC      ;CHANGE 04 TRAP VECTOR
26 054434 005777 125062          TST      @LLCSR          ;TEST FOR L CLOCK
27 054440 012737 054600 001536  MOV      #LCLOCK,CLOCK    ;LOAD SUBROUTINE ADDRESS
28 054446 012737 054712 001540  MOV      #LSTOP,STOPCL    ;LOAD STOP ADDRESS
29 054454 012777 054650 125042  MOV      #LCOUNT,@LLVEC   ;LOAD L CLOCK INTERRUPT VECTOR
30 054462 012777 000300 125036  MOV      #PR6,@LLVEC+2
31 054470 013777 001520 125020  MOV      $LPVEC+2,$LPVEC  ;CLEAR P CLOCK INTERRUPT VECTOR
32 054476 005077 125016          CLR      @LPVEC+2
33 054502 000422          BR      30$
34 054504 012716 054512 20$:  MOV      #25$,(SP)      ;DUMMY RTI ADDRESS
35 054510 000002          RTI      ;RESTORE PRIORITY
36
37          ;NO CLOCK AVAILABLE - AUGMENT RETURN ADDRESS
38 054512          25$:
39 054512 005037 001536          CLR      CLOCK          ;CLEAR SUBROUTINE ADDRESS
40 054516 012737 001520 001516  MOV      #$LPVEC+2,$LPVEC ;CLEAR P CLOCK INTERRUPT VECTOR
41 054524 005037 001520          CLR      $LPVEC+2
42 054530 012737 001526 001524  MOV      #$LLVEC+2,$LLVEC ;CLEAR L CLOCK INTERRUPT VECTOR
43 054536 005037 001526          CLR      $LLVEC+2
44 054542 062766 000002 000004  ADD      #2,4(SP)        ;CHANGE RETURN ADDRESS
45 054550          30$:
46 054550 012637 000006          MOV      (SP)+,ERRVEC+2  ;;POP STACK INTO ERRVEC+2
47 054554 012637 000004          MOV      (SP)+,ERRVEC   ;;POP STACK INTO ERRVEC
48 054560 000207          RTS      PC
49
50          ;ROUTINES TO START THE CLOCK (KW11-L OR KW11-P)
51 054562 012777 177777 124724 PCLOCK: MOV      #-1,@SLPCSB  ;LOAD COUNT SET BUFFER
52 054570 012777 000135 124714  MOV      #135,@SLPCSR    ;LOAD CONTROL REGISTER
53 054576 000403          BR      PLCLK          ;GO TO COMMON CODE
54
55 054600 012777 000100 124714 LCLOCK: MOV      #100,@$LLCSR ;LOAD CONTROL REGISTER
56

```

```

57 054606 005037 001532      PLCLK: CLR      TIME      ;CLEAR TIMER COUNT
58 054612 104400              TRAP                    ;:PUSH OLD PSW AND PC ON STACK
    054614 012605              MOV      (SP)+,R5      ;:SAVE THE PSW IN R5
59 054616 010537 001530              MOV      R5,$PSW      ;:SAVE PRIORITY
60 054622 042705 177437              BIC      #^CPR7,R5    ;:MASK X
61 054626 022705 000300              CMP      #PR6,R5     ;:IS PRIORITY TOO HIGH??
62 054632 101005              BHI      40$          ;:NO!!
63 054634 012746 000240              MOV      #PR5,-(SP)   ;:PUT NEW PS ON STACK
    054640 012746 054646              MOV      #30$,-(SP)  ;:PUT NEW PC ON STACK
    054644 000002              RTI                    ;:POP NEW PC AND PS
    054646 000207              30$:
    054646 000207              40$: RTS      PC

64 054646 000207
65
66
67
68 054650
69 054650 062737 000021 001532      PCOUNT:
70 054656 103003              LCOUNT: ADD      #17.,TIME ;ADD 17MS TO ELAPSED TIME
71 054660 012737 177777 001532              BCC      10$          ;:BRANCH IF NO OVERFLOW
72 054666 162737 000021 001534              MOV      #-1.,TIME   ;:RESTORE MAXIMUM COUNT
73 054674 100002              10$: SUB      #17.,WATCH ;DECREMENT REMAINING TIME
74 054676 005037 001534              BPL      20$          ;:BRANCH IF POSITIVE
75 054702              CLR      WATCH      ;CLEAR REMAINING TIME
76
77
78
79 054702 000002              20$:
    054702 000002              IRP:   RTI                    ;RETURN TO USER
80
81
82
83 054704 005077 124602              ;ROUTINES TO STOP THE CLOCK (KW11-L OR KW11-P)
84 054710 000402              PSTOP: CLR      @SLPCSR ;STOP P CLOCK
    054710 000402              BR      PLSTP          ;GO TO COMMON STOP CODE
85
86 054712 005077 124604              LSTOP: CLR      @LLCSR ;STOP L CLOCK
87
88 054716
89 054716 013746 001530              PLSTP: MOV      $PSW,-(SP) ;:PUT NEW PS ON STACK
    054722 012746 054730              MOV      #10$,-(SP)  ;:PUT NEW PC ON STACK
    054726 000002              RTI                    ;:POP NEW PC AND PS
    054730
90 054730 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(R0)    ;LOAD RMMR1
MOV    #DMD!MUR,RMMR1(R0) ;LOAD RMMR1
MOV    #0,RMER1(R0)      ;LOAD RMER1
MOV    #0,RMER2(R0)      ;LOAD RMER2
MOV    #PACACK!GO,RMCS1(R0) ;LOAD RMCS1
MOV    RMDS(R0),SBDDAT   ;STORE RMDS AT SBDDAT
BIC    #^CVV,SBDDAT
BNE    10$               ;BRANCH IF VOLUME VALID SET
MOV    R0,SBDADR         ;SETUP FOR ERROR MSG
ADD    #RMDS,SBDADR
MOV    #VV,$GDDAT
ADD    #2,(SP)           ;MOVE RETURN ADDRESS TO ERROR
MOVB   #170,a(SP)       ;WRITE ERROR NUMBER
MOV    #PACACK,$TMPO
10$:   RTS               ;RETURN
PC
```

```
054732 004737 055156
054732 012760 000001 000024
054744 012760 001001 000024
054752 012760 000000 000014
054760 012760 000000 000042
054766 012760 000023 000000
054774 016037 000012 001142
055002 042737 177677 001142
055010 001020
055012 010037 001136
055016 062737 000012 001136
055024 012737 000100 001140
055032 062716 000002
055036 112776 000170 000000
055044 012737 000022 001174
055052 000207
```

```

1      .SBTTL SET OFFSET MODE SUBROUTINE
2
3      ;THIS SUBROUTINE EXECUTES AN OFFSET COMMAND AND VERIFIES THAT OFFSET
4      ;MODE SETS. THE DRIVE SHOULD BE IN DIAGNOSTIC MODE WHEN CALLING THE
5      ;SUBROUTINE, WHICH WILL LEAVE DMD ON. THE SUBROUTINE RETURNS TO THE
6      ;WORD FOLLOWING THE CALL UNLESS THERE IS AN ERROR, IN WHICH CASE IT
7      ;RETURNS TO THE SECOND WORD FOLLOWING THE CALL
8
9      ;CALL: JSR      PC,SETOM      JUMP TO SUBROUTINE
10     BR      ??                  RETURN HERE IF NO ERROR
11     ERROR   RETURN HERE IF ERROR
12
13     SETOM:
14     055054 012760 001001 000024  MOV      #DMD!MUR,RMMR1(RO)      ;LOAD RMMR1
15     055062 012760 000000 000014  MOV      #0,RMER1(RO)           ;LOAD RMER1
16     055070 012760 000000 000042  MOV      #0,RMER2(RO)           ;LOAD RMER2
17     055076 012760 000015 000000  MOV      #OFFSET!GO, RMCS1(RO)   ;LOAD RMCS1
18     055104 016037 000012 001142  MOV      RMD5(RO), $BDDAT       ;STORE RMD5 AT $BDDAT
19     055112 042737 177776 001142  BIC      #^COM,$BDDAT
20     055120 001015                BNE      10$                    ;BRANCH IF OFFSET ON
21     055122 012737 000001 001140  MOV      #OM,$GDDAT
22     055130 010037 001136                MOV      RO,$BDADR
23     055134 062737 000012 001136  ADD      #RMD5,$BDADR
24     055142 062716 000002                ADD      #2,(SP)                ;MOVE RETURN ADDRESS TO ERROR
25     055146 112776 000200 000000  MOV      #200,@(SP)            ;WRITE ERROR NUMBER
26     055154                10$:
27     055154 000207                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          055156 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
10         055160 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
11         055162 013700 001276    MOV      $BASE,R0      ;R0 = UNIBUS BASE ADDRESS
12         055166 012760 000040 000010 MOV      #CLR,RMCS2(R0) ;CLEAR MASSBUS
13         055174 013701 001466    MOV      TSTQUE,R1      ;GET DEVICE UNDER TEST
14         055200 111160 000010    MOV      (R1),RMCS2(R0) ;SELECT DEVICE
15         055204 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
16         055206 012600          MOV      (SP)+,R0      ;;POP STACK INTO R0
17         055210 000207          RTS      PC          ;RETURN
```

1

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

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

```

055212
055212 010046
055214 010146
055216 010246
055220 010346
055222 010446
055224 010546
055226 016646 000022
055232 016646 000022
055236 016646 000022
055242 016646 000022
055246 000002
    
```

```

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

```

*RESTORE R0-R5
*CALL:
*   RESREG
    
```

```

055250
055250 012666 000022
055254 012666 000022
055260 012666 000022
055264 012666 000022
055270 012605
055272 012604
055274 012603
055276 012602
055300 012601
055302 012600
055304 000002
    
```

```

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

.SBTTL BINARY TO ASCII AND TYPE ROUTINE

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

```

*      MOV      NUMBER,-(SP)      ;;NUMBER TO BE TYPED
*      TYPBN
*
055306 010146          $TYPBN: MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
055310 016601 000006  MOV      6(SP),R1      ;;GET THE INPUT NUMBER
055314 000261          SEC                          ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
055316 112737 000060 055360 1$:  MOVB     #'0,$BIN      ;;SET CHARACTER TO AN ASCII '0'.
055324 006101          ROL      R1                ;;GET THIS BIT
055326 001406          BEQ      2$                ;;DONE?
055330 105537 055360  ADCB     $BIN          ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
055334 104401 055360  TYPE     ,$BIN          ;;GO TYPE THIS BIT
055340 000241          CLC                          ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
055342 000765          BR       1$                ;;GO DO THE NEXT BIT
055344 012601          2$:  MOV      (SP)+,R1      ;;POP THE STACK INTO R1
055346 016666 000002 000004  MOV      2(SP),4(SP)  ;;ADJUST THE STACK
055354 012616          MOV      (SP)+,(SP)
055356 000002          RTI                          ;;RETURN TO USER
055360      000      000  $BIN:  .BYTE  0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
  
```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

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

055362				\$TYPDS:	MOV R0,-(SP)	::PUSH R0 ON STACK
055362	010046				MOV R1,-(SP)	::PUSH R1 ON STACK
055364	010146				MOV R2,-(SP)	::PUSH R2 ON STACK
055366	010246				MOV R3,-(SP)	::PUSH R3 ON STACK
055370	010346				MOV R5,-(SP)	::PUSH R5 ON STACK
055372	010546				MOV #20200,-(SP)	::SET BLANK SWITCH AND SIGN
055374	012746	020200			MOV 20(SP),R5	::GET THE INPUT NUMBER
055400	016605	000020			BPL 1\$	::BR IF INPUT IS POS.
055404	100004				NEG R5	::MAKE THE BINARY NUMBER POS.
055406	005405				MOV #'-,1(SP)	::MAKE THE ASCII NUMBER NEG.
055410	112766	000055	000001	1\$:	CLR R0	::ZERO THE CONSTANTS INDEX
055416	005000				MOV #SDBLK,R3	::SETUP THE OUTPUT POINTER
055420	012703	055576			MOVB #' ,(R3)+	::SET THE FIRST CHARACTER TO A BLANK
055424	112723	000040		2\$:	CLR R2	::CLEAR THE BCD NUMBER
055430	005002				MOV \$DTBL(R0),R1	::GET THE CONSTANT
055432	016001	055566		3\$:	SUB R1,R5	::FORM THIS BCD DIGIT
055436	160105				BLT 4\$	::BR IF DONE
055440	002402				INC R2	::INCREASE THE BCD DIGIT BY 1
055442	005202				BR 3\$	
055444	000774			4\$:	ADD R1,R5	::ADD BACK THE CONSTANT
055446	060105				TST R2	::CHECK IF BCD DIGIT=0
055450	005702				BNE 5\$	::FALL THROUGH IF 0
055452	001002				TSTB (SP)	::STILL DOING LEADING 0'S?
055454	105716				BMI 7\$	::BR IF YES
055456	100 07			5\$:	ASLB (SP)	::MSD?
055460	106316				BCC 6\$	::BR IF NO
055462	103003				MOVB 1(SP),-1(R3)	::YES--SET THE SIGN
055464	116663	000001	177777	6\$:	BIS #'0,R2	::MAKE THE BCD DIGIT ASCII
055472	052702	000060		7\$:	BIS #' ,R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
055476	052702	000040			MOVB R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
055502	110223				TST (R0)+	::JUST INCREMENTING
055504	005720				CMP R0,#10	::CHECK THE TABLE INDEX
055506	020027	000010			BLT 2\$	::GO DO THE NEXT DIGIT
055512	002746				BGT 8\$	::GO TO EXIT
055514	003002				MOV R5,R2	::GET THE LSD
055516	010502				BR 6\$	::GO CHANGE TO ASCII
055520	000764			8\$:	TSTB (SP)+	::WAS THE LSD THE FIRST NON-ZERO?
055522	105726				BPL 9\$	::BR IF NO
055524	100003				MOVB -1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
055526	116663	177777	177776	9\$:	CLRB (R3)	::SET THE TERMINATOR
055534	105013				MOV (SP)+,R5	::POP STACK INTO R5
055536	012605				MOV (SP)+,R3	::POP STACK INTO R3
055540	012603				MOV (SP)+,R2	::POP STACK INTO R2
055542	012602				MOV (SP)+,R1	::POP STACK INTO R1
055544	012601					



055546	012600		MOV	(SP)+,R0	::POP STACK INTO R0
055550	104401	055576	TYPE	\$DBLK	::NOW TYPE THE NUMBER
055554	016666	000002 000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
055562	012616		MOV	(SP)+,(SP)	
055564	000002		RTI		::RETURN TO USER
055566	023420	\$DTBL:	10000.		
055570	001750		1000.		
055572	000144		100.		
055574	000012		10.		
055576		\$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

```

```

055406 017646 000000          $TYPOS: MOV     @ (SP),-(SP)      ;;PICKUP THE MODE
055612 116637 000001 056031  MOVB   i (SP), $OFILL      ;;LOAD ZERO FILL SWITCH
055620 112637 056023          MOVB   (SP)+, $OMODE+1    ;;NUMBER OF DIGITS TO TYPE
055624 062716 000002          ADD     #2, (SP)        ;;ADJUST RETURN ADDRESS
055630 000406          BR     $TYPON
055632 112737 000001 056031  $TYPOC: MOVB   #1, $OFILL      ;;SET THE ZERO FILL SWITCH
055640 112737 000006 056033  MOVB   #6, $OMODE+1    ;;SET FOR SIX(6) DIGITS
055646 112737 000005 056030  $TYPON: MOVB   #5, $OCNT      ;;SET THE ITERATION COUNT
055654 010346          MOV     R3, -(SP)      ;;SAVE R3
055656 010446          MOV     R4, -(SP)      ;;SAVE R4
055660 010546          MOV     R5, -(SP)      ;;SAVE R5
055662 113704 056033          MOVB   $OMODE+1, R4    ;;GET THE NUMBER OF DIGITS TO TYPE
055666 005404          NEG     R4
055670 062704 000006          ADD     #6, R4        ;;SUBTRACT IT FOR MAX. ALLOWED
055674 110437 056032          MOVB   R4, $OMODE     ;;SAVE IT FOR USE
055700 113704 056031          MOVB   $OFILL, R4     ;;GET THE ZERO FILL SWITCH
055704 016605 000012          MOV     12(SP), R5    ;;PICKUP THE INPUT NUMBER
055710 005003          CLR     R3           ;;CLEAR THE OUTPUT WORD
055712 006105          1$:   ROL     R5      ;;ROTATE MSB INTO 'C'
055714 000404          BR     3$           ;;GO DO MSB
055716 006105          2$:   ROL     R5      ;;FORM THIS DIGIT
055720 006105          ROL     R5
055722 006105          ROL     R5
055724 010503          MOV     R5, R3
055726 006103          3$:   ROL     R3      ;;GET LSB OF THIS DIGIT
055730 105337 056032          DECB   $OMODE        ;;TYPE THIS DIGIT?
055734 100016          BPL    7$           ;;BR IF NO
055736 042703 177770          BIC    #177770, R3   ;;GET RID OF JUNK
055742 001002          BNE    4$           ;;TEST FOR 0
055744 005704          TST    R4           ;;SUPPRESS THIS 0?
055746 001403          BEQ    5$           ;;BR IF YES
055750 005204          4$:   INC     R4      ;;DON'T SUPPRESS ANYMORE 0'S

```

055752	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
055756	052703	000040	5\$:	BIS	#' R3	::MAKE ASCII IF NOT ALREADY
055762	110337	056026		MOVB	R3,8\$	::SAVE FOR TYPING
055766	104401	056026		TYPE	8\$	::GO TYPE THIS DIGIT
055772	105337	056030	7\$:	DECB	\$OCNT	::COUNT BY 1
055776	003347			BGT	2\$	::BR IF MORE TO DO
056000	002402			BLT	6\$	::BR IF DONE
056002	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
056004	000744			BR	2\$	::GO DO THE LAST DIGIT
056006	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
056010	012604			MOV	(SP)+,R4	::RESTORE R4
056012	012603			MOV	(SP)+,R3	::RESTORE R3
056014	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
056022	012616			MOV	(SP)+,(SP)	
056024	000002			RTI		::RETURN
056026	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
056027	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
056030	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
056031	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
056032	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

```

```

056034 105737 001173 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
056040 100002          BPL 1$ ;; BR IF YES
056042 000000          HALT ;; HALT HERE IF NO TERMINAL
056044 000430          BR 3$ ;; LEAVE
056046 010046          1$: MOV RO,-(SP) ;; SAVE RO
056050 017600 000002 MOV @2(SP),RO ;; GET ADDRESS OF ASCIZ STRING
056054 122737 000001 001242 CMPB #APTENV,$ENV ;; RUNNING IN APT MODE
056062 001011          BNE 62$ ;; NO, GO CHECK FOR APT CONSOLE
056064 132737 000100 001243 BITB #APTSPool,$ENVM ;; SPOOL MESSAGE TO APT
056072 001405          BEQ 62$ ;; NO, GO CHECK FOR CONSOLE
056074 010037 056104 MOV RO,61$ ;; SETUP MESSAGE ADDRESS FOR APT
056100 004737 062606 JSR PC,$ATY3 ;; SPOOL MESSAGE TO APT
056104 000000          61$: .WORD 0 ;; MESSAGE ADDRESS
056106 132737 000040 001243 62$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
056114 001003          BNE 60$ ;; YES, SKIP TYPE OUT
056116 112046          2$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
056120 001005          BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
056122 005726          TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
056124 012600          60$: MOV (SP)+,RO ;; RESTORE RO
056126 062716 000002 3$: ADD #2,(SP) ;; ADJUST RETURN PC
056132 000002          RTI ;; RETURN
056134 122716 000011 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
056140 001430          BEQ 8$
056142 122716 000200 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
056146 001006          BNE 5$
056150 005726          TST (SP)+ ;; POP <CR><LF> EQUIV
056152 104401          TYPE ;; TYPE A CR AND LF
056154 001217          $CRLF
056156 105037 056364 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
056162 000755          BR 2$ ;; GET NEXT CHARACTER
056164 004737 056246 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
056170 123726 001172 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
056174 001350          BNE 2$ ;; IF NO GO GET NEXT CHAR.
056176 013746 001170 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
;; AND THE NULL CHAR.
056202 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
056206 002770          BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
056210 004737 056246 JSR PC,$TYPEC ;; GO TYPE A NULL
056214 105337 056364 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
056220 000770          BR 7$ ;; LOOP

```

:HORIZONTAL TAB PROCESSOR

```

056222 112716 000040      8$:  MOVB  #' (SP)      ;;REPLACE TAB WITH SPACE
056226 004737 056246      9$:  JSR  PC,$TYPEC      ;;TYPE A SPACE
056232 132737 000007 056364 BITB  #7,$CHARCNT      ;;BRANCH IF NOT AT
056240 001372              BNE  9$                ;;TAB STOP
056242 005726              TST  (SP)+             ;;POP SPACE OFF STACK
056244 000724              BR   2$                ;;GET NEXT CHARACTER
056246                                $TYPEC:
056246 105777 122706      TSTB  @STKS            ;;CHAR IN KYBD BUFFER?
056252 100022              BPL  10$              ;;BR IF NOT
056254 017746 122702      MOV   @STKB,-(SP)      ;;GET CHAR
056260 042716 177600      BIC  #177600,(SP)    ;;STRIP EXTRANEIOUS BITS
056264 122716 000023      CMPB #SXOFF,(SP)    ;;WAS CHAR XOFF
056270 001012              BNE  102$            ;;BR IF NOT
056272                                101$:
056272 105777 122662      TSTB  @STKS            ;;WAIT FOR CHAR
056276 100375              BPL  101$            ;;BR IF NOT
056300 117716 122656      MOVB  @STKB,(SP)     ;;GET CHAR
056304 042716 177600      BIC  #177600,(SP)    ;;STRIP IT
056310 122716 000021      CMPB #SXON,(SP)     ;;WAS IT XON?
056314 001366              BNE  101$            ;;BR IF NOT
056316                                102$:
056316 005726              TST  (SP)+             ;;FIX STACK
056320                                10$:
056320 105777 122640      TSTB  @STPS            ;;WAIT UNTIL PRINTER IS READY
056324 100375              BPL  10$                ;;BR IF NOT
056326 116677 000002 122632 MOVB  2(SP),@STPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
056334 122766 000015 000002 CMPB  #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
056342 001003              BNE  1$                ;;BRANCH IF NO
056344 105037 056364      CLRB  $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
056350 000406              BR   $TYPEX          ;;EXIT
056352 122766 000012 000002 1$:  CMPB  #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
056360 001402              BEQ  $TYPEX          ;;BRANCH IF YES
056362 105227              INCB (PC)+           ;;COUNT THE CHARACTER
056364 000000      $CHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE
056366 000207      $TYPEX: RTS         PC
  
```

.SBTTL SCOPE HANDLER ROUTINE

```

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

```

056370          $SCOPE:
056370 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
056372 032777 040000 122554 1$: BIT #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
056400 001402          BEQ 9$          ;;NO IF SW14=0
056402 000137 057032          JMP $OVER          ;;JUMP OVER SCOPE ROUTINE
056406          9$:
:#####START OF CODE FOR THE XOR TESTER#####
056406 000416 $XTSTR: BR 6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
:THIS INSTRUCTION TO A 'NOP' (NOP=240)
056410 013746 000004          MOV @ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
056414 012737 056434 000004          MOV #5,@ERRVEC          ;;SET FOR TIMEOUT
056422 005737 177060          TST @177060          ;;TIME OUT ON XOR?
056426 012637 000004          MOV (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
056432 000561          BR $SVLAD          ;;GO TO THE NEXT TEST
056434 022626          5$: CMP (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
056436 012637 000004          MOV (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
056442 000521          BR 7$          ;;LOOP ON THE PRESENT TEST
056444          6$:#####END OF CODE FOR THE XOR TESTER#####
056444 032777 000400 122502          BIT #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
056452 001421          BEQ 2$          ;;BR IF NO
056454 005046          CLR -(SP)          ;;CLEAR A TEMP. LOCATION
056456 117716 122472          MOVB @SWR,(SP)          ;;PICKUP THE DESIRED TEST NUMBER
056462 001414          BEQ 8$          ;;BRANCH IF BAD TEST NUMBER IN SWR
056464 022716 000120          CMP #120,(SP)          ;;CHECK THE NUMBER IN THE SWR
056470 002411          BLT 8$          ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
056472 011637 001116          MOV (SP),$STNM          ;;UPDATE THE TEST NUMBER
056476 005316          DEC (SP)          ;;BACKUP BY ONE
056500 006316          ASL (SP)          ;;SCALE THE TEST NUMBER AS AN INDEX
056502 062716 057050          ADD #$$W08TBL,(SP)          ;;FORM THE ADDRESS OF TEST POINTER
056506 013637 001122          MOV @ (SP)+,$LPADR          ;;SET LOOP ADDRESS TO DESIRED TEST
056512 000547          BR $OVER          ;;GO LOOP ON THE TEST
056514 005726          8$: TST (SP)+          ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
056516 105737 001117          2$: TSTB $ERFLG          ;;HAS AN ERROR OCCURRED?
056522 001502          BEQ 3$          ;;BR IF NO
056524 022737 177777 057700          CMP #-1,CPSAVE          ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
056532 001455          BEQ 2003$          ;;KICK AROUND ROUTINE IF SO
056534 013746 000004          MOV ERRVEC,-(SP)          ;;SAVE CONTENTS OF ERROR VECTOR
056540 012737 056556 000004          MOV #2000$,ERRVEC          ;;SETUP 'TRAP' RETURN ADDRESS
056546 013737 177766 057700          MOV 177766,CPSAVE          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
056554 000406          BR 2001$
056556 012737 177777 057700 2000$: MOV #-1,CPSAVE          ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
056564 012716 056572          MOV #2001$,(SP)          ;;SETUP RETURN ADDRESS
056570 000002          RTI
    
```

```

056572 012637 000004          2001$: MOV    (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR
056576 022737 177777 057700 2002$: CMP    #-1,CPSAVE    ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
056604 001430                BEQ    2003$          ;;BRANCH IF SO
056606 032737 000001 057700    BIT    #BIT00,CPSAVE    ;;SEE IF THE POWER MONITOR BIT IS ON
056614 001424                BEQ    2003$          ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
056616 042737 000001 177766    BIC    #BIT00,177766    ;;CLEAR THE BIT FOUND TO BE SET
056624 013746 001154          MOV    SWR,-(SP)        ;;SAVE SWR ADDRESS
056630 017646 000000          MOV    @ (SP),-(SP)     ;;SAVE SWR VALUE
056634 012737 000176 001154    MOV    #176,SWR        ;;GET SOFTWARE SWR ADDRESS
056642 011677 122306          MOV    (SP),@SWR        ;;GET CURRENT SWR VALUE
056646 042777 001000 122300    BIC    #BIT09,@SWR      ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
056654 104177                EMT    177             ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
056656 012676 000000          MOV    (SP)+,@(SP)     ;;RESTORE SWR TO ORIGINAL VALUE
056662 012637 001154          MOV    (SP)+,SWR       ;;RESTORE SWR ADDRESS
056666                2003$:
056666 123737 001131 001117    CMPB   $ERMAX,$ERFLG    ;;MAX. ERRORS FOR THIS TEST OCCURRED?
056674 101015                BHI    3$              ;;BR IF NO
056676 032777 001000 122250    BIT    #BIT09,@SWR      ;;LOOP ON ERROR?
056704 001404                BEQ    4$              ;;BR IF NO
056706 013737 001124 001122 7$:  MOV    $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
056714 000446                BR     $OVER           ;;
056716 105037 001117 4$:    CLR    $ERFLG          ;;ZERO THE ERROR FLAG
056722 005057 001206          CLR    $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
056726 000415                BR     1$              ;;ESCAPE TO THE NEXT TEST
056730 032777 004000 122216 3$:  BIT    #BIT11,@SWR      ;;INHIBIT ITERATIONS?
056736 001011                BNE    1$              ;;BR IF YES
056740 005737 001230          TST    $PASS           ;;IF FIRST PASS OF PROGRAM
056744 001406                BEQ    1$              ;;INHIBIT ITERATIONS
056746 005237 001120          INC    $ICNT           ;;INCREMENT ITERATION COUNT
056752 023737 001206 001120    CMP    $TIMES,$ICNT     ;;CHECK THE NUMBER OF ITERATIONS MADE
056760 002024                BGE    $OVER           ;;BR IF MORE ITERATION REQUIRED
056762 012737 000001 001120 1$:  MOV    #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
056770 013737 057046 001206    MOV    $MXCNT,$TIMES    ;;SET NUMBER OF ITERATIONS TO DO
056776 105237 001116          $SVLAD: INCB   $STNM      ;;COUNT TEST NUMBERS
057002 113737 001116 001226    MOV    $STNM,$TESTN     ;;SET TEST NUMBER IN APT MAILBOX
057010 011637 001122          MOV    (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
057014 011637 001124          MOV    (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
057020 005037 001210          CLR    $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
057024 112737 000001 001131    MOV    #1,$ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
057032 013777 001116 122116 $OVER: MOV    $STNM,@DISPLAY    ;;DISPLAY TEST NUMBER
057040 013716 001122          MOV    $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
057044 000002                RTI                    ;;FIXES PS
057046 000005                $MXCNT: 5.             ;;MAX. NUMBER OF ITERATIONS
057050                $SWOBTBL:
                .REPT  $TN-1
057050 007160                .WORD  TST1+2          ;;STARTING ADDRESS OF TEST 1
057052 007470                .WORD  TST2+2          ;;STARTING ADDRESS OF TEST 2
057054 007660                .WORD  TST3+2          ;;STARTING ADDRESS OF TEST 3
057056 010030                .WORD  TST4+2          ;;STARTING ADDRESS OF TEST 4
057060 010160                .WORD  TST5+2          ;;STARTING ADDRESS OF TEST 5
057062 011264                .WORD  TST6+2          ;;STARTING ADDRESS OF TEST 6
057064 012360                .WORD  TST7+2          ;;STARTING ADDRESS OF TEST 7
057066 012454                .WORD  TST10+2         ;;STARTING ADDRESS OF TEST 10
057070 012540                .WORD  TST11+2         ;;STARTING ADDRESS OF TEST 11
057072 013000                .WORD  TST12+2         ;;STARTING ADDRESS OF TEST 12
057074 013346                .WORD  TST13+2         ;;STARTING ADDRESS OF TEST 13

```

057076	014130	.WORD	TST14+2	::STARTING	ADDRESS	OF	TEST	14
057100	014236	.WORD	TST15+2	::STARTING	ADDRESS	OF	TEST	15
057102	014670	.WORD	TST16+2	::STARTING	ADDRESS	OF	TEST	16
057104	015402	.WORD	TST17+2	::STARTING	ADDRESS	OF	TEST	17
057106	015504	.WORD	TST20+2	::STARTING	ADDRESS	OF	TEST	20
057110	015770	.WORD	TST21+2	::STARTING	ADDRESS	OF	TEST	21
057112	016106	.WORD	TST22+2	::STARTING	ADDRESS	OF	TEST	22
057114	016234	.WORD	TST23+2	::STARTING	ADDRESS	OF	TEST	23
057116	016506	.WORD	TST24+2	::STARTING	ADDRESS	OF	TEST	24
057120	017006	.WORD	TST25+2	::STARTING	ADDRESS	OF	TEST	25
057122	017430	.WORD	TST26+2	::STARTING	ADDRESS	OF	TEST	26
057124	017524	.WORD	TST27+2	::STARTING	ADDRESS	OF	TEST	27
057126	020012	.WORD	TST30+2	::STARTING	ADDRESS	OF	TEST	30
057130	020326	.WORD	TST31+2	::STARTING	ADDRESS	OF	TEST	31
057132	020616	.WORD	TST32+2	::STARTING	ADDRESS	OF	TEST	32
057134	021334	.WORD	TST33+2	::STARTING	ADDRESS	OF	TEST	33
057136	021646	.WORD	TST34+2	::STARTING	ADDRESS	OF	TEST	34
057140	022140	.WORD	TST35+2	::STARTING	ADDRESS	OF	TEST	35
057142	022462	.WORD	TST36+2	::STARTING	ADDRESS	OF	TEST	36
057144	023120	.WORD	TST37+2	::STARTING	ADDRESS	OF	TEST	37
057146	023626	.WORD	TST40+2	::STARTING	ADDRESS	OF	TEST	40
057150	024146	.WORD	TST41+2	::STARTING	ADDRESS	OF	TEST	41
057152	024464	.WORD	TST42+2	::STARTING	ADDRESS	OF	TEST	42
057154	025066	.WORD	TST43+2	::STARTING	ADDRESS	OF	TEST	43
057156	025236	.WORD	TST44+2	::STARTING	ADDRESS	OF	TEST	44
057160	025616	.WORD	TST45+2	::STARTING	ADDRESS	OF	TEST	45
057162	026326	.WORD	TST46+2	::STARTING	ADDRESS	OF	TEST	46
057164	026622	.WORD	TST47+2	::STARTING	ADDRESS	OF	TEST	47
057166	027224	.WORD	TST50+2	::STARTING	ADDRESS	OF	TEST	50
057170	027460	.WORD	TST51+2	::STARTING	ADDRESS	OF	TEST	51
057172	027666	.WORD	TST52+2	::STARTING	ADDRESS	OF	TEST	52
057174	030520	.WORD	TST53+2	::STARTING	ADDRESS	OF	TEST	53
057176	030772	.WORD	TST54+2	::STARTING	ADDRESS	OF	TEST	54
057200	031174	.WORD	TST55+2	::STARTING	ADDRESS	OF	TEST	55
057202	031406	.WORD	TST56+2	::STARTING	ADDRESS	OF	TEST	56
057204	031610	.WORD	TST57+2	::STARTING	ADDRESS	OF	TEST	57
057206	031776	.WORD	TST60+2	::STARTING	ADDRESS	OF	TEST	60
057210	032114	.WORD	TST61+2	::STARTING	ADDRESS	OF	TEST	61
057212	032362	.WORD	TST62+2	::STARTING	ADDRESS	OF	TEST	62
057214	032606	.WORD	TST63+2	::STARTING	ADDRESS	OF	TEST	63
057216	032722	.WORD	TST64+2	::STARTING	ADDRESS	OF	TEST	64
057220	033110	.WORD	TST65+2	::STARTING	ADDRESS	OF	TEST	65
057222	033340	.WORD	TST66+2	::STARTING	ADDRESS	OF	TEST	66
057224	033556	.WORD	TST67+2	::STARTING	ADDRESS	OF	TEST	67
057226	034004	.WORD	TST70+2	::STARTING	ADDRESS	OF	TEST	70
057230	034254	.WORD	TST71+2	::STARTING	ADDRESS	OF	TEST	71
057232	034462	.WORD	TST72+2	::STARTING	ADDRESS	OF	TEST	72
057234	035006	.WORD	TST73+2	::STARTING	ADDRESS	OF	TEST	73
057236	035234	.WORD	TST74+2	::STARTING	ADDRESS	OF	TEST	74
057240	035342	.WORD	TST75+2	::STARTING	ADDRESS	OF	TEST	75
057242	035466	.WORD	TST76+2	::STARTING	ADDRESS	OF	TEST	76
057244	035616	.WORD	TST77+2	::STARTING	ADDRESS	OF	TEST	77
057246	035750	.WORD	TST100+2	::STARTING	ADDRESS	OF	TEST	100
057250	036102	.WORD	TST101+2	::STARTING	ADDRESS	OF	TEST	101
057252	036272	.WORD	TST102+2	::STARTING	ADDRESS	OF	TEST	102
057254	036474	.WORD	TST103+2	::STARTING	ADDRESS	OF	TEST	103
057256	036622	.WORD	TST104+2	::STARTING	ADDRESS	OF	TEST	104



057260 037026  
057262 037156  
057264 037352  
057266 037550  
057270 040026  
057272 040332  
057274 041724  
057276 043452  
057300 045632  
057302 046212  
057304 050354  
057306 052642

.WORD TST105+2  
.WORD TST106+2  
.WORD TST107+2  
.WORD TST110+2  
.WORD TST111+2  
.WORD TST112+2  
.WORD TST113+2  
.WORD TST114+2  
.WORD TST115+2  
.WORD TST116+2  
.WORD TST117+2  
.WORD TST120+2

:: STARTING ADDRESS OF TEST 105  
:: STARTING ADDRESS OF TEST 106  
:: STARTING ADDRESS OF TEST 107  
:: STARTING ADDRESS OF TEST 110  
:: STARTING ADDRESS OF TEST 111  
:: STARTING ADDRESS OF TEST 112  
:: STARTING ADDRESS OF TEST 113  
:: STARTING ADDRESS OF TEST 114  
:: STARTING ADDRESS OF TEST 115  
:: STARTING ADDRESS OF TEST 116  
:: STARTING ADDRESS OF TEST 117  
:: STARTING ADDRESS OF TEST 120

.SBTTL ERROR HANDLER ROUTINE

```

:*****
:*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
:*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
:*AND GO TO 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
    
```

```

057310 105037 057702 $ERROR: CLR      IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
057314 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
057316 105237 001117 7$:      INCB      $ERFLG          ;;SET THE ERROR FLAG
057322 001775          BEQ      7$          ;;DON'T LET THE FLAG GO TO ZERO
057324 013777 001116 121624 MOV      $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
057332 032777 002000 121614 BIT      #BIT10,@SWR    ;;BELL ON ERROR?
057340 001402          BEQ      1$          ;;NO - SKIP
057342 104401 001212          TYPE      $BELL          ;;RING BELL
057346 005237 001126 1$:      INC      $ERTTL          ;;COUNT THE NUMBER OF ERRORS
057352 011637 001132          MOV      (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
057356 162737 000002 001132 SUB      #2, $ERRPC
057364 117737 121542 001130 MOV      @ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
057372 032777 001000 121554 BIT      #BIT09,@SWR   ;;SEE IF LOOP ON ERROR IS SET
057400 001060          BNE      1004$      ;;BRANCH AROUND ROUTINE IF SO
057402 122737 000177 001130 CMP      #177, $ITEMB  ;;SEE IF THIS IS THE POWER FAIL CALL
057410 001454          BEQ      1004$      ;;BRANCH AROUND ROUTINE IF IT IS
057412 105737 057702          TSTB     IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
057416 001047          BNE      1003$      ;;BRANCH IF SO
057420 022737 177777 057700 CMP      #-1, CPSAVE   ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
057426 001445          BEQ      1004$      ;;BRANCH IF SO
057430 013746 000004          MOV      ERRVEC, -(SP) ;;SAVE CONTENTS OF ERROR VECTOR
057434 012737 057452 000304 MOV      #1000$, ERRVEC ;;SETUP 'TRAP' RETURN ADDRESS
057442 013737 177766 057700 MOV      177766, CPSAVE ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
057450 000406          BR      1001$
057452 012737 177777 057700 1000$: MOV      #-1, CPSAVE   ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
057460 012716 057466          MOV      #1001$, (SP) ;;SETUP RETURN ADDRESS
057464 000002          RTI
057466 012637 000004 1001$: MOV      (SP)+, ERRVEC ;;RESTORE CONTENTS OF ERROR VECTOR

057472 022737 177777 057700 1002$: CMP      #-1, CPSAVE   ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
057500 001420          BEQ      1004$      ;;BRANCH IF SO
057502 032737 000001 057700 BIT      #BIT00, CPSAVE ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
057510 001414          BEQ      1004$      ;;BRANCH IF OK
057512 042737 000001 177766 BIC      #BIT00, 177766 ;;CLEAR THE BIT FOUND SET
057520 113737 001130 057702 MOV      $ITEMB, IBSAVE ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
057526 112737 000177 001130 MOV      #177, $ITEMB  ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
057534 000402          BR      1004$      ;;BRANCH OVER IBSAVE CLEARING

057536 105037 057702 1003$: CLR      IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
057542          1004$:
057542 032777 020000 121404 BIT      #BIT13,@SWR   ;;SKIP TYPEOUT IF SET
057550 001004          BNE      20$          ;;SKIP TYPEOUTS
057552 004737 057704          JSR      PC, ERRYP      ;;GO TO USER ERROR ROUTINE
    
```

057556	104401	001217		TYPE	,\$CRLF	
057562			20\$:			
057562	122737	000001	001242	CMPB	#APTENV,\$ENV	::RUNNING IN APT MODE
057570	001007			BNE	2\$	::NO SKIP APT ERROR REPORT
057572	113737	001130	057604	MOVB	\$ITEMB,21\$	::SET ITEM NUMBER AS ERROR NUMBER
057600	004737	062616		JSR	PC,\$ATY4	::REPORT FATAL ERROR TO APT
057604	000		21\$:	.BYTE	0	
057605	000			.BYTE	0	
057606	000777		22\$:	BR	22\$	::APT ERROR LOOP
057610	105737	057702	2\$:	TSTB	IBSAVE	::SEE IF IBSAVE IS LOADED
057614	001005			BNE	3\$	::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
057616	005777	121332		TST	@SWR	::HALT ON ERROR
057622	100002			BPL	3\$	::SKIP IF CONTINUE
057624	000000			HALT		::HALT ON ERROR!
057626	104410			CKSWR		::TEST FOR CHANGE IN SOFT-SWR
057630			3\$:			
057630	032777	001000	121316	BIT	#BIT09,@SWR	::LOOP ON ERROR SWITCH SET?
057636	001402			BEQ	4\$	::BR IF NO
057640	013716	001124		MOV	\$LPERR,(SP)	::FUDGE RETURN FOR LOOPING
057644	005737	001210	4\$:	TST	\$ESCAPE	::CHECK FOR AN ESCAPE ADDRESS
057650	001402			BEQ	5\$	::BR IF NONE
057652	013716	001210		MOV	\$ESCAPE,(SP)	::FUDGE RETURN ADDRESS FOR ESCAPE
057656			5\$:			
057656	022737	054302	000042	CMP	#SENDAD,@#42	::ACT-11 AUTO-ACCEPT?
057664	001001			BNE	6\$	::BRANCH IF NO
057666	000000			HALT		::YES
057670			6\$:			
057670	105737	057702		TSTB	IBSAVE	::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
057674	001210			BNE	7\$	::BRANCH BACK TO CALL ORIGINAL ERROR
057676	000002			RTI		::RETURN
057700	000000			CPSAVE: .WORD	0	::LOCATION TO SAVE CPU ERROR REG CONTENTS
057702	000000			IBSAVE: .WORD	0	::LOCATION TO SAVE ITEM BYTE

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

```

057704 104414
057706 032777 020000 121240
057714 001402
057716 000137 060514

057722 104401 001217
057726 104401 060530
057732 013746 001234

057736 104403
057740 003
057741 000

057742 013700 001276
057746 016000 000026
057752 042700 177740
057756 012737 063512 060000
057764 022700 000026
057770 001004
057772 104401 060565
057776 104401
060000 000000

060002 005037 060520
060006 013737 001226 060520
060014 104401 060535
060020 013746 060520

060024 104403
060026 003
060027 000
060030 005037 060522
060034 113737 001130 060522
060042 001406
060044 104401 060545
060050 013746 060522

060054 104403
060056 003
060057 000
060060 104401 060554
  
```

.SBTTL ERROR TYPEOUT ROUTINE

```

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

```

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

;TYPE UNIT NUMBER, DRIVE TYPE, TEST NUMBER, ERROR NUMBER, AND
;PROGRAM COUNTER
1$: TYPE ,SCLRF ;TYPE 'DRV#'
TYPE ,ERTY00 ;:SAVE $UNIT FOR TYPEOUT
MOV $UNIT,-(SP) ;:TYPE DRIVE NUMBER
;:GO TYPE--OCTAL ASCII
;:TYPE 3 DIGIT(S)
;:SUPPRESS LEADING ZEROS
TYPOS ;:TYPE 'DRIVE TYPE', RM80 FOR UNIT UNDER TEST
.BYTE 3 ;GET RM BASE ADDRESS
.BYTE 0 ;GET DRIVE TYPE REGISTER
;SAVE DRIVE TYPE BITS AND
;GET ASCII DRIVE TYPE
;IS DEVICE AN RM80 ?
2$: TYPE ,ERTY05 ;NO !!
TYPE " - " ;TYPE " - "
;TYPE DRIVE TYPE
;DRIVE TYPE MESSAGE IS STORED HERE
3$: .WORD 0

;TYPE TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER
4$: CLR TSTNMB ;LOAD TEST NUMBER FOR
MOV $TSTN,TSTNMB ;TYPE 'TST#'
TYPE ,ERTY01 ;:SAVE TSTNMB FOR TYPEOUT
MOV TSTNMB,-(SP) ;:TYPE TEST NUMBER
;:GO TYPE--OCTAL ASCII
;:TYPE 3 DIGIT(S)
;:SUPPRESS LEADING ZEROS
;LOAD ERROR NUMBER FOR
;TYPEOUT
;SKIP IF NO ERROR CALLED
5$: CLR ERRNMB ;TYPE 'ERR#'
MOV $ITEMB,ERRNMB ;:SAVE ERRNMB FOR TYPEOUT
TYPE ,ERTY02 ;:TYPE ERROR NUMBER
MOV ERRNMB,-(SP) ;:GO TYPE--OCTAL ASCII
;:TYPE 3 DIGIT(S)
;:SUPPRESS LEADING ZEROS
5$: TYPE ,ERTY03 ;TYPE 'PC='
  
```

```

46 060064 013746 001132      MOV      $ERRPC,-(SP)      ;;SAVE $ERRPC FOR TYPEOUT
                                ;;TYPE PROGRAM COUNTER
                                ;;GO TYPE--OCTAL ASCII
                                ;;TYPE 6 DIGIT(S)
                                ;;TYPE LEADING ZEROS
      060070 104403      TYPOS
      060072      006      .BYTE 6
      060073      001      .BYTE 1

47
48      ;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
49 060074 00573, 060522 6$: TST      ERRNMB      ;WAS AN ERROR CALLED?
50 060100 001002      BNE      7$          ;BR IF YES
51 060102 000137 060514      JMP      27$        ;NO--EXIT
52
53 060106 104401 001217 7$: TYPE      , $CRLF      ;YES-TYPE CRLF
54 060112 105037 060526      CLRB     BOTFLG      ;CLEAR BOT FLAG
55 060116 105037 060527      CLRB     CHRCNT      ;CLEAR CHARACTER COUNTER
56 060122 013700 060522      MOV      ERRNMB,RO   ;RO POINTS TO FIRST OF
57 060126 122700 000177      CMPB    #177,RO      ;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
58 060132 001003      BNE      8$          ;BRANCH IF NOT
59 060134 012700 060572      MOV      #PFECH,RO   ;MOVE POWER FAIL ERROR CALL TABLE TO RO
60 060140 000405      BR      9$
61 060142 006300      8$: ASL      RO          ;FOUR ENTRIES IN ERROR
62 060144 006300      ASL      RO          ;TABLE
63 060146 006300      ASL      RO
64 060150 062700 001532      ADD      #$ERRTB-8.,RO
65 060154 011001      9$: MOV      (RO),R1      ;R1 POINTS TO ERROR MESSAGE
66                                     ;TABLE
67 060156 001507      BEQ      19$        ;BRANCH IF NO ERROR MESSAGE
68
69      ;TYPE THE ERROR MESSAGE
70 060160 012102 10$: MOV      (R1)+,R2      ;R2=ADDRESS OF MESSAGE STRING
71 060162 001505      BEQ      19$        ;BRANCH IF END OF MESSAGE
72 060164 010237 060332      MOV      R2,18$     ;LOAD ADDRESS OF STRING
73 060170 005037 060524      CLR      BOTADR      ;CLEAR BOT ADDRESS
74 060174 112203 11$: MOVB     (R2)+,R3      ;END OF STRING??
75 060176 001454      BEQ      17$        ;YES!!
76 060200 122703 000015      CMPB    #CR,R3      ;CARRIAGE RETURN??
77 060204 001003      BNE      12$        ;NO!!
78 060206 105037 060527      CLRB     CHRCNT      ;YES-CLEAR CHAR COUNT
79 060212 000770      BR      11$        ;GET NEXT CHARACTER
80 060214 122703 000012 12$: CMPB    #LF,R3      ;LINE FEED??
81 060220 001765      BEQ      11$        ;YES-GET NEXT CHARACTER
82 060222 122703 000011      CMPB    #HT,R3      ;HORIZONTAL TAB??
83 060226 001007      BNE      14$        ;NO!!
84 060230 105237 060527 13$: INCB     CHRCNT      ;ADJUST CHARACTER COUNT
85 060234 132737 000007 060527      BITB    #7,CHRCNT
86 060242 001372      BNE      13$
87 060244 000407      BR      15$
88 060246 105237 060527 14$: INCB     CHRCNT      ;INCREMENT CHARACTER COUNT
89 060252 122703 000040      CMPB    #' ,R3      ;SPACE??
90 060256 001002      BNE      15$        ;NO!!
91 060260 010237 060524      MOV      R2,BOTADR   ;SAVE ADDRESS OF SPACE
92 060264 122737 000100 060527 15$: CMPB    #64.,CHRCNT ;END OF LINE??
93 060272 103340      BHIS    11$        ;NO!!
94 060274 013704 060524      MOV      BOTADR,R4   ;GET ADDRESS OF LAST SPACE
95 060300 001007      BNE      16$        ;BRANCH IF SPACE DETECTED
96 060302 104401 001217      TYPE      , $CRLF      ;TYPE CRLF
97 060306 105037 060527      CLRB     CHRCNT      ;CLEAR CHARACTER COUNT
98 060312 013702 060332      MOV      18$,R2      ;SET UP R2 FOR TESTING

```

99	060316	000726		BR	11\$	
100	060320	105044		CLRB	-(R4)	:REPLACE SPACE
101	060322	112737	177777 060526	MOVB	#-1,BOTFLG	:SET BOT FLAG
102	060330	104401		TYPE		:TYPE ERROR MESSAGE STRING
103	060332	000000		.WORD		:STRING ADDRESS GOES HERE
104	060334	105737	060526	TSTB	BOTFLG	:WAS STRING TRUNCATED??
105	060340	001707		BEQ	10\$	:NO!!
106	060342	104401	001217	TYPE	,\$CRLF	:YES-TYPE CRLF
107	060346	105037	060526	CLRB	BOTFLG	:CLEAR BOT FLAG
108	060352	105037	060527	CLRB	CHRCNT	:CLEAR CHARACTER COUNT
109	060356	013702	060524	MOV	BOTADR,R2	:SETUP R2 FOR TESTING
110	060362	010237	060332	MOV	R2,18\$	:SETUP 18\$ FOR TYPING
111	060366	112742	000040	MOVB	#',-(R2)	:RESTORE SPACE
112	060372	105722		TSTB	(R2)+	:RESTORE R2
113	060374	000677		BR	11\$	:TYPE REST OF STRING
114						
115						:TYPE ERROR HEADER AND ERROR DATA
116	060376	016001	000002	19\$: MOV	2(R0),R1	:R1 POINTS TO ERROR HEADER TABLE
117	060402	001444		BEQ	27\$	:BRANCH IF NO HEADER
118	060404	104401	001217	TYPE	,\$CRLF	: (ASSUME NO DATA)
119	060410	016002	000004	MOV	4(R0),R2	:R2 POINTS TO DATA ADDRESS TABLE
120	060414	016003	000006	MOV	6(R0),R3	:R3 POINTS TO FORMAT TABLE
121	060420	012137	060430	20\$: MOV	(R1)+,21\$	:PUT HEADER ADDRESS FOR TYPE
122	060424	001433		BEQ	27\$	:BRANCH IF END OF HEADERS
123						: (ASSUME END OF DATA)
124	060426	104401		TYPE		
125	060430	000000		.WORD	0	:HEADER ADDRESS GOES HERE
126	060432	104401	001217	TYPE	,\$CRLF	
127	060436	005702		TST	R2	:DATA WITH HEADER??
128	060440	001767		BEQ	20\$	:NO!!
129	060442	012204		MOV	(R2)+,R4	:R4 POINTS TO DATA ADDRESS
130	060444	012305		MOV	(R3)+,R5	:R5 POINTS TO FORMAT
131	060446	105725		22\$: TSTB	(R5)+	:WHAT KIND OF DATA??
132	060450	100407		BMI	24\$	:BINARY
133	060452	001403		BEQ	23\$	:OCTAL
134	060454	013446		MOV	@(R4)+,-(SP)	:DECIMAL
135	060456	104405		TYPDS		
136	060460	000405		BR	25\$	
137	060462	013446		23\$: MOV	@(R4)+,-(SP)	
138	060464	104402		TYPOC		
139	060466	000402		BR	25\$	
140	060470	013446		24\$: MOV	@(R4)+,-(SP)	
141	060472	104406		TYPBN		
142	060474	005714		25\$: TST	(R4)	:MORE DATA??
143	060476	001403		BEQ	26\$	:NO!!
144	060500	104401	060562	TYPE	ERTY04	:YES-TYPE 2 SPACES
145	060504	000760		BR	22\$	:AND CONTINUE
146	060506	104401	001217	26\$: TYPE	,\$CRLF	:TYPE ONE BLANK LINE
147	060512	000742		BR	20\$	:BEFORE NEXT HEADER
148	060514	104415		27\$: RESREG		
149	060516	000207		RTS	PC	
150						
151	060520	000000		TSTNMB:	.WORD 0	:TEST NUMBER
152	060522	000000		ERRNMB:	.WORD 0	:ERROR NUMBER
153	060524	000000		BOTADR:	.WORD 0	:BEGINNING OF TEXT ADDRESS
154	060526	000		BOTFLG:	.BYTE 0	:BOT FLAG
155	060527	000		CHRCNT:	.BYTE 0	:CHARACTER COUNT

156									
157	060530	104	122	126	ERTY00:	.ASCIZ	@DRV#@		
158	060535	054	040	124	ERTY01:	.ASCIZ	@, TEST#@		
159	060545	054	040	105	ERTY02:	.ASCIZ	@, ERR#@		
160	060554	054	040	120	ERTY03:	.ASCIZ	@, PC=@		
161	060562	040	040	000	ERTY04:	.ASCIZ	@ @		
162	060565	040	055	040	ERTY05:	.ASCIZ	@ - @		
163					.EVEN				
164	060572	060602	060670	060706	PFECH:	PFECH1,PFECH2,PFECH3,PFECH4	;WORDS DEFINING TABLES BELOW		
165	060602	060606	000000		PFECH1:	+.4,0			
166	060606	120	117	127		.ASCIZ	?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?		
167					.EVEN				
168	060670	060674	000000		PFECH2:	+.4,0			
169	060674	103	120	125		.ASCIZ	?CPUERREG?		
170					.EVEN				
171	060706	060710			PFECH3:	+.2			
172	060710	057700	000000			.WORD	CPSAVE,0		
173	060714	060716			PFECH4:	+.2			
174	060716	000	000			.BYTE	0,0		

1

.SBTTL TTY INPUT ROUTINE

```

*****
ENABL  LSB
060720 000000 $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
060722 000000 $TKQIN: .WORD 0     ;;INPUT POINTER
060724 000000 $TKQOUT: .WORD 0    ;;OUTPUT POINTER
060726 060727 $TKQSR: .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
060730 005037 060720 $TKINT: CLR  $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
060734 012737 060726 060722 MOV  #$TKQSR,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
060742 013737 060722 060724 MOV  $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
060750 012737 061000 000060 MOV  #$TKSRV,@TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
060756 012737 000200 000062 MOV  #200,@TKVEC+2  ;;'BR' LEVEL 4
060764 005777 120172 000062 TST  @TKB           ;;CLEAR DONE FLAG
060770 012777 000100 120162 MOV  #100,@TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
060776 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)
.
061000 117746 120156 $TKSRV: MOVB  @TKB,-(SP)  ;;PICKUP THE CHARACTER
061004 042716 177600 BIC  #^C177,(SP)      ;;STRIP THE JUNK
061010 021627 000021 CMP  (SP),#$XON      ;;IS IT A RANDOM XON?
061014 001002      BNE  30$      ;;BRANCH IF NO
061016 005726      TST  (SP)+      ;;CLEAN RANDOM XON OFF STACK
061020 000002      RTI           ;;RETURN
061022      30$:
061022 021627 000003 CMP  (SP),#3         ;;IS IT A CONTROL C?
061026 001007      BNE  1$      ;;BRANCH IF NO
061030 104401 062126 TYPE  $CNTLC          ;;TYPE A CONTROL-C (^C)
061034 004737 060730 JSR  PC,$TKINT      ;;INIT THE KEYBOARD
061040 005726      TST  (SP)+      ;;CLEAN UP STACK
061042 000137 062170 JMP  SHUT           ;;CONTROL C RESTART
061046 021627 000007 1$:  CMP  (SP),#7         ;;IS IT A CONTROL G?
061052 001004      BNE  2$      ;;BRANCH IF NO
061054 022737 000176 001154 CMP  #SWREG,SWR     ;;IS SOFT-SWR SELECTED?
061062 001500      BEQ  6$      ;;GO TO SWR CHANGE

061064      2$:
061064 022737 000001 060720 CMP  #1,$TKCNT      ;;IS THE QUEUE FULL?
061072 001004      BNE  3$      ;;BRANCH IF NO
061074 104401 001212 TYPE  $BELL          ;;RING THE TTY BELL

```



```

061100 005726          TST      (SP)+      ;; CLEAN CHARACTER OFF OF STACK
061102 000451          BR        5$          ;; EXIT
061104 021627 000023  3$:    CMP      (SP),#23      ;; IS IT A CONTROL-S?
061110 001021          BNE      32$          ;; BRANCH IF NO
061112 005077 120042          CLR      @STKS      ;; DISABLE TTY KEYBOARD INTERRUPTS
061116 005726          TST      (SP)+      ;; CLEAN CHAR OFF STACK
061120 105777 120034  31$:    TSTB     @STKS      ;; WAIT FOR A CHAR
061124 100375          BPL      31$          ;; LOOP UNTIL ITS THERE
061126 117746 120030          MOVB     @STKB,-(SP)  ;; GET THE CHARACTER
061132 042716 177600          BIC      #'C177,(SP) ;; MAKE IT 7-BIT ASCII
061136 022627 000021          CMP      (SP)+,#21  ;; IS IT A CONTROL-Q?
061142 001366          BNE      31$          ;; BRANCH IF NO
061144 012777 000100 120006  MOV      #100,@STKS  ;; REENABLE TTY KEYBOARD INTERRUPTS
061152 000002          RTI          ;; RETURN
061154 005237 060720  32$:    INC      $TKCNT     ;; COUNT THIS CHARACTER
061160 021627 000140          CMP      (SP),#140  ;; IS IT UPPER CASE?
061164 002405          BLT      4$          ;; BRANCH IF YES
061166 021627 000175          CMP      (SP),#175  ;; IS IT A SPECIAL CHAR?
061172 003002          BGT      4$          ;; BRANCH IF YES
061174 042716 000040          BIC      #40,(SP)   ;; MAKE IT UPPER CASE
061200 112677 177516  4$:    MOVB     (SP)+,@STKQIN ;; AND PUT IT IN QUEUE
061204 005237 060722          INC      $TKQIN     ;; UPDATE THE POINTER
061210 023727 060722 060727  CMP      $TKQIN,$STKQEND ;; GO OFF THE END?
061216 001003          BNE      5$          ;; BRANCH IF NO
061220 012737 060726 060722  MOV      #$STKQRT,$TKQIN ;; RESET THE POINTER
061226 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.

```

061230 022737 000176 001154 $CKSWR: CMP      #SWREG,SWR  ;; IS THE SOFT-SWR SELECTED
061236 001124          BNE      15$          ;; EXIT IF NOT
061240 105777 117714          TSTB     @STKS      ;; IS A CHAR WAITING?
061244 100121          BPL      15$          ;; IF NOT, EXIT
061246 117746 117710          MOVB     @STKB,-(SP) ;; YES
061252 042716 177600          BIC      #'C177,(SP) ;; MAKE IT 7-BIT ASCII
061256 021627 000007          CMP      (SP),#7    ;; IS IT A CONTROL-G?
061262 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.

```

061264 123727 001150 000001 6$:    CMPEB   $AUTOB,#1  ;; ARE WE RUNNING IN AUTO-MODE?
061272 001674          BEQ      2$          ;; BRANCH IF YES
061274 005726          TST      (SP)+      ;; CLEAR CONTROL-G OFF STACK
061276 004737 060730          JSR      PC,$TKINT  ;; FLUSH THE TTY INPUT QUEUE
061302 005077 117652          CLR      @STKS      ;; DISABLE TTY KEYBOARD INTERRUPTS
061306 112737 000001 001151  MOVB     #1,$INTAG   ;; SET INTERRUPT MODE INDICATOR

061314 104401 062140          TYPE     ,SCNTLG    ;; ECHO THE CONTROL-G (^G)
061320 104401 062145          TYPE     ,MSWR     ;; TYPE CURRENT CONTENTS
061324 013746 000176          MOV      SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
061330 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)

```

061332	104401	062156		TYPE	.SMNEW	::PROMPT FOR NEW SWR
061336	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
061340	005046			CLR	-(SP)	::THE NEW SWR
061342	105777	117612	7\$:	TSTB	@\$TKS	::CHAR THERE?
061346	100375			BPL	7\$	::IF NOT TRY AGAIN
061350	117746	117606		MOVB	@\$TKB, -(SP)	::PICK UP CHAR
061354	042716	177600		BIC	#^C177, (SP)	::MAKE IT 7-BIT ASCII
061360	021627	000003		CMP	(SP), #3	::IS IT A CONTROL-C?
061364	001015			BNE	9\$	::BRANCH IF NOT
061366	104401	062126		TYPE	.\$CNTLC	::YES, ECHO CONTROL-C (^C)
061372	062706	000006		ADD	#6, SP	::CLEAN UP STACK
061376	123727	001151	000001	CMPB	\$INTAG, #1	::REENABLE TTY KEYBOARD INTERRUPTS?
061404	001003			BNE	8\$	::BRANCH IF NO
061406	012777	000100	117544	MOV	#100, @\$TKS	::ALLOW TTY KEYBOARD INTERRUPTS
061414	000137	062170	8\$:	JMP	SHUT	::CONTROL-C RESTART
061420	021627	000025	9\$:	CMP	(SP), #25	::IS IT A CONTROL-U?
061424	001005			BNE	10\$	::BRANCH IF NOT
061426	104401	062133		TYPE	.\$CNTLU	::YES, ECHO CONTROL-U (^U)
061432	062706	000006	20\$:	ADD	#6, SP	::IGNORE PREVIOUS INPUT
061436	000737			BR	19\$	::LET'S TRY IT AGAIN
061440	021627	000015	10\$:	CMP	(SP), #15	::IS IT A <CR>?
061444	001022			BNE	16\$	::BRANCH IF NO
061446	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
061452	001403			BEQ	11\$	::BRANCH IF YES
061454	016677	000002	117472	MOV	2(SP), @SWR	::SAVE NEW SWR
061462	062706	000006	11\$:	ADD	#6, SP	::CLEAR UP STACK
061466	104401	001217	14\$:	TYPE	.\$CRLF	::ECHO <CR> AND <LF>
061472	123727	001151	000001	CMPB	\$INTAG, #1	::RE-ENABLE TTY KBD INTERRUPTS?
061500	001003			BNE	15\$	::BRANCH IF NOT
061502	012777	000100	117450	MOV	#100, @\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
061510	000002		15\$:	RTI		::RETURN
061512	004737	056246	16\$:	JSR	PC, \$TYPEC	::ECHO CHAR
061516	021627	000060		CMP	(SP), #60	::CHAR < 0?
061522	002420			BLT	18\$	::BRANCH IF YES
061524	021627	000067		CMP	(SP), #67	::CHAR > 7?
061530	003015			BGT	18\$	::BRANCH IF YES
061532	042726	000060		BIC	#60, (SP)+	::STRIP-OFF ASCII
061536	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
061542	001403			BEQ	17\$	::BRANCH IF YES
061544	006316			ASL	(SP)	::NO, SHIFT PRESENT
061546	006316			ASL	(SP)	::CHAR OVER TO MAKE
061550	006316			ASL	(SP)	::ROOM FOR NEW ONE.
061552	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
061556	056616	177776		BIS	-2(SP), (SP)	::SET IN NEW CHAR
061562	000667			BR	7\$	::GET THE NEXT ONE
061564	104401	001216	18\$:	TYPE	.\$QJES	::TYPE ?<CR><LF>
061570	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  
 ;;\*

061572	011646			\$RDCHR:	MOV	(SP),-(SP)	;;PUSH DOWN THE PC AND
061574	016666	000004	000002		MOV	4(SP),2(SP)	;;THE PS
061602	005066	000004			CLR	4(SP)	;;GET READY FOR A CHARACTER
061606	005046				CLR	-(SP)	;;PUT NEW PS ON STACK
061610	012746	061616			MOV	#64\$,-(SP)	;;PUT NEW PC ON STACK
061614	000002				RTI		;;POP NEW PC AND PS
061616				64\$:			
061616	005737	060720		1\$:	TST	\$TKCNT	;;WAIT ON A CHARACTER
061622	001775				BEQ	1\$	
061624	065337	060720			DEC	\$TKCNT	;;DECREMENT THE COUNTER
061630	117766	177070	000004		MOVB	@\$TKQOUT,4(SP)	;;GET ONE CHARACTER
061636	005237	060724			INC	\$TKQOUT	;;UPDATE THE POINTER
061642	023727	060724	060727		CMP	\$TKQOUT,#\$TKQEND	;;DID IT GO OFF OF THE END?
061650	001003				BNE	2\$	;;BRANCH IF NO
061652	012737	060726	060724		MOV	#\$TKQSRST,\$TKQOUT	;;RESET THE POINTER
061660	000002			2\$:	RTI		;;RETURN

\*\*\*\*\*

;;\*THIS ROUTINE WILL INPUT A STRING FROM THE TTY

;;\*CALL:  
 ;;\* RDLIN ;;INPUT A STRING FROM THE TTY  
 ;;\* RETURN HERE ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK  
 ;;\* ;;TERMINATOR WILL BE A BYTE OF ALL 0'S  
 ;;\*

061662	010346			\$RDLIN:	MOV	R3,-(SP)	;;SAVE R3
061664	005046				CLR	-(SP)	;;CLEAR THE RUBOUT KEY
061666	012703	062116		1\$:	MOV	#\$TTYIN,R3	;;GET ADDRESS
061672	022703	062126		2\$:	CMP	#\$TTYIN+8.,R3	;;BUFFER FULL?
061676	101456				BLOS	4\$	;;BR IF YES
061700	104411				RDCHR		;;GO READ ONE CHARACTER FROM THE TTY
061702	112613				MOVB	(SP)+,(R3)	;;GET CHARACTER
061704	122713	000177		10\$:	CMPEB	#177,(R3)	;;IS IT A RUBOUT
061710	001022				BNE	5\$	;;BR IF NO
061712	005716				TST	(SP)	;;IS THIS THE FIRST RUBOUT?
061714	001007				BNE	6\$	;;BR IF NO
061716	112737	000134	062114		MOVB	#'\,9\$	;;TYPE A BACK SLASH
061724	104401	062114			TYPE	,9\$	
061730	012716	177777			MOV	#-1,(SP)	;;SET THE RUBOUT KEY
061734	005303			6\$:	DEC	R3	;;BACKUP BY ONE
061736	020327	062116			CMP	R3,#\$TTYIN	;;STACK EMPTY?
061742	103434				BLO	4\$	;;BR IF YES
061744	111337	062114			MOVB	(R3),9\$	;;SETUP TO TYPEOUT THE DELETED CHAR.
061750	104401	062114			TYPE	,9\$	;;GO TYPE
061754	000746				BR	2\$	;;GO READ ANOTHER CHAR.
061756	005716			5\$:	TST	(SP)	;;RUBOUT KEY SET?
061760	001406				BEQ	7\$	;;BR IF NO
061762	112737	000134	062114		MOVB	#'\,9\$	;;TYPE A BACK SLASH
061770	104401	062114			TYPE	,9\$	
061774	005016				CLR	(SP)	;;CLEAR THE RUBOUT KEY
061776	122713	000025		7\$:	CMPEB	#25,(R3)	;;IS CHARACTER A CTRL U?
062002	001003				BNE	8\$	;;BR IF NO

062004	104401	062133		TYPE	\$CNTLU	::TYPE A CONTROL 'U'
062010	000726			BR	1\$	::GO START OVER
062012	122713	000022	8\$:	CMPB	#22,(R3)	::IS CHARACTER A "'R'?"
062016	001011			BNE	3\$	::BRANCH IF NO
062020	105013			CLRB	(R3)	::CLEAR THE CHARACTER
062022	104401	001217		TYPE	,\$CRLF	::TYPE A 'CR' & 'LF'
062026	104401	062116		TYPE	,\$TTYIN	::TYPE THE INPUT STRING
062032	000717			BR	2\$	::GO PICKUP ANOTHER CHACTER
062034	104401	001216	4\$:	TYPE	,\$QUES	::TYPE A '?'
062040	000712			BR	1\$	::CLEAR THE BUFFER AND LOOP
062042	111337	062114	3\$:	MOVB	(R3),9\$	::ECHO THE CHARACTER
062046	104401	062114		TYPE	,\$9\$	
062052	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
062056	001305			BNE	2\$	::LOOP IF NOT RETURN
062060	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
062064	104401	001220		TYPE	,\$LF	::TYPE A LINE FEED
062070	005726			TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
062072	012603			MOV	(SP)+,R3	::RESTORE R3
062074	011646			MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
062076	016666	000004	000002	MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
062104	012766	062116	000004	MOV	#TTYIN,4(SP)	
062112	000002			RTI		::RETURN
062114	000		9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
062115	000			.BYTE	0	::TERMINATOR
062116				.BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT
062126	136	103	015	\$CNTLC:	.\$ASCIZ /^C/<15><12>	::CONTROL 'C'
062133	136	125	015	\$CNTLU:	.\$ASCIZ /^U/<15><12>	::CONTROL 'U'
062140	136	107	015	\$CNTLG:	.\$ASCIZ /^G/<15><12>	::CONTROL 'G'
062145	015	012	123	\$MSWR:	.\$ASCIZ <15><12>/SWR = /	
062156	040	040	116	\$MNEW:	.\$ASCIZ / NEW = /	
				.EVEN		
2						
3	062170	005737	000042	SHUT:	TST @#42	::ANY MONITOR PRESENT ?
4	062174	001002			1\$	::BR IF YES
5	062176	000137	004652		JMP START	::GO TO START
6	062202	005037	001300	1\$:	CLR \$DEVN	::FUDGE NO DRIVES IN MAP
7	062206	000137	054104		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  
 ::\* RETURN HERE

:::READ AN OCTAL NUMBER  
 :::LOW ORDER BITS ARE ON TOP OF THE STACK  
 :::HIGH ORDER BITS ARE IN \$HIOCT

062212 011646  
 062214 016666 000004 000002  
 062222 010046  
 062224 010146  
 062226 010246  
 062230 104412  
 062232 012600  
 062234 005001  
 062236 005002  
 062240 112046  
 062242 001412  
 062244 006301  
 062246 006102  
 062250 006301  
 062252 006102  
 062254 006301  
 062256 006102  
 062260 042716 177770  
 062264 062601  
 062266 000764  
 062270 005726  
 062272 010166 000012  
 062276 010237 062312  
 062302 012602  
 062304 012601  
 062306 012600  
 062310 000002  
 062312 000000

\$RDOCT: MOV (SP),-(SP)  
 MOV 4(SP),2(SP)  
 MOV R0,-(SP)  
 MOV R1,-(SP)  
 MOV R2,-(SP)  
 1\$: RDLIN  
 MOV (SP)+,R0  
 CLR R1  
 CLR R2  
 2\$: MOVB (R0)+,-(SP)  
 BEQ 3\$  
 ASL R1  
 ROL R2  
 ASL R1  
 ROL R2  
 ASL R1  
 ROL R2  
 BIC #^C7,(SP)  
 ADD (SP)+,R1  
 BR 2\$  
 3\$: TST (SP)+  
 MOV R1,12(SP)  
 MOV R2,\$HIOCT  
 MOV (SP)+,R2  
 MOV (SP)+,R1  
 MOV (SP)+,R0  
 RTI  
 \$HIOCT: .WORD 0

:::PROVIDE SPACE FOR THE  
 :::INPUT NUMBER  
 :::PUSH R0 ON STACK  
 :::PUSH R1 ON STACK  
 :::PUSH R2 ON STACK  
 :::READ AN ASCII LINE  
 :::GET ADDRESS OF 1ST CHARACTER  
 :::CLEAR DATA WORD  
 :::PICKUP THIS CHARACTER  
 :::IF ZERO GET OUT  
 :::\*2  
 :::\*4  
 :::\*8  
 :::STRIP THE ASCII JUNK  
 :::ADD IN THIS DIGIT  
 :::LOOP  
 :::CLEAN TERMINATOR FROM STACK  
 :::SAVE THE RESULT  
 :::POP STACK INTO R2  
 :::POP STACK INTO R1  
 :::POP STACK INTO R0  
 :::RETURN  
 :::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.

062314	016646	000002	\$TRAP:	MOV	2(SP),-(SP)	::ASSUME THE STATUS OF
062320	042716	000020		BIC	#20,(SP)	:: THE CALLER--DO NOT ALLOW
062324	012746	062332		MOV	#1\$,-(SP)	:: T-BIT TRAPS
062330	000002			RTI		::SET THE NEW STATUS
062332	010046		1\$:	MOV	R0,-(SP)	::SAVE R0
062334	016600	000002		MOV	2(SP),R0	::GET TRAP ADDRESS
062340	005740			TST	-(R0)	::BACKUP BY 2
062342	111000			MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
062344	006300			ASL	R0	::POSITION FOR INDEXING
062346	016000	062366		MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
062352	000200			RTS	R0	::GO TO ROUTINE

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

062354	011646	000004	000002	\$TRAP2:	MOV	(SP),-(SP)	::MOVE THE PC DOWN
062356	016666				MOV	4(SP),2(SP)	::MOVE THE PSW DOWN
062364	000002				RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

				:	ROUTINE		
				:	-----		
062366	062354			\$TRPAD:	.WORD	\$TRAP2	
062370	056034				\$TYPE	::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
062372	055632				\$TYPOC	::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
062374	055606				\$TYPOS	::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
062376	055646				\$TYPON	::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
062400	055362				\$TYPDS	::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
062402	055306				\$TYPBN	::CALL=TYPBN	TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
062404	061320				\$GTSWR	::CALL=GTSWR	TRAP+7(104407) GET SOFT-SWR SETTING
062406	061230				\$CKSWR	::CALL=CKSWR	TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
062410	061572				\$RDCHR	::CALL=RDCHR	TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
062412	061662				\$RDLIN	::CALL=RDLIN	TRAP+12(104412) TTY TYPEIN STRING ROUTINE
062414	062212				\$RDOCT	::CALL=RDOCT	TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
062416	055212				\$SAVREG	::CALL=SAVREG	TRAP+14(104414) SAVE R0-R5 ROUTINE
062420	055250				\$RESREG	::CALL=RESREG	TRAP+15(104415) RESTORE R0-R5 ROUTINE

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
062422 012737 062562 000024 $PWRDN: MOV    #SILLUP,@#PWRVEC  ;;SET FOR FAST UP
062430 012737 000340 000026      MOV    #340,@#PWRVEC+2  ;;PRIO:7
062436 010046      MOV    R0,-(SP)        ;;PUSH R0 ON STACK
062440 010146      MOV    R1,-(SP)        ;;PUSH R1 ON STACK
062442 010246      MOV    R2,-(SP)        ;;PUSH R2 ON STACK
062444 010346      MOV    R3,-(SP)        ;;PUSH R3 ON STACK
062446 010446      MOV    R4,-(SP)        ;;PUSH R4 ON STACK
062450 010546      MOV    R5,-(SP)        ;;PUSH R5 ON STACK
062452 017746      MOV    @SWR,-(SP)      ;;PUSH @SWR ON STACK
062456 010637 062566      MOV    SP,$SAVR6     ;;SAVE SP
062462 012737 062474 000024      MOV    #SPWRUP,@#PWRVEC ;;SET UP VECTOR
062470 000000      HALT
062472 000776      BR     .-2           ;;HANG UP

*****
:POWER UP ROUTINE
062474 012737 062562 000024 $PWRUP: MOV    #SILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
062502 013706 062566      MOV    $SAVR6,SP     ;;GET SP
062506 005037 062566      CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
062512 005237 062566      1$:   INC    $SAVR6     ;;WAIT FOR THE INC
062516 001375      BNE    1$           ;;OF WORD
062520 012677 116430      MOV    (SP)+,@SWR    ;;POP STACK INTO @SWR
062524 012605      MOV    (SP)+,R5     ;;POP STACK INTO R5
062526 012604      MOV    (SP)+,R4     ;;POP STACK INTO R4
062530 012603      MOV    (SP)+,R3     ;;POP STACK INTO R3
062532 012602      MOV    (SP)+,R2     ;;POP STACK INTO R2
062534 012601      MOV    (SP)+,R1     ;;POP STACK INTO R1
062536 012600      MOV    (SP)+,R0     ;;POP STACK INTO R0
062540 012737 062422 000024      MOV    #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
062546 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
062554 104401      TYPE $PWRMG: .WORD $POWER ;;REPORT THE POWER FAILURE
062556 062570      RTI $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
062560 000002      $SILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
062562 000000      BR     .-2         ;; BEFORE THE POWER DOWN WAS COMPLETE
062564 000776      $SAVR6: 0          ;;PUT THE SP HERE
062566 000000      $POWER: .ASCIZ <15><12>'POWER'
062570      015      012      120 .EVEN
    
```

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
062600 112737 000001 063044 $AT.1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
062606 112737 000001 063042 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
062614 000403 BR $ATYC
062616 112737 000001 063044 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
062624 $ATYC:
062624 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
062626 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
062630 105737 063042 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
062634 001450 BEQ 5$ ;;IF NOT: BR
062636 122737 000001 001242 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
062644 001031 BNE 3$ ;;IF NOT: BR
062646 132737 000100 001243 BITB #APTSPool,$ENVM ;;SHOULD SPOOL MESSAGES?
062654 001425 BEQ 3$ ;;IF NOT: JR
062656 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
062662 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
062670 005737 001222 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
062674 001375 BNE 1$ ;;IF NOT: WAIT
062676 010037 001236 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
062702 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
062704 001376 BNE 2$
062706 163700 001236 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
062712 006200 ASR R0 ;;GET MESSAGE LGTH IN WORDS
062714 010037 001240 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
062720 012737 000004 001222 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
062726 000413 BR 5$
062730 017637 000004 062754 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
062736 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
062744 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
062750 004737 056034 JSR PC,$TYPE ;;CALL TYPE MACRO
062754 000000 4$: .WORD 0
062756 5$:
062756 105737 063044 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
062762 001416 BEQ 12$ ;;IF NOT: BR
062764 005737 001242 TST $ENV ;;RUNNING UNDER APT?
062770 001413 BEQ 12$ ;;IF NOT: BR
062772 005737 001222 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
062776 001375 BNE 11$ ;;IF NOT: WAIT
063000 017637 000004 001224 MOV @4(SP),$FATAL ;;GET ERROR #
063006 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
063014 005237 001222 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
063020 105037 063044 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
063024 105037 063043 CLRB $LFLG ;;CLEAR LOG FLAG
063030 105037 063042 CLRB $MFLG ;;CLEAR MESSAGE FLAG
063034 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
063036 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
063040 000207 RTS ;;RETURN
063042 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
063043 000 $LFLG: .BYTE 0 ;;LOG FLAG
063044 000 $FFLG: .BYTE 0 ;;FATAL FLAG
.EVEN
000200 APTSIZE = 200
000001 APTENV = 001
000100 APTSPool = 100
000040 APTCSUP = 040

```



.SBTTL CONSOLE MESSAGES

2					
3	063046	075	000	EQUALS:	.ASCIZ @=@
4	063050	101	114	ALL:	.ASCIZ @ALL@<CRLF>
5	063055	040	077	040	QUES: .ASCIZ @ ? @
6	063061	054	040	000	COMMA: .ASCIZ @, @
7	063064	200	124	131	MSHELP: .ASCIZ <CRLF>@TYPE HELP TEXT (L) N ? @
8	063115	200	122	115	CNSLO1: .ASCIZ <CRLF>@RMCS1=@
9	063125	040	114	111	CNSLO2: .ASCIZ @ LIMITS - LO= 160000, HI= 17XXXX@<CRLF>
10	063167	122	115	126	CNSLO3: .ASCIZ @RMVEC=@
11	063176	040	114	111	CNSLO4: .ASCIZ @ LIMITS - LO= 0, HI= 1000@<CRLF><LF>
12	063232	200	124	131	CNSLO7: .ASCII <CRLF>@TYPE 'A' TO TEST ALL DRIVES, OR TYPE DRIVE NUMBER(S)@
13	063317	200	101	116	.ASCIZ <CRLF>@AND TERMINATE INPUT WITH A CARRIAGE RETJRN.@
14	063374	200			CNSLO8: .ASCII <CRLF>
15	063375	040	077	111	CNSLO9: .ASCIZ @ ?ILLEGAL INPUT@<CRLF>
16	063416	200	104	122	DRIVES: .ASCIZ <CRLF>/DRIVE(S) TO BE TESTED/
17	063445	116	117	116	NONE: .ASCIZ /NONE/
18	063452	200	104	122	MSDRVS: .ASCIZ <CRLF>/DRIVE(S): /
19	063466	104	122	111	MSGDRV: .ASCIZ /DRIVE/
20	063474	200	125	116	SYSTAT: .ASCIZ <CRLF>/UNIT STATUS: /
21	063512	122	115	070	SRM80: .ASCIZ /RM80/
22	063517	040	116	117	NOTRM: .ASCIZ / NOT AN RM80/
23	063534	040	114	117	LODEV: .ASCIZ / LOAD DEVICE/
24	063551	040	116	117	NOTPRS: .ASCIZ / NOT PRESENT/
25	063566	040	116	117	NOTAVL: .ASCIZ / NOT AVAILABLE/
26	063605	040	117	106	UNTOFF: .ASCIZ / OFFLINE/
27	063616	040	117	116	UNTON: .ASCIZ / ONLINE/
28	063626	116	000		N: .ASCIZ /N/
29	063630	131	000		Y: .ASCIZ /Y/
30	063632	040			BLNKS4: .ASCII / /
31	063633	040			BLNKS3: .ASCII / /
32	063634	040			BLNKS2: .ASCII / /
33	063635	040	000		BLNKS1: .ASCIZ / /
34					.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.

063640

063640 020000

FNCDTB:

.WORD OPI

:FUNCTION CODE TABLE

:NOP

58	063642	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (2)
59	063644	132000	.WORD	ATA!OPI!IVC!IAE	:SEEK
60	063646	130000	.WORD	ATA!OPI!IVC	:RECALIBRATE
61	063650	020000	.WORD	OPI	:DRIVE CLEAR
62	063652	030000	.WORD	OPI!IVC	:RELEASE
63	063654	130000	.WORD	OPI!ATA!IVC	:OFFSET
64	063656	130000	.WORD	OPI!ATA!IVC	:RETURN TO CENTERLINE
65	063660	020000	.WORD	OPI	:READ IN PRESET
66	063662	020000	.WORD	OPI	:PACK ACKNOWLEDGE
67	063664	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (24)
68	063666	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (26)
69	063670	132000	.WORD	ATA!OPI!IVC!IAE	:SEARCH
70	063672	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (32)
71	063674	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (34)
72	063676	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (36)
73	063700	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (40)
74	063702	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (42)
75	063704	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (44)
76	063706	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (46)
77	063710	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK DATA
78	063712	073300	.WORD	WCE!OPI!IVC!IAE!AOE!HCE!ECH	:WRITE CHECK HEADER AND DATA
79	063714	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (54)
80	063716	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (56)
81	063720	037200	.WORD	OPI!IVC!WLE!IAE!AOE!HCE	:WRITE DATA
82	063722	037000	.WORD	OPI!IVC!WLE!IAE!AOE	:WRITE HEADER AND DATA
83	063724	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (64)
84	063726	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (66)
85	063730	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ DATA
86	063732	033300	.WORD	OPI!IVC!IAE!AOE!HCE!ECH	:READ HEADER AND DATA
87	063734	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (74)
88	063736	130001	.WORD	OPI!ATA!ILF!IVC	:ILLEGAL FUNCTION (76)

1		
2		
3	063740	001
4	063741	002
5	063742	004
6	063743	010
7	063744	020
8	063745	040
9	063746	100
10	063747	200

.SBTTL ATTENTION (ATA) TABLE

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

Line	Address	Value	Category
1			
2			
3	063750		
4	063750		
5	063750	000000	
6	063752	C00001	
7	063754	000003	
8	063756	000007	
9	063760	000017	
10	063762	000037	
11	063764	000077	
12	063766	000177	
13	063770	000377	
14	063772	000777	
15	063774	001777	
16	063776	003777	
17	064000	007777	
18	064002	017777	
19	064004	037777	
20	064006	077777	
21	064010	177777	
22	064012	177777	
23	064014	077777	
24	064016	037777	
25	064020	017777	
26	064022	007777	
27	064024	003777	
28	064026	001777	
29	064030	000777	
30	064032	000377	
31	064034	000177	
32	064036	000077	
33	064040	000037	
34	064042	000017	
35	064044	000007	
36	064046	000003	
37	064050	000001	
38	064052	000000	
39	064054	000000	
40	064056	000001	
41	064060	000002	
42	064062	000004	
43	064064	000010	
44	064066	000020	
45	064070	000040	
46	064072	000100	
47	064074	000200	
48	064076	000400	
49	064100	001000	
50	064102	002000	
51	064104	004000	
52	064106	010000	
53	064110	020000	
54	064112	040000	
55	064114	100000	
56	064116	100000	
57	064120	040000	

.SBTTL DATA PATTERN TABLE

Category	Word Count
RGDTPT:	
MIXED:	
.WORD	0.
.WORD	1.
.WORD	3.
.WORD	7.
.WORD	15.
.WORD	31.
.WORD	63.
.WORD	127.
.WORD	255.
.WORD	511.
.WORD	1023.
.WORD	2047.
.WORD	4095.
.WORD	8191.
.WORD	16383.
.WORD	32767.
ONES:	
.WORD	65535.
.WORD	65535.
.WORD	32767.
.WORD	16383.
.WORD	8191.
.WORD	4095.
.WORD	2047.
.WORD	1023.
.WORD	511.
.WORD	255.
.WORD	127.
.WORD	63.
.WORD	31.
.WORD	15.
.WORD	7.
.WORD	3.
.WORD	1.
ZEROS:	
.WORD	0.
.WORD	0.
.WORD	1.
.WORD	2.
.WORD	4.
.WORD	8.
.WORD	16.
.WORD	32.
.WORD	64.
.WORD	128.
.WORD	256.
.WORD	512.
.WORD	1024.
.WORD	2048.
.WORD	4096.
.WORD	8192.
.WORD	16384.
.WORD	32768.
.WORD	32768.
.WORD	16384.

58	064122	020000	.WORD	8192.
59	064124	010000	.WORD	4096.
60	064126	004000	.WORD	2048.
61	064130	002000	.WORD	1024.
62	064132	001000	.WORD	512.
63	064134	000400	.WORD	256.
64	064136	000200	.WORD	128.
65	064140	000100	.WORD	64.
66	064142	000040	.WORD	32.
67	064144	000020	.WORD	16.
68	064146	000010	.WORD	8.
69	064150	000004	.WORD	4.
70	064152	000002	.WORD	2.
71	064154	000001	.WORD	1.
72	064156	000000	.WORD	0.
73	064160	177777	.WORD	65535.
74	064162	177776	.WORD	65534.
75	064164	177774	.WORD	65532.
76	064166	177770	.WORD	65528.
77	064170	177760	.WORD	65520.
78	064172	177740	.WORD	65504.
79	064174	177700	.WORD	65472.
80	064176	177600	.WORD	65408.
81	064200	177400	.WORD	65280.
82	064202	177000	.WORD	65024.
83	064204	176000	.WORD	64512.
84	064206	174000	.WORD	63488.
85	064210	170000	.WORD	61440.
86	064212	160000	.WORD	57344.
87	064214	140000	.WORD	49152.
88	064216	100000	.WORD	32768.
89	064220	000000	.WORD	0.
90	064222	000000	.WORD	0.
91	064224	100000	.WORD	32768.
92	064226	140000	.WORD	49152.
93	064230	160000	.WORD	57344.
94	064232	170000	.WORD	61440.
95	064234	174000	.WORD	63488.
96	064236	176000	.WORD	64512.
97	064240	177000	.WORD	65024.
98	064242	177400	.WORD	65280.
99	064244	177600	.WORD	65408.
100	064246	177700	.WORD	65472.
101	064250	177740	.WORD	65504.
102	064252	177760	.WORD	65520.
103	064254	177770	.WORD	65528.
104	064256	177774	.WORD	65532.
105	064260	177776	.WORD	65534.
106	064262	177777	.WORD	65535.
107	064264	125252	.WORD	43690.
108	064266	152525	.WORD	43690./2
109	064270	125252	.WORD	43690.
110	064272	177777	.WORD	65535.
111	064274	177776	.WORD	65534.
112	064276	177775	.WORD	65533.
113	064300	177773	.WORD	65531.
114	064302	177767	.WORD	65527.

115	064304	177757	.WORD	65519.
116	064306	177737	.WORD	65503.
117	064310	177677	.WORD	65471.
118	064312	177577	.WORD	65407.
119	064314	177377	.WORD	65279.
120	064316	176777	.WORD	65023.
121	064320	175777	.WORD	64511.
122	064322	173777	.WORD	63487.
123	064324	167777	.WORD	61439.
124	064326	157777	.WORD	57343.
125	064330	137777	.WORD	49151.
126	064332	077777	.WORD	32767.
127	064334	077777	.WORD	32767.
128	064336	137777	.WORD	49151.
129	064340	157777	.WORD	57343.
130	064342	167777	.WORD	61439.
131	064344	173777	.WORD	63487.
132	064346	175777	.WORD	64511.
133	064350	176777	.WORD	65023.
134	064352	177377	.WORD	65279.
135	064354	177577	.WORD	65407.
136	064356	177677	.WORD	65471.
137	064360	177737	.WORD	65503.
138	064362	177757	.WORD	65519.
139	064364	177767	.WORD	65527.
140	064366	177773	.WORD	65531.
141	064370	177775	.WORD	65533.
142	064372	177776	.WORD	65534.
143	064374	177777	.WORD	65535.
144	064376			
145				

ENRGDT:  
.EVEN

				.SBTTL ERROR MESSAGE TABLE		
1						
2						
3	064376	077527	072436	000000	EMT1:	.WORD EMS300,EMS1,0
4	064404	077545	077570	077615	EMT2:	.WORD EMS301,EMS302,EMS303,EMS1,EMS304
5	064416	102545	101754	102115		.WORD EMS511,EMS500,EMS501,EMS502,EMS503,0
6	064432	077545	077635	077570	EMT3:	.WORD EMS301,EMS306,EMS302
7	064440	102545	102272	102115		.WORD EMS511,EMS505,EMS501,EMS502,0
8	064452	077527	077570	077651	EMT4:	.WORD EMS300,EMS302,EMS307,EMS2
9	064462	102545	102142	102115		.WORD EMS511,EMS502,EMS501,EMS503,0
10	064474	077545	077712	077727	EMT5:	.WORD EMS301,EMS310,EMS311
11	064502	102545	102142	102115		.WORD EMS511,EMS502,EMS501,EMS503,EMS504
12	064514	077773	000000			.WORD EMS312,0
13	064520	077545	077635	077727	EMT6:	.WORD EMS301,EMS306,EMS311
14	064526	102545	102142	102115		.WORD EMS511,EMS502,EMS501,EMS503,EMS504,0
15	064542	077545	100031	077570	EMT7:	.WORD EMS301,EMS313,EMS302
16	064550	102545	102115	102142		.WORD EMS511,EMS501,EMS502,EMS504,EMS503,0
17	064564	100137	100160	100062	EMT10:	.WORD EMS316,EMS317,EMS314
18	064572	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
19	064602	100137	100160	100111	EMT11:	.WORD EMS316,EMS317,EMS315
20	064610	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
21	064620	100137	100200	100062	EMT12:	.WORD EMS316,EMS320,EMS314
22	064626	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
23	064636	100137	100200	100111	EMT13:	.WORD EMS316,EMS320,EMS315
24	064644	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
25	064654	100137	100220	100062	EMT14:	.WORD EMS316,EMS321,EMS314
26	064662	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
27	064672	100137	100220	100111	EMT15:	.WORD EMS316,EMS321,EMS315
28	064700	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
29	064710	100137	100240	100062	EMT16:	.WORD EMS316,EMS322,EMS314
30	064716	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
31	064726	100137	100240	100111	EMT17:	.WORD EMS316,EMS322,EMS315
32	064734	102545	102115	102142		.WORD EMS511,EMS501,EMS502,0
33	064744	077545	100277	077015	EMT20:	.WORD EMS301,EMS324,EMS250
34	064752	102545	102237			.WORD EMS511,EMS504
35	064756	077773	100330	000000		.WORD EMS312,EMS326,0
36	064764	077545	100260	077015	EMT21:	.WORD EMS301,EMS323,EMS250
37	064772	102545	102237			.WORD EMS511,EMS504
38	064776	077773	100317	000000		.WORD EMS312,EMS325,0
39	065004	077545	100031	077015	EMT22:	.WORD EMS301,EMS313,EMS250
40	065012	102545	102237	000000		.WORD EMS511,EMS504,0
41	065020	077545	100277	077053	EMT23:	.WORD EMS301,EMS324,EMS251
42	065026	102545	102115			.WORD EMS511,EMS501
43	065032	077773	100330	000000		.WORD EMS312,EMS326,0
44	065040	077545	100260	077053	EMT24:	.WORD EMS301,EMS323,EMS251
45	065046	102545	102115			.WORD EMS511,EMS501
46	065052	077773	100317	000000		.WORD EMS312,EMS325,0
47	065060	077545	100031	077053	EMT25:	.WORD EMS301,EMS313,EMS251
48	065066	102545	102115	000000		.WORD EMS511,EMS501,0
49	065074	077545	077635	077117	EMT26:	.WORD EMS301,EMS306,EMS252,EMS253,EMS327,EMS254
50	065110	102545	102212	102115		.WORD EMS511,EMS503,EMS501,EMS502
51	065120	100346	100062	000000		.WORD EMS330,EMS314,0
52	065126	077527	077117		EMT27:	.WORD EMS300,EMS252
53	065132	102545	102115	000000		.WORD EMS511,EMS501,0
54	065140	077527	077117		EMT30:	.WORD EMS300,EMS252
55	065144	102545	102115	102237		.WORD EMS511,EMS501,EMS504,0
56	065154	077527	077117		EMT31:	.WORD EMS300,EMS252
57	065160	102545	102115	102212		.WORD EMS511,EMS501,EMS503,0



58	065170	077545	100277	077117	EMT32:	.WORD	EMS301,EMS324,EMS252
59	065176	102545	102115	000000		.WORD	EMS511,EMS501,0
60	065204	077545	100277	077117	EMT33:	.WORD	EMS301,EMS324,EMS252
61	065212	102545	102115	102237		.WORD	EMS511,EMS501,EMS504,0
62	065222	077545	100277	077117	EMT34:	.WORD	EMS301,EMS324,EMS252
63	065230	102545	102115	102212		.WORD	EMS511,EMS501,EMS503,0
64	065240	077545	100260	077117	EMT35:	.WORD	EMS301,EMS323,EMS252
65	065246	102545	102115	000000		.WORD	EMS511,EMS501,0
66	065254	077545	100031	077117	EMT36:	.WORD	EMS301,EMS313,EMS252
67	065262	102545	102115	000000		.WORD	EMS511,EMS501,0
68	065270	077545	100277	077246	EMT37:	.WORD	EMS301,EMS324,EMS255
69	065276	077773	100330	000000		.WORD	EMS312,EMS326,0
70	065304	077545	100260	077246	EMT40:	.WORD	EMS301,EMS323,EMS255
71	065312	102545	102237			.WORD	EMS511,EMS504
72	065316	077773	100317	000000		.WORD	EMS312,EMS325,0
73	065324	077545	100031	077246	EMT41:	.WORD	EMS301,EMS313,EMS255
74	065332	102545	102237	000000		.WORD	EMS511,EMS504,0
75	065340	077545	100260	077015	EMT42:	.WORD	EMS301,EMS323,EMS250,EMS327,EMS255
76	065352	102545	102237	102115		.WORD	EMS511,EMS504,EMS501,EMS503,0
77	065364	077527	072556		EMT43:	.WORD	EMS300,EMS3
78	065370	102545	102115	000000		.WORD	EMS511,EMS501,0
79	065376	100365	072623	077152	EMT44:	.WORD	EMS331,EMS4,EMS253
80	065404	102545	102115	000000		.WORD	EMS511,EMS501,0
81	065412	077527	077152		EMT45:	.WORD	EMS300,EMS253
82	065416	102545	102115	102212		.WORD	EMS511,EMS501,EMS503,0
83	065426	077527	077152		EMT46:	.WORD	EMS300,EMS253
84	065432	102545	102115	102237		.WORD	EMS511,EMS501,EMS504,0
85	065442	077545	100277	077152	EMT47:	.WORD	EMS301,EMS324,EMS253
86	065450	102545	102115	102212		.WORD	EMS511,EMS501,EMS503
87	065456	077773	100330	000000		.WORD	EMS312,EMS326,0
88	065464	077545	100277	077152	EMT50:	.WORD	EMS301,EMS324,EMS253
89	065472	102545	102115	102237		.WORD	EMS511,EMS501,EMS504
90	065500	077773	100330	000000		.WORD	EMS312,EMS326,0
91	065506	077545	100260	077152	EMT51:	.WORD	EMS301,EMS323,EMS253
92	065514	102545	102115			.WORD	EMS511,EMS501
93	065520	077773	100317	000000		.WORD	EMS312,EMS325,0
94	065526	077545	100031	077152	EMT52:	.WORD	EMS301,EMS313,EMS253
95	065534	102545	102115	000000		.WORD	EMS511,EMS501,0
96	065542	100365	072623	077310	EMT53:	.WORD	EMS331,EMS4,EMS256
97	065550	102545	102115	000000		.WORD	EMS511,EMS501,0
98	065556	077545	100277	077310	EMT54:	.WORD	EMS301,EMS324,EMS256
99	065564	102545	102115			.WORD	EMS511,EMS501
100	065570	077773	100330	000000		.WORD	EMS312,EMS326,0
101	065576	077545	100260	077310	EMT55:	.WORD	EMS301,EMS323,EMS256
102	065604	102545	102115			.WORD	EMS511,EMS501
103	065610	077773	100317	000000		.WORD	EMS312,EMS325,0
104	065616	077545	100031	077310	EMT56:	.WORD	EMS301,EMS313,EMS256
105	065624	102545	102115	000000		.WORD	EMS511,EMS501,0
106	065632	077340	100415		EMT57:	.WORD	EMS257,EMS332
107	065636	102545	102345	102115		.WORD	EMS511,EMS506,EMS501,0
108	065646	072654	100433		EMT60:	.WORD	EMS5,EMS333
109	065652	102545	102402	102115		.WORD	EMS511,EMS507,EMS501,0
110	065662	077545	100277	077374	EMT61:	.WORD	EMS301,EMS324,EMS260
111	065670	102545	102115			.WORD	EMS511,EMS501
112	065674	077773	100330	000000		.WORD	EMS312,EMS326,0
113	065702	077545	100260	077374	EMT62:	.WORD	EMS301,EMS323,EMS260
114	065710	102545	102115			.WORD	EMS511,EMS501

115	065714	077773	100317	000000		.WORD	EMS312,EMS325,0
116	065722	077545	100031	077374	EMT63:	.WORD	EMS301,EMS313,EMS260
117	065730	102545	102115	000000		.WORD	EMS511,EMS501,0
118	065736	077527	073175		EMT64:	.WORD	EMS300,EMS12
119	065742	102545	102115	000000		.WORD	EMS511,EMS501,0
120	065750	073175	100457	100555	EMT65:	.WORD	EMS12,EMS335,EMS342
121	065756	102545	102115	000000		.WORD	EMS511,EMS501,0
122	065764	073175	100502	100555	EMT66:	.WORD	EMS12,EMS336,EMS342
123	065772	102545	102115	000000		.WORD	EMS511,EMS501,0
124	066000	077527	072724		EMT67:	.WORD	EMS300,EMS6
125	066004	102545	102115			.WORD	EMS511,EMS501
126	066010	072770	100433	000000		.WORD	EMS7,EMS333,0
127	066016	077527	072724	100340	EMT70:	.WORD	EMS300,EMS6,EMS327,EMS7
128	066026	102545	102115	102237		.WORD	EMS511,EMS501,EMS504,0
129	066036	072724	100457	100535	EMT71:	.WORD	EMS6,EMS335,EMS340,EMS10,EMS333,EMS342
130	066052	102545	102115	102142		.WORD	EMS511,EMS501,EMS502,0
131	066062	072724	100502	100535	EMT72:	.WORD	EMS6,EMS336,EMS340,EMS10,EMS334,EMS342
132	066076	102545	102115	102142		.WORD	EMS511,EMS501,EMS502,0
133	066106	077545	077374	077615	EMT73:	.WORD	EMS301,EMS260,EMS303,EMS11
134	066116	102545	102115	102142		.WORD	EMS511,EMS501,EMS502,0
135	066126	100607	100623	100555	EMT74:	.WORD	EMS343,EMS344,EMS342,0
136	066136	077527	073253		EMT75:	.WORD	EMS300,EMS13
137	066142	102545	102212	000000		.WORD	EMS511,EMS503,0
138	066150	100700	073253	100652	EMT76:	.WORD	EMS346,EMS13,EMS345
139	066156	102545	102212	000000		.WORD	EMS511,EMS503,0
140	066164	100521	073253	100652	EMT77:	.WORD	EMS337,EMS13,EMS345
141	066172	102545	102212	000000		.WORD	EMS511,EMS503,0
142	066200	077545	100031	077205	EMT100:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS13
143	066212	102545	102212	000000		.WORD	EMS511,EMS503,0
144	066220	100700	073322	100546	EMT101:	.WORD	EMS346,EMS14,EMS341,EMS15
145	066230	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
146	066240	100521	073322	100546	EMT102:	.WORD	EMS337,EMS14,EMS341,EMS15
147	066250	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
148	066260	077545	100031	077205	EMT103:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS15
149	066272	102545	102212			.WORD	EMS511,EMS503
150	066276	073322	100415	000000		.WORD	EMS14,EMS332,0
151	066304	100700	073526	100546	EMT104:	.WORD	EMS346,EMS17,EMS341,EMS16
152	066314	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
153	066324	100521	073526	100546	EMT105:	.WORD	EMS337,EMS17,EMS341,EMS16
154	066334	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
155	066344	077545	100031	077205	EMT106:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS16
156	066356	102545	102212			.WORD	EMS511,EMS503
157	066362	073526	100415	000000		.WORD	EMS17,EMS332,0
158	066370	100700	073567	100546	EMT107:	.WORD	EMS346,EMS20,EMS341,EMS21
159	066400	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
160	066410	073567	100744	073633	EMT110:	.WORD	EMS20,EMS351,EMS21,EMS350,EMS22,EMS315
161	066424	102545	102115	000000		.WORD	EMS511,EMS501,0
162	066432	073567	100447	100737	EMT111:	.WORD	EMS20,EMS334,EMS350,EMS22,EMS333
163	066444	102545	102115	000000		.WORD	EMS511,EMS501,0
164	066452	100521	073567	100546	EMT112:	.WORD	EMS337,EMS20,EMS341,EMS21
165	066462	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
166	066472	100521	073567	100546	EMT113:	.WORD	EMS337,EMS20,EMS341,EMS21,EMS350,EMS22,EMS334
167	066510	102545	102115	000000		.WORD	EMS511,EMS501,0
168	066516	073567	100762	073633	EMT114:	.WORD	EMS20,EMS352,EMS21,EMS350,EMS22,EMS333
169	066532	102545	102115	000000		.WORD	EMS511,EMS501,0
170	066540	077545	100031	077205	EMT115:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS21
171	066552	102545	102212			.WORD	EMS511,EMS503

172	066556	073567	100415	000000		.WORD	EMS20,EMS332,0
173	066564	100700	073757	100546	EMT116:	.WORD	EMS346,EMS23,EMS341,EMS24
174	066574	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
175	066604	100521	073757	100546	EMT117:	.WORD	EMS337,EMS23,EMS341,EMS24
176	066614	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
177	066624	077545	100031	077205	EMT120:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS24
178	066636	102545	102212			.WORD	EMS511,EMS503
179	066642	073757	100415	000000		.WORD	EMS23,EMS332,0
180	066650	100700	074114	100546	EMT121:	.WORD	EMS346,EMS25,EMS341,EMS26
181	066660	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
182	066670	100521	074114	100546	EMT122:	.WORD	EMS337,EMS25,EMS341,EMS26
183	066700	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
184	066710	077545	100031	077205	EMT123:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS26
185	066722	102545	102212			.WORD	EMS511,EMS503
186	066726	074114	100415	000000		.WORD	EMS25,EMS332,0
187	066734	077527	074251	077651	EMT124:	.WORD	EMS300,EMS27,EMS307,EMS2
188	066744	102545	102212	000000		.WORD	EMS511,EMS503,0
189	066752	100365	074251	100775	EMT125:	.WORD	EMS331,EMS27,EMS353
190	066760	102545	102212			.WORD	EMS511,EMS503
191	066764	074315	100111	000000		.WORD	EMS30,EMS315,0
192	066772	100521	074251	100546	EMT126:	.WORD	EMS337,EMS27,EMS341,EMS30
193	067002	102545	102212	000000		.WORD	EMS511,EMS503,0
194	067010	077545	100031	077205	EMT127:	.WORD	EMS301,EMS313,EMS254,EMS347,EMS30
195	067022	102545	102212			.WORD	EMS511,EMS503
196	067026	074251	100415	000000		.WORD	EMS27,EMS332,0
197	067034	074375	101022	077015	EMT130:	.WORD	EMS31,EMS354,EMS250
198	067042	102545	102237	102212		.WORD	EMS511,EMS504,EMS503,0
199	067052	074446	101022	077015	EMT131:	.WORD	EMS32,EMS354,EMS250
200	067060	102545	102237	102212		.WORD	EMS511,EMS504,EMS503,0
201	067070	101055	074526	077015	EMT132:	.WORD	EMS355,EMS33,EMS250,EMS341,EMS30
202	067102	102545	102237	000000		.WORD	EMS511,EMS504,0
203	067110	101055	074556	077015	EMT133:	.WORD	EMS355,EMS34,EMS250,EMS341,EMS30
204	067122	102545	102237	000000		.WORD	EMS511,EMS504,0
205	067130	100365	072623	077246	EMT134:	.WORD	EMS331,EMS4,EMS255
206	067136	102545	102237	000000		.WORD	EMS511,EMS504,0
207	067144	074605	101117	101141	EMT135:	.WORD	EMS35,EMS357,EMS360,EMS15
208	067154	102545	102115	000000		.WORD	EMS511,EMS501,0
209	067162	077425	101215		EMT136:	.WORD	EMS261,EMS362
210	067166	102545	102212	000000		.WORD	EMS511,EMS503,0
211	067174	077527	074647	077651	EMT137:	.WORD	EMS300,EMS36,EMS307,EMS2
212	067204	102545	102115	000000		.WORD	EMS511,EMS501,0
213	067212	101055	074677	077246	EMT140:	.WORD	EMS355,EMS37,EMS255,EMS341,EMS30
214	067224	102545	102237	000000		.WORD	EMS511,EMS504,0
215	067232	100700	074730	100652	EMT141:	.WORD	EMS346,EMS40,EMS345
216	067240	102545	102237	000000		.WORD	EMS511,EMS504,0
217	067246	100521	074730	100546	EMT142:	.WORD	EMS337,EMS40,EMS341,EMS30
218	067256	102545	102237	000000		.WORD	EMS511,EMS504,0
219	067264	101236	077712	075006	EMT143:	.WORD	EMS363,EMS310,EMS41
220	067272	102545	102115	000000		.WORD	EMS511,EMS501,0
221	067300	100521	075006	100546	EMT144:	.WORD	EMS337,EMS41,EMS341,EMS252,EMS327,EMS253
222	067314	102545	102115	000000		.WORD	EMS511,EMS501,0
223	067322	075006	101022	101253	EMT145:	.WORD	EMS41,EMS354,EMS364,EMS252,EMS365,EMS253
224	067336	102545	102115	000000		.WORD	EMS511,EMS501,0
225	067344	077545	077635	074647	EMT146:	.WORD	EMS301,EMS306,EMS36
226	067352	102545	102115	102212		.WORD	EMS511,EMS501,EMS503,0
227	067362	101301	075054		EMT147:	.WORD	EMS366,EMS42
228	067366	102545	102212	000000		.WORD	EMS511,EMS503,0

229	067374	101324	100776	101274	EMT150:	.WORD	EMS367,EMS353,EMS365,EMS42,EMS354,EMS3
230	067410	102545	102212	000000		.WORD	EMS511,EMS503,0
231	067416	100521	074647		EMT151:	.WORD	EMS337,EMS36
232	067422	102545	102115	000000		.WORD	EMS511,EMS501,0
233	067430	075136	101022	074647	EMT152:	.WORD	EMS43,EMS354,EMS36
234	067436	102545	102115	000000		.WORD	EMS511,EMS501,0
235	067444	101324	100776	101274	EMT153:	.WORD	EMS367,EMS353,EMS365,EMS36,EMS370
236	067456	102545	102212	000000		.WORD	EMS511,EMS503,0
237	067464	101324	100776	101274	EMT154:	.WORD	EMS367,EMS353,EMS365,EMS36,EMS371
238	067476	102545	102212	000000		.WORD	EMS511,EMS503,0
239	067504	077527	075207	077651	EMT155:	.WORD	EMS300,EMS44,EMS307,EMS2
240	067514	102545	102212	000000		.WORD	EMS511,EMS503,0
241	067522	101324	100776	101274	EMT156:	.WORD	EMS367,EMS353,EMS365,EMS44,EMS354,EMS3
242	067536	102545	102212	000000		.WORD	EMS511,EMS503,0
243	067544	077527	075250	077651	EMT157:	.WORD	EMS300,EMS45,EMS307,EMS2
244	067554	102545	102212	000000		.WORD	EMS511,EMS503,0
245	067562	101324	100776	101274	EMT160:	.WORD	EMS367,EMS353,EMS365,EMS45,EMS354,EMS3
246	067576	102545	102212	102115		.WORD	EMS511,EMS503,EMS501
247	067604	074605	100433	000000		.WORD	EMS35,EMS333,0
248	067612	077527	075325	077651	EMT161:	.WORD	EMS300,EMS46,EMS307,EMS2
249	067622	102545	102212	000000		.WORD	EMS511,EMS503,0
250	067630	100521	075325	100776	EMT162:	.WORD	EMS337,EMS46,EMS353
251	067636	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
252	067646	074605	100457	100521	EMT163:	.WORD	EMS35,EMS335,EMS337,EMS41,EMS334,EMS372
253	067662	102545	102115	000000		.WORD	EMS511,EMS501,0
254	067670	075407	100457	100521	EMT164:	.WORD	EMS47,EMS335,EMS337,EMS41,EMS335,EMS372
255	067704	102545	102115	000000		.WORD	EMS511,EMS501,0
256	067712	100521	074605	100340	EMT165:	.WORD	EMS337,EMS35,EMS327,EMS47
257	067722	102545	102115			.WORD	EMS511,EMS501
258	067726	075006	100433	101440		.WORD	EMS41,EMS333,EMS372,0
259	067736	077527	075407	077651	EMT166:	.WORD	EMS300,EMS47,EMS307,EMS2
260	067746	102545	102115	102212		.WORD	EMS511,EMS501,EMS503,0
261	067756	075447	100457	100535	EMT167:	.WORD	EMS50,EMS335,EMS340,EMS36,EMS333
262	067770	102545	102115	102212		.WORD	EMS511,EMS501,EMS503,0
263	070000	100521	074605		EMT170:	.WORD	EMS337,EMS35
264	070004	102545	102115	000000		.WORD	EMS511,EMS501,0
265	070012	075447	074556	072556	EMT171:	.WORD	EMS50,EMS34,EMS3
266	070020	102545	102115	000000		.WORD	EMS511,EMS501,0
267	070026	077545	077635	075516	EMT172:	.WORD	EMS301,EMS306,EMS51
268	070034	102545	102115	102237		.WORD	EMS511,EMS501,EMS504,0
269	070044	101324	100776	101274	EMT173:	.WORD	EMS367,EMS353,EMS365,EMS47,EMS354,EMS3
270	070060	102545	102115	000000		.WORD	EMS511,EMS501,0
271	070066	077527	077015	100340	EMT174:	.WORD	EMS300,EMS250,EMS327,EMS255,EMS327,EMS256
272	070102	100546	102644			.WORD	EMS341,EMS600
273	070106	102545	102115	102237		.WORD	EMS511,EMS501,EMS504,0
274	070116	077527	077310	100546	EMT175:	.WORD	EMS300,EMS256,EMS341,EMS600
275	070126	102545	102115	102237		.WORD	EMS511,EMS501,EMS504,0
276	070136	077527	077015	100546	EMT176:	.WORD	EMS300,EMS250,EMS341,EMS600
277	070146	102545	102237	000000		.WORD	EMS511,EMS504,0
278	070154	000000			EMT177:	.WORD	
279	070156	100521	075565	100546	EMT200:	.WORD	EMS337,EMS52,EMS341,EMS601
280	070166	102545	102115	000000		.WORD	EMS511,EMS501,0
281	070174	100700	075565	100546	EMT201:	.WORD	EMS346,EMS52,EMS341,EMS602
282	070204	102545	102115	000000		.WORD	EMS511,EMS501,0
283	070212	100700	075516	100546	EMT202:	.WORD	EMS346,EMS51,EMS341,EMS602
284	070222	102545	102115	000000		.WORD	EMS511,EMS501,0
285	070230	100700	075565	100652	EMT203:	.WORD	EMS346,EMS52,EMS345,EMS373,EMS255

286	070242	102545	102237	102115	.WORD	EMS511,EMS504,EMS501,0
287	070252	100700	075565	100546	EMT204: .WORD	EMS346,EMS52,EMS341,EMS27
288	070262	102545	102237	102115	.WORD	EMS511,EMS504,EMS501,0
289	070272	075626	101022	072556	EMT205: .WORD	EMS53,EMS354,EMS3
290	070300	102545	102212	102142	.WORD	EMS511,EMS503,EMS502,EMS510,0
291	070312	100521	075667	101476	EMT206: .WORD	EMS337,EMS54,EMS374,EMS250,EMS327,EMS255
292	070326	100340	072556		.WORD	EMS327,EMS3
293	070332	102545	102237	102115	.WORD	EMS511,EMS504,EMS501,0
294	070342	075667	101022	072556	EMT207: .WORD	EMS54,EMS354,EMS3
295	070350	102545	102115	000000	.WORD	EMS511,EMS501,0
296	070356	075667	101022	101253	EMT210: .WORD	EMS54,EMS354,EMS364,EMS250
297	070366	102545	102237	000000	.WORD	EMS511,EMS504,0
298	070374	075667	101022	101253	EMT211: .WORD	EMS54,EMS354,EMS364,EMS255
299	070404	102545	102237	000000	.WORD	EMS511,EMS504,0
300	070412	100521	075744		EMT212: .WORD	EMS337,EMS55
301	070416	102545	102237	000000	.WORD	EMS511,EMS504,0
302	070424	076022	100447	100652	EMT213: .WORD	EMS56,EMS334,EMS345,EMS373,EMS262,EMS327,EMS251
303	070442	102545	102115		.WORD	EMS511,EMS501
304	070446	076022	100457	000000	.WORD	EMS56,EMS335,0
305	070454	100521	076022		EMT214: .WORD	EMS337,EMS56
306	070460	102545	102115	000000	.WORD	EMS511,EMS501,0
307	070466	076115	100433	100737	EMT215: .WORD	EMS57,EMS333,EMS350,EMS60,EMS334
308	070500	102545	102115	000000	.WORD	EMS511,EMS501,0
309	070506	101236	077635	072654	EMT216: .WORD	EMS363,EMS306,EMS5
310	070514	102545	102115	000000	.WORD	EMS511,EMS501,0
311	070522	101236	077635	076253	EMT217: .WORD	EMS363,EMS306,EMS61
312	070530	102545	102115	000000	.WORD	EMS511,EMS501,0
313	070536	076326	100433	101527	EMT220: .WORD	EMS62,EMS333,EMS375,EMS251
314	070546	102545	102115	000000	.WORD	EMS511,EMS501,0
315	070554	076326	100433	101543	EMT221: .WORD	EMS62,EMS333,EMS376,EMS262
316	070564	102545	102115	000000	.WORD	EMS511,EMS501,0
317	070572	076326	100433	101543	EMT222: .WORD	EMS62,EMS333,EMS376,EMS250
318	070602	102545	102115	000000	.WORD	EMS511,EMS501,0
319	070610	100700	076326	100546	EMT223: .WORD	EMS346,EMS62,EMS341,EMS603
320	070620	102545	102115	000000	.WORD	EMS511,EMS501,0
321	070626	100700	076407	101543	EMT224: .WORD	EMS346,EMS63,EMS376,EMS262
322	070636	102545	102115	102212	.WORD	EMS511,EMS501,EMS503,0
323	070646	076407	100433	100737	EMT225: .WORD	EMS63,EMS333,EMS350,EMS363,EMS310,EMS262
324	070662	102545	102115	000000	.WORD	EMS511,EMS501,0
325	070670	076407	101117	074647	EMT226: .WORD	EMS63,EMS357,EMS36,EMS372
326	070700	102545	102115	000000	.WORD	EMS511,EMS501,0
327	070706	076407	101077	101141	EMT227: .WORD	EMS63,EMS356,EMS360,EMS15
328	070716	102545	102115	000000	.WORD	EMS511,EMS501,0
329	070724	076407	101077	101167	EMT230: .WORD	EMS63,EMS356,EMS361,EMS15
330	070734	102545	102115	000000	.WORD	EMS511,EMS501,0
331	070742	076407	101077	075006	EMT231: .WORD	EMS63,EMS356,EMS41
332	070750	102545	102115	000000	.WORD	EMS511,EMS501,0
333	070756	075006	101557	100555	EMT232: .WORD	EMS41,EMS377,EMS342,EMS365,EMS63,EMS332
334	070772	102545	102115	000000	.WORD	EMS511,EMS501,0
335	071000	101324	100776	101274	EMT233: .WORD	EMS367,EMS353,EMS365,EMS63,EMS401
336	071012	102545	102212	102115	.WORD	EMS511,EMS503,EMS501,0
337	071022	073445	101557	101440	EMT234: .WORD	EMS16,EMS377,EMS372,EMS365,EMS64,EMS354,EMS3
338	071040	102545	102212	102115	.WORD	EMS511,EMS503,EMS501,0
339	071050	077527	076517	077651	EMT235: .WORD	EMS300,EMS65,EMS307,EMS2
340	071060	102545	102142	102212	.WORD	EMS511,EMS502,EMS503,0
341	071070	076022	101557	101543	EMT236: .WORD	EMS56,EMS377,EMS376,EMS252,EMS372,EMS350
342	071104	076517	101605		.WORD	EMS65,EMS401

ERROR MESSAGE TABLE

343	071110	102545	102142	102212		.WORD	EMS511,EMS502,EMS503,EMS501,0
344	071122	077527	076560	077651	EMT237:	.WORD	EMS300,EMS66,EMS307,EMS2
345	071132	102545	102115	102212		.WORD	EMS511,EMS501,EMS503,0
346	071142	073367	101570	101440	EMT240:	.WORD	EMS15,EMS400,EMS372,EMS350,EMS66,EMS401
347	071156	102545	102212	102115		.WORD	EMS511,EMS503,EMS501,0
348	071166	076560	100502	100535	EMT241:	.WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS604
349	071204	102545	102212	000000		.WORD	EMS511,EMS503,0
350	071212	101631	102776	101622	EMT242:	.WORD	EMS403,EMS604,EMS402,EMS21,EMS377
351	071224	102545	102212			.WORD	EMS511,EMS503
352	071231	074647	100457	000000		.WORD	EMS36,EMS335,0
353	071236	076560	100502	100535	EMT243:	.WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS604
354	071254	102545	102212	000000		.WORD	EMS511,EMS503,0
355	071262	076407	101605	101711	EMT244:	.WORD	EMS63,EMS401,EMS405,EMS604
356	071272	102545	102212	000000		.WORD	EMS511,EMS503,0
357	071300	074647	101374	101711	EMT245:	.WORD	EMS36,EMS370,EMS405,EMS604
358	071310	102545	102212	000000		.WORD	EMS511,EMS503,0
359	071316	101631	102776	101622	EMT246:	.WORD	EMS403,EMS604,EMS402,EMS24,EMS377
360	071330	102545	102212			.WORD	EMS511,EMS503
361	071334	074647	100457	000000		.WORD	EMS36,EMS335,0
362	071342	076635	100415	101711	EMT247:	.WORD	EMS67,EMS332,EMS405,EMS604
363	071352	102545	102212	000000		.WORD	EMS511,EMS503,0
364	071360	076560	100502	100535	EMT250:	.WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS605
365	071376	102545	102212	000000		.WORD	EMS511,EMS503,0
366	071404	101631	103023	101622	EMT251:	.WORD	EMS403,EMS605,EMS402,EMS21,EMS377
367	071416	102545	102212			.WORD	EMS511,EMS503
368	071422	074647	100457	000000		.WORD	EMS36,EMS335,0
369	071430	076560	100502	100535	EMT252:	.WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS605
370	071446	102545	102212	000000		.WORD	EMS511,EMS503,0
371	071454	076407	101605	101711	EMT253:	.WORD	EMS63,EMS401,EMS405,EMS605
372	071464	102545	102212	000000		.WORD	EMS511,EMS503,0
373	071472	074647	101374	101711	EMT254:	.WORD	EMS36,EMS370,EMS405,EMS605
374	071502	102545	102212	000000		.WORD	EMS511,EMS503,0
375	071510	101631	103023	101622	EMT255:	.WORD	EMS403,EMS605,EMS402,EMS24,EMS377
376	071522	102545	102212			.WORD	EMS511,EMS503
377	071526	074647	100457	000000		.WORD	EMS36,EMS335,0
378	071534	076635	100415	101711	EMT256:	.WORD	EMS67,EMS332,EMS405,EMS605
379	071544	102545	102212	000000		.WORD	EMS511,EMS503,0
380	071552	076560	100502	100535	EMT257:	.WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS606
381	071570	102545	102212	000000		.WORD	EMS511,EMS503,0
382	071576	101631	103041	101622	EMT260:	.WORD	EMS403,EMS606,EMS402,EMS21,EMS377
383	071610	102545	102212			.WORD	EMS511,EMS503
384	071614	074647	100457	000000		.WORD	EMS36,EMS335,0
385	071622	076560	100502	100535	EMT261:	.WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS606
386	071640	102545	102212	000000		.WORD	EMS511,EMS503,0
387	071646	076407	101605	101711	EMT262:	.WORD	EMS63,EMS401,EMS405,EMS606
388	071656	102545	102212	000000		.WORD	EMS511,EMS503,0
389	071664	074647	101374	101711	EMT263:	.WORD	EMS36,EMS370,EMS405,EMS606
390	071674	102545	102212	000000		.WORD	EMS511,EMS503,0
391	071702	101631	103041	101622	EMT264:	.WORD	EMS403,EMS606,EMS402,EMS24,EMS377
392	071714	102545	102212			.WORD	EMS511,EMS503
393	071720	074647	100457	000000		.WORD	EMS36,EMS335,0
394	071726	076747	101605	101711	EMT265:	.WORD	EMS70,EMS401,EMS405,EMS606
395	071736	102545	102212	000000		.WORD	EMS511,EMS503,0
396	071744	076635	100415	101711	EMT266:	.WORD	EMS67,EMS332,EMS405,EMS606
397	071754	102545	102212	000000		.WORD	EMS511,EMS503,0
398	071762	076560	101077	101734	EMT267:	.WORD	EMS66,EMS356,EMS407
399	071770	102545	102212	000000		.WORD	EMS511,EMS503,0

400	071776	076560	100502	100535	EMT270:	.WORD	EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS607
401	072014	102545	102212	000000		.WORD	EMS511,EMS503,0
402	072022	101631	103061	101622	EMT271:	.WORD	EMS403,EMS607,EMS402,EMS21,EMS377
403	072034	102545	102212			.WORD	EMS511,EMS503
404	072040	074251	100502	000000		.WORD	EMS27,EMS336,0
405	072046	074251	101374	101711	EMT272:	.WORD	EMS27,EMS370,EMS405,EMS607
406	072056	102545	102212	000000		.WORD	EMS511,EMS503,0
407	072064	074251	101411	101711	EMT273:	.WORD	EMS27,EMS371,EMS405,EMS607
408	072074	102545	102212	000000		.WORD	EMS511,EMS503,0
409	072102	074647	101672	101711	EMT274:	.WORD	EMS36,EMS404,EMS405,EMS607
410	072112	102545	102212	000000		.WORD	EMS511,EMS503,0
411	072120	075626	101605	101711	EMT275:	.WORD	EMS53,EMS401,EMS405,EMS607
412	072130	102545	102212	102142		.WORD	EMS511,EMS503,EMS502,0
413	072140	076635	100415	101711	EMT276:	.WORD	EMS67,EMS332,EMS405,EMS607
414	072150	102545	102212	000000		.WORD	EMS511,EMS503,0
415	072156	076560	100502	100535	EMT277:	.WORD	EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS607
416	072174	102545	102212	000000		.WORD	EMS511,EMS503,0
417	072202	101631	103061	101622	EMT300:	.WORD	EMS403,EMS607,EMS402,EMS24,EMS377
418	072214	102545	102212			.WORD	EMS511,EMS503
419	072220	074251	100502	000000		.WORD	EMS27,EMS336,0
420	072226	076747	101605	101711	EMT301:	.WORD	EMS70,EMS401,EMS405,EMS607
421	072236	102545	102212	000000		.WORD	EMS511,EMS503,0
422	072244	077527	077246	100546	EMT302:	.WORD	EMS300,EMS255,EMS341,EMS600
423	072254	102545	102237	000000		.WORD	EMS511,EMS504,0

1	072262	103077	000000	EHT1:	.WORD	EH1,0
2	072266	103155	000000	EHT2:	.WORD	EH2,0
3	072272	103172	000000	EHT5:	.WORD	EH5,0
4	072276	103230	000000	EHT7:	.WORD	EH7,0
5	072302	103247	000000	EHT57:	.WORD	EH57,0
6	072306	103305	000000	EHT65:	.WORD	EH65,0
7	072312	103361	000000	EHT71:	.WORD	EH71,0
8	072316	103163	000000	EHT74:	.WORD	EH3,0
9	072322	103436	000000	EHT115:	.WORD	EH115,0
10	072326	103533	000000	EHT130:	.WORD	EH130,0
11	072332	103651	000000	EHT132:	.WORD	EH132,0
12	072336	103747	000000	EHT142:	.WORD	EH142,0
13	072342	104045	000000	EHT145:	.WORD	EH145,0
14	072346	104162	000000	EHT150:	.WORD	EH150,0
15	072352	104260	000000	EHT213:	.WORD	EH213,0
16	072356	104355	000000	EHT220:	.WORD	EH220,0



1	072362	104414
2	072364	104424
3	072366	104430
4	072370	104436
5	072372	104444
6	072374	104454
7	072376	104424
8	072400	104464
9	072402	104476
10	072404	104464
11	072406	104512

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

1	072410	104516
2	072412	104521
3	072414	104522
4	072416	104524
5	072420	104516
6	072422	104516
7	072424	104521
8	072426	104524
9	072430	104530
10	072432	104524
11	072434	104522

EFT1:	.WORD	EF1
EFT2:	.WORD	EF2
EFT5:	.WORD	EF5
EFT57:	.WORD	EF57
EFT65:	.WORD	EF1
EFT71:	.WORD	EF1
EFT74:	.WORD	EF2
EFT115:	.WORD	EF57
EFT130:	.WORD	EF130
EFT132:	.WORD	EF57
EFT220:	.WORD	EF5

				.SBTTL	ERROR MESSAGE STRINGS		
1							
2							
3	072436	116	117	116	EMS1:	.ASCIZ	@NONEXISTENT DEVICE 'NED' (RMCS2,BIT 12) a
4	072507	103	117	116	EMS2:	.ASCIZ	@CONTROLLER CLEAR 'CLR' (RMCS2,BIT 05) a
5	072556	106	125	116	EMS3:	.ASCIZ	@FUNCTION CODE (RMCS1, BITS 01 - 05) a
6	072623	125	116	125	EMS4:	.ASCIZ	@MUSED BIT POSITIONS OF a
7	072654	104	105	126	EMS5:	.ASCIZ	@DEVICE AVAILABLE 'DVA' (RMCS1, BIT 11) a
8	072724	120	101	122	EMS6:	.ASCIZ	@PARTY ERROR 'PAR' (RMER1, BIT 03) a
9	072770	104	101	124	EMS7:	.ASCIZ	@DATA PARITY ERROR 'DPE' (RMER2, BIT 03) a
10	073041	120	101	122	EMS10:	.ASCIZ	@PARITY TEST 'PAT' (RMCS2, BIT 04) a
11	073104	115	101	123	EMS11:	.ASCII	@MASSBUS CONTROL BUS PARITY ERROR 'MCPE' a
12	073154	050	122	115		.ASCIZ	a(RMCS1, BIT 13) a
13	073175	111	114	114	EMS12:	.ASCIZ	@ILLEGAL REGISTER ERROR 'ILR' (RMER1, BIT 01) a
14	073253	104	111	101	EMS13:	.ASCIZ	@DIAGNOSTIC MODE 'DMD' (RMPR1, BIT 00) a
15	073322	115	105	104	EMS14:	.ASCIZ	@MEDIUM ON LINE 'MOL' (RMDS, BIT 12) a
16	073367	115	101	111	EMS15:	.ASCIZ	@MAINTENANCE UNIT READY 'MUR' (RMPR1, BIT 09) a
17	073445	115	101	111	EMS16:	.ASCIZ	@MAINTENANCE WRITE PROTECT 'MWP' (RMPR1, BIT 03) a
18	073526	127	122	111	EMS17:	.ASCIZ	@WRITE LOCK 'WRL' (RMDS, BIT 11) a
19	073567	104	105	126	EMS20:	.ASCIZ	@DEVICE CHECK 'DVC' (RMER2, BIT 07) a
20	073633	115	101	111	EMS21:	.ASCIZ	@MAINTENANCE DRIVE FAULT 'MDF' (RMPR1, BIT 06) a
21	073712	125	116	123	EMS22:	.ASCIZ	@UNSAFE STATUS 'UNS' (RMER1, BIT 14) a
22	073757	123	105	105	EMS23:	.ASCIZ	@SEEK INCOMPLETE STATUS 'SKI' (RMER2, BIT 14) a
23	074035	115	101	111	EMS24:	.ASCIZ	@MAINTENANCE SEEK ERROR 'MSER' (RMPR1, BIT 07) a
24	074114	120	117	123	EMS25:	.ASCIZ	@POSITIONING IN PROGRESS 'PIP' (RMDS, BIT 13) a
25	074172	115	101	111	EMS26:	.ASCIZ	@MAINTENANCE ON CYLINDER 'MOC' (RMPR1, BIT 08) a
26	074251	105	116	104	EMS27:	.ASCIZ	@END OF BLOCK 'EBL' (RMPR1, BIT 13) a
27	074315	104	111	101	EMS30:	.ASCIZ	@DIAGNOSTIC END OF BLOCK 'DEBL' (RMPR1, BIT 13) a
28	074375	114	101	123	EMS31:	.ASCIZ	@LAST SECTOR STATUS 'LS' (RMPR1, BIT 02) a
29	074446	114	101	123	EMS32:	.ASCIZ	@LAST SECTOR/TRACK STATUS 'LST' (RMPR1, BIT 01) a
30	074526	123	105	103	EMS33:	.ASCIZ	@SECTOR ADDRESS BITS OF a
31	074556	124	122	101	EMS34:	.ASCIZ	@TRACK ADDRESS BITS OF a
32	074605	126	117	114	EMS35:	.ASCIZ	@VOLUME VALID 'VV' (RMDS, BIT 06) a
33	074647	107	117	040	EMS36:	.ASCIZ	@GO BIT (RMCS1, BIT 00) a
34	074677	103	131	114	EMS37:	.ASCIZ	@CYLINDER ADDRESS BIT OF a
35	074730	114	101	123	EMS40:	.ASCIZ	@LAST BLOCK TRANSFERRED, 'LBT' (RMDS, BIT 10) a
36	075006	103	117	115	EMS41:	.ASCIZ	@COMPOSITE ERROR 'ERR' (RMDS, BIT 14) a
37	075054	103	117	115	EMS42:	.ASCIZ	@COMMAND SEQUENCER TEST BIT 'TST' (RMPR2, BIT 12) a
38	075136	104	122	111	EMS43:	.ASCIZ	@DRIVE READY STATUS 'DRY' (RMDS, BIT 07) a
39	075207	103	117	116	EMS44:	.ASCIZ	@CONTINUE 'CONT' (RMPR1, BIT 06) a
40	075250	111	116	126	EMS45:	.ASCIZ	@INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) a
41	075325	114	117	123	EMS46:	.ASCIZ	@LOSS OF SYSTEM CLOCK ERROR 'LSC' (RMER2, BIT 11) a
42	075407	117	103	103	EMS47:	.ASCIZ	@OCCUPIED 'OCC' (RMPR1, BIT 15) a
43	075447	111	114	114	EMS50:	.ASCIZ	@ILLEGAL FUNCTION 'ILF' (RMER1, BIT 0) a
44	075516	117	106	106	EMS51:	.ASCIZ	@OFFSET DIRECTION 'OFD' (RMOF, BIT 07) a
45	075565	117	106	106	EMS52:	.ASCIZ	@OFFSET MODE 'OM' (RMDS, BIT 00) a
46	075626	122	125	116	EMS53:	.ASCIZ	@RUN AND GO 'RG' (RMPR1, BIT 14) a
47	075667	111	116	126	EMS54:	.ASCIZ	@INVALID ADDRESS ERROR 'IAE' (RMER1, BIT 10) a
48	075744	101	104	104	EMS55:	.ASCIZ	@ADDRESS OVERFLOW ERROR 'AOE' (RMER1, BIT 09) a
49	076022	122	105	107	EMS56:	.ASCII	@REGISTER MODIFICATION REFUSED ERROR a
50	076066	042	122	115		.ASCIZ	a'RMR' (RMER1, BIT 02) a
51	076115	104	122	111	EMS57:	.ASCIZ	@DRIVE REQUEST REQUIRED STATUS 'DRQ' (RMDT, BIT 11) a
52	076201	120	122	117	EMS60:	.ASCIZ	@PROGRAMMABLE STATUS 'PGM' (RMDS, BIT 09) a
53	076253	104	122	111	EMS61:	.ASCIZ	@DRIVE PRESENT STATUS 'DPR' (RMDS, BIT 08) a
54	076326	120	117	122	EMS62:	.ASCIZ	@PORT REQUEST FLOP 'RQA,RQB' (RMPR2, BITS 15,14) a
55	076407	101	124	124	EMS63:	.ASCIZ	@ATTENTION 'ATA' (RMDS, BIT 15) a
56	076447	127	122	111	EMS64:	.ASCIZ	@WRITE LOCK ERROR 'WLE' (RMER1, BIT 11) a
57	076517	105	130	103	EMS65:	.ASCIZ	@EXCEPTION 'REX' (RMPR1, BIT 12) a

58	076560	111	116	126	EMS66:	.ASCIZ	@INVALID COMMAND ERROR "IVC" (RMR2, BIT 12) @
59	076635	124	101	107	EMS67:	.ASCIZ	@TAG BUS (RMMR2, BITS 00-09) OR TAG CONTROL @
60	076711	114	111	116		.ASCIZ	@LINES (RMMR2, BITS 10,11,13) @
61	076747	123	105	101	EMS70:	.ASCIZ	@SEARCH ENABLE "ESRC" (RMMR1, BIT 11) @
62							
63	077015	104	111	123	EMS250:	.ASCIZ	@DISK ADDRESS REGISTER (RMDA) @
64	077053	103	117	116	EMS251:	.ASCIZ	@CONTROL STATUS REGISTER #1 (RMCS1) @
65	077117	105	122	122	EMS252:	.ASCIZ	@ERROR REGISTER #1 (RMR1) @
66	077152	105	122	122	EMS253:	.ASCIZ	@ERROR REGISTER #2 (RMR2) @
67	077205	115	101	111	EMS254:	.ASCIZ	@MAINTENANCE REGISTER #1 (RMMR1) @
68	077246	104	105	123	EMS255:	.ASCIZ	@DESIRED CYLINDER REGISTER (RMDC) @
69	077310	117	106	106	EMS256:	.ASCIZ	@OFFSET REGISTER (RMOF) @
70	077340	104	122	111	EMS257:	.ASCIZ	@DRIVE TYPE REGISTER (RMDT) @
71	077374	110	117	114	EMS260:	.ASCIZ	@HOLDING REGISTER (RMMR) @
72	077425	123	105	122	EMS261:	.ASCIZ	@SERIAL NUMBER REGISTER (RMSN) @
73	077464	101	124	124	EMS262:	.ASCIZ	@ATTENTION SUMMARY REGISTER (RMAS) @
74							
75	077527	103	101	116	EMS300:	.ASCIZ	@CANNOT CLEAR @
76	077545	103	101	116	EMS301:	.ASCIZ	@CANNOT WRITE/READ @
77	077570	101	116	131	EMS302:	.ASCIZ	@ANY DEVICE REGISTER @
78	077615	127	111	124	EMS303:	.ASCIZ	@WITHOUT @
79	077626	105	122	122	EMS304:	.ASCIZ	@ERROR @
80	077635	101	040	117	EMS306:	.ASCIZ	@A ONE FROM @
81	077651	125	123	111	EMS307:	.ASCIZ	@USING MASSBUS INITIALIZE, I.E., @
82	077712	101	040	132	EMS310:	.ASCIZ	@A ZERO FROM @
83	077727	105	126	105	EMS311:	.ASCIZ	@EVERY DEVICE REGISTER BIT POSITION @
84	077773	124	110	105	EMS312:	.ASCIZ	@THE FOLLOWING BITS ARE STUCK @
85	100031	101	040	123	EMS313:	.ASCIZ	@A SHIFTING ONE BIT FROM @
86	100062	101	120	120	EMS314:	.ASCIZ	@APPEARS STUCK AT ZERO @
87	100111	101	120	120	EMS315:	.ASCIZ	@APPEARS STUCK AT ONE @
88	100137	122	105	107	EMS316:	.ASCIZ	@REGISTER SELECT @
89	100160	061	040	050	EMS317:	.ASCIZ	@1 (1,2,4,8,16) @
90	100200	062	040	050	EMS320:	.ASCIZ	@2 (1,2,4,8,16) @
91	100220	064	040	050	EMS321:	.ASCIZ	@4 (1,2,4,8,16) @
92	100240	070	040	050	EMS322:	.ASCIZ	@8 (1,2,4,8,16) @
93	100260	101	114	114	EMS323:	.ASCIZ	@ALL ONES FROM @
94	100277	101	114	114	EMS324:	.ASCIZ	@ALL ZEROS FROM @
95	100317	101	124	040	EMS325:	.ASCIZ	@AT ZERO @
96	100330	101	124	040	EMS326:	.ASCIZ	@AT ONE @
97	100340	054	040	117	EMS327:	.ASCIZ	@, OR @
98	100346	015	012	103	EMS330:	.ASCIZ	<CR><LF>@CS MBA CLRL @
99	100365	103	101	116	EMS331:	.ASCIZ	@CANNOT READ ZEROS FROM @
100	100415	111	123	040	EMS332:	.ASCIZ	@IS INCORRECT @
101	100433	111	123	040	EMS333:	.ASCIZ	@IS NOT SET @
102	100447	111	123	040	EMS334:	.ASCIZ	@IS SET @
103	100457	123	110	117	EMS335:	.ASCIZ	@SHOULD NOT BE SET @
104	100502	123	110	117	EMS336:	.ASCIZ	@SHOULD BE SET @
105	100521	103	101	116	EMS337:	.ASCIZ	@CANNOT SET @
106	100535	102	105	103	EMS340:	.ASCIZ	@BECAUSE @
107	100546	125	123	111	EMS341:	.ASCIZ	@USING @
108	100555	104	125	122	EMS342:	.ASCIZ	@DURING REGISTER TRANSFER @
109	100607	125	116	105	EMS343:	.ASCIZ	@UNEXPECTED @
110	100623	102	125	123	EMS344:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
111	100652	102	131	040	EMS345:	.ASCIZ	@BY REGISTER TRANSFER @
112	100700	103	101	116	EMS346:	.ASCIZ	@CANNOT RESET @
113	100716	127	111	124	EMS347:	.ASCIZ	@WITHOUT SETTING @
114	100737	102	125	124	EMS350:	.ASCIZ	@BUT @

115	100744	127	101	123	EMS351:	.ASCIZ	@WAS RESET BY @
116	100762	127	101	123	EMS352:	.ASCIZ	@WAS SET BY @
117	100776	111	116	040	EMS353:	.ASCIZ	@IN DIAGNOSTIC MODE @
118	101022	111	123	040	EMS354:	.ASCIZ	@IS INCORRECT ACCORDING TO @
119	101055	103	101	116	EMS355:	.ASCIZ	@CANNOT INCREMENT @
120	101077	127	101	123	EMS356:	.ASCIZ	@WAS NOT SET BY @
121	101117	127	101	123	EMS357:	.ASCIZ	@WAS NOT RESET BY @
122	101141	060	040	124	EMS360:	.ASCIZ	@0 TO 1 TRANSITION OF @
123	101167	061	040	124	EMS361:	.ASCIZ	@1 TO 0 TRANSITION OF @
124	101215	111	123	040	EMS362:	.ASCIZ	@IS INCONSISTENT @
125	101236	103	101	116	EMS363:	.ASCIZ	@CANNOT READ @
126	101253	124	105	123	EMS364:	.ASCIZ	@TEST PATTERN IN @
127	101274	101	116	104	EMS365:	.ASCIZ	@AND @
128	101301	103	101	116	EMS366:	.ASCIZ	@CANNOT INITIALIZE @
129	101324	124	110	105	EMS367:	.ASCIZ	@THE COMMAND SEQUENCER HAS BEEN CLOCKED @
130	101374	122	105	123	EMS370:	.ASCIZ	@RESET EARLY @
131	101411	104	111	104	EMS371:	.ASCIZ	@DID NOT RESET ON TIME @
132	101440	104	125	122	EMS372:	.ASCIZ	@DURING COMMAND EXECUTION @
133	101472	124	117	040	EMS373:	.ASCIZ	@TO @
134	101476	127	111	124	EMS374:	.ASCIZ	@WITH ANY COMBINATION OF @
135	101527	102	131	040	EMS375:	.ASCIZ	@BY READING @
136	101543	102	131	040	EMS376:	.ASCIZ	@BY WRITING @
137	101557	127	101	123	EMS377:	.ASCIZ	@WAS SET @
138	101570	127	101	123	EMS400:	.ASCIZ	@WAS NOT SET @
139	101605	104	111	104	EMS401:	.ASCIZ	@DID NOT SET @
140	101622	127	110	111	EMS402:	.ASCIZ	@WHILE @
141	101631	103	117	115	EMS403:	.ASCIZ	@COMMAND SEQUENCER DID NOT ABORT @
142	101672	127	101	123	EMS404:	.ASCIZ	@WAS NOT RESET @
143	101711	104	125	122	EMS405:	.ASCIZ	@DURING @
144	101721	127	101	123	EMS406:	.ASCIZ	@WAS RESET @
145	101734	123	105	101	EMS407:	.ASCIZ	@SEARCH TIMEOUT @
146							
147	101754	011	104	105	EMS500:	.ASCII	@ DEVICE IS NON-EXISTENT,@<CR><LF>
148	102006	011	104	105		.ASCII	@ DEVICE IS SWITCHED TO OTHER PORT@<CR><LF>
149	102051	011	124	122		.ASCIZ	@ TRANSCEIVER ENABLE SWITCH IS OFF@<CR><LF>
150	102115	011	111	106	EMS501:	.ASCIZ	@ IF MODULE, M7686,@<CR><LF>
151	102142	011	115	101	EMS502:	.ASCIZ	@ MASSBUS TRANSCEIVER, M5922 OR M5923 @<CR><LF>
152	102212	011	103	123	EMS503:	.ASCIZ	@ CS MODULE, M7684,@<CR><LF>
153	102237	011	104	123	EMS504:	.ASCIZ	@ DS MODULE, M8685/M7685,@<CR><LF>
154	102272	011	104	105	EMS505:	.ASCIZ	@ DEVICE IS SWITCHED TO A/B PORT POSITION@<CR><LF>
155	102345	011	104	105	EMS506:	.ASCIZ	@ DEVICE IS NOT AN RM80, OR@<CR><LF>
156	102402	011	104	105	EMS507:	.ASCIZ	@ DEVICE IS SWITCHED TO PROGRAMMABLE PORT POSITION, OR@<CR><LF>
157	102472	011	101	123	EMS510:	.ASCIZ	@ ASSUMING THE RH CONTROLLER HAS NO FAULT@<CR><LF>
158	102545	015	012	011	EMS511:	.ASCII	@<CR><LF>@ PROBABLE FAULT(S):@<CR><LF>
159	102574	011	050	116		.ASCIZ	@ (NOT INCLUDING CABLES OR CONNECTORS)@<CR><LF>
160							
161	102644	122	105	101	EMS600:	.ASCIZ	@READ IN PRESET COMMAND @
162	102674	117	106	106	EMS601:	.ASCIZ	@OFFSET COMMAND @
163	102714	122	105	124	EMS602:	.ASCIZ	@RETURN TO CENTER COMMAND @
164	102755	122	105	114	EMS603:	.ASCIZ	@RELEASE COMMAND @
165	102776	122	105	103	EMS604:	.ASCIZ	@RECALIBRATE COMMAND @
166	103023	123	105	105	EMS605:	.ASCIZ	@SEEK COMMAND @
167	103041	123	105	101	EMS606:	.ASCIZ	@SEARCH COMMAND @
168	103061	104	101	124	EMS607:	.ASCIZ	@DATA COMMAND @

1	103077	105	130	120	EH1:	.ASCII	@EXPCTD	RECEVD	REGSTR@<CRLF>
2	103126	123	124	101		.ASCIIZ	@STATUS	STATUS	ADRESS@
3	103155	040	102	101	EH2:	.ASCII	@BASE@<CRLF>		
4	103163	101	104	122	EH3:	.ASCIIZ	@ADRESS@		
5	103172	105	130	120	EH5:	.ASCII	@EXPCTD	STUCK@<CRLF>	
6	103211	122	105	123		.ASCIIZ	@RESULT	BIT(S)@	
7	103230	105	130	120	EH7:	.ASCIIZ	@EXPCTD	RECEVD@	
8	103247	122	105	103	EH57:	.ASCII	@RECEVD	DRVTYP@<CRLF>	
9	103266	104	122	126		.ASCIIZ	@DRVTYP	REGADR@	
10	103305	105	130	120	EH65:	.ASCII	@EXPCTD	RECEVD	TEST@<CRLF>
11	103332	123	124	101		.ASCIIZ	@STATUS	STATUS	REGSTR@
12	103361	105	130	120	EH71:	.ASCII	@EXPCTD	RECEVD	TEST@<CRLF>
13	103407	123	124	101		.ASCIIZ	@STATUS	STATUS	PATTRN@
14	103436	105	130	120	EH115:	.ASCII	@EXPCTD	RECEVD	REGSTR
15	103474	123	124	101		.ASCIIZ	@STATUS	STATUS	ADRESS
16	103533	105	130	120	EH130:	.ASCII	@EXPCTD	RECEVD	REGSTR
17	103602	123	124	101		.ASCIIZ	@STATUS	STATUS	ADRESS
18	103651	105	130	120	EH132:	.ASCII	@EXPCTD	ACTUAL	REGSTR
19	103710	103	117	125		.ASCIIZ	@COUNT	COUNT	ADRESS
20	103747	105	130	120	EH142:	.ASCII	@EXPCTD	RECEVD	REGSTR
21	104006	123	124	101		.ASCIIZ	@STATUS	STATUS	ADRESS
22	104045	105	130	120	EH145:	.ASCII	@EXPCTD	ACTUAL	REGSTR
23	104113	103	115	120		.ASCIIZ	@CMPERR	CMPERR	ADRESS
24									
25	104162	105	130	120	EH150:	.ASCII	@EXPCTD	ACTUAL	REGSTR
26	104222	122	105	123		.ASCIIZ	@RESULT	RESULT	ADRESS
27	104260	105	130	120	EH213:	.ASCII	@EXPCTD	ACTUAL	STATUS
28	104316	122	105	123		.ASCIIZ	@RESULT	RESULT	ADRESS
29	104355	101	103	124	EH220:	.ASCII	@ACTUAL	REGSTR@<CRLF>	
30	104374	122	105	123		.ASCIIZ	@RESULT	ADRESS@	
31									

.EVEN

1	104414	001140	001142	001136	ED1:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,0
2	104424	001136	000000		ED2:	.WORD	\$BDADR,0
3	104430	001140	001142	000000	ED5:	.WORD	\$GDDAT,\$BDDAT,0
4	104436	001142	001136	000000	ED57:	.WORD	\$BDDAT,\$BDADR,0
5	104444	001140	001142	001174	ED7:	.WORD	\$GDDAT,\$BDDAT,\$TMPO,0
6	104454	001140	001142	001450	ED71:	.WORD	\$GDDAT,\$BDDAT,\$RMHRO,0
7	104464	001140	001142	001136	ED115:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,\$TMPO,0
8	104476	001140	001142	001136	ED130:	.WORD	\$GDDAT,\$BDDAT,\$BDADR,\$TMPO,\$TMP1,0
9	104512	001142	001136		ED220:	.WORD	\$BDDAT,\$BDADR
10							
11	104516	000	000	000	EF1:	.BYTE	0,0,0
12	104521	000			EF2:	.BYTE	0
13	104522	000	000		EF5:	.BYTE	0,0
14	104524	000	000	000	EF57:	.BYTE	0,0,0,0
15	104530	000	000	000	EF130:	.BYTE	0,0,0,0,0
16					.EVEN		

1	104536				BUFFER:	
2	104536				BUFOONE: .BLKW	258.
3	105542				BUFTWO: .BLKW	258.
4						
5	104536				.=BUFFER	
6						
7	104536				HELP:	
8	104536	200			.ASCII	<CRLF>
9	104537	200			.ASCII	<CRLF>
10	104540	114	111	123	.ASCII	@LIST OF TESTS@<CRLF>
11	104556	055	055	055	.ASCII	@-----@<CRLF>
12	104574	124	061	011	.ASCII	@T1 TRANSFER TEST@<CRLF>
13	104615	124	062	011	.ASCII	@T2 CTOD TEST@<CRLF>
14	104632	124	063	011	.ASCII	@T3 MASSBUS INITIALIZE TEST@<CRLF>
15	104665	124	064	011	.ASCII	@T4 CLEAR STUCK ACTIVE TEST@<CRLF>
16	104720	124	065	011	.ASCII	@T5 TRISTATE TRANSFER TEST@<CRLF>
17	104752	124	066	011	.ASCII	@T6 REGISTER SELECT TEST@<CRLF>
18	105002	124	067	011	.ASCII	@T7 DRIVE TYPE TEST@<CRLF>
19	105025	124	061	060	.ASCII	@T10 DEVICE AVAILABLE TEST@<CRLF>
20	105057	124	061	061	.ASCII	@T11 HOLDING REGISTER TRANSFER TEST@<CRLF>
21	105122	124	061	062	.ASCII	@T12 CONTROL STATUS #1 TRANSFER TEST@<CRLF>
22	105166	124	061	063	.ASCII	@T13 ERROR REGISTER #1 TRANSFER TEST@<CRLF>
23	105232	124	061	064	.ASCII	@T14 CLEAR OFFSET STUCK ACTIVE TEST@<CRLF>
24	105275	124	061	065	.ASCII	@T15 OFFSET REGISTER TRANSFER TEST@<CRLF>
25	105337	124	061	066	.ASCII	@T16 ERROR REGISTER #2 TRANSFER TEST@<CRLF>
26	105403	124	061	067	.ASCII	@T17 SERIAL NUMBER TEST@<CRLF>
27	105432	124	062	060	.ASCII	@T20 CONTROL BUS PARITY DETECTION TEST@<CRLF>
28	105500	124	062	061	.ASCII	@T21 CONTROL BUS PARITY GENERATION TEST@<CRLF>
29	105547	124	062	062	.ASCII	@T22 RMDA,RMDC FAULT TEST@<CRLF>
30	105600	124	062	063	.ASCII	@T23 DISK ADDRESS TRANSFER TEST@<CRLF>
31	105637	124	062	064	.ASCII	@T24 DESIRED CYLINDER TRANSFER TEST@<CRLF>
32	105702	124	062	065	.ASCII	@T25 ILLEGAL REGISTER TEST@<CRLF>
33	105734	124	062	066	.ASCII	@T26 RESET GO BY INIT TEST@<CRLF>
34	105766	124	062	067	.ASCII	@T27 DIAGNOSTIC MODE TEST@<CRLF>
35	106017	124	063	060	.ASCII	@T30 MOL TEST@<CRLF>
36	106034	124	063	061	.ASCII	@T31 WRITE LOCK TEST@<CRLF>
37	106060	124	063	062	.ASCII	@T32 DRIVE FAULT TEST@<CRLF>
38	106105	124	063	063	.ASCII	@T33 SEEK ERROR TEST@<CRLF>
39	106131	124	063	064	.ASCII	@T34 PIP TEST@<CRLF>
40	106146	124	063	065	.ASCII	@T35 EBL TEST@<CRLF>
41	106163	124	063	066	.ASCII	@T36 LAST SECTOR, LAST TRACK TEST@<CRLF>
42	106224	124	063	067	.ASCII	@T37 RMDA COUNT TEST@<CRLF>
43	106250	124	064	060	.ASCII	@T40 RMDC COUNT TEST@<CRLF>
44	106274	124	064	061	.ASCII	@T41 LBT TEST@<CRLF>
45	106311	124	064	062	.ASCII	@T42 COMPOSITE ERROR TEST@<CRLF>
46	106342	124	064	063	.ASCII	@T43 WRITE GO TEST@<CRLF>
47	106364	124	064	064	.ASCII	@T44 BRANCH MULTIPLEXOR TEST@<CRLF>
48	106420	124	064	065	.ASCII	@T45 SET/RESET GO TEST@<CRLF>
49	106446	124	064	066	.ASCII	@T46 END 1 RESET GO TEST@<CRLF>
50	106476	124	064	067	.ASCII	@T47 SET PULSE TEST@<CRLF>
51	106521	124	065	060	.ASCII	@T50 SET/RESET IVC TEST@<CRLF>
52	106550	124	065	061	.ASCII	@T51 SET LSC TEST@<CRLF>
53	106571	124	065	062	.ASCII	@T52 DECODE TEST@<CRLF>
54	106611	124	065	063	.ASCII	@T53 SET/RESET VOLUME VALID TEST@<CRLF>
55	106651	124	065	064	.ASCII	@T54 ILLEGAL FUNCTION TEST@<CRLF>
56	106703	124	065	065	.ASCII	@T55 OCCUPIED TEST@<CRLF>
57	106725	124	065	066	.ASCII	@T56 READ IN PRESET TEST@<CRLF>



58	106755	124	065	067	.ASCII	@T57	RIP/RMOF TEST@<CRLF>
59	106777	124	066	060	.ASCII	@T60	RMDA/RMDC/RIP TEST@<CRLF>
60	107026	124	066	061	.ASCII	@T61	OFFSET COMMAND TEST@<CRLF>
61	107056	124	066	062	.ASCII	@T62	RETURN TO CENTER TEST@<CRLF>
62	107110	124	066	063	.ASCII	@T63	RMDC CLEAR OFFSET TEST@<CRLF>
63	107143	124	066	064	.ASCII	@T64	EBL CLEAR OFFSET TEST@<CRLF>
64	107175	124	066	065	.ASCII	@T65	RUN AND GO TEST@<CRLF>
65	107221	124	066	066	.ASCII	@T66	SET IAE TEST@<CRLF>
66	107242	124	066	067	.ASCII	@T67	SEARCH, SEEK, READ, WRITE TEST@<CRLF>
67	107305	124	067	060	.ASCII	@T70	INVALID TRACK/SECTOR TEST@<CRLF>
68	107343	124	067	061	.ASCII	@T71	INVALID CYLINDER TEST@<CRLF>
69	107375	124	067	062	.ASCII	@T72	SET AOE TEST@<CRLF>
70	107416	124	067	063	.ASCII	@T73	SET RMR TEST@<CRLF>
71	107437	124	067	064	.ASCII	@T74	PGM STATUS CHECK@<CRLF>
72	107464	124	067	065	.ASCII	@T75	DVA/DPR STATUS CHECK@<CRLF>
73	107515	124	067	066	.ASCII	@T76	PORT REQUEST TEST, PART 1@<CRLF>
74	107553	124	067	067	.ASCII	@T77	PORT REQUEST TEST, PART 2@<CRLF>
75	107611	124	061	060	.ASCII	@T100	PORT REQUEST TEST, PART 3@<CRLF>
76	107650	124	061	060	.ASCII	@T101	RELEASE TEST@<CRLF>
77	107672	124	061	060	.ASCII	@T102	WRITE ATA TEST@<CRLF>
78	107716	124	061	060	.ASCII	@T103	RESET ATA BY GO TEST@<CRLF>
79	107750	124	061	060	.ASCII	@T104	UNIT READY ATA TEST@<CRLF>
80	110001	124	061	060	.ASCII	@T105	ERROR ATA TEST@<CRLF>
81	110025	124	061	060	.ASCII	@T106	REGISTER TRANSFER ATA TEST@<CRLF>
82	110065	124	061	060	.ASCII	@T107	P SET ATA TEST@<CRLF>
83	110111	124	061	061	.ASCII	@T110	SET WLE TEST@<CRLF>
84	110133	124	061	061	.ASCII	@T111	EXCEPTION TEST@<CRLF>
85	110157	124	061	061	.ASCII	@T112	RECALIBRATE TEST@<CRLF>
86	110205	124	061	061	.ASCII	@T113	SEEK TEST@<CRLF>
87	110224	124	061	061	.ASCII	@T114	SEARCH TEST@<CRLF>
88	110245	124	061	061	.ASCII	@T115	SEARCH TIMEOUT TEST@<CRLF>
89	110276	124	061	061	.ASCII	@T116	DATA COMMAND TESTS (1)@<CRLF>
90	110332	124	061	061	.ASCII	@T117	DATA COMMAND TESTS (2)@<CRLF>
91	110366	124	061	062	.ASCII	@T120	DATA COMMAND TESTS (3)@<CRLF>
92	110422	200			.ASCII	<CRLF>	
93	110423	117	120	105	.ASCII	@OPERATIONAL SWITCH SETTINGS@<CRLF>	
94	110457	055	055	055	.ASCII	@-----@<CRLF>	
95	110513	123	127	111	.ASCII	@SWITCH	USE@<CRLF>
96	110530	055	055	055	.ASCII	@-----@<CRLF>	
97	110565	040	040	061	.ASCII	@ 15	HALT ON ERROR@<CRLF>
98	110611	040	040	061	.ASCII	@ 14	LOOP ON TEST@<CRLF>
99	110634	040	040	061	.ASCII	@ 13	INHIBIT ERROR TYPEOUTS@<CRLF>
100	110671	040	040	061	.ASCII	@ 12	@<CRLF>
101	110700	040	040	061	.ASCII	@ 11	INHIBIT ITERATIONS@<CRLF>
102	110731	040	040	061	.ASCII	@ 10	BELL ON ERROR@<CRLF>
103	110755	040	040	040	.ASCII	@ 9	LOOP ON ERROR@<CRLF>
104	111001	040	040	040	.ASCII	@ 8	LOOP ON TEST IN SWR<7:0>@<CRLF>
105	111040	040	040	040	.ASCII	@ 7	TN128@<CRLF>
106	111054	040	040	040	.ASCII	@ 6	TN64@<CRLF>
107	111067	040	040	040	.ASCII	@ 5	TN32@<CRLF>
108	111102	040	040	040	.ASCII	@ 4	TN16@<CRLF>
109	111115	040	040	040	.ASCII	@ 3	TN8@<CRLF>
110	111127	040	040	040	.ASCII	@ 2	TN4@<CRLF>
111	111141	040	040	040	.ASCII	@ 1	TN2@<CRLF>
112	111153	040	040	040	.ASCII	@ 0	TN1@<CRLF>
113							
114	000200			.END		200	

## SYMBOL TABLE

ABASE = 176700	AUTSIZ = 001326	CHRCNT = 060527	ED1 = 104414	EH71 = 103361
ACDW1 = 000000	AVECT1 = 120254	CKSWR = 104410	ED115 = 104464	EMS1 = 072436
ACDW2 = 000000	AVECT2 = 000000	CLOCK = 001536	ED130 = 104476	EMS10 = 073041
ACPUOP = 000000	A16 = 000400	CLR = 000040	ED2 = 104424	EMS11 = 073104
ADDW0 = 000000	A17 = 001000	CMNSTA = 006612	ED220 = 104512	EMS12 = 073175
ADDW1 = 000000	BADTMO = 004562	CNSL01 = 063115	ED5 = 104430	EMS13 = 073253
ADDW10 = 000000	BAI = 000010	CNSL02 = 063125	ED57 = 104436	EMS14 = 073322
ADDW11 = 000000	BB00 = 000001	CNSL03 = 063167	ED65 = 104444	EMS15 = 073367
ADDW12 = 000000	BB01 = 000002	CNSL04 = 063176	ED71 = 104457	EMS16 = 073445
ADDW13 = 000000	BB02 = 000004	CNSL07 = 063232	EECC = 000020	EMS17 = 073526
ADDW14 = 000000	BB03 = 000010	CNSL08 = 063374	EFT1 = 072410	EMS2 = 072507
ADDW15 = 000000	BB04 = 000020	CNSL09 = 063375	EFT115 = 072426	EMS20 = 073567
ADDW2 = 000000	BB05 = 000040	CNTCLR = 055156	EFT130 = 072430	EMS21 = 073633
ADDW3 = 000000	BB06 = 000100	COMMA = 063061	EFT132 = 072432	EMS22 = 073712
ADDW4 = 000000	BB07 = 000200	CONT = 000100	EFT2 = 072412	EMS23 = 073757
ADDW5 = 000000	BB08 = 000400	CPSAVE = 057700	EFT220 = 072434	EMS24 = 074035
ADDW6 = 000000	BB09 = 001000	CR = 000015	EFT5 = 072414	EMS25 = 074114
ADDW7 = 000000	BIT0 = 000001	CRLF = 000200	EFT57 = 072416	EMS250 = 077015
ADDW8 = 000000	BIT00 = 000001	CYLSK = 001777	EFT65 = 072420	EMS251 = 077053
ADDW9 = 000000	BIT01 = 000002	DBCK = 100000	EFT71 = 072422	EMS252 = 077117
ADEVCT = 000000	BIT02 = 000004	DBEN = 040000	EFT74 = 072424	EMS253 = 077152
ADEVN = 000000	BIT03 = 000010	DBL = 002000	EF1 = 104516	EMS254 = 077205
ADR = 000001	BIT04 = 000020	DCK = 100000	EF130 = 104530	EMS255 = 077246
AENV = 000000	BIT05 = 000040	DDISP = 177570	EF2 = 104521	EMS256 = 077310
AFNMV = 000000	BIT06 = 000100	DEBL = 020000	EF5 = 104522	EMS257 = 077340
AFATAL = 000000	BIT07 = 000200	DISPLA = 001156	EF57 = 104524	EMS26 = 074172
ALL = 063050	BIT08 = 000400	DISPRE = 000174	EHT1 = 072262	EMS260 = 077374
AMADR1 = 000000	BIT09 = 001000	DLT = 100000	EHT115 = 072322	EMS261 = 077425
AMADR2 = 000000	BIT1 = 000002	DMD = 000001	EHT130 = 072326	EMS262 = 077464
AMADR3 = 000000	BIT10 = 002000	DPE = 000010	EHT132 = 072332	EMS27 = 074251
AMADR4 = 000000	BIT11 = 004000	DPEHI = 040000	EHT142 = 072336	EMS3 = 072556
AMAMS1 = 000000	BIT12 = 010000	DPELO = 020000	EHT145 = 072342	EMS30 = 074315
AMAMS2 = 000000	BIT13 = 020000	DPR = 000400	EHT150 = 072346	EMS300 = 077527
AMAMS3 = 000000	BIT14 = 040000	DRIVES = 063416	EHT2 = 072266	EMS301 = 077545
AMAMS4 = 000000	BIT15 = 100000	DRQ = 004000	EHT213 = 072352	EMS302 = 077570
AMSGAD = 000000	BIT2 = 000004	DRVCLR = 000010	EHT220 = 072356	EMS303 = 077615
AMSGLG = 000000	BIT3 = 000010	DRY = 000200	EHT5 = 072272	EMS304 = 077626
AMSGTY = 000000	BIT4 = 000020	DSWR = 177570	EHT57 = 072302	EMS306 = 077635
AMTYP1 = 000000	BIT5 = 000040	DTE = 010000	EHT65 = 072306	EMS307 = 077651
AMTYP2 = 000000	BIT6 = 000100	DULPRT = 024026	EHT7 = 072276	EMS31 = 074375
AMTYP3 = 000000	BIT7 = 000200	DVA = 004000	EHT71 = 072312	EMS310 = 077712
AMTYP4 = 000000	BIT8 = 000400	DVC = 000200	EHT74 = 072316	EMS311 = 077727
AOE = 001000	BIT9 = 001000	EVL = 020000	EH1 = 103077	EMS312 = 077773
APASS = 000000	BLNKS1 = 063635	ECH = 000100	EH115 = 103436	EMS313 = 100031
APE = 100000	BLNKS2 = 063634	ECI = 004000	EH130 = 103533	EMS314 = 100062
APRIOR = 000000	BLNKS3 = 063633	ECRC = 001000	EH132 = 103651	EMS315 = 100111
APTCSU = 000040	BLNKS4 = 063632	EDT1 = 072362	EH142 = 103747	EMS316 = 100137
APTENV = 000001	BOTADR = 060524	EDT115 = 072400	EH145 = 104045	EMS317 = 100160
APTSIZ = 000200	BOTFLG = 060526	EDT130 = 072402	EH150 = 104162	EMS32 = 074446
APTSPO = 000100	BPTVEC = 000014	FDT132 = 072404	EH2 = 103155	EMS320 = 100200
ASWREG = 000000	BSE = 100000	EDT2 = 072364	EH213 = 104260	EMS321 = 100220
ATA = 100000	BUFFER = 104536	EDT220 = 072406	EH220 = 104355	EMS322 = 100240
ATESTN = 000000	BUFONE = 104536	EDT5 = 072366	EH3 = 103163	EMS323 = 100260
ATNMSK = 000377	BUFTWO = 105542	EDT57 = 072370	EH5 = 103172	EMS324 = 100277
ATNTBL = 063740	CC = 004000	EDT65 = 072372	EH57 = 103247	EMS325 = 100317
AUNIT = 000000	CH = 002000	EDT71 = 072374	EH65 = 103305	EMS326 = 100330
AUSWR = 000000	CHGADR = 001330	EDT74 = 072376	EH7 = 103230	EMS327 = 100340

## SYMBOL TABLE

EMS33	074526	EMS43	075136	EMT112	066452	EMT175	070116	EMT257	071552
EMS330	100346	EMS44	075207	EMT113	066472	EMT176	070136	EMT26	065074
EMS331	100365	EMS45	075250	EMT114	066516	EMT177	070154	EMT260	071576
EMS332	100415	EMS46	075325	EMT115	066540	EMT2	064404	EMT261	071622
EMS333	100433	EMS47	075407	EMT116	066564	EMT20	064744	EMT262	071646
EMS334	100447	EMS5	072654	EMT117	066604	EMT200	070156	EMT263	071664
EMS335	100457	EMS50	075447	EMT12	064620	EMT201	070174	EMT264	071702
EMS336	100502	EMS500	101754	EMT120	066624	EMT202	070212	EMT265	071726
EMS337	100521	EMS501	102115	EMT121	066650	EMT203	070230	EMT266	071744
EMS34	074556	EMS502	102142	EMT122	066670	EMT204	070252	EMT267	071762
EMS340	100535	EMS503	102212	EMT123	066710	EMT205	070272	EMT27	065126
EMS341	100546	EMS504	102237	EMT124	066734	EMT206	070312	EMT270	071776
EMS342	100555	EMS505	102272	EMT125	066752	EMT207	070342	EMT271	072022
EMS343	100607	EMS506	102345	EMT126	066772	EMT21	064764	EMT272	072046
EMS344	100623	EMS507	102402	EMT127	067010	EMT210	070356	EMT273	072064
EMS345	100652	EMS51	075516	EMT13	064636	EMT211	070374	EMT274	072102
EMS346	100700	EMS510	102472	EMT130	067034	EMT212	070412	EMT275	072120
EMS347	100716	EMS511	102545	EMT131	067052	EMT213	070424	EMT276	072140
EMS35	074605	EMS52	075565	EMT132	067070	EMT214	070454	EMT277	072156
EMS350	100737	EMS53	075626	EMT133	067110	EMT215	070466	EMT3	064432
EMS351	100744	EMS54	075667	EMT134	067130	EMT216	070506	EMT30	065140
EMS352	100762	EMS55	075744	EMT135	067144	EMT217	070522	EMT300	072202
EMS353	100776	EMS56	076022	EMT136	067162	EMT22	065004	EMT301	072226
EMS354	101022	EMS57	076115	EMT137	067174	EMT220	070536	EMT302	072244
EMS355	101055	EMS6	072724	EMT14	064654	EMT221	070554	EMT31	065154
EMS356	101077	EMS60	076201	EMT140	067212	EMT222	070572	EMT32	065170
EMS357	101117	EMS600	102644	EMT141	067232	EMT223	070610	EMT33	065204
EMS36	074647	EMS601	102674	EMT142	067246	EMT224	070626	EMT34	065222
EMS360	101141	EMS602	102714	EMT143	067264	EMT225	070646	EMT35	065240
EMS361	101167	EMS603	102755	EMT144	067300	EMT226	070670	EMT36	065254
EMS362	101215	EMS604	102776	EMT145	067322	EMT227	070706	EMT37	065270
EMS363	101236	EMS605	103023	EMT146	067344	EMT23	065020	EMT4	064452
EMS364	101253	EMS606	103041	EMT147	067362	EMT230	070724	EMT40	065304
EMS365	101274	EMS607	103061	EMT15	064672	EMT231	070742	EMT41	065324
EMS366	101301	EMS61	076253	EMT150	067374	EMT232	070756	EMT42	065340
EMS367	101324	EMS62	076326	EMT151	067416	EMT233	071000	EMT43	065364
EMS37	074677	EMS63	076407	EMT152	067430	EMT234	071022	EMT44	065376
EMS370	101374	EMS64	076447	EMT153	067444	EMT235	071050	EMT45	065412
EMS371	101411	EMS65	076517	EMT154	067464	EMT236	071070	EMT46	065426
EMS372	101440	EMS66	076560	EMT155	067504	EMT237	071122	EMT47	065442
EMS373	101472	EMS67	076635	EMT156	067522	EMT24	065040	EMT5	064474
EMS374	101476	EMS7	072770	EMT157	067544	EMT240	071142	EMT50	065464
EMS375	101527	EMS70	076747	EMT16	064710	EMT241	071166	EMT51	065506
EMS376	101543	EMTVEC=	000030	EMT160	067562	EMT242	071212	EMT52	065526
EMS377	101557	EMT1	064376	EMT161	067612	EMT243	071236	EMT53	065542
EMS4	072623	EMT10	064564	EMT162	067630	EMT244	071262	EMT54	065556
EMS40	074730	EMT100	066200	EMT163	067646	EMT245	071300	EMT55	065576
EMS400	101570	EMT101	066220	EMT164	067670	EMT246	071316	EMT56	065616
EMS401	101605	EMT102	066240	EMT165	067712	EMT247	071342	EMT57	065632
EMS402	101622	EMT103	066260	EMT166	067736	EMT25	065060	EMT6	064520
EMS403	101631	EMT104	066304	EMT167	067756	EMT250	071360	EMT60	065646
EMS404	101672	EMT105	066324	EMT17	064726	EMT251	071404	EMT61	065662
EMS405	101711	EMT106	066344	EMT170	070000	EMT25	071430	EMT62	065702
EMS406	101721	EMT107	066370	EMT171	070012	EMT253	071454	EMT63	065722
EMS407	101734	EMT11	064602	EMT172	070026	EMT254	071472	EMT64	065736
EMS41	075006	EMT110	066410	EMT173	070044	EMT255	071510	EMT65	065750
EMS42	075054	EMT111	066432	EMT174	070066	EMT256	071534	EMT66	065764

SYMBOL TABLE

EMT67	066000	ILF54	= 000054	MUR	= 001000	QUES	= 063055	RMER1I	001352
EMT7	064542	ILF56	= 000056	MWD	= 000010	RD	= 000070	RMER10	001426
EMT70	066016	ILF64	= 000064	MWP	= 000010	RDCHR	= 104411	RMER2	= 000042
EMT71	066036	ILF66	= 000066	MXF	= 001000	RDLIN	= 104412	RMER2I	001400
EMT72	066062	ILF74	= 000074	N	= 063626	RDOCT	= 104413	RMER20	001454
EMT73	066106	ILF76	= 000076	NDTMSK	= 115760	RDY	= 000200	RMHR	= 000036
EMT74	066126	ILR	= 000002	NED	= 010000	READY	= 007006	RMHRI	001374
EMT75	066136	ILRG50	= 000050	NEM	= 004000	READY1	= 007034	RMHRO	001450
EMT76	066150	ILRG52	= 000052	NONE	= 063445	RECAL	= 000006	RMLA	= 000020
EMT77	066164	ILRG54	= 000054	NOP	= 000000	RESREG	= 104415	RMLAI	001356
ENRGDT	064376	ILRG56	= 000056	NOTAVL	= 063566	RESVEC	= 000010	RMLAO	001432
EQUALS	063046	ILRG60	= 000060	NOTPRS	= 063551	REX	= 010000	RMPR1	= 000024
ERR	= 040000	ILRG62	= 000062	NOTRM	= 063517	RG	= 040000	RMPR1I	001362
ERRNMB	060522	ILRG64	= 000064	NSA	= 100000	RGDPT	= 063750	RMPR10	001436
ERROR	= 104000	ILRG66	= 000066	OCC	= 100000	RH	= 000072	RMPR2	= 000040
ERRTYP	057704	ILRG70	= 000070	OFD	= 000200	RIP	= 000020	RMPR2I	001376
ERRVEC	= 000004	ILRG72	= 000072	OFFSET	= 000014	RELEASE	= 000012	RMPR20	001452
ERTY00	060530	ILRG74	= 000074	OM	= 000001	RMAS	= 000016	RMOF	= 000032
ERTY01	060535	ILRG76	= 000076	ONES	= 064010	RMASI	= 001354	RMOFI	001370
ERTY02	060545	IOTVEC	= 000020	OPE	= 020000	RMASO	= 001430	RMOFO	001444
ERTY03	060554	IPCK0	= 000001	OPI	= 020000	RMB	= 000004	RMR	= 000004
ERTY04	060562	IPCK1	= 000002	OR	= 000200	RMBAE	= 000050	RMSN	= 000030
ERTY05	060565	IPCK2	= 000004	PACACK	= 000022	RMBAEI	= 001406	RMSNI	001366
ESRC	= 004000	IPCK3	= 000010	PAKACK	= 000022	RMBAEO	= 001462	RMSNO	001442
FER	= 000020	IR	= 000100	PAR	= 000010	RMBAI	= 001342	RMWC	= 000002
FMT16	= 010000	IRP	= 054702	PAT	= 000020	RMBAO	= 001416	RMWCI	001340
FNCDTB	063640	IVC	= 010000	PCLOCK	= 054562	RMC	= 000000	RMWCO	001414
FNCMSK	= 000077	LBC	= 002000	PCOUNT	= 054650	RMCS1I	= 001336	RQA	= 100000
FO	= 000002	LBT	= 002000	PDA	= 000400	RMCS10	= 001412	RQB	= 040000
F1	= 000004	LCLOCK	= 054600	PFECH	= 060572	RMCS2	= 000010	RTC	= 000016
F2	= 000010	LCOUNT	= 054650	PFECH1	= 060602	RMCS2I	= 001346	R6	= 000006
F3	= 000020	LF	= 000012	PFECH2	= 060670	RMCS20	= 001422	R7	= 000007
F4	= 000040	LODEV	= 063534	PFECH3	= 060706	RMCS3	= 000052	SADMSK	= 000377
GETBUF	= 001336	LS	= 000004	PFECH4	= 060714	RMCS3I	= 001410	SAVREG	= 104414
GO	= 000001	LSC	= 004000	PGE	= 002000	RMCS30	= 001464	SA1	= 000001
GTSWR	= 104407	LST	= 000002	PGM	= 001000	RMDA	= 000006	SA16	= 000020
HCE	= 000200	LSTOP	= 054712	PHA	= 000200	RMDAI	= 001344	SA2	= 000002
HCI	= 002000	LSTRK	= 001334	PIP	= 020000	RMDAO	= 001420	SA4	= 000004
HCRC	= 000400	MCLK	= 004000	PIRQ	= 177772	RMDB	= 000022	SAB	= 000010
HELP	= 104536	MCPE	= 020000	PIRQVE	= 000240	RMDBI	= 001360	SC	= 100000
HT	= 000011	MDF	= 000100	PLCLK	= 054606	RMDBO	= 001434	SCOPE	= 000004
IAE	= 002000	MDPE	= 000400	PLFS	= 002000	RMDC	= 000034	SCTMSK	= 003700
IBSAVE	= 057702	MI	= 000004	PLSTP	= 054716	RMDCI	= 001372	SCO	= 000100
IDXMSK	= 000077	MIXED	= 063750	PRO	= 000000	RMDCO	= 001446	SC1	= 000200
IE	= 000100	MOC	= 000400	PR1	= 000040	RMDS	= 000012	SC2	= 000400
ILF	= 000001	MOH	= 020000	PR2	= 000100	RMDSI	= 001350	SC3	= 001000
ILF02	= 000002	MOL	= 010000	PR3	= 000140	RMDSO	= 001424	SC4	= 002000
ILF24	= 000024	MRD	= 002000	PR4	= 000200	RMDT	= 000026	SEARCH	= 000030
ILF26	= 000026	MR1AAA	= 051401	PR5	= 000240	RMDTI	= 001364	SEEK	= 000004
ILF30	= 000030	MS	= 000040	PR6	= 000300	RMDTO	= 001440	SETOM	= 055054
ILF32	= 000032	MSC	= 000002	PR7	= 000340	RMEC1	= 000044	SETVV	= 054732
ILF34	= 000034	MSDRVS	= 063452	PS	= 177776	RMEC1I	= 001402	SHUT	= 062170
ILF36	= 000036	MSE	= 100000	PSEL	= 002000	RMEC10	= 001456	SIZCLK	= 054322
ILF40	= 000040	MSEN	= 010000	PSTOP	= 054704	RMEC2	= 000046	SKI	= 040000
ILF42	= 000042	MSER	= 000200	PSW	= 177776	RMEC2I	= 001404	SNGPRT	= 020026
ILF44	= 000044	MSGDRV	= 063466	PUTBUF	= 001412	RMEC20	= 001460	SSE	= 000040
ILF46	= 000046	MSHELP	= 063064	PURVEC	= 000024	RMER1	= 000014	SSEI	= 001000

## SYMBOL TABLE

SSF = 020000	TST100 035746	TST55 031172	\$ATYC 062624	\$GDDAT 001140
STACK = 001100	TST101 036100	TST56 031404	\$ATY1 062600	\$GET42 054272
STANDA 006074	TST102 036270	TST57 031606	\$ATY3 062606	\$GTSWR 061320
START 004652	TST103 036472	TST6 011262	\$ATY4 062616	\$GT42P 054266
START1 004642	TST104 036620	TST60 031774	\$AUTOB 001150	\$HD = 000000
START2 004656	TST105 037024	TST61 032212	\$BASE 001276	\$HIBTS 001100
STKLMT= 177774	TST106 037154	TST62 032360	\$BADR 001136	\$HIOCT 062312
STOPCL 001540	TST107 037350	TST63 032604	\$BDDAT 001142	\$ICNT 001120
SWR 001154	TST11 012536	TST64 032720	\$BELL 001212	\$ILLUP 062562
SWREG 000176	TST110 037546	TST65 033106	\$BIN 055360	\$INTAG 001151
SWO = 000001	TST111 040024	TST66 033336	\$CDW1 001302	\$ITEMB 001130
SWO0 = 000001	TST112 040330	TST67 033554	\$CDW2 001304	\$LF 001220
SW01 = 000002	TST113 041722	TST7 012356	\$CHARC 056364	\$LFLG 063043
SW02 = 000004	TST114 043450	TST70 034002	\$CKSWR 061230	\$LLCSR 001522
SW03 = 000010	TST115 045630	TST71 034252	\$CMTAG 001114	\$LLVEC 001524
SW04 = 000020	TST116 046210	TST72 034460	\$CM3 = 000000	\$LPADR 001122
SW05 = 000040	TST117 050352	TST73 035004	\$CM4 = 000005	\$LPCSB 001514
SW06 = 000100	TST12 012776	TST74 035232	\$CNTLC 062126	\$LPCSR 001512
SW07 = 000200	TST120 052640	TST75 035340	\$CNTLG 062140	\$LPERR 001124
SW08 = 000400	TST13 013344	TST76 035464	\$CNTLU 062133	\$LPVEC 001516
SW09 = 001000	TST14 014126	TST77 035614	\$CPUOP 001250	\$MADR1 001254
SW1 = 000002	TST15 014234	TYPBN = 104406	\$CRLF 001217	\$MADR2 001260
SW10 = 002000	TST16 014666	TYPDS = 104405	\$DBLK 055576	\$MADR3 001264
SW11 = 004000	TST17 015400	TYPE = 104401	\$DDW0 001306	\$MADR4 001270
SW12 = 010000	TST2 007466	TYPDC = 104402	\$DDW1 001310	\$MAIL 001222
SW13 = 020000	TST20 015502	TYPDN = 104404	\$DDW2 001312	\$MAMS1 001252
SW14 = 040000	TST21 015766	TYPOS = 104403	\$DDW3 001314	\$MAMS2 001256
SW15 = 100000	TST22 016104	UNS = 040000	\$DDW4 001316	\$MAMS3 001262
SW2 = 000004	TST23 016232	UNTMSK= 000007	\$DDW5 001320	\$MAMS4 001266
SW3 = 000010	TST24 016504	UNTOFF 063605	\$DDW6 001322	\$MBADR 001102
SW4 = 000020	TST25 017004	UNTON 063616	\$DDW7 001324	\$MFLG 063042
SW5 = 000040	TST26 017426	UPE = 020000	\$DEVCT 001232	\$MNEW 062156
SW6 = 000100	TST27 017522	USE = 040000	\$DEVM 001300	\$MSGAD 001236
SW7 = 000200	TST3 007656	UO = 000001	\$DOAGN 054312	\$MSGLG 001240
SW8 = 000400	TST30 020010	U1 = 000002	\$DTBL 055566	\$MSGTY 001222
SW9 = 001000	TST31 020324	U2 = 000004	\$ENDAD 054302	\$MSWR 062145
SYSTAT 063474	TST32 020614	VV = 000100	\$ENDCT 054140	\$MTYP1 001253
TADMSK= 177400	TST33 021332	WATCH 001534	\$ENULL 054316	\$MTYP2 001257
TAG = 020000	TST34 021644	WC = 000040	\$ENV 001242	\$MTYP3 001263
TAGADR= 001114	TST35 022136	WCD = 000050	\$ENVM 001243	\$MTYP4 001267
TAP = 040000	TST36 022460	WCE = 040000	\$EOP 054104	\$MXCNT 057046
TA1 = 000400	TST37 023116	WCEHI = 010000	\$EOPCT 054132	\$NULL 001170
TA2 = 001000	TST4 010026	WCELO = 004000	\$EJSP 054046	\$NWTST= 000001
TA4 = 002000	TST40 023624	WCF = 000040	\$ERFLG 001117	\$OCNT 056030
TAB = 004000	TST41 024144	WCH = 000052	\$ERMAX 001131	\$OMODE 056032
TBITVE= 000014	TST42 024462	WD = 000060	\$ERROR 057310	\$OVER 057032
TIME 001532	TST43 025064	WH = 000062	\$ERRPC 001132	\$PASS 001230
TKVEC = 000060	TST44 025234	WLE = 004000	\$ERRTB 001542	\$PASTM 001106
TPVEC = 000064	TST45 025614	WRL = 004000	\$ERTTL 001126	\$POWER 062570
TRAPVE= 000034	TST46 026324	XNUDC = 176000	\$ESCAP 001210	\$PSW 001530
TRE = 040000	TST47 026620	XNUER2= 001527	\$ETABL 001242	\$PWDRN 062422
TRTVEC= 000014	TST5 010156	XNUOF = 160577	\$ETEND 001326	\$PWRMG 062556
TST = 010000	TST50 027222	XSIZ 005604	\$FATAL 001224	\$PWUP 062474
TSTNMB 060520	TST51 027456	XXDP 001332	\$FFLG 063044	\$QUES 001216
TSTQUE 001466	TST52 027664	Y 06363^	\$FILLC 001172	\$RDCHR 061572
TST1 007156	TST53 030516	ZEROS 064052	\$FILLS 001171	\$RDLIN 061662
TST10 012452	TST54 030770	\$APTHD 001100	\$GDADR 001134	\$RDOCT 062212

\$RDSZ = 000010	\$SWRMK= 000000	\$TMP0 001174	\$TRPAD 062366	\$UNIT 001234
\$RESRE 055250	\$SWOBT 057050	\$TMP1 001176	\$STSM 001104	\$UNITM 001110
\$RM80 063512	\$TESTN 001226	\$TMP2 001200	\$STSTM 001116	\$USWR 001246
\$RTNAD 054314	\$TIMES 001206	\$TMP3 001202	\$TTYIN 062116	\$VECT1 001272
\$SAVRE 055212	\$TKB 001162	\$TMP4 001204	\$TYPBN 055306	\$VECT2 001274
\$SAVR6 062566	\$TKCNT 060720	\$TN = 000121	\$TYPDS 055362	\$XOFF = 000023
\$SCOPE 056370	\$TKINT 060730	\$TPB 001166	\$TYPE 056034	\$XON = 000021
\$SETUP= 000137	\$TKQFN= 060727	\$TPFLG 001173	\$YPEC 056246	\$XTSTR 056406
\$STUP = 177777	\$TKQIN 060722	\$TPS 001164	\$YPEX 056366	\$GET4= 000000
\$SVLAD 056776	\$TKQOU 060724	\$TRAP 062314	\$YPOC 055632	\$SW08= 000121
\$SVPC = 000210	\$TKQSR 060726	\$TRAP2 062354	\$YPON 055646	\$FILL 056031
\$SWR = 167400	\$TKS 001160	\$TRP = 000016	\$YPOS 055606	\$.X = 001100
\$SWREG C01244	\$TKSRV 061000			

. ABS. 111166 000  
000000 001  
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 62208 WORDS ( 243 PAGES)  
DYNAMIC MEMORY AVAILABLE FOR 71 PAGES  
.A:CZRNB/C=A:CZRNB.DOC,CZRNB,SYSMAC/M









\$ILLUP	30-1	30-1	30-1#													
\$INTAG	6-0#	27-1	27-1	27-1	27-1	27-1*										
\$ITEMB	6-0#	25-1	25-1	25-1	25-1	25-1	25-1*	25-1*	26-41							
\$LF	6-0#	23-1	23-1	25-1	25-1	27-1	27-1	27-1								
\$LFLG	31-1#	31-1*														
\$LLCSR	7-0#	15-26	15-55*	15-86*												
\$LLVEC	7-0#	15-17	15-17*	15-18*	15-29*	15-30*	15-42	15-42*	15-43*							
\$LPADR	6-0#	9-23*	24-1	24-1	24-1	24-1*	24-1*	24-1*								
\$LPCSB	7-0#	15-51*														
\$LPCSR	7-0#	15-12	15-52*	15-83*												
\$LPERR	6-0#	9-23*	13-240*	13-278*	13-[09*	13-[51*	13-\08*	13-J08*	13-J49*	13-J83*	13-^28*	13-^88*	13-'00*	13-'44*		
	13-78*	13-a40*	13-b00*	13-b71*	13-d73*	13-e65*	13-f24*	13-f96*	13-h76*	13-i56*	13-j21*	13-k12*	24-1	24-1		
	24-1	24-1*	25-1													
\$LPVEC	7-0#	15-15*	15-16*	15-31	15-31*	15-32*	15-40	15-40*	15-41*							
\$MADR1	6-0#															
\$MADR2	6-0#															
\$MADR3	6-0#															
\$MADR4	6-0#															
\$MAIL	5-13	5-13	6-0#	9-23	9-28	13-1	13-73	13-106	13-137	13-159	13-332	13-524	13-538	13-549		
	13-602	13-670	13-774	13-787	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-=15	13-=32		
	13-90	13->55	13-?16	13-@18	13-@79	13-A39	13-B20	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64		
	13-114	13-189	13-K28	13-K74	13-L08	13-M40	13-N00	13-N36	13-N84	13-O23	13-O53	13-O93	13-P19	13-P64		
	13-P85	13-Q12	13-Q56	13-R33	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11		
	13-V48	13-V86	13-W06	13-W39	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-\66	13-_58	13-c65	13-d33	13-h32		
	13-L08	23-1	24-1	25-1												
\$MAMS1	6-0#															
\$MAMS2	6-0#															
\$MAMS3	6-0#															
\$MAMS4	6-0#															
\$MBADR	5-13#															
\$MFLG	31-1	31-1#	31-1*	31-1*												
\$MNEW	27-1	27-1#														
\$MSGAD	6-0#	31-1	31-1*													
\$MSGLG	6-0#	31-1*														
\$MSGTY	6-0#	31-1	31-1	31-1*	31-1*											
\$MSWR	27-1	27-1#														
\$MTYP1	6-0#															
\$MTYP2	6-0#															
\$MTYP3	6-0#															
\$MTYP4	6-0#															
\$MXCNT	24-1	24-1	24-1	24-1#												
\$NULL	6-0#	23-1	23-1	23-1												
\$MNTST	13-1	13-1	13-1#	13-1#	13-73	13-73	13-73#	13-73#	13-106	13-106	13-106#	13-106#	13-137	13-137		
	13-137#	13-137#	13-159	13-159	13-159#	13-159#	13-332	13-332	13-332#	13-332#	13-524	13-524	13-524#	13-524#		
	13-538	13-538	13-538#	13-538#	13-549	13-549	13-549#	13-549#	13-602	13-602	13-602#	13-602#	13-670	13-670		
	13-670#	13-670#	13-774	13-774	13-774#	13-774#	13-787	13-787	13-787#	13-787#	13-877	13-877	13-877#	13-877#		
	13-986	13-986	13-986#	13-986#	13-:09	13-:09	13-:09#	13-:09#	13-:62	13-:62	13-:62#	13-:62#	13-:90	13-:90		
	13-:90#	13-:90#	13-:10	13-:10	13-:10#	13-:10#	13-:54	13-:54	13-:54#	13-:54#	13-<16	13-<16	13-<16#	13-<16#		
	13-=15	13-=15	13-=15#	13-=15#	13-=32	13-=32	13-=32#	13-=32#	13-=90	13-=90	13-=90#	13-=90#	13->55	13->55		
	13->55#	13->55#	13-?16	13-?16	13-?16#	13-?16#	13-@18	13-@18	13-@18#	13-@18#	13-@79	13-@79	13-@79#	13-@79#		
	13-A39	13-A39	13-A39#	13-A39#	13-B20	13-B20	13-B20#	13-B20#	13-C03	13-C03	13-C03#	13-C03#	13-D10	13-D10		
	13-D10#	13-D10#	13-D69	13-D69	13-D69#	13-D69#	13-E28	13-E28	13-E28#	13-E28#	13-F08	13-F08	13-F08#	13-F08#		
	13-F43	13-F43	13-F43#	13-F43#	13-G64	13-G64	13-G64#	13-G64#	13-I14	13-I14	13-I14#	13-I14#	13-I89	13-I89		
	13-I89#	13-I89#	13-K28	13-K28	13-K28#	13-K28#	13-K74	13-K74	13-K74#	13-K74#	13-L08	13-L08	13-L08#	13-L08#		
	13-M40	13-M40	13-M40#	13-M40#	13-N00	13-N00	13-N00#	13-N00#	13-N36	13-N36	13-N36#	13-N36#	13-N84	13-N84		
	13-N84#	13-N84#	13-O23	13-O23	13-O23#	13-O23#	13-O53	13-O53	13-O53#	13-O53#	13-O93	13-O93	13-O93#	13-O93#		

	13-P19	13-P19	13-P19#	13-P19#	13-P64	13-P64	13-P64#	13-P64#	13-P85	13-P85	13-P85#	13-P85#	13-Q12	13-Q12
	13-Q12#	13-Q12#	13-Q56	13-Q56	13-Q56#	13-Q56#	13-R33	13-R33	13-R33#	13-R33#	13-R81	13-R81	13-R81#	13-R81#
	13-S32	13-S32	13-S32#	13-S32#	13-S69	13-S69	13-S69#	13-S69#	13-T25	13-T25	13-T25#	13-T25#	13-T82	13-T82
	13-T82#	13-T82#	13-U00	13-U00	13-U00#	13-U00#	13-U23	13-U23	13-U23#	13-U23#	13-U52	13-U52	13-U52#	13-U52#
	13-U81	13-U81	13-U81#	13-U81#	13-V11	13-V11	13-V11#	13-V11#	13-V48	13-V48	13-V48#	13-V48#	13-V86	13-V86
	13-V86#	13-V86#	13-W06	13-W06	13-W06#	13-W06#	13-W39	13-W39	13-W39#	13-W39#	13-W60	13-W60	13-W60#	13-W60#
	13-X09	13-X09	13-X09#	13-X09#	13-X54	13-X54	13-X54#	13-X54#	13-Y24	13-Y24	13-Y24#	13-Y24#	13-Z01	13-Z01
	13-Z01#	13-Z01#	13-166	13-166	13-166#	13-166#	13-58	13-58	13-58#	13-58#	13-c65	13-c65	13-c65#	13-c65#
	13-d33	13-d33	13-d33#	13-d33#	13-h32	13-h32	13-h32#	13-h32#	13-T08	13-T08	13-L08#	13-L08#		
SOCNT	22-1#	22-1*	22-1*											
SOMODE	22-1	22-1#	22-1*	22-1*	22-1*	22-1*								
SOVER	24-1	24-1	24-1	24-1	24-1#									
SPASS	6-0#	9-23*	14-20	14-20	14-20	14-20*	14-20*	24-1	24-1	24-1				
SPASTM	5-13#													
SPOWER	30-1	30-1#												
SPSW	7-0#	15-59*	15-89											
SPWRDN	9-23	30-1	30-1#											
SPWRMG	30-1#													
SPWRUP	30-1	30-1#												
SQUES	6-0#	23-1	23-1	25-1	25-1	27-1	27-1	27-1	27-1					
SR2A	29-1													
SRDCHR	27-1#	29-1	29-1											
SRDEEC	29-1													
SRDLIN	27-1#	29-1	29-1											
SRDOCT	28-1#	29-1	29-1											
SRDSZ	27-1	27-1#												
SRRESZ	19-1#	29-1												
SRM80	9-92	26-28	32-21#											
SRTNAD	14-20#													
SSAVR6	30-1	30-1#	30-1*	30-1*	30-1*									
SSAVRE	19-1#	29-1	29-1											
SSCOPE	9-23	24-1#												
SSETUP	4-998	4-998	4-998	4-998	4-998	4-998	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	9-23
	9-23	9-23	9-23	9-23	9-23	9-23	9-23	9-23	9-23	9-23	9-23	9-28	9-28	9-28
	14-20	14-20	24-1	25-1	25-1	25-1	25-1	27-1	27-1	27-1	27-1	27-1	27-1	
SSSTUP	4-998	4-998	4-998	4-998	4-998	4-998	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#
	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#	4-998#
SSVLAD	24-1	24-1#												
SSVPC	5-5	5-5#												
SSWOBT	24-1	24-1#												
SSWR	4-685#	4-696	4-697	4-697	4-697	4-697	4-697	4-697	4-697	4-697	6-0	6-0	6-0	9-23
	9-23	9-23	9-23	9-23	13-1	13-73	13-106	13-137	13-159	13-332	13-524	13-538	13-549	13-602
	13-670	13-774	13-787	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-=15	13-=32	13-=90
	13->55	13-?16	13-@18	13-@79	13-A39	13-B20	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64	13-I14
	13-189	13-K28	13-K74	13-L08	13-M40	13-N00	13-N36	13-N84	13-O23	13-O53	13-O93	13-P19	13-P64	13-P85
	13-Q12	13-Q56	13-R33	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11	13-V48
	13-V86	13-W06	13-W39	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-166	13-58	13-c65	13-d33	13-h32	13-l08
	14-20	14-20	14-20	14-20	14-20	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1
	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	25-1	25-1
	25-1	25-1	25-1	25-1	25-1	25-1	25-1	25-1	30-1					
SSWREG	6-0#	9-23												
SSWRPK	4-697	4-697	4-697	4-697	4-697	4-697	4-697	4-697	4-697	24-1	24-1	24-1	24-1	24-1
	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1	24-1
STESTN	6-0#	13-1*	13-73*	13-106*	13-137*	13-159*	13-332*	13-524*	13-538*	13-549*	13-602*	13-670*	13-774*	13-787*
	13-877*	13-986*	13-:09*	13-:62*	13-:90*	13-:10*	13-:54*	13-<16*	13-=15*	13-=32*	13-=90*	13->55*	13-?16*	13-@18*
	13-@79*	13-A39*	13-B20*	13-C03*	13-D10*	13-D69*	13-E28*	13-F08*	13-F43*	13-G64*	13-I14*	13-K28*	13-K74*	



STRPAD	29-1	29-1#													
STSTM	5-13#														
STSTNM	6-0#	11-46*	14-20*	24-1	24-1	24-1	24-1	24-1	24-1*	24-1*	25-1	25-1	25-1		
STTYIN	27-1	27-1	27-1	27-1	27-1	27-1#									
STYPBN	20-1#	29-1	29-1												
STYPDS	21-1#	29-1	29-1												
STYPE	23-1#	29-1	29-1	31-1											
STYPEC	23-1	23-1	23-1	23-1#	27-1										
STYPEX	23-1	23-1	23-1#												
STYPOC	22-1#	29-1	29-1												
STYPON	22-1	22-1#	29-1												
STYPOS	22-1#	29-1													
SUNIT	6-0#	11-50*	11-63	26-22											
SUNITM	5-13#														
SUSWR	6-0#														
SVECT1	6-0#	10-46	10-56	10-63	10-64*										
SVECT2	6-0#														
SXOFF	23-1	23-1													
SXON	23-1	23-1	27-1												
SXTSTR	24-1#														
.SASTA	31-1	31-1													
.SX	5-13	5-13#													
A16	4-944#	13-<98													
A17	4-943#	13-<98													
ABASE	4-995#	6-0	6-0												
ACDW1	6-0	6-0													
ACDW2	6-0	6-0													
ACPLUOP	6-0	6-0													
ADDW0	6-0	6-0													
ADDW1	6-0	6-0													
ADDW10	6-0														
ADDW11	6-0														
ADDW12	6-0														
ADDW13	6-0														
ADDW14	6-0														
ADDW15	6-0														
ADDW2	6-0	6-0													
ADDW3	6-0	6-0													
ADDW4	6-0	6-0													
ADDW5	6-0	6-0													
ADDW6	6-0	6-0													
ADDW7	6-0	6-0													
ADDW8	6-0														
ADDW9	6-0														
ADEVCT	6-0	6-0													
ADEVVM	6-0	6-0													
ADR	13-1	13-1#	13-1#	13-73	13-73#	13-73#	13-73#	13-106	13-106#	13-106#	13-137	13-137#	13-137#	13-159	13-159#
	13-159#	13-332	13-332#	13-332#	13-524	13-524#	13-524#	13-538	13-538#	13-538#	13-538#	13-549	13-549#	13-549#	13-602
	13-602#	13-602#	13-670	13-670#	13-670#	13-774	13-774#	13-774#	13-787	13-787#	13-787#	13-787#	13-877	13-877#	13-877#
	13-986	13-986#	13-986#	13-:09	13-:09#	13-:09#	13-:62	13-:62#	13-:62#	13-:62#	13-:90	13-:90#	13-:90#	13-:10	13-:10#
	13-:10#	13-:54	13-:54#	13-:54#	13-<16	13-<16#	13-<16#	13-=15	13-=15#	13-=15#	13-=15#	13-=32	13-=32#	13-=32#	13-=90
	13-=90#	13-=90#	13->55	13->55#	13->55#	13-?16	13-?16#	13-?16#	13-@18	13-@18#	13-@18#	13-@18#	13-@79	13-@79#	13-@79#
	13-A39	13-A39#	13-A39#	13-B20	13-B20#	13-B20#	13-C03	13-C03#	13-C03#	13-C03#	13-D10	13-D10#	13-D10#	13-D69	13-D69#
	13-D69#	13-E28	13-E28#	13-E28#	13-F08	13-F08#	13-F08#	13-F43	13-F43#	13-F43#	13-F43#	13-G64	13-G64#	13-G64#	13-114
	13-114#	13-114#	13-189	13-189#	13-189#	13-K28	13-K28#	13-K28#	13-K74	13-K74#	13-K74#	13-K74#	13-L08	13-L08#	13-L08#
	13-M40	13-M40#	13-M40#	13-N00	13-N00#	13-N00#	13-N36	13-N36#	13-N36#	13-N36#	13-N84	13-N84#	13-N84#	13-N84#	13-023

	13-023#	13-053	13-053#	13-053#	13-093	13-093#	13-093#	13-P19	13-P19#	13-P19#	13-P64	13-P64#	13-P64#	13-P85
	13-P85#	13-P85#	13-Q12	13-Q12#	13-Q12#	13-Q56	13-Q56#	13-Q56#	13-R33	13-R33#	13-R33#	13-R81	13-R81#	13-R81#
	13-S32	13-S32#	13-S32#	13-S69	13-S69#	13-S69#	13-T25	13-T25#	13-T25#	13-T82	13-T82#	13-T82#	13-U00	13-U00#
	13-U00#	13-U23	13-U23#	13-U23#	13-U52	13-U52#	13-U52#	13-U81	13-U81#	13-U81#	13-V11	13-V11#	13-V11#	13-V48
	13-V48#	13-V48#	13-V86	13-V86#	13-V86#	13-W06	13-W06#	13-W06#	13-W39	13-W39#	13-W39#	13-W60	13-W60#	13-W60#
	13-X09	13-X09#	13-X09#	13-X54	13-X54#	13-X54#	13-Y24	13-Y24#	13-Y24#	13-Z01	13-Z01#	13-Z01#	13-166	13-166#
	13-166#	13-58	13-58#	13-58#	13-c65	13-c65#	13-c65#	13-d33	13-d33#	13-d33#	13-h32	13-h32#	13-h32#	13-108
AENV	6-0	6-0												
AENV#	6-0	6-0												
AFATAL	6-0	6-0												
ALL	10-77	32-4#												
AMADR1	6-0	6-0												
AMADR2	6-0	6-0												
AMADR3	6-0	6-0												
AMADR4	6-0	6-0												
AMAMS1	6-0	6-0												
AMAMS2	6-0	6-0												
AMAMS3	6-0	6-0												
AMAMS4	6-0	6-0												
AMSGAD	6-0	6-0												
AMSGLG	6-0	6-0												
AMSGTY	6-0	6-0												
AMTYP1	6-0	6-0												
AMTYP2	6-0	6-0												
AMTYP3	6-0	6-0												
AMTYP4	6-0	6-0												
AOE	4-787#	4-798	13-688	13-717	13-N64	13-T04	13-T08	33-77	33-78	33-81	33-82	33-85	33-86	
APASS	6-0	6-0												
APE	4-973#													
APRIOR	6-0													
APTCSU	23-1	31-1#												
APTENV	23-1	25-1	31-1	31-1#										
APTSIZ	9-23	31-1#												
APTSPO	23-1	31-1	31-1#											
ASWREG	6-0	6-0												
ATA	4-767#	13-V66	13-V97	13-W18	13-W22	13-W30	13-W34	13-W51	13-W53	13-W76	13-W99	13-X31	13-X35	13-144
	13-146	13-21	13-23	13-a33	13-a35	33-58	33-59	33-60	33-63	33-64	33-67	33-68	33-69	33-70
	33-71	33-72	33-73	33-74	33-75	33-76	33-79	33-80	33-83	33-84	33-87	33-88		
ATESTN	6-0	6-0												
ATNMSK	4-804#	13-V61												
ATNTBL	9-79	9-109	10-105	11-17	13-V60	34-3#								
AUNIT	6-0	6-0												
AUSWR	6-0	6-0												
AUTSIZ	7-0#	9-65*	9-107	10-108*										
AVECT1	4-996#	6-0	6-0											
AVECT2	6-0	6-0												
BADTMO	9-3#	9-25												
BAI	4-962#													
BB00	4-889#													
BB01	4-889#													
BB02	4-889#	13-m40												
BB03	4-889#	13-m36	13-m39											
BB04	4-889#													
BB05	4-889#													
BB06	4-889#	13-150	13-151	13-152	13-153	13-154	13-155	13-156	13-157	13-158	13-159			

BB07	4-889#													
BB08	4-889#													
BB09	4-889#													
BIT0	4-700#	13-211	13-567	13-632	13-715	13-723	13-804	13-812	13-825	13-898	13-906	13-914	13-927	13-935
	13-947	13-25	13-68	13-78	13-55	13-04	13-69	13-33	13-33	13-94	13-A52	13-A63	13-J33	
BIT00	4-700	4-700#	4-712	4-759	4-777	4-796	4-633	4-851	4-889	4-965	4-983	24-1	24-1	25-1
	25-1													
BIT01	4-700	4-700#	4-711	4-758	4-795	4-832	4-850	4-899	4-965	4-982				
BIT02	4-700	4-700#	4-710	4-757	4-794	4-831	4-849	4-889	4-965	4-981				
BIT03	4-700	4-700#	4-709	4-756	4-793	4-830	4-848	4-889	4-901	4-962	4-980			
BIT04	4-700	4-700#	4-708	4-755	4-792	4-847	4-889	4-961						
BIT05	4-700	4-700#	4-707	4-791	4-829	4-846	4-889	4-900	4-960					
BIT06	4-700	4-700#	4-776	4-790	4-812	4-828	4-845	4-889	4-946	4-959	4-979			
BIT07	4-700	4-700#	4-775	4-789	4-811	4-827	4-844	4-870	4-889	4-879	4-945	4-958		
BIT08	4-700	4-700#	4-752	4-774	4-788	4-810	4-826	4-843	4-889	4-944	4-957	24-1		
BIT09	4-700	4-700#	4-751	4-773	4-787	4-809	4-825	4-842	4-869	4-889	4-943	4-956	24-1	24-1
	25-1	25-1												
BIT1	4-700#	13-256	13-579	13-646	13-745	13-835	13-957	13-36	13-90	13-66	13-73	13-15	13-80	13-62
	13-244	13-A04	13-A74											
BIT10	4-700#	4-750	4-772	4-786	4-808	4-824	4-841	4-868	4-886	4-898	4-942	4-955	4-978	25-1
BIT11	4-700#	4-706	4-749	4-771	4-785	4-823	4-840	4-859	4-867	4-885	4-897	4-954	4-977	24-1
BIT12	4-700#	4-770	4-784	4-822	4-839	4-866	4-884	4-896	4-953	4-976				
BIT13	4-700#	4-769	4-783	4-821	4-838	4-858	4-883	4-895	4-908	4-941	4-952	4-975	25-1	
BIT14	4-700#	4-768	4-782	4-820	4-837	4-857	4-882	4-894	4-907	4-940	4-951	4-974	24-1	
BIT15	4-700#	4-767	4-781	4-819	4-836	4-856	4-881	4-893	4-906	4-939	4-950	4-973		
BIT2	4-700#													
BIT3	4-700#													
BIT4	4-700#													
BIT5	4-700#													
BIT6	4-700#													
BIT7	4-700#	9-72												
BIT8	4-700#													
BIT9	4-700#													
BLNKS1	32-33#													
BLNKS2	9-123	10-34	10-48	32-32#										
BLNKS3	32-31#													
BLNKS4	9-83	32-30#												
BOTADR	26-73*	26-91*	26-94	26-109	26-153#									
BOTFLG	26-54*	26-101*	26-104	26-107*	26-154#									
BPTVEC	4-700#													
BSE	4-893#													
BUFFER	13-587	13-120	43-1#	43-5										
BUFONE	43-2#													
EIF TWO	43-3#													
CC	4-885#	13-150	13-151	13-152	13-153	13-154	13-155	13-156	13-157	13-158	13-159	13-42	13-43	13-44
	13-45	13-46	13-47	13-48	13-49	13-50	13-51	13-c43	13-c44	13-c45	13-c46	13-c48	13-c49	13-c50
	13-c51	13-c55	13-f17	13-f85	13-f86	13-m20	13-m21	13-m32	13-m80	13-m81	13-n05	13-n06	13-n07	13-n08
	13-n09	13-n10	13-n11	13-n12	13-n13	13-n14	13-n48	13-n49	13-n50	13-n51	13-n52	13-n53	13-n54	
CH	4-886#	13-150	13-151	13-152	13-153	13-154	13-155	13-156	13-157	13-158	13-159	13-f13	13-g93	13-g94
	13-g95	13-g96	13-g98	13-g99	13-h00	13-h01	13-h05	13-l85	13-l86	13-m20	13-m21	13-m30	13-m80	13-m81
	13-n16	13-n17	13-n18	13-n19	13-n20	13-n21	13-n22	13-n23	13-n24	13-n25	13-n48	13-r49	13-n50	13-n51
	13-n52	13-n53	13-n54											
CHGADR	7-0#	9-14*	9-17*	10-25	10-27*									
CHRCNT	26-55*	26-78*	26-84*	26-85	26-88*	26-92	26-97*	26-108*	26-155#					
CKSWR	24-1	25-1	25-1	29-1#										
CLOCK	7-0#	13-K93	13-Q30	13-Y68	13-d17	13-f42	13-g14	13-h49	13-h93	13-i80	13-j43	13-k29	13-l62	15-13*

CLR	15-27*	15-39*													
CMNSTA	4-960#	9-85	11-54	18-10											
CNSLO1	9-132	11-2#													
CNSLO2	10-32	32-8#													
CNSLO3	10-40	32-9#													
CNSLO4	10-44	32-10#													
CNSLO7	10-54	32-11#													
CNSLO8	10-69	32-12#													
CNSLO9	10-100	32-14#													
CNTCLR	32-15#														
	13-7	13-28	13-75	13-108	13-116	13-139	13-162	13-213	13-262	13-370	13-386	13-408	13-430	13-447	
	13-469	13-485	13-500	13-540	13-551	13-605	13-611	13-677	13-754	13-776	13-889	13-965	13-988	13-:21	
	13-:73	13-:92	13-:13	13-:57	13-<23	13-=17	13-=23	13-=39	13-=92	13->61	13-?18	13-@20	13-@81	13-A47	
	13-B29	13-C19	13-C63	13-D19	13-D29	13-D85	13-E35	13-E60	13-E84	13-F15	13-F51	13-G71	13-I21	13-I97	
	13-K36	13-K80	13-L98	13-M48	13-N87	13-O30	13-O58	13-Q21	13-T33	13-T85	13-U03	13-V51	13-V88	13-W09	
	13-W42	13-W66	13-Y42	16-14	18-8#										
COMMA	10-94	11-6	11-14	32-6#											
CONT	4-845#	13-J05	13-J31	13-J35											
CPSAVE	24-1	24-1	24-1	24-1*	24-1*	25-1	25-1	25-1	25-1	25-1	25-1#	25-1*	25-1*	26-172	
CR	4-700#	10-81	10-92	23-1	23-1	26-76	40-98	40-147	40-148	40-149	40-150	40-151	40-152	40-153	
	40-154	40-155	40-156	40-157	40-158	40-158	40-159								
CRLF	4-700#	9-6	9-28	9-28	9-42	9-53	9-59	11-30	23-1	23-1	32-4	32-7	32-8	32-9	
	32-11	32-12	32-13	32-14	32-15	32-16	32-18	32-20	41-1	41-3	41-5	41-8	41-10	41-:2	
	41-14	41-16	41-18	41-20	41-22	41-25	41-27	41-29	43-8	43-9	43-10	43-11	43-?2	43-13	
	43-14	43-15	43-16	43-17	43-18	43-19	43-20	43-21	43-22	43-23	43-24	43-25	43-26	43-27	
	43-28	43-29	43-30	43-31	43-32	43-33	43-34	43-35	43-36	43-37	43-38	43-39	43-40	43-41	
	43-42	43-43	43-44	43-45	43-46	43-47	43-48	43-49	43-50	43-51	43-52	43-53	43-54	43-55	
	43-56	43-57	43-58	43-59	43-60	43-61	43-62	43-63	43-64	43-65	43-66	43-67	43-68	43-69	
	43-70	43-71	43-72	43-73	43-74	43-75	43-76	43-77	43-78	43-79	43-80	43-81	43-82	43-83	
	43-84	43-85	43-86	43-87	43-88	43-89	43-90	43-91	43-92	43-93	43-94	43-95	43-96	43-97	
	43-98	43-99	43-100	43-101	43-102	43-103	43-104	43-105	43-106	43-107	43-108	43-109	43-110	43-111	
	43-112														
CYLMASK	4-875#	13-80													
DBCK	4-819#	13-F79	13-H10	13-I47	13-J28	13-K54	13-L18	13-L39	13-L60	13-M67	13-N13	13-N55	13-O02	13-O40	
	13-069	13-P07	13-P41	13-Q80	13-R55	13-S01	13-S46	13-S93	13-V23	13-X24	13-X72	13-X74	13-Y61	13-Z18	
	13-Z26	13-Z57	13-Z67	13-Z95	13-[25	13-[37	13-[68	13-[79	13-[97	13-^41	13-^86	13-^94	13-]28	13-]38	
	13-J69	13-^02	13-^14	13-^48	13-^59	13-^77	13-^26	13-^78	13-^86	13-^20	13-^30	13-^64	13-^98	13-a13	
	13-a26	13-a60	13-a71	13-a89	13-b19	13-b34	13-b45	13-b60	13-b98	13-c24	13-c81	13-c92	13-d00	13-d51	
	13-d61	13-d91	13-e01	13-e27	13-e54	13-e83	13-f01	13-f61	13-f71	13-g30	13-g48	13-g67	13-h65	13-i09	
	13-i19	13-i26	13-i36	13-i96	13-j08	13-j59	13-j71	13-j81	13-j91	13-k45	13-k56	13-k64	13-k83	13-l96	
	13-m53	13-m86													
DBEN	4-820#	4-852	13-C26	13-C31	13-C32	13-C67	13-C71	13-C72	13-D34	13-D38	13-D39	13-E06	13-E07	13-F17	
	13-F53	13-F79	13-F80	13-G73	13-H10	13-H11	13-I23	13-I47	13-I48	13-I99	13-J28	13-J29	13-K38	13-K54	
	13-K55	13-K82	13-L18	13-L19	13-L39	13-L40	13-L60	13-L61	13-M00	13-M50	13-M67	13-M68	13-N09	13-N13	
	13-N14	13-N47	13-N55	13-N56	13-N89	13-O02	13-O03	13-O32	13-O40	13-O41	13-O60	13-O69	13-O70	13-P04	
	13-P07	13-P08	13-P38	13-P41	13-P42	13-Q01	13-Q03	13-Q04	13-Q23	13-Q70	13-Q80	13-Q81	13-R48	13-R55	
	13-R56	13-R97	13-S01	13-S02	13-S43	13-S46	13-S47	13-S81	13-S93	13-S94	13-S99	13-T00	13-T35	13-T43	
	13-V20	13-V23	13-V24	13-V52	13-X18	13-X24	13-X25	13-X64	13-Y58	13-Y61	13-Y62	13-Z11	13-Z18	13-Z19	
	13-Z24	13-Z26	13-Z27	13-Z49	13-Z57	13-Z58	13-Z63	13-Z67	13-Z68	13-Z87	13-Z95	13-Z96	13-[18	13-[25	
	13-[26	13-[31	13-[32	13-[37	13-[38	13-[60	13-[68	13-[69	13-[74	13-[79	13-[80	13-[93	13-[97	13-[98	
	13-^17	13-^41	13-^42	13-^77	13-^86	13-^87	13-^92	13-^94	13-^95	13-]18	13-]28	13-]29	13-]34	13-]38	
	13-]39	13-]59	13-]69	13-]70	13-]93	13-^02	13-^03	13-^08	13-^09	13-^14	13-^15	13-^38	13-^48	13-^49	
	13-^54	13-^59	13-^60	13-^73	13-^77	13-^78	13-^99	13-^26	13-^27	13-^69	13-^78	13-^79	13-^84	13-^86	
	13-^87	13-^10	13-^20	13-^21	13-^26	13-^30	13-^31	13-^54	13-^64	13-^65	13-^89	13-^98	13-^99	13-a05	
	13-a08	13-a13	13-a14	13-a20	13-a21	13-a22	13-a26	13-a27	13-a50	13-a60	13-a61	13-a66	13-a71	13-a72	
	13-a85	13-a89	13-a90	13-b10	13-b19	13-b20	13-b26	13-b29	13-b34	13-b35	13-b45	13-b46	13-b56	13-b60	





DULPRT	4-862#	13-529												
DVA	4-706#	9-90	13-542	13-544	13-U05	13-U07								
DVC	4-899#	13-152	13-972	13-974	13-?30	13-?53	13-?57	13-?94	13-?98					
EBL	4-838#	13-A49	13-A60	13-A70	13-A72	13-A87	13-A89	13-e12	13-e17	13-e43	13-f82	13-f87	13-147	13-152
	13-k02	13-k07	13-k96	13-L01										
ECH	4-790#	4-798	13-688	13-717	33-77	33-78	33-85	33-86						
ECI	4-867#													
ECRC	4-842#													
ED1	38-1	42-1#												
ED115	38-8	38-10	42-7#											
ED130	38-9	42-8#												
ED2	38-2	38-7	42-2#											
ED220	38-11	42-9#												
ED5	38-3	42-3#												
ED57	38-4	42-4#												
ED65	38-5	42-5#												
ED71	38-6	42-6#												
EDT1	8-3	8-49	8-52	8-55	8-58	8-61	8-64	8-70	8-73	8-76	8-79	8-82	8-85	8-88
	8-91	8-94	8-97	8-100	8-106	8-109	8-112	8-115	8-118	8-121	8-124	8-127	8-130	8-133
	8-136	8-139	8-145	8-148	8-151	8-154	8-157	8-166	8-169	8-184	8-187	8-190	8-196	8-199
	8-205	8-208	8-214	8-217	8-220	8-223	8-226	8-229	8-235	8-238	8-244	8-247	8-253	8-256
	8-259	8-277	8-280	8-283	8-286	8-298	8-301	8-307	8-310	8-316	8-319	8-322	8-325	8-328
	8-334	8-340	8-343	8-346	8-349	8-355	8-358	8-367	8-376	8-379	8-385	8-388	8-391	8-394
	8-397	8-424	8-427	8-430	8-445	8-448	8-451	8-454	8-457	8-460	8-473	8-479	8-485	8-488
	8-492	8-495	8-498	8-501	8-504	8-508	8-511	8-515	8-518	8-521	8-524	8-527	8-530	8-533
	8-537	8-540	8-543	8-546	8-549	8-552	8-555	8-558	8-561	8-564	8-567	8-570	8-573	8-576
	8-580	8-583	8-586	8-589	38-1#									
EDT115	8-232	8-241	8-250	8-262	8-313	8-331	8-337	8-361	8-364	8-370	8-400	8-406	8-409	8-412
	8-418	8-421	8-464	8-467	8-470	8-476	8-482	38-8#						
EDT130	8-265	8-268	8-304	38-9#										
EDT132	8-271	8-274	8-289	8-292	8-295	8-415	38-10#							
EDT2	8-6	38-2#												
EDT220	8-433	8-436	8-439	8-442	38-11#									
EDT5	8-15	8-18	8-22	38-3#										
EDT57	8-142	38-4#												
EDT65	8-160	8-163	8-193	8-202	8-211	38-5#								
EDT71	8-172	8-175	8-178	38-6#										
EDT74	8-181	38-7#												
EECC	4-847#													
E:1	39-1	39-5	39-6	42-11#										
EF130	39-9	42-15#												
EF2	39-2	39-7	42-12#											
EF5	39-3	39-11	42-13#											
EF57	39-4	39-8	39-10	42-14#										
EFT1	8-3	8-49	8-52	8-55	8-58	8-61	8-64	8-70	8-73	8-76	8-79	8-82	8-85	8-88
	8-91	8-94	8-97	8-100	8-106	8-109	8-112	8-115	8-118	8-121	8-124	8-127	8-130	8-133
	8-136	8-139	8-145	8-148	8-151	8-154	8-157	8-166	8-169	8-184	8-187	8-190	8-196	8-199
	8-205	8-208	8-214	8-217	8-220	8-223	8-226	8-229	8-235	8-238	8-244	8-247	8-253	8-256
	8-259	8-277	8-280	8-283	8-286	8-298	8-301	8-307	8-310	8-316	8-319	8-322	8-325	8-328
	8-334	8-340	8-343	8-346	8-349	8-355	8-358	8-367	8-376	8-379	8-385	8-388	8-391	8-394
	8-397	8-424	8-427	8-430	8-445	8-448	8-451	8-454	8-457	8-460	8-473	8-479	8-485	8-488
	8-492	8-495	8-498	8-501	8-504	8-508	8-511	8-515	8-518	8-521	8-524	8-527	8-530	8-533
	8-537	8-540	8-543	8-546	8-549	8-552	8-555	8-558	8-561	8-564	8-567	8-570	8-573	8-576
	8-580	8-583	8-586	8-589	39-1#									
EFT115	8-232	8-241	8-250	8-262	8-313	8-331	8-337	8-361	8-364	8-370	8-400	8-406	8-409	8-412
	8-418	8-421	8-464	8-467	8-470	8-476	8-482	39-8#						





EMS330	36-51	40-98#												
EMS331	36-79	36-96	36-189	36-205	40-99#									
EMS332	36-106	36-150	36-157	36-172	36-179	36-186	36-196	36-333	36-362	36-378	36-396	36-413	40-100#	
EMS333	36-108	36-126	36-129	36-162	36-168	36-247	36-258	36-261	36-307	36-313	36-315	36-317	36-323	40-101#
EMS334	36-131	36-162	36-166	36-252	36-302	36-307	40-102#							
EMS335	36-120	36-129	36-252	36-254	36-254	36-261	36-304	36-352	36-361	36-368	36-377	36-384	36-393	40-103#
EMS336	36-122	36-131	36-348	36-353	36-364	36-369	36-380	36-385	36-400	36-404	36-415	36-419	40-104#	
EMS337	36-140	36-146	36-153	36-164	36-166	36-175	36-182	36-192	36-217	36-221	36-231	36-250	36-252	36-254
	36-256	36-263	36-279	36-291	36-300	36-305	40-105#							
EMS34	36-203	36-265	40-31#											
EMS340	36-129	36-131	36-261	36-348	36-353	36-364	36-369	36-380	36-385	36-400	36-415	40-106#		
EMS341	36-144	36-146	36-151	36-153	36-158	36-164	36-166	36-173	36-175	36-180	36-182	36-192	36-201	36-203
	36-213	36-217	36-221	36-272	36-274	36-276	36-279	36-281	36-283	36-287	36-319	36-422	40-107#	
EMS342	36-120	36-122	36-129	36-131	36-135	36-333	40-108#							
EMS343	36-135	40-109#												
EMS344	36-135	40-110#												
EMS345	36-138	36-140	36-215	36-285	36-302	40-111#								
EMS346	36-138	36-144	36-151	36-158	36-173	36-180	36-215	36-281	36-283	36-285	36-287	36-319	36-321	40-112#
EMS347	36-142	36-148	36-155	36-170	36-177	36-184	36-194	40-113#						
EMS35	36-207	36-247	36-252	36-256	36-263	40-32#								
EMS350	36-160	36-162	36-166	36-168	36-307	36-323	36-341	36-346	40-114#					
EMS351	36-160	40-115#												
EMS352	36-168	40-116#												
EMS353	36-189	36-229	36-235	36-237	36-241	36-245	36-250	36-269	36-335	40-117#				
EMS354	36-197	36-199	36-223	36-229	36-233	36-241	36-245	36-269	36-289	36-294	36-296	36-298	36-337	40-118#
EMS355	36-201	36-203	36-213	40-119#										
EMS356	36-327	36-329	36-331	36-398	40-120#									
EMS357	36-207	36-325	40-121#											
EMS36	36-211	36-225	36-231	36-233	36-235	36-237	36-261	36-325	36-352	36-357	36-361	36-368	36-373	36-377
	36-384	36-389	36-393	36-409	40-33#									
EMS360	36-207	36-327	40-122#											
EMS361	36-329	40-123#												
EMS362	36-209	40-124#												
EMS363	36-219	36-309	36-311	36-323	40-125#									
EMS364	36-223	36-296	36-298	40-126#										
EMS365	36-223	36-229	36-235	36-237	36-241	36-245	36-269	36-333	36-335	36-337	40-127#			
EMS366	36-227	40-128#												
EMS367	36-229	36-235	36-237	36-241	36-245	36-269	36-335	40-129#						
EMS37	36-213	40-34#												
EMS370	36-235	36-357	36-373	36-389	36-405	40-130#								
EMS371	36-237	36-407	40-131#											
EMS372	36-252	36-254	36-258	36-325	36-337	36-341	36-346	40-132#						
EMS373	36-285	36-302	40-133#											
EMS374	36-291	40-134#												
EMS375	36-317	40-135#												
EMS376	36-315	36-317	36-321	36-341	40-136#									
EMS377	36-333	36-337	36-341	36-350	36-359	36-366	36-375	36-382	36-391	36-402	36-417	40-137#		
EMS4	36-79	36-96	36-205	40-6#										
EMS40	36-215	36-217	40-35#											
EMS400	36-346	40-138#												
EMS401	36-335	36-342	36-346	36-355	36-371	36-387	36-394	36-411	36-420	40-139#				
EMS402	36-350	36-359	36-366	36-375	36-382	36-391	36-402	36-417	40-140#					
EMS403	36-350	36-359	36-366	36-375	36-382	36-391	36-402	36-417	40-141#					
EMS404	36-353	36-369	36-385	36-409	36-415	40-142#								
EMS405	36-348	36-353	36-355	36-357	36-362	36-364	36-369	36-371	36-373	36-378	36-380	36-385	36-387	36-389
	36-394	36-396	36-400	36-405	36-407	36-409	36-411	36-413	36-415	36-420	40-143#			





EMT136	8-283	36-209#
EMT137	8-286	36-211#
EMT14	8-37	36-25#
EMT140	8-289	36-213#
EMT141	8-292	36-215#
EMT142	8-295	36-217#
EMT143	8-298	36-219#
EMT144	8-301	36-221#
EMT145	8-304	36-223#
EMT146	8-307	36-225#
EMT147	8-310	36-227#
EMT15	8-40	36-27#
EMT150	8-313	36-229#
EMT151	8-316	36-231#
EMT152	8-319	36-233#
EMT153	8-322	36-235#
EMT154	8-325	36-237#
EMT155	8-328	36-239#
EMT156	8-331	36-241#
EMT157	8-334	36-243#
EMT16	8-43	36-29#
EMT160	8-337	36-245#
EMT161	8-340	36-248#
EMT162	8-343	36-250#
EMT163	8-346	36-252#
EMT164	8-349	36-254#
EMT165	8-352	36-256#
EMT166	8-355	36-259#
EMT167	8-358	36-261#
EMT17	8-46	36-31#
EMT170	8-361	36-263#
EMT171	8-364	36-265#
EMT172	8-367	36-267#
EMT173	8-370	36-269#
EMT174	8-373	36-271#
EMT175	8-376	36-274#
EMT176	8-379	36-276#
EMT177	36-278#	
EMT2	8-6	36-4#
EMT20	8-49	36-33#
EMT200	8-385	36-279#
EMT201	8-388	36-281#
EMT202	8-391	36-283#
EMT203	8-394	36-285#
EMT204	8-397	36-287#
EMT205	8-400	36-289#
EMT206	8-403	36-291#
EMT207	8-406	36-294#
EMT21	8-52	36-36#
EMT210	8-409	36-296#
EMT211	8-412	36-298#
EMT212	8-415	36-300#
EMT213	8-418	36-302#
EMT214	8-421	36-305#
EMT215	8-424	36-307#
EMT216	8-427	36-309#



EMT217	8-430	36-311#	
EMT22	8-55	36-39#	
EMT220	8-433	36-313#	
EMT221	8-436	36-315#	
EMT222	8-439	36-317#	
EMT223	8-442	36-319#	
EMT224	8-445	36-321#	
EMT225	8-448	36-323#	
EMT226	8-451	36-325#	
EMT227	8-454	36-327#	
EMT23	8-58	36-41#	
EMT230	8-457	36-329#	
EMT231	8-460	36-331#	
EMT232	8-464	36-333#	
EMT233	8-467	36-335#	
EMT234	8-470	36-337#	
EMT235	8-473	36-339#	
EMT236	8-476	36-341#	
EMT237	8-479	8-482	36-344#
EMT24	8-61	36-44#	
EMT240	36-346#		
EMT241	8-485	36-348#	
EMT242	8-488	36-350#	
EMT243	8-492	36-353#	
EMT244	8-495	36-355#	
EMT245	8-498	36-357#	
EMT246	8-501	36-359#	
EMT247	8-504	36-362#	
EMT25	8-64	36-47#	
EMT250	8-508	36-364#	
EMT251	8-511	36-366#	
EMT252	8-515	36-369#	
EMT253	8-518	36-371#	
EMT254	8-521	36-373#	
EMT255	8-524	36-375#	
EMT256	8-527	36-378#	
EMT257	8-530	36-380#	
EMT26	8-67	36-49#	
EMT260	8-533	36-382#	
EMT261	8-537	36-385#	
EMT262	8-540	36-387#	
EMT263	8-543	36-389#	
EMT264	8-546	36-391#	
EMT265	8-549	36-394#	
EMT266	8-552	36-396#	
EMT267	8-555	36-398#	
EMT27	8-70	36-52#	
EMT270	8-558	36-400#	
EMT271	8-561	36-402#	
EMT272	8-564	36-405#	
EMT273	8-567	36-407#	
EMT274	8-570	36-409#	
EMT275	8-573	36-411#	
EMT276	8-576	36-413#	
EMT277	8-580	36-415#	
EMT3	8-9	36-6#	

EMT30	8-73	36-54#													
EMT300	8-583	36-417#													
EMT301	8-586	36-420#													
EMT302	8-589	36-422#													
EMT31	8-76	36-56#													
EMT32	8-79	36-58#													
EMT33	8-82	36-60#													
EMT34	8-85	36-62#													
EMT35	8-88	36-64#													
EMT36	8-91	36-66#													
EMT37	8-94	36-68#													
EMT4	8-12	36-8#													
EMT40	8-97	36-70#													
EMT41	8-100	36-73#													
EMT42	8-103	36-75#													
EMT43	8-106	36-77#													
EMT44	8-109	36-79#													
EMT45	8-112	36-81#													
EMT46	8-115	36-83#													
EMT47	8-118	36-85#													
EMT5	8-15	36-10#													
EMT50	8-121	36-88#													
EMT51	8-124	36-91#													
EMT52	8-127	36-94#													
EMT53	8-130	36-96#													
EMT54	8-133	36-98#													
EMT55	8-136	36-101#													
EMT56	8-139	36-104#													
EMT57	8-142	36-106#													
EMT6	8-18	36-13#													
EMT60	8-145	36-108#													
EMT61	8-148	36-110#													
EMT62	8-151	36-113#													
EMT63	8-154	36-116#													
EMT64	8-157	36-118#													
EMT65	8-160	36-120#													
EMT66	8-163	36-122#													
EMT67	8-166	36-124#													
EMT7	8-22	36-15#													
EMT70	8-169	36-127#													
EMT71	8-172	36-129#													
EMT72	8-175	36-131#													
EMT73	8-178	36-133#													
EMT74	8-181	36-135#													
EMT75	8-184	36-136#													
EMT76	8-187	36-138#													
EMT77	8-190	36-140#													
EMTVEC	4-700#	9-23*	9-23*												
ENRGDT	35-144#														
EQUALS	32-3#														
ERP	4-768#	13-E40	13-E44	13-E50	13-E66	13-E91	13-E92	13-M07							
ERRNMB	26-40*	26-41*	26-44	26-49	26-56	26-152#									
ERROR	4-700#														
ERRTYP	25-1	26-13#													
ERRVEC	4-700#	9-23	9-23*	9-23*	9-25*	9-26*	13-<37	13-<38	13-<39*	13-<40*	13-<46*	13-<47*	13-<51*	13-<52*	
	13-<80	13-<81	13-<82*	13-<83*	13-=03*	13-=04*	13-=11*	13-=12*	15-6	15-7	15-8*	15-9*	15-25*	15-45*	



ILF64	4-739#	13-nj7													
ILF66	4-740#	13-100													
ILF74	4-743#	13-103													
ILF76	4-744#	13-78	13-93	13-111	13-124	13-165	13-189	13-222	13-238	13-433	13-437	13-609	13-613	13-623	
	13-626	13-637	13-639	13-640	13-655	13-659	13-F38	13-G79	13-106	13-129	13-K68	13-M62	13-N31	13-N78	
	13-047	13-072	13-R76												
	4-795#	13-680	13-709	13-<25	13-<60	13-<64	13-<96								
ILR															
ILRG50	4-930#														
ILRG52	4-930#														
ILRG54	4-930#														
ILRG56	4-930#														
ILRG60	4-930#														
ILRG62	4-930#														
ILRG64	4-930#														
ILRG66	4-930#														
ILRG70	4-930#														
ILRG72	4-930#														
ILRG74	4-930#														
ILRG76	4-930#														
IOTVEC	4-700#	9-23*	9-23*												
IPCK0	4-983#														
IPCK1	4-982#														
IPCK2	4-981#														
IPCK3	4-980#														
IR	4-959#	13-13	13-34												
IRP	5-8	10-62	15-79#												
IVC	4-896#	13-903	13-924	13-K42	13-K57	13-K59	33-58	33-59	33-60	33-62	33-63	33-64	33-67	33-68	
	33-69	33-70	33-71	33-72	33-73	33-74	33-75	33-76	33-77	33-78	33-79	33-80	33-81	33-82	
	33-83	33-84	33-85	33-86	33-87	33-88									
LBC	4-898#	13-911	13-932												
LBT	4-772#	13-D91	13-E09	13-E11											
LCLOCK	15-27	15-55#													
LCOUNT	15-29	15-69#													
LF	4-700#	23-1	23-1	26-80	32-11	40-98	40-147	40-148	40-149	40-150	40-151	40-152	40-153	40-154	
	40-155	40-156	40-157	40-158	40-158	40-159									
LODEV	9-119	32-23#													
LS	4-849#	13-B35	13-B50												
LSC	4-897#	13-903	13-924	13-K86	13-L01	13-L03									
LST	4-850#	13-B58	13-B72												
LSTOP	15-28	15-86#													
LSTRK	7-0#	11-56*	13-B26	13-C84	13-D16	13-D75	13-P87	13-R83	13-S72	13- _79	13-L15				
MCLK	4-823#	13-e09	13-e35	13-f79	13-144	13-j99	13-k91								
MCPE	4-941#	13-:79													
MDF	4-828#	13-?55	13-?96	13-263	13-267	13-268	13-J34	13-J38	13-J39	13-'26	13-'30	13-'31	13-b56	13-b60	
	13-b61	13-d97	13-e01	13-e02	13-e09	13-e10	13-e27	13-e28	13-e35	13-e36	13-e54	13-e55	13-f67	13-f71	
	13-f72	13-f79	13-f80	13-144	13-145	13-j87	13-j91	13-j92	13-j99	13-k00	13-k79	13-k83	13-k84	13-k91	
	13-k92														
MDPE	4-957#														
MI	4-831#	13-a05	13-b26												
MIXED	35-4#														
MOC	4-826#	4-852	13-a88	13-A19	13-[18	13-[25	13-[26	13-[32	13-[37	13-[38	13-[60	13-[68	13-[69	13-\17	
	13-\42	13-J93	13-^02	13-^03	13-^09	13-^14	13-^15	13-^38	13-^48	13-^49	13-^99	13- 27	13-'89	13-'98	
	13-'99	13-a08	13-a13	13-a14	13-a20	13-a21	13-a22	13-a26	13-a27	13-a50	13-a60	13-a61	13-b10	13-b19	
	13-b20	13-b29	13-b34	13-b35	13-b45	13-b46	13-b56	13-b60	13-b61	13-b82	13-b98	13-b99	13-c24	13-c25	
	13-c72	13-c81	13-c82	13-c88	13-c92	13-c93	13-d00	13-d01	13-d16	13-d42	13-d51	13-d52	13-d57	13-d61	
	13-d62	13-d82	13-d91	13-d92	13-d97	13-e01	13-e02	13-e09	13-e10	13-e27	13-e28	13-e35	13-e36	13-e54	





RECAL	4-719#	13-H61	13-J70	13-214	13-252	13-290	13-[21	13-[63	13-\20					
RESREG	26-148	29-1#												
RESVEC	4-700#													
REX	4-839#	13-Y44	13-Y70	13-Y75										
RG	4-837#	13-Q33	13-Q51	13-e90	13-e92	13-f44	13-f49	13-g16	13-g21	13-h51	13-h56	13-h95	13-100	13-182
	13-187	13-j45	13-j50	13-k31	13-k36	13-l64	13-l69							
RGDTPT	35-3#													
RH	4-742#	13-G55	13-R16											
RIP	4-724#	13-G27	13-J90	13-L37	13-M89	13-N99	13-038	13-067	13-Y15					
RELEASE	4-721#	13-G19	13-176	13-J78	13-U30	13-U59	13-U88	13-V21						
RMAS	4-916#	13-58	13-390*	13-T68	13-U67*	13-V62*	13-V76	13-V82	13-W13*	13-W27*	13-W46*	13-W69*	13-W95	
RMAS1	7-0#													
RMASO	7-0#													
RMBA	4-989#	13-54	13-S87*	13-L53*										
RMBAE	4-992#													
RMBAEI	7-0#	13-<99*	13-=00*											
RMBAEO	7-0#													
RMBAI	7-0#													
RMBAO	7-0#	13-L20*	13-L53											
RMCS1	4-912#	4-987#	9-90	13-78*	13-85	13-111*	13-119	13-165*	13-172*	13-180	13-222*	13-230	13-433*	13-541
	13-609*	13-612	13-617	13-78*	13-85	13-111*	13-119	13-165*	13-172*	13-180	13-222*	13-230	13-433*	13-541
	13-664	13-:78	13-<98*	13-623*	13-624*	13-625	13-630	13-636*	13-637*	13-638	13-644	13-656*	13-657*	13-658
	13-H12	13-132*	13-133	13-21*	13-24	13-C27*	13-C68*	13-D35*	13-E05*	13-F24*	13-F25	13-F73*	13-G81*	13-G82
	13-N99*	13-038*	13-067*	13-149	13-J16*	13-K53*	13-K92*	13-L16*	13-L37*	13-L58*	13-M27*	13-M64*	13-N12*	13-N50*
	13-U30*	13-U38*	13-U59*	13-P05*	13-P39*	13-Q02*	13-Q31*	13-Q74*	13-R51*	13-R98*	13-S44*	13-S88*	13-T40*	13-U04
	13-[21*	13-L63*	13-[85	13-U88*	13-V21*	13-V95*	13-X01	13-X21*	13-X70*	13-Y59*	13-Z14*	13-Z52*	13-Z71	13-Z90*
	13-74*	13-15*	13-34	13-101	13-120*	13-182*	13-J23*	13-J42	13-J64*	13-J98*	13-^43*	13-^65	13-^81	13-04*
	13-e58	13-e79*	13-f39*	13-59*	13-94*	13-a55*	13-a77	13-a93	13-b15*	13-b64	13-b87*	13-c77*	13-d47*	13-d87*
	7-0#	13-85*	13-93*	13-g11*	13-h46*	13-h90*	13-i77*	13-j40*	13-k26*	13-l59*	16-19*	17-17*		
RMCS11	7-0#	13-85*	13-93*	13-g11*	13-h46*	13-h90*	13-i77*	13-j40*	13-k26*	13-l59*	16-19*	17-17*		
RMCS10	7-0#	13-L18*	13-L59	13-119*	13-124*	13-126	13-180*	13-189*	13-193*	13-199	13-230*	13-238*	13-243	
RMCS2	4-990#	9-85*	9-86*	9-88	11-54*	11-55*	13-8	13-15	13-25	13-29	13-36	13-44	13-56	13-:37*
	13-<42	18-10*	18-12*											
RMCS21	7-0#													
RMCS20	7-0#	13-:13*	13-:14*	13-:37	13-:43	13-:52	13-:55*							
RMCS3	4-993#													
RMCS31	7-0#													
RMCS30	7-0#													
RMDA	4-913#	13-79*	13-86	13-166*	13-173*	13-181	13-217*	13-223*	13-231	13-265*	13-272	13-387*	13-393	13-409*
	13-415	13-470*	13-474	13-487*	13-491	13-506*	13-560*	13-561*	13-571*	13-572*	13-587*	13-588*	13-:95*	13-:97*
	13-:99	13-:16*	13-:17*	13-:18	13-:23	13-:27*	13-:28*	13-:29	13-:34	13-:41*	13-:42*	13-:43	13-:48	13-832*
	13-C06	13-C23*	13-C33	13-C65*	13-C75	13-D32*	13-D87*	13-N94*	13-008	13-063*	13-075	13-079	13-P98*	13-Q65*
	13-R43*	13-R93*	13-S40*	13-S85*	13-T74	13-U96*	13-W98	13-Y56*	13-178*	13-J19*	13-J60*	13-J94*	13-^39*	13-00*
	13-70*	13-11*	13-55*	13-90*	13-a51*	13-b11*	13-b83*	13-c76*	13-d45*	13-d85*	13-e77*	13-f37*	13-g09*	13-F44*
	13-F89*	13-166*	13-j30*	13-k24*	13-l43*									
RMDAI	7-0#	13-86*	13-95	13-181*	13-194*	13-200	13-231*	13-244	13-272*	13-281	13-393*	13-396	13-415*	13-418
	13-474*	13-476	13-491*	13-494	13-:99*	13-:01								
RMDAG	7-0#	13-C54*	13-C65	13-C87*	13-C89*	13-D16*	13-D17*	13-D32	13-D59*	13-D65*	13-D75*	13-D76*	13-D87	13-E18*
	13-E24*	13-P87*	13-P88*	13-P98	13-R83*	13-R84*	13-R93	13-S13	13-S19	13-S21*	13-S22	13-S24*	13-S26*	13-S27
	13-S72*	13-S73*	13-S85	13-T15*	13-T21*	13-L15*	13-L16*	13-L43	13-m45					
RMDB	4-991#	13-60												
RMDBI	7-0#													
RMDBO	7-0#													
RMDC	4-922#	13-80*	13-87	13-176*	13-184	13-219*	13-225*	13-233	13-267*	13-274	13-372*	13-376	13-432*	13-436
	13-450*	13-456	13-488*	13-502*	13-508	13-:96*	13-:98*	13-:00	13-:60	13-:63*	13-:64	13-:72*	13-:73*	13-:74
	13-:82*	13-:83*	13-:84	13-<00*	13-<01*	13-<04	13-D14	13-D21*	13-D40	13-D86*	13-N95*	13-011	13-064*	13-084
	13-088	13-P01*	13-P76*	13-Q66*	13-R44*	13-R92*	13-S39*	13-S84*	13-Y57*	13-179*	13-J20*	13-J61*	13-J95*	13-^40*

	13-01*	13-71*	13-12*	13-56*	13-91*	13-a52*	13-b12*	13-b84*	13-c75*	13-d46*	13-d86*	13-e78*	13-f38*	13-g10*
RMDCI	13-F45*	13-F88*	13-i67*	13-j31*	13-k25*	13-L44*								
	7-0#	13-87*	13-97*	13-184*	13-191*	13-197*	13-203	13-233*	13-240*	13-246	13-274*	13-291	13-376*	13-379*
RMDCO	13-380	13-436*	13-440*	13-441	13-456*	13-462*	13-463	13-508*	13-512	13-513	13-:00*	13-:03*	13-:04	
RMDS	7-0#	13-534*	13-539	13-558	13-564*	13-565	13-L14*	13-L44	13-L92					
	4-914#	9-87	9-99	13-472*	13-504*	13-594	13-599	13->10	13->22	13->31	13->59	13->65	13->75	13->92
	13-a84	13-a89	13-a99	13-A16	13-D72	13-D90	13-E08	13-E31	13-E39	13-E49	13-E65	13-E90	13-G97	13-H02
	13-H37	13-H40	13-L22	13-L29	13-L43	13-L50	13-M05	13-M06	13-M11	13-M43	13-M53	13-M71	13-O96	13-P13
	13-P22	13-P47	13-P67	13-P77	13-P90	13-Q05	13-T87	13-T96	13-U13	13-U18	13-U41	13-U70	13-U99	13-V65
	13-V69	13-V96	13-W01	13-W17	13-W21	13-W29	13-W33	13-W50	13-W55	13-W75	13-W82	13-X30	13-X34	13-L43
	13-L48	13-A20	13-A25	13-a32	13-a37	16-20	16-24	17-18	17-23					
RMDSI	7-0#													
RMDSO	7-0#													
RMDT	4-919#	9-93	9-95	13-451*	13-526	13-532	13-T86	13-V29	26-26					
RMDTI	7-0#													
RMDTO	7-0#													
RMEC1	4-926#	13-453*												
RMEC11	7-0#													
RMEC10	7-0#													
RMEC2	4-927#	13-62	13-414*	13-<53										
RMEC2I	7-0#													
RMEC20	7-0#													
RMER1	4-915#	13-112*	13-120	13-142*	13-147	13-167*	13-174*	13-182	13-226*	13-234	13-268*	13-275	13-371*	13-375
	13-431*	13-435	13-448*	13-454	13-486*	13-490	13-505*	13-675*	13-678	13-684	13-692	13-700	13-705*	13-706*
	13-707	13-713	13-721	13-730	13-736*	13-737*	13-738	13-743	13-755*	13-756	13-767	13-:22	13-:27	13-:39
	13-<24	13-<29	13-<59	13->28*	13->38	13->43	13->71	13->76	13-C21*	13-E03*	13-E37*	13-E47*	13-E62*	13-E64*
	13-E86*	13-F18*	13-F54*	13-G74*	13-I24*	13-J00*	13-K39*	13-K83*	13-L15*	13-L36*	13-L57*	13-M01*	13-M28	13-M32
	13-M51*	13-N17	13-N24	13-N90*	13-O33*	13-O61*	13-O62*	13-Q24*	13-Q86	13-R61	13-R68	13-S07	13-S12	13-S52
	13-S57	13-T03	13-T07	13-T28	13-T36*	13-T46	13-V53*	13-V90*	13-W11*	13-W44*	13-W49*	13-W68*	13-X80	13-X86
	13-Y67*	13-Z12*	13-Z32	13-Z37	13-Z50*	13-Z88*	13-[01	13-[05	13-[19*	13-[61*	13-118*	13-180*	13-100	13-105
	13-121*	13-162*	13-175	13-179	13-196*	13-241*	13-22*	13-72*	13-92	13-97	13-13*	13-57*	13-70	13-74
	13-92*	13-a53*	13-b13*	13-b85*	13-c73*	13-d18	13-d26	13-d43*	13-d65	13-d70	13-d83*	13-e75*	13-f35*	13-g07*
	13-h42*	13-h69	13-h74	13-h86*	13-i75*	13-j14	13-j19	13-j38*	13-k22*	13-l57*	16-17*	17-15*		
RMER1I	7-0#	13-120*	13-128	13-147*	13-150*	13-182*	13-195*	13-201	13-234*	13-247	13-275*	13-297	13-375*	13-377
	13-435*	13-437*	13-438	13-454*	13-457	13-490*	13-492	13-678*	13-679	13-687	13-695	13-707*	13-708	13-716
	13-725													
RMER10	7-0#	13-L10*	13-L15	13-L25	13-L36	13-L46	13-L57	13-L67	13-L77	13-L86*				
RMER2	4-925#	13-113*	13-121	13-143*	13-148	13-169*	13-177*	13-185	13-227*	13-235	13-269*	13-276	13-389*	13-395
	13-411*	13-417	13-452*	13-471*	13-475	13-489*	13-503*	13-509	13-881	13-887*	13-891	13-918*	13-919*	13-920
	13-939*	13-940*	13-941	13-967*	13-968	13-:23	13-?21	13-?27*	13-?29	13-?56	13-?60	13-?91*	13-?93	13-?02
	13-a23	13-a27*	13-a28	13-a38*	13-a40	13-a54*	13-a56	13-C22*	13-E04*	13-E38*	13-E48*	13-E63*	13-E87*	13-E88*
	13-F19*	13-F55*	13-G75*	13-I25*	13-J01*	13-K31	13-K40*	13-K41	13-K56	13-K77	13-K84*	13-K85	13-L00	13-M02*
	13-M52*	13-N91*	13-O34*	13-Q25*	13-T37*	13-V54*	13-V91*	13-W12*	13-W45*	13-Z13*	13-Z51*	13-Z89*	13-[20*	13-[62*
	13-119*	13-181*	13-J22*	13-J63*	13-J97*	13-42*	13-03*	13-73*	13-14*	13-58*	13-93*	13-a54*	13-b14*	13-b86*
	13-c74*	13-d44*	13-d84*	13-e76*	13-f36*	13-g08*	13-h43*	13-h87*	13-176*	13-139*	13-k23*	13-L58*	16-18*	17-16*
RMER2I	7-0#	13-121*	13-125*	13-130	13-148*	13-152*	13-185*	13-192*	13-198*	13-204	13-235*	13-241*	13-248	13-276*
	13-301	13-395*	13-401*	13-402	13-417*	13-423*	13-424	13-475*	13-478*	13-479	13-509*	13-515*	13-516	13-891*
	13-894	13-902	13-910	13-920*	13-923	13-931	13-941*	13-943	13-951	13-:23*	13-:28			
RMER20	7-0#													
RPHR	4-923#	13-374*	13-413*	13-554	13-562	13-573	13-589	13-:38*	13-:74*	13-:77				
RPHRI	7-0#	13-:77*												
RPHRO	7-0#	13-:15*	13-:17*	13-:38	13-:50	13-:57*	13-:65*	13-:71*	13-:74	13-:86	42-6			
RPLA	4-917#	13-412*												
RPLAI	7-0#													
RPLAO	7-0#													
RPMR1	4-918#	13-144*	13-149	13-373*	13-35	13-40	13-48*	13-49*	13-50	13-59*	13-60*	13-61	13-74*	13-75*





RMSNO	7-0#														
RMWC	4-988#	13-52	13-S86*	13-L52*											
RMWCI	7-0#														
RMWCO	7-0#	13-L19*	13-L52												
RQA	4-881#	13-U34	13-U45	13-U63	13-U74	13-U92	13-V03	13-V31	13-V35	13-V25	13-_09	13-c07	13-f13	13-g58	
	13-L90	13-m28													
RQB	4-882#	13-U34	13-U45	13-U63	13-U74	13-U92	13-V03	13-V31	13-V37	13-V25	13-_09	13-c07	13-f13	13-g58	
	13-L90	13-m28													
RTC	4-723#	13-H67	13-J86	13-P39	13-X50										
SA1	4-759#														
SA16	4-755#														
SA2	4-758#														
SA4	4-757#														
SAB	4-756#														
SADMSK	4-763#														
SAVREG	26-13	29-1#													
SC	4-939#														
SCO	4-812#														
SC1	4-811#														
SC2	4-810#														
SC3	4-809#														
SC4	4-808#														
SCOPE	4-700#	13-1	13-73	13-106	13-137	13-159	13-332	13-524	13-538	13-549	13-602	13-670	13-774	13-787	
	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-=15	13-=32	13-=90	13->55	13-?16	13-a18	
	13-a79	13-A39	13-B20	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64	13-I14	13-I89	13-K28	13-K74	
	13-L08	13-M40	13-N00	13-N36	13-N84	13-O23	13-O53	13-O93	13-P19	13-P64	13-P85	13-Q12	13-Q56	13-R33	
	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11	13-V48	13-V86	13-W06	13-W39	
	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-^66	13-_58	13-c65	13-d33	13-h32	13-l08	14-9			
SCTMSK	4-814#														
SEARCH	4-729#	13-G43	13-I79	13-K06	13-R04	13-R98	13-S44	13-_74	13-'15	13-'59	13-'94	13-a55	13-b15	13-b87	
	13-c77														
SEEK	4-718#	13-H58	13-J66	13-R07	13-^82	13-J23	13-J64	13-J98	13-^43	13-_04					
SETOM	13-P31	13-P72	13-P95	13-i69	13-j33	13-L47	17-13#								
SETVV	13-N04	13-N42	13-O99	13-P24	13-P35	13-P69	13-P92	13-Q60	13-R37	13-R86	13-S36	13-S78	13-U25	13-U54	
	13-U83	13-V15	13-X13	13-X58	13-Y51	13-Z06	13-Z44	13-Z82	13-[13	13-[55	13-^12	13-^71	13-J12	13-J53	
	13-J87	13-^32	13-^93	13-^63	13-'04	13-'48	13-'83	13-a44	13-b04	13-b76	13-c67	13-d37	13-d77	13-e69	
	13-f29	13-g01	13-h36	13-h80	13-i61	13-j25	13-k16	13-l38	16-13#						
SHUT	27-1	27-1	27-3#												
SIZCLK	11-28	15-5#													
SKI	4-894#	13-152	13-969	13-971	13-a29	13-a41	13-a43	13-a57	13-a61						
SNGPRT	4-861#	13-527													
SSE	4-900#														
SSEI	4-869#	13-849	13-839	13-862	13-890	13-892	13-C98	13-D00	13-D62	13-D64	13-E21	13-E23	13-T18	13-T20	
SSF	4-908#														
STACK	4-700#	9-23	13-1	13-73	13-106	13-137	13-159	13-332	13-524	13-538	13-549	13-602	13-670	13-774	
	13-787	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-=15	13-=32	13-=90	13->55	13-?16	
	13-a18	13-a79	13-A39	13-B20	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64	13-I14	13-I89	13-K28	
	13-K74	13-L08	13-M40	13-N00	13-N36	13-N84	13-O23	13-O53	13-O93	13-P19	13-P64	13-P85	13-Q12	13-Q56	
	13-R33	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11	13-V48	13-V86	13-W06	
	13-W39	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-^66	13-_58	13-c65	13-d33	13-h32	13-l08			
STAMDA	9-67	10-3#													
START	5-1	9-17#	11-33	11-42	27-5										
START1	5-3	9-14#													
START2	9-15	9-18#													
STKLMT	4-700#														
STOPCL	7-0#	13-K96	13-Q38	13-Q43	13-Y74	13-Y84	13-d23	13-d29	13-f48	13-f54	13-g20	13-g26	13-h55	13-h61	



TST103	13-V86#	24-1
TST104	13-W06#	24-1
TST105	13-W39#	24-1
TST106	13-W60#	24-1
TST107	13-X09#	24-1
TST11	13-549#	24-1
TST110	13-X54#	24-1
TST111	13-Y24#	24-1
TST112	13-Z01#	24-1
TST113	13-166#	24-1
TST114	13-58#	24-1
TST115	13-c65#	24-1
TST116	13-d33#	24-1
TST117	13-h32#	24-1
TST12	13-602#	24-1
TST120	13-L08#	24-1
TST13	13-670#	24-1
TST14	13-774#	24-1
TST15	13-787#	24-1
TST16	13-877#	24-1
TST17	13-986#	24-1
TST2	13-73#	24-1
TST20	13-:09#	24-1
TST21	13-:62#	24-1
TST22	13-:90#	24-1
TST23	13-:10#	24-1
TST24	13-:54#	24-1
TST25	13-<16#	24-1
TST26	13-=15#	24-1
TST27	13-=32#	24-1
TST3	13-106#	24-1
TST30	13-=90#	24-1
TST31	13->55#	24-1
TST32	13-?16#	24-1
TST33	13-a18#	24-1
TST34	13-a79#	24-1
TST35	13-A39#	24-1
TST36	13-B20#	24-1
TST37	13-C03#	24-1
TST4	13-137#	24-1
TST40	13-D10#	24-1
TST41	13-D69#	24-1
TST42	13-E28#	24-1
TST43	13-F08#	24-1
TST44	13-F43#	24-1
TST45	13-G64#	24-1
TST46	13-I14#	24-1
TST47	13-I89#	24-1
TST5	13-159#	24-1
TST50	13-K28#	24-1
TST51	13-K74#	24-1
TST52	13-L08#	24-1
TST53	13-M40#	24-1
TST54	13-N00#	24-1
TST55	13-N36#	24-1
TST56	13-N84#	24-1





\$\$LARE	5-15#													
\$\$CMTM	5-15#	6-0	6-0	6-0	6-0	6-0								
\$\$ESCA	4-700#													
\$\$NEWT	4-700#	13-1	13-73	13-106	13-137	13-159	13-332	13-524	13-538	13-549	13-602	13-670	13-774	13-787
	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-=15	13-=32	13-=90	13->55	13-?16	13-@18
	13-@79	13-A39	13-B20	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64	13-I14	13-I89	13-K28	13-K74
	13-LC8	13-M40	13-NJ0	13-N36	13-N84	13-O23	13-O53	13-O93	13-P19	13-P64	13-P85	13-Q12	13-Q56	13-R33
	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11	13-V48	13-V86	13-W06	13-W39
	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-166	13-58	13-c65	13-d33	13-h32	13-l08			
\$\$SET	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1#
\$\$SETM	9-23	9-23#												
\$\$SKIP	4-700#													
\$.SACT1	4-692#	5-5												
\$.SAPT8	4-692#	6-0	6-0#											
\$.SAPTH	4-692#	5-13												
\$.SAPTY	4-692#	31-1												
\$.SCATC	4-688#	5-1												
\$.SCMTA	4-689#	5-15												
\$.SEOP	4-689#	14-20												
\$.SERPO	4-689#	25-1												
\$.SPOWC	4-691#	30-1												
\$.SADDE	4-690#													
\$.SADOC	4-690#	28-1												
\$.SREAD	4-690#	27-1												
\$.SSAVE	4-691#	19-1												
\$.SSCOP	4-689#	24-1												
\$.SSIZE	4-691#													
\$.STRAP	4-691#	29-1												
\$.STYPB	4-690#	20-1												
\$.STYPD	4-690#	21-1												
\$.STYPE	4-689#	23-1												
\$.STYPO	4-690#	22-1												
\$.EQUAT	4-688#	4-700												
\$.HEADE	4-688#	4-696												
\$.SETUP	4-688#	4-998												
\$.SWRHI	4-688#	4-697												
\$.SWRLO	4-688#	4-697#	4-698											
CLEAR	4-497#	13-7	13-28	13-75	13-108	13-116	13-139	13-162	13-213	13-262	13-370	13-386	13-408	13-430
	13-447	13-469	13-485	13-500	13-540	13-551	13-605	13-611	13-677	13-754	13-776	13-889	13-965	13-988
	13-:21	13-:73	13-:92	13-:13	13-:57	13-<23	13-=17	13-=23	13-=39	13-=92	13->61	13-?18	13-@20	13-@81
	13-A47	13-B29	13-C19	13-C63	13-D19	13-D29	13-D85	13-E35	13-E60	13-E84	13-F15	13-F51	13-G71	13-I21
	13-197	13-K36	13-K80	13-L98	13-M48	13-N87	13-O30	13-O58	13-Q21	13-T33	13-T85	13-U03	13-V51	13-V88
	13-W09	13-W42	13-W66	13-Y42	16-14									
CLKOFF	4-612#	13-K96	13-Q38	13-Q43	13-Y74	13-Y84	13-d23	13-d29	13-f48	13-f54	13-g20	13-g26	13-h55	13-h61
	13-h99	13-i05	13-i86	13-i92	13-j49	13-j55	13-k35	13-k41	13-l68	13-l74				
CLKON	4-603#	13-K93	13-Q30	13-Y68	13-d17	13-f42	13-g14	13-h49	13-h93	13-i60	13-j43	13-k29	13-l62	
CLKSNC	4-672#													
COMMEN	4-700#													
ENBSCH	4-638#													
ENDCOM	4-700#													
ERR	4-568#	8-3	8-6	8-9	8-12	8-15	8-18	8-22	8-25	8-28	8-31	8-34	8-37	8-40
	8-43	8-46	8-49	8-52	8-55	8-58	8-61	8-64	8-67	8-70	8-73	8-76	8-79	8-82
	8-85	8-88	8-91	8-94	8-97	8-100	8-103	8-106	8-109	8-112	8-115	8-118	8-121	8-124
	8-127	8-130	8-133	8-136	8-139	8-142	8-145	8-148	8-151	8-154	8-157	8-160	8-163	8-166
	8-169	8-172	8-175	8-178	8-181	8-184	8-187	8-190	8-193	8-196	8-199	8-202	8-205	8-208
	8-211	8-214	8-217	8-220	8-223	8-226	8-229	8-232	8-235	8-238	8-241	8-244	8-247	8-250

	8-253	8-256	8-259	8-262	8-265	8-268	8-271	8-274	8-277	8-280	8-283	8-286	8-289	8-292
	8-295	8-298	8-301	8-304	8-307	8-310	8-313	8-316	8-319	8-322	8-325	8-328	8-331	8-334
	8-337	8-340	8-343	8-346	8-349	8-352	8-355	8-358	8-361	8-364	8-367	8-370	8-373	8-376
	8-379	8-382	8-385	8-388	8-391	8-394	8-397	8-400	8-403	8-406	8-409	8-412	8-415	8-418
	8-421	8-424	8-427	8-430	8-433	8-436	8-439	8-442	8-445	8-448	8-451	8-454	8-457	8-460
	8-464	8-467	8-470	8-473	8-476	8-479	8-482	8-485	8-488	8-492	8-495	8-498	8-501	8-504
	8-508	8-511	8-515	8-518	8-521	8-524	8-527	8-530	8-533	8-537	8-540	8-543	8-546	8-549
	8-552	8-555	8-558	8-561	8-564	8-567	8-570	8-573	8-576	8-580	8-583	8-586	8-589	
ERROR	4-700#	13-16	13-37	13-68	13-103	13-134	13-156	13-210	13-255	13-316	13-382	13-404	13-426	13-443
	13-465	13-481	13-496	13-518	13-533	13-546	13-566	13-578	13-594	13-618	13-631	13-645	13-665	13-685
	13-693	13-701	13-714	13-722	13-731	13-744	13-768	13-784	13-803	13-811	13-824	13-834	13-852	13-870
	13-897	13-905	13-913	13-926	13-934	13-946	13-956	13-979	13-:01	13-:30	13-:32	13-:45	13-:47	13-:81
	13-:06	13-:24	13-:35	13-:49	13-:67	13-:77	13-:89	13-<08	13-<30	13-<56	13-<66	13-<68	13-:28	13-:44
	13-:54	13-:65	13-:82	13->03	13->14	13->26	13->41	13->68	13->79	13-?02	13-?32	13-?46	13-?49	13-?61
	13-?79	13-?82	13-a04	13-a32	13-a65	13-a93	13-A03	13-A25	13-A51	13-A62	13-A73	13-A94	13-B54	13-B76
	13-C36	13-C78	13-D43	13-D94	13-E12	13-E43	13-E52	13-E70	13-F00	13-F32	13-F64	13-F90	13-G88	13-M03
	13-H22	13-H30	13-H43	13-I37	13-I55	13-I63	13-J08	13-J38	13-K45	13-K63	13-K89	13-L04	13-L30	13-L51
	13-L72	13-L81	13-M09	13-M14	13-M21	13-M33	13-M57	13-M76	13-N04	13-N26	13-N42	13-N72	13-O19	13-O49
	13-O80	13-O89	13-O99	13-P16	13-P24	13-P31	13-P35	13-P51	13-P61	13-P69	13-P72	13-P81	13-P92	13-P95
	13-Q09	13-Q40	13-Q60	13-Q98	13-R37	13-R70	13-R86	13-S14	13-S36	13-S59	13-S78	13-T09	13-T54	13-T56
	13-197	13-U09	13-U19	13-U25	13-U49	13-U54	13-U78	13-U83	13-V07	13-V15	13-V45	13-V71	13-V83	13-W02
	13-W23	13-W35	13-W56	13-W83	13-X13	13-X37	13-X58	13-X89	13-Y49	13-Y51	13-Y79	13-Z06	13-Z38	13-Z44
	13-276	13-282	13-[07	13-[13	13-[49	13-[55	13-[90	13-106	13-112	13-131	13-171	13-J06	13-J12	13-J47
	13-J53	13-J81	13-J87	13-^26	13-^32	13-^70	13-^86	13-^93	13- 16	13- 63	13- 98	13- 04	13- 39	13- 48
	13- 76	13- 83	13-a38	13-a44	13-a82	13-a98	13-b04	13-b53	13-b69	13-b76	13-c14	13-c67	13-d11	13-d27
	13-d37	13-d71	13-d77	13-e20	13-e48	13-e63	13-e69	13-e95	13-f22	13-f29	13-f52	13-f90	13-g01	13-g24
	13-g87	13-h36	13-h59	13-h75	13-h80	13-i03	13-i55	13-i61	13-i69	13-i90	13-j20	13-j25	13-j33	13-j53
	13-k10	13-k16	13-k39	13-k75	13-L04	13-L38	13-L47	13-L72	13-m03	13-m61	13-m93	24-1		
FSCAPE	4-700#													
GETAS	4-274#	13-V76												
GETBA	4-138#													
GETBAE	4-282#													
GETCS1	4-122#	13-85	13-119	13-180	13-230	13-541	13-612	13-625	13-638	13-658	13-:78	13-:24	13-F25	13-G82
	13-H12	13-133	13-149	13-U04	13-U38	13-271	13-[85	13-101	13-J42	13-^65	13-^81	13- 34	13-a77	13- a93
	13-b64	13-e58												
GETCS2	4-154#	13-8	13-29											
GETDA	4-178#	13-86	13-181	13-231	13-272	13-393	13-415	13-474	13-491	13-:99	13-;18	13-;29	13-;43	13-C33
	13-C75	13-008	13-075											
GETDB	4-146#													
GETDC	4-186#	13-87	13-184	13-233	13-274	13-376	13-436	13-456	13-508	13-:00	13-;64	13-;74	13-;84	13-<04
	13-D40	13-011	13-084											
GETDS	4-162#	13-:99	13->10	13->22	13->31	13->65	13->75	13->92	13-a89	13-a99	13-A16	13-D90	13-E08	13-E39
	13-E49	13-E65	13-E90	13-G97	13-H37	13-L22	13-L43	13-M06	13-M11	13-M53	13-M71	13-P13	13-P47	13-P77
	13-Q05	13-T87	13-U13	13-U41	13-U70	13-U99	13-V65	13-V96	13-W17	13-W29	13-W50	13-W75	13-X30	13-[43
	13-^20	13-a32	16-20	17-18										
GETDT	4-210#	13-526	13-T86	13-V29										
GETEC1	4-226#													
GETEC2	4-234#													
GETER1	4-170#	13-120	13-147	13-182	13-234	13-275	13-375	13-435	13-454	13-490	13-678	13-707	13-738	13-756
	13-:22	13-:39	13-<24	13-<59	13-?38	13-?71	13-M28	13-M17	13-Q86	13-R61	13-S07	13-S52	13-T03	13-T46
	13-X80	13-232	13-[01	13-J00	13-J75	13- 92	13- 70	13-d18	13-d65	13-h69	13-j14			
GETER2	4-202#	13-121	13-148	13-185	13-235	13-276	13-395	13-417	13-475	13-509	13-891	13-920	13-941	13-968
	13-:23	13-?29	13-?56	13-?93	13-a28	13-a40	13-a56	13-K41	13-K56	13-K85	13-L00			
GETHR	4-242#	13-562	13-573	13-589	13-:77									
GETLA	4-194#													
GETMR1	4-250#	13-149	13-:40	13-:50	13-:61	13-:76	13-A48	13-A59	13-A69	13-A86	13-B33	13-J04	13-J30	13-L64



	13-K16	13-N61	13-Q32	13-Y43	13-Y69	13-b42	13-d02	13-e11	13-e42	13-e89	13-f43	13-f81	13-g15	13-h50
GETMR2	13-h94	13-146	13-181	13-j44	13-k01	13-k30	13-k66	13-k95	13-l63					
	4-258#	13-F59	13-F85	13-U33	13-U44	13-U62	13-U73	13-U91	13-V02	13-V30	13-V24	13-V08	13-c06	13-f05
GETOF	13-g57	13-L79	13-m14	13-m72										
	4-266#	13-88	13-183	13-232	13-273	13-394	13-416	13-455	13-507	13-778	13-797	13-818	13-846	13-864
GETPRI	13-831	13-c25	13-D31	13-D89	13-014	13-046	13-P55	13-S83						
GETSN	4-700#	15-58												
GETSWR	4-218#	13-994	13-998											
GETWC	4-700#	9-28	9-28#											
GETX	4-130#													
MSG	4-114#													
	4-8#	13-1	13-73	13-106	13-137	13-159	13-326#	13-332	13-521#	13-524	13-538	13-549	13-602	13-670
	13-774	13-787	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-:15	13-:32	13-:90	13->55
	13-?16	13-a18	13-a79	13-A39	13-B08#	13-B20	13-C00#	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64
	13-114	13-189	13-K28	13-K74	13-L08	13-M40	13-N00	13-N36	13-N84	13-023	13-053	13-093	13-P19	13-P64
	13-P85	13-Q12	13-Q56	13-R33	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11
	13-V48	13-V86	13-W06	13-W39	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-166	13-58	13-c65	13-d33	13-h32
	13-L08													
MULT	4-700#													
NEWTST	4-700#	13-1	13-73	13-106	13-137	13-159	13-332	13-524	13-538	13-549	13-602	13-670	13-774	13-787
	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-:15	13-:32	13-:90	13->55	13-?16	13-a18
	13-a79	13-A39	13-B20	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64	13-I14	13-I89	13-K28	13-K74
	13-L08	13-M40	13-N00	13-N36	13-N84	13-023	13-053	13-093	13-P19	13-P64	13-P85	13-Q12	13-Q56	13-R33
	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11	13-V48	13-V86	13-W06	13-W39
	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-166	13-58	13-c65	13-d33	13-h32	13-L08			
NWTST	4-511#	13-1	13-73	13-106	13-137	13-159	13-332	13-524	13-538	13-549	13-602	13-670	13-774	13-787
	13-877	13-986	13-:09	13-:62	13-:90	13-:10	13-:54	13-<16	13-:15	13-:32	13-:90	13->55	13-?16	13-a18
	13-a79	13-A39	13-B20	13-C03	13-D10	13-D69	13-E28	13-F08	13-F43	13-G64	13-I14	13-I89	13-K28	13-K74
	13-L08	13-M40	13-N00	13-N36	13-N84	13-023	13-053	13-093	13-P19	13-P64	13-P85	13-Q12	13-Q56	13-R33
	13-R81	13-S32	13-S69	13-T25	13-T82	13-U00	13-U23	13-U52	13-U81	13-V11	13-V48	13-V86	13-W06	13-W39
	13-W60	13-X09	13-X54	13-Y24	13-Z01	13-166	13-58	13-c65	13-d33	13-h32	13-L08			
POP	4-700#	13-<46	13-<47	13-<51	13-<52	15-45	15-46	18-13	19-1	21-1	28-1	30-1	30-1	31-1
	31-1													
PUSH	4-700#	13-<37	13-<38	13-<80	13-<81	15-6	15-7	18-8	19-1	21-1	28-1	30-1	30-1	31-1
	31-1	31-1												
PUTAS	4-470#	13-390	13-U67	13-V62	13-W13	13-W27	13-W46	13-W69						
PUTBA	4-329#	13-S87	13-L53											
PUTBAE	4-478#													
PUTCS1	4-314#	13-78	13-111	13-165	13-172	13-222	13-433	13-609	13-623	13-624	13-636	13-637	13-656	13-657
	13-:21	13-C27	13-C68	13-D35	13-E05	13-F24	13-F73	13-G81	13-I32	13-J16	13-K53	13-K92	13-L16	13-L37
	13-L58	13-M27	13-M64	13-N12	13-N50	13-N99	13-O38	13-O67	13-P05	13-P39	13-Q02	13-Q31	13-Q74	13-R51
	13-R98	13-S44	13-S88	13-T40	13-U30	13-U59	13-U88	13-V21	13-V95	13-X21	13-X70	13-Y59	13-Z14	13-Z52
	13-290	13-[21	13-[63	13-\20	13-\82	13-J23	13-J64	13-J98	13-^43	13- 04	13- 74	13- 15	13- 59	13- 94
	13-a55	13-b15	13-b87	13-c77	13-d47	13-d87	13-e79	13-f39	13-g11	13-h46	13-h90	13-i77	13-j40	13-k26
	13-L59	16-19	17-17											
PUTCS2	4-345#	13-:37												
PUTDA	4-369#	13-79	13-166	13-173	13-217	13-223	13-265	13-387	13-409	13-470	13-487	13-506	13-560	13-561
	13-571	13-572	13-587	13-588	13-:95	13-:97	13-:16	13-:17	13-:27	13-:28	13-:41	13-:42	13-B32	13-C23
	13-c65	13-D32	13-D87	13-N94	13-063	13-P98	13-Q65	13-R43	13-R93	13-S40	13-S85	13-U96	13-Y56	13-V78
	13-J19	13-J60	13-J94	13-^39	13- 00	13- 70	13- 11	13- 55	13- 90	13-a51	13-b11	13-b83	13-c76	13-d45
	13-d85	13-e77	13-f37	13-g09	13-h44	13-h89	13-i66	13-j30	13-k24	13-l43				
PUTDB	4-337#													
PUTDC	4-377#	13-80	13-176	13-219	13-225	13-267	13-372	13-432	13-450	13-488	13-502	13-:96	13-:98	13-:63
	13- 72	13-:73	13-:82	13-:83	13-<00	13-<01	13-D21	13-D86	13-N95	13-064	13-P01	13-P76	13-Q66	13-R44
	13-R92	13-S39	13-S84	13-Y57	13-\79	13-J20	13-J61	13-J95	13-^40	13- 01	13- 71	13- 12	13- 56	13- 91
	13-a52	13-b12	13-b84	13-c75	13-d46	13-d86	13-e78	13-f38	13-g10	13-h45	13-h88	13-i67	13-j31	13-k25



RGBFMC	4-12#	7-0	7-0											
SCTCMP	4-650#													
SETLFS	4-661#													
SETOM	4-627#	13-P31	13-P72	13-P95	13-i69	13-j33	13-l47							
SETPRI	4-700#	11-49	15-63	15-89	27-1									
SETTRA	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1	29-1#	
SETUP	4-700#	9-23												
SETVV	4-616#	13-N04	13-N42	13-099	13-P24	13-P35	13-P69	13-P92	13-Q60	13-R37	13-R86	13-S36	13-S78	13-U25
	13-U54	13-U83	13-V15	13-X13	13-X58	13-Y51	13-Z06	13-Z44	13-Z82	13-[13	13-[55	13-\12	13-\71	13-J12
	13-J53	13-J87	13-^32	13-^93	13- 63	13-'04	13-'48	13-'83	13-a44	13-b04	13-b76	13-c67	13-d37	13-d77
	13-e69	13-f29	13-g01	13-h36	13-F80	13-i61	13-j25	13-k16	13-l38					
SKIP	4-700#													
SLASH	4-700#													
STARS	4-700#	5-5	5-13	5-13	5-13	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	13-332	13-332	13-524	13-524	13-538	13-538	13-549	13-549	13-602	13-602
	13-670	13-670	13-774	13-774	13-787	13-787	13-877	13-877	13-986	13-986	13-:09	13-:09	13-:62	13-:62
	13-:90	13-:90	13-:10	13-:10	13-:54	13-:54	13-<16	13-<16	13-=15	13-=15	13-=32	13-=32	13-=90	13-=90
	13->55	13->55	13-?16	13-?16	13-a18	13-a18	13-a79	13-a79	13-A39	13-A39	13-B20	13-B20	13-C03	13-C03
	13-D10	13-D10	13-D69	13-D69	13-E28	13-E28	13-F08	13-F08	13-F43	13-F43	13-G64	13-G64	13-I14	13-I14
	13-I89	13-I89	13-K28	13-K28	13-K74	13-K74	13-L08	13-L08	13-L91	13-L93	13-M40	13-M40	13-N00	13-N00
	13-N36	13-N36	13-N84	13-N84	13-O23	13-O23	13-O53	13-O53	13-O93	13-O93	13-P19	13-P19	13-P64	13-P64
	13-P85	13-P85	13-Q12	13-Q12	13-Q56	13-Q56	13-R33	13-R33	13-R81	13-R81	13-S32	13-S32	13-S69	13-S69
	13-T25	13-T25	13-T82	13-T82	13-U00	13-U00	13-U23	13-U23	13-U52	13-U52	13-U81	13-U81	13-V11	13-V11
	13-V48	13-V48	13-V86	13-V86	13-W06	13-W06	13-W39	13-W39	13-W60	13-W60	13-X09	13-X09	13-X54	13-X54
	13-Y24	13-Y24	13-Z01	13-Z01	13-\66	13-\66	13- 58	13- 58	13-b89	13-c03	13-c65	13-c65	13-d33	13-d33
	13-f07	13-f15	13-g39	13-g53	13-h32	13-h32	13-T08	13-T08	13-L33	13-L35	13-L81	13-L88	13-m16	13-m26
	13-m74	13-m82	14-20	19-1	20-1	21-1	22-1	23-1	24-1	25-1	27-1	27-1	27-1	27-1
	27-1	28-1	29-1	30-1	30-1	31-1								
SWRSU	4-700#	9-23	9-23#											
TAGS	4-46#	6-0												
TRMTRP	29-1#													
TYPBIN	4-700#													
TYPDEC	4-700#	14-20	14-20											
TYPNAM	4-688#	4-700#	9-28											
TYPNUM	4-700#													
TYPOCS	4-700#	9-82	11-16	11-63	26-22	26-39	26-44	26-46						
TYPOCT	4-700#	10-33	27-1											
TYPTXT	4-700#	9-6	9-42	9-53	9-59	11-30	14-20	14-20						