

RM03

DUAL PORT LOGIC TEST PART 2
MD-11-DZRMH-A

EP-DZRMH-A-DL-A

COPYRIGHT © 1977

FICHE 1 OF 1

OCT 1977

digital

MADE IN USA

The image displays a grid of 100 small test pattern cards, arranged in 10 rows and 10 columns. Each card contains a different test pattern, likely for a logic test. The patterns are small and dense, consisting of various symbols, lines, and text. The cards are set against a dark background.

100

B01

EOF1DZRMGASEQ
PBR000002
DZRMHA.P11

00010000 770804
MD-11-DZRMH-A, RMO3 DUAL

PDP10 411
CONTRLR LOGIC TEST - PART 2

GAHDR1DZRMHASEQ
MACY11 30(1046) 01-AUG-77

00010000 770804
11:14 PAGE 1

01-AUG-77 11:02

.REM ↑

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

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZRMH-A-D
PRODUCT NAME:	RMO3 DUAL PORT LOGIC TEST, PART 2
DATE CREATED:	15 JULY 77
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	DOUG RIIKONEN

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

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

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

COPYRIGHT (C) 1977, DIGITAL EQUIPMENT CORPORATION

53
54
55
56
57
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

RMO3 DUAL PORT LOGIC TEST, PART 2

CONTENTS

- 1. ABSTRACT
- 2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 PREREQUISITE PROGRAMS
 - 2.3 OTHER PROGRAMS
- 3. LOADING PROCEDURES
- 4. STARTING PROCEDURES
 - 4.1 STARTING ADDRESSES
 - 4.2 UNIBUS & VECTOR ADDRESSES
 - 4.3 OPERATOR ACTION
- 5. OPERATING PROCEDURES
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 'SOFTWARE' SWITCH REGISTER
 - 5.3 TEST SELECTION
 - 5.4 DUAL PORT TEST CABLE CONNECTION
- 6. ERRORS
- 7. MISCELLANEOUS
 - 7.1 RESTRICTIONS
 - 7.2 LIMITATIONS
 - 7.3 EXECUTION TIME
 - 7.4 REQUIRED TESTS
 - 7.5 DISK SURFACE USAGE
 - 7.6 LOOP ON ERROR OPTION
- 8. TEST DESCRIPTIONS

1. ABSTRACT

THE RMO3 DUAL PORT LOGIC TEST PERFORMS A SERIES OF TESTS WHICH VERIFY THAT THE RMO3 DUAL PORT LOGIC IS FUNCTIONING PROPERLY. ONLY THE CONTROL LOGIC IS TESTED BY THIS PROGRAM; DATA HANDLING IN THE DUAL PORT MODE IS NOT TESTED BY THIS PROGRAM.

BOTH PORTS OF THE DRIVE ARE CABLED TO THE SAME MASSBUS BY A SPECIAL ADAPTER CABLE. THIS ARRANGEMENT ALLOWS THE DUAL PORT LOGIC TO BE TESTED FROM ONE PDP-11/RH11 OR RH70.

THIS PROGRAM IS THE SECOND PART OF THE RMO3 DUAL PORT OPTION LOGIC TEST, AND IS USED TO TEST THE "PORT SELECT" SWITCH.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
16K OF MEMORY
KW11-L OR KW11-P CLOCK
TELETYPE
RH11 OR RH70 WITH AN RMO3
RMO3 DUAL PORT TEST CABLE

2.2 PREREQUISITE PROGRAMS

- A. RMO3 DISKLESS DIAGNOSTIC
- B. RMO3 FUNCTIONAL TEST

THE PRELIMINARY PROGRAMS MUST BE RUN TWICE: ONCE FROM EACH CONTROLLER (PORT).

- C. RMO3 DUAL PORT LOGIC TEST
PART 1

2.3 OTHER PROGRAMS

DYNAMIC OPERATION OF THE DUAL PORT OPTION IS TESTED BY THE RMO3 PERFORMANCE EXERCISER PROGRAM.

3. LOADING PROCEDURES

THE PROGRAM MAY BE LOADED BY THE ABSOLUTE PAPER TAPE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE MEDIA USING THE ASSOCIATED 'XDP' LOADER. THE PROGRAM MAY NOT

95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150

BE INCLUDED IN AN 'XXDP' CHAIN.

4. STARTING PROCEDURES

4.1 STARTING ADDRESSES

- A. THE NORMAL STARTING ADDRESS OF THE PROGRAM IS LOCATION 200(8). STARTING AT THIS ADDRESS ALLOWS THE OPERATOR TO SELECT (OR RESELECT) THE ADDRESS OF THE DRIVE TO BE TESTED.
- B. THE RESTART ADDRESS IS LOCATION 204(8). THE PROGRAM WILL USE THE CURRENT DRIVE (DCL) ADDRESS.
- C. THE PROGRAM CAN BE STARTED AT LOCATION 210(8) TO ALLOW THE RH11 OR RH70 ADDRESS TO BE CHANGED.

4.2 UNIBUS & VECTOR ADDRESS

THE PROGRAM ASSUMES THE FOLLOWING UNIBUS AND VECTOR ADDRESSES. THESE ADDRESSES MAY BE CHANGED PRIOR TO STARTING THE PROGRAM FROM ANY OF THE STARTING LOCATIONS.

MEMORY LOCATION	CONTENTS	FUNCTION
-----	-----	-----
1142	177560	TTY KEYBOARD STATUS REG
1144	177562	TTY KEYBOARD BUFFER REG
1146	177564	TTY PRINTER STATUS REG
1150	177566	TTY PRINTER BUFFER REG
1210	172540	KW11-P STATUS REG
1212	172542	KW11-L COUNTER BUFFER
1214	104	KW11-P VECTOR ADDRESS
1216	177546	KW11-L STATUS REGISTER
1220	100	KW11-L VECTOR ADDRESS

4.3 OPERATOR ACTION

- A. CONNECT THE DUAL PORT TEST CABLE BETWEEN BUS A & BUS B ON THE DRIVE BEING TESTED. (SEE SECTION 5.4)
- B. LOAD THE PROGRAM INTO MEMORY IN THE PROCESSOR CONTROLLING THE MASSBUS USED FOR TESTING.
- C. SWITCH THE 'PORT SELECT' SWITCH ON THE DRIVE TO BE TESTED TO THE 'A/B' POSITION. CYCLE THE DRIVE UP.
- D. LOAD THE APPROPRIATE STARTING ADDRESS (200(8) OR 210(8)) INTO THE SWITCH REGISTER (OR THE 'SOFTWARE' SWITCH REGISTER, SEE SECTION 5.2.)
- E. PRESS START.
- F. ENTER THE DRIVE NUMBER.
- G. ENTER THE NUMBER OF THE TEST TO BE RUN. ('CARRIAGE RETURN' OR '0' WILL RUN ALL TESTS.)
- H. THE PROGRAM MAY BE STOPPED AT ANY TIME AND RESTARTED

151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
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

FROM LOCATION 204.

5. OPERATING PROCEDURES

5.1 OPERATIONAL SWITCH SETTINGS

WITH ALL SWITCHES SET TO ZERO, THE PROGRAM WILL TYPE ALL ERRORS AND CONTINUE TESTING.

THE SWITCH SETTINGS ARE:

SW<15>=1...HALT ON ERROR
SW<14>=1...LOOP ON TEST
SW<13>=1...INHIBIT ERROR TYPEOUTS
SW<11>=1...INHIBIT TEST ITERATIONS
SW<10>=1...RING TTY BELL ON ERROR
SW<09>=1...LOOP ON ERROR

5.2 'SOFTWARE' SWITCH REGISTER

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RMO3 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

5.3 TEST SELECTION

INDIVIDUAL TESTS ARE SELECTED IN RESPONSE TO THE 'ENTER TEST NUMBER:' MESSAGE. ANY VALID TEST NUMBER CAN BE ENTERED. EACH ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN (CR). THE LOOP ON TEST SWITCH, SW<15>, MUST BE SET TO ALL CONTINUOUS EXECUTION OF THE SELECTED TEST.

TO RUN ALL TESTS IN SEQUENCE, ENTER EITHER A '0' FOLLOWED

207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262

263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318

BY A CARRIAGE RETURN OR A CARRIAGE RETURN BY ITSELF. THE PROGRAM WILL THEN EXECUTE ALL TESTS IN SEQUENCE.

THE 'RUBOUT KEY' (RO) CAN BE USED TO DELETE THE LAST CHARACTER ENTERED. SUCCESSIVELY STRIKING THE RO KEY WILL DELETE CHARACTERS UNTIL THE PREVIOUS CHARACTERS HAVE BEEN DELETED. CHARACTERS DELETED BY THE RO KEY WILL BE TYPED AND WILL BE SEPARATED BY '\ ' FROM THE CHARACTERS ENTERED BY THE OPERATOR.

THE OPERATOR CAN DELETE AN ENTIRE ENTRY BY TYPING A 'CONTROL U' .

5.4 TEST CABLE CONNECTION

TO TEST THE RMD3 DUAL PORT OPTION WITH THIS PROGRAM, A SPECIAL TEST CABLE MUST BE USED. (THE TEST CABLE IS P/N 7010507-02). THE TEST CABLE CONNECTS MASSBUS A & MASSBUS B TOGETHER AT THE DRIVE BEING TESTED AND IS CONSTRUCTED SO THAT BIT 0 OF THE MASSBUS UNIT SELECT LINES IS COMPLEMENTED.

WITH THE TEST CABLE CONNECTED TO THE DRIVE UNDER TEST, THE DRIVE APPEARS AS TWO UNITS ON THE MASSBUS: EACH PORT OF THE RMD3 WILL RESPOND TO A DIFFERENT MASSBUS ADDRESS.

THE ADDRESS OF EACH PORT WILL DEPEND UPON THE DRIVE'S ADDRESS PLUG.

THE PROGRAM WILL TYPEOUT THE APPARENT ADDRESSES OF BOTH PORTS. (ONE PORT WILL HAVE THE ADDRESS OF THE DRIVE; THE OTHER PORT WILL HAVE THE ADDRESS DEVELOPED BY THE CABLE).

 * ANY OTHER DRIVE ON THE MASSBUS WHICH HAS AN ADDRESS IN *
 * CONFLICT WITH EITHER OF THE TEST ADDRESSES MUST BE *
 * POWERED DOWN. *

THE TEST CABLE CONNECTION TO THE DRIVE UNDER TEST WILL DEPEND ON WHICH PROCESSOR/RH11 IS TO TEST THE DRIVE. IF THE DRIVE IS TO BE TESTED BY THE PROCESSOR ON PORT A, THE TEST CABLE IS CONNECTED FROM 'BUS A OUT' TO 'BUS B IN'. IF THE DRIVE IS TO BE TESTED BY THE PORT B PROCESSOR, THE TEST CABLE IS CONNECTED FROM 'BUS B OUT' TO 'BUS A IN'.

WHEN THE DUAL PORT TEST CABLE IS CONNECTED, THE ATTENTION BITS FOR PORTS A & B ARE ASSERTED IN THE SAME BIT POSITION WHEN 'RMS' (ATTENTION SUMMARY REGISTER) IS READ. THE ATTENTION BIT POSITION IS DETERMINED BY THE ADDRESS OF THE DRIVE. THE ATTENTION BIT THAT APPEARS FOR THE DRIVE IS THE INCLUSIVE 'OR' OF THE PORT A & PORT B ATTENTION BITS. BECAUSE OF THIS, THE PROGRAM LOOKS AT ONLY THE ATTENTION BIT IN 'RMS1' (DRIVE STATUS REGISTER) TO DETERMINE THE STATE OF THE SELECTED PORT'S ATTENTION BIT.

319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
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
3746. ERRORS

WHEN THE PROGRAM ENCOUNTERS AN ERROR, THE ERROR ROUTINE IS CALLED AND IF SW<13> IS NOT SET, THE ERROR MESSAGE PERTAINING TO THE ERROR WILL BE TYPED. EACH ERROR TYPEOUT WILL CONTAIN THE FOLLOWING:

- A. AN ERROR MESSAGE
- B. A DATA HEADER LINE
- C. A DATA LINE CONTAINING:
 - 1. THE TEST NUMBER
 - 2. THE PC (PROGRAM COUNTER VALUE) WHERE THE ERROR CALL WAS MADE
 - 3. CONTENTS OF THE APPROPRIATE REGISTERS

7. MISCELLANEOUS

7.1 RESTRICTIONS

TO RUN THIS PROGRAM, THE SYSTEM MUST HAVE EITHER A KW11-P OR A KW11-L CLOCK. ADDITIONALLY, THE RMO3 UNDER TEST MUST HAVE THE DUAL PORT TEST CABLE CONNECTED.

7.2 LIMITATIONS

THIS PROGRAM DOES NOT TEST DATA TRANSFERS THROUGH EITHER PORT AND DOES NOT TEST THE DYNAMIC OPERATION OF THE DUAL PORT OPTION.

7.3 EXECUTION TIME

THE PROGRAM TAKES ABOUT 60 SECONDS PER PASS (DEPENDING ON OPERATOR INTERVENTION EFFICIENCY).

7.4 REQUIRED TESTS

IF THE PROGRAM IS BEING EXECUTED IN SINGLE TEST MODE, THE OPERATOR MUST CALL AND RUN THE FOLLOWING TESTS BEFORE OTHER TESTS ARE RUN:

- A. TEST 2 AND TEST 3. THESE TESTS SET 'VV-A' AND 'VV-B' RESPECTIVELY. THESE TESTS MUST BE PERFORMED AT LEAST ONCE BEFORE TESTS 4 - 10 ARE RUN.
- B. TEST 4 AND TEST 5. THESE TESTS DETERMINE AND STORE FOR LATER USE THE TIMEOUT ONE-SHOT VALUE MEASURED THROUGH EACH PORT. THESE TESTS MUST BE PERFORMED AT LEAST ONCE BEFORE TESTS 6 - 10 ARE RUN.

7.5 DISK SURFACE USAGE

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
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

THE DIAGNOSTIC DOES NOT USE THE DISK SURFACE. HOWEVER, THE DRIVE MUST BE CYCLED UP AND ON LINE FOR THE DIAGNOSTIC TO BE RUN.

7.8 LOOP ON ERROR OPTION

IF SW<09> IS SET, THE PROGRAM WILL LOOP ON A FAILING TEST UNTIL EITHER THE SWITCH IS RESET OR THE ERROR STOPS OCCURING. BECAUSE THE PROGRAM MUST RESET THE RMD3 TO A KNOWN STATE BEFORE LOOPING ON THE ERROR, THE TEST FOR SW<09> IS PERFORMED AT THE END OF THE TEST - NOT AT THE POINT WHERE THE ERROR WAS DETECTED.

8. TEST DESCRIPTIONS

8.1 METHOD USED TO VERIFY THAT DRIVE IS IN NEUTRAL

THE PROGRAM DETERMINES THE THE DRIVE IS IN NEUTRAL BY CHECKING THE CONTENTS OF THE DRIVE STATUS REGISTER (RMD51) THROUGH BOTH PORTS. THE PROGRAM MASKS OUT THE PORT DEPENDENT BITS ('ATA' & 'VV') AND VERIFIES THAT CORRECT STATUS IS READ THROUGH BOTH PORTS. (THE CORRECT STATUS IS 'MOL', 'PGM', 'DPR', & 'DRY'.) IF NEITHER PORT SEES ALL ZEROS FROM RMD51, THE PROGRAM CONCLUDES THAT THE DRIVE IS IN NEUTRAL AND THAT ANY BIT DISCREPANCY BETWEEN PORTS INDICATES A FAILURE IN THE PATH FOR THAT BIT.

8.2 METHOD USED TO VERIFY THAT THE DRIVE HAS BEEN SEIZED

THE PROGRAM VERIFIES THAT THE DRIVE HAS BEEN SEIZED BY CHECKING THE DRIVE STATUS REGISTER (RMD51) THROUGH THE SEIZING PORT AND VERIFYING THAT CORRECT STATUS IS SEEN. WHEN RMD51 IS READ THROUGH THE OPPOSITE PORT, ZEROS SHOULD BE SEEN. IF BOTH CONDITIONS EXIST. (I.E. THE OPPOSITE PORT), THE PROGRAM CONCLUDES THAT THE DRIVE HAS BEEN SEIZED BY THE SPECIFIED PORT.

TEST 1 DRIVE ACCESS TEST

VERIFY THAT THE DRIVE CAN BE ACCESSED THROUGH BOTH PORTS

- A. SELECT DRIVE, VERIFY THAT THE DRIVE IS PRESENT, THAT THE DRIVE IS A DUAL PORT RMD3, THAT THE DRIVE IS ONLINE (RMD51 HAS 'MOL', 'PGM', 'DPR', & 'DRY' BITS SET), AND THE THE DRIVE SERIAL NUMBER READ THROUGH BOTH PORTS IS THE SAME.
- B. THE TEST IS REPEATED THROUGH BOTH PORTS.

43
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
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

TEST 2 SET 'VV' FOR PORT A

SET VOLUME VALID

- A. ISSUE A DRIVE CLEAR COMMAND THROUGH PORT A.
- B. ISSUE A READIN PRESET COMMAND THROUGH PORT A. VERIFY THAT THE 'VV' BIT IS SET FOR PORT A.
- C. ISSUE A RELEASE COMMAND THROUGH PORT A. VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION BIT IS SET.

TEST 3 SET 'VV' FOR PORT B

SET VOLUME VALID

- A. ISSUE A DRIVE CLEAR COMMAND THROUGH PORT B.
- B. ISSUE A READIN PRESET COMMAND THROUGH PORT B. VERIFY THAT THE 'VV' BIT IS SET FOR PORT B.
- C. ISSUE A RELEASE COMMAND THROUGH PORT B. VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION BIT IS SET.

TEST 4 MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A

MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT A

- A. WRITE 0'S INTO RMO31 THROUGH PORT A AND VERIFY THAT THE DRIVE HAS BEEN SEIZED.
- B. WAIT FOR TIMEOUT TO OCCUR. MEASURE THE DURATION OF THE TIMEOUT ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
- C. VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS TO NEUTRAL

TEST 5 MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT B

MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT B

- A. WRITE 0'S INTO RMO31 THROUGH PORT B AND VERIFY THAT THE DRIVE HAS BEEN SEIZED.

K01

487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542

- B. WAIT FOR TIMEOUT TO OCCUR. MEASURE THE DURATION OF THE TIMEOUT ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
- C. VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS TO NEUTRAL

TEST 6 TEST 'CONTROLLER SELECT' SWITCH, DRIVE CYCLED UP

TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED UP).

- A. SWITCH TO CONTROLLER 'A' POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMD51, AS READ THROUGH BOTH PORTS, ARE CORRECT.
- B. SWITCH TO CONTROLLER 'B' POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMD51, AS READ THROUGH BOTH PORTS, ARE CORRECT.
- C. RETURN THE 'CONTROLLER SELECT' SWITCH TO THE 'A/B' POSITION. VERIFY THE DRIVE STATE.

TEST 7 TEST 'CONTROLLER SELECT' SWITCH LOCKED ON PORT A

TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).

- A. CYCLE THE DRIVE DOWN.
- B. SWITCH TO CONTROLLER A POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMD51, AS READ THROUGH BOTH PORTS, ARE CORRECT.
- C. SWITCH THE 'CONTROLLER SELECT' SWITCH TO A; CYCLE THE DRIVE UP.
- D. WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-A IS RESET, AND THAT 'ATA-A IS SET.
- E. ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH PORT A.
- F. VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT B AND 'NED' SETS WHEN ATTEMPTING TO ACCESS THE DRIVE THROUGH PORT B. ATTEMPT TO SET PORT REQUEST BY WRITING 0'S INTO RMD51 THROUGH PORT B.
- G. ISSUE A RELEASE COMMAND THROUGH PORT A. VERIFY THAT THE DRIVE REMAINS LOCKED ON PORT A.

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
571
572
573
574
575
576
577
578

TEST 10 TEST 'CONTROLLER SELECT' SWITCH LOCKED ON PORT B

TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).

- A. CYCLE THE DRIVE DOWN.
- B. SWITCH TO CONTROLLER B POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMD51, AS READ THROUGH BOTH PORTS, ARE CORRECT.
- C. SWITCH THE 'CONTROLLER SELECT' SWITCH TO B; CYCLE THE DRIVE UP.
- D. WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-B IS RESET, AND THAT 'ATA-B IS SET.
- E. ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH PORT B.
- F. VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT A AND 'NED' SETS WHEN ATTEMPTING TO ACCESS THE DRIVE THROUGH PORT A. ATTEMPT TO SET PORT REQUEST BY WRITING 0'S INTO RMD51 THROUGH PORT A.
- G. ISSUE A RELEASE COMMAND THROUGH PORT B. VERIFY THAT THE DRIVE REMAINS LOCKED ON PORT B.
- H. CYCLE THE DRIVE DOWN. CHANGE THE 'CONTROLLER SELECT' SWITCH TO A/B; CYCLE THE DRIVE UP.
- I. VERIFY THAT BOTH PORTS CAN ACCESS THE DRIVE, THAT BOTH ATTENTION BITS ARE SET, AND THAT BOTH 'VV' BITS ARE RESET.

↑

579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634

;PROGRAM REVISION #001

```
.TITLE MD-11-DZRMH-A, RM03 DUAL CONTRLR LOGIC TEST - PART 2
;*COPYRIGHT (C) 1977
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY D. RIIKONEN
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
;*
```

.SBTTL OPERATIONAL SWITCH SETTINGS

```
*
* SWITCH USE
*-----
* 15 HALT ON ERROR
* 14 LOOP ON TEST
* 13 INHIBIT ERROR TYPEOUTS
* 11 INHIBIT ITERATIONS
* 10 BELL ON ERROR
* 9 LOOP ON ERROR
```

.SBTTL BASIC DEFINITIONS

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

;*MISCELLANEOUS DEFINITIONS

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

;*GENERAL PURPOSE REGISTER DEFINITIONS

```
R0= %0 ;;GENERAL REGISTER
R1= %1 ;;GENERAL REGISTER
R2= %2 ;;GENERAL REGISTER
R3= %3 ;;GENERAL REGISTER
R4= %4 ;;GENERAL REGISTER
R5= %5 ;;GENERAL REGISTER
R6= %6 ;;GENERAL REGISTER
R7= %7 ;;GENERAL REGISTER
SP= %6 ;;STACK POINTER
```

001100

000011
000012
000015
000200
177776
177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007
000006

```

635      000007      PC=      %7      ;;PROGRAM COUNTER
636
637
638      000000      ;*PRIORITY LEVEL DEFINITIONS
639      000040      PR0=      0      ;;PRIORITY LEVEL 0
640      000100      PR1=      40     ;;PRIORITY LEVEL 1
641      000140      PR2=      100    ;;PRIORITY LEVEL 2
642      000200      PR3=      140    ;;PRIORITY LEVEL 3
643      000240      PR4=      200    ;;PRIORITY LEVEL 4
644      000300      PR5=      240    ;;PRIORITY LEVEL 5
645      000340      PR6=      300    ;;PRIORITY LEVEL 6
646
647
648      100000      ;*"SWITCH REGISTER" SWITCH DEFINITIONS
649      040000      SW15=     100000
650      020000      SW14=     40000
651      010000      SW13=     20000
652      004000      SW12=     10000
653      002000      SW11=     4000
654      001000      SW10=     2000
655      000400      SW09=     1000
656      000200      SW08=     400
657      000100      SW07=     200
658      000040      SW06=     100
659      000020      SW05=     40
660      000010      SW04=     20
661      000004      SW03=     10
662      000002      SW02=     4
663      000001      SW01=     2
664      .EQUIV      SW00=     1
665      .EQUIV      SW09, SW9
666      .EQUIV      SW08, SW8
667      .EQUIV      SW07, SW7
668      .EQUIV      SW06, SW6
669      .EQUIV      SW05, SW5
670      .EQUIV      SW04, SW4
671      .EQUIV      SW03, SW3
672      .EQUIV      SW02, SW2
673      .EQUIV      SW01, SW1
674      .EQUIV      SW00, SW0
675
676      100000      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
677      040000      BIT15=    100000
678      020000      BIT14=    40000
679      010000      BIT13=    20000
680      004000      BIT12=    10000
681      002000      BIT11=    4000
682      001000      BIT10=    2000
683      000400      BIT09=    1000
684      000200      BIT08=    400
685      000100      BIT07=    200
686      000040      BIT06=    100
687      000020      BIT05=    40
688      000010      BIT04=    20
689      000004      BIT03=    10
690      000002      BIT02=    4
        000001      BIT01=    2
    
```

```

691          000001      BIT00= 1
692          .EQUIV      BIT09,BIT9
693          .EQUIV      BIT08,BIT8
694          .EQUIV      BIT07,BIT7
695          .EQUIV      BIT06,BIT6
696          .EQUIV      BIT05,BIT5
697          .EQUIV      BIT04,BIT4
698          .EQUIV      BIT03,BIT3
699          .EQUIV      BIT02,BIT2
700          .EQUIV      BIT01,BIT1
701          .EQUIV      BIT00,BIT0
702
703          ;*BASIC "CPU" TRAP VECTOR ADDRESSES
704          000004      ERRVEC= 4          ;: TIME OUT AND OTHER ERRORS
705          000010      RESVEC= 10         ;: RESERVED AND ILLEGAL INSTRUCTIONS
706          000014      TBITVEC=14        ;: "T" BIT
707          000014      TRTVEC= 14         ;: TRACE TRAP
708          000014      BPTVEC= 14         ;: BREAKPOINT TRAP (BPT)
709          000020      IOTVEC= 20         ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
710          000024      PMRVEC= 24         ;: POWER FAIL
711          000030      EMTVEC= 30         ;: EMULATOR TRAP (EMT) **ERROR**
712          000034      TRAPVEC=34        ;: "TRAP" TRAP
713          000060      TKVEC= 60          ;: TTY KEYBOARD VECTOR
714          000064      TPVEC= 64         ;: TTY PRINTER VECTOR
715          000240      PIRQVEC=240       ;: PROGRAM INTERRUPT REQUEST VECTOR
716
717          ;:*****
718
719          ;:*****
720
721          .SBTTL      RH11 REGISTERS
722
723          ;:*****
724
725          ;CONTROL AND STATUS REGISTER 1 (RMCS1)
726
727          000100      IE= 100             ;: INTERRUPT ENABLE (BIT #6)
728          000200      RDY= 200           ;: READY (BIT #7)
729          000400      A16= 400           ;: HIGH ORDER BUS ADDRESS BIT (BIT #8)
730          001000      A17= 1000         ;: HIGH ORDER BUS ADDRESS BIT (BIT #9)
731          002000      PSEL= 2000        ;: PORT SELECT (BIT #10)
732          020000      MCPE= 20000       ;: MASSBUSS PARITY ERROR (BIT #13)
733          040000      TRE= 40000        ;: TRANSFER ERROR (BIT #14)
734          100000      SC= 100000        ;: SPECIAL CONDITION (BIT #15)
735
736          ;WORD COUNT REGISTER (RMC)
737          ;(EACH BIT IS CALLED BY BIT NUMBER)
738
739          ;BUS ADDRESS REGISTER (RMBA)
740          ;(EACH BIT IS CALLED BY BIT NUMBER)
741
742          ;CONTROL AND STATUS REGISTER 2 (RMCS2)
743
744          000001      U0= 1               ;: UNIT SELECT (BIT #0)
745          000002      U1= 2             ;: UNIT SELECT (BIT #1)
746          000004      U3= 4             ;: UNIT SELECT (BIT #2)

```

RH11 REGISTERS

747 000010
748 000020
749 000040
750 000100
751 000200
752 000400
753 001000
754 002000
755 004000
756 010000
757 020000
758 040000
759 100000

BAI= 10
PAT= 20
CLR= 40
IR= 100
OR= 200
MOPE= 400
MXF= 1000
PGE= 2000
NEM= 4000
NED= 10000
UPE= 20000
WCE= 40000
DLT= 100000

;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
;MASSBUS PARITY TEST (BIT #4)
;CLEAR (BIT #5)
;INPUT READY (BIT #6)
;OUTPUT READY (BIT #7)
;MASS BUS PARITY ERROR (BIT #8)
;MISSED TRANSFER ERROR (BIT #9)
;PROGRAM ERROR (BIT #10)
;NON EXISTENT MEMORY (BIT #11)
;NON EXISTENT DRIVE (BIT #12)
;UNIBUS PARITY ERROR (BIT #13)
;WRITE CHECK ERROR (BIT #14)
;DATA LATE (BIT #15)

;DATA BUFFER REGISTER (RMD8)
;(EACH BIT IS CALLED BY BIT NUMBER)

;*****

.SBTTL RMD3 REGISTERS

;*****

;CONTROL AND STATUS 1 REGISTER. (#00)

773 000001
774 000002
775 000004
776 000010
777 000020
778 000040
779 004000

GO= 1
FO= 2
F1= 4
F2= 10
F3= 20
F4= 40
DVA= 4000

;GO BIT (BIT #0)
;FUNCTION CODE BIT #1
;FUNCTION CODE BIT #2
;FUNCTION CODE BIT #3
;FUNCTION CODE BIT #4
;FUNCTION CODE BIT #5
;DEVICE AVAILABLE (BIT #11)

;DRIVE STATUS REGISTER (RMD51) (#01)

783 000002
784 000004
785 000010
786 000020
787 000040
788 000100
789 000200
790 000400
791 001000
792 002000
793 004000
794 010000
795 020000
796 040000
797 100000

DFS= 1
DFF20= 2
DIGB= 4
GRV= 10
DL64= 20
DE1= 40
VV= 100
DRY= 200
DPR= 400
PGM= 1000
LBT= 2000
WRL= 4000
MOL= 10000
PIP= 20000
ERR= 40000
ATA= 100000

DRIVE FORWARD 5"/SEC. (BIT #0)
;DRIVE FORWARD 20"/SEC. (BIT #1)
;DRIVE TO INNER GUARD BAND (BIT #2)
;GO REVERSE (BIT #3)
;DIFFERENCE LESS THAN 64 (BIT #4)
;DIFFERENCE EQUALS 1 (BIT #5)
;VOLUME VALID (BIT #6)
;DRIVE READY (BIT #7)
;DRIVE PRESENT (BIT #8)
;PROGRAMABLE (BIT #9)
;LAST SECTOR TRANSFERRED (BIT #10)
;WRITE LOCK (BIT #11)
;MEDIUM ON-LINE (BIT #12)
;POSITIONING OPERATION IN PROGRESS (BIT #13)
;COMPOSITE ERROR (BIT #14)
;ATTENTION ACTIVE (BIT #15)

;ERROR REGISTER #01 (RMER1) (#02)

802 000001

ILF= 1

;ILLEGAL FUNCTION (BIT #0)

803	000002	ILR=	2	; ILLEGAL REGISTER (BIT #1)
804	000004	RMR=	4	; REGISTER MODIFICATION REFUSED (BIT #2)
805	000010	PAR=	10	; PARITY ERROR (BIT #3)
806	000020	FER=	20	; FORMAT ERROR (BIT #4)
807	000040	WCF=	40	; WRITE CLOCK FAIL (BIT #5)
808	000100	ECH=	100	; ECC HARD ERROR (BIT #6)
809	000200	HCE=	200	; HEADER COMPARE ERROR (BIT #7)
810	000400	HCRC=	400	; HEADER CRC ERROR (BIT #8)
811	001000	AOE=	1000	; ADDRESS OVERFLOW ERROR (BIT #9)
812	002000	IAE=	2000	; INVALID ADDRESS ERROR (BIT #10)
813	004000	WLE=	4000	; WRITE LOCK ERROR (BIT #11)
814	010000	DTE=	10000	; DRIVE TIMING ERROR (BIT #12)
815	020000	OPI=	20000	; OPERATION INCOMPLETE (BIT #13)
816	040000	UNS=	40000	; DRIVE UNSAFE (BIT #14)
817	100000	DCK=	100000	; DATA CHECK ERROR (BIT 15)
818				
819				; MAINTAINABILITY REGISTER (RMMR1) (#03)
820				
821	000001	DMD=	1	; DIAGNOSTIC MODE (BIT #0)
822				
823				; ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
824				
825	000001	AT0=	1	; DEVICE 0 (BIT #0)
826	000002	AT1=	2	; DEVICE 1 (BIT #1)
827	000004	AT2=	4	; DEVICE 2 (BIT #2)
828	000010	AT3=	10	; DEVICE 3 (BIT #3)
829	000020	AT4=	20	; DEVICE 4 (BIT #4)
830	000040	AT5=	40	; DEVICE 5 (BIT #5)
831	000100	AT6=	100	; DEVICE 6 (BIT #6)
832	000200	AT7=	200	; DEVICE 7 (BIT #7)
833				
834				; DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
835				; (EACH BIT IS CALLED BY BIT NUMBER)
836				
837				; DRIVE TYPE REGISTER (RMDT) (#06)
838				
839	000001	DT00=	1	; DRIVE TYPE NUMBER BIT 1
840	000002	DT01=	2	; DRIVE TYPE NUMBER BIT 2
841	000004	DT02=	4	; DRIVE TYPE NUMBER BIT 3
842	000010	DT03=	10	; DRIVE TYPE NUMBER BIT 4
843	000020	DT04=	20	; DRIVE TYPE NUMBER BIT 5
844	000040	DT05=	40	; DRIVE TYPE NUMBER BIT 6
845	000100	DT06=	100	; DRIVE TYPE NUMBER BIT 7
846	000200	DT07=	200	; DRIVE TYPE NUMBER BIT 8
847	000400	DT08=	400	; DRIVE TYPE NUMBER BIT 9
848	004000	DRQ=	4000	; DRIVE REQUEST REQUIRED (BIT #11)
849	020000	MOH=	20000	; MOVING HEAD (BIT #13)
850	040000	TAP=	40000	; TAPE DRIVE (BIT #14)
851	100000	NBA=	100000	; NOT BLOCK ADDRESSED (BIT #15)
852				
853				; LOOK-AHEAD REGISTER (RMLA) (#07)
854				
855	000100	SC0=	100	; SECTOR COUNT FIELD 0 (BIT #6)
856	000200	SC1=	200	; SECTOR COUNT FIELD 1 (BIT #7)
857	000400	SC2=	400	; SECTOR COUNT FIELD 2 (BIT #8)
858	001000	SC3=	1000	; SECTOR COUNT FIELD 3 (BIT #9)

RMO3 REGISTERS

```

859      002000      SC4=      2000      ;SECTOR COUNT FIELD 4 (BIT #10)
860
861      ;RMO3 ERROR REGISTER #2 (RMR2) (#10)
862
863      000010      DPE=      10      ;DATA PARITY ERROR (BIT #3)
864      000200      DVC=      200     ;DEVICE CHECK (BIT #7)
865      002000      LBC=      2000    ;LOSS OF BIT CLOCK (BIT #10)
866      004000      LSC=      4000    ;LOSS OF SYSTEM CLOCK (BIT #11)
867      010000      IVC=      10000   ;INVALID COMMAND (BIT #12)
868      020000      OPE=      20000   ;OPERATOR ERROR (BIT #13)
869      100000      SKI=      100000  ;SEEK INCOMPLETE (BIT #14)
870
871      ;OFFSET REGISTER (RMOF) (#11)
872
873      000200      OFD=      200     ;OFFSET FORWARD (BIT #5)
874      002000      HCI=      2000    ;HEADER COMPARE INHIBIT (BIT #10)
875      004000      ECI=      4000    ;ERROR CORRECTION CODE INHIBIT (BIT #11)
876      010000      FMT16= 10000   ;FORMAT BIT (BIT #12)
877
878      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
879      ;(EACH BIT IS CALLED BY BIT NUMBER)
880
881      ;SERIAL NUMBER REGISTER (RMSN) (#14)
882      ;(EACH IS CALLED BY BIT NUMBER)
883
884      ;ECC POSITION REGISTER (RMEC1) (#16)
885      ;(EACH BIT IS CALLED BY BIT NUMBER)
886
887      ;ECC PATTERN REGISTER (RMEC2) (#17)
888      ;(EACH BIT IS CALLED BY BIT NUMBER)
889
890      ;*****
891
892      .SBTTL  DEFINITIONS OF THE RH11/RMO3 ADDRESS INDEXES
893
894      ;*****
895
896      000000      RMCS1=0      ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
897      000002      RMWC=2      ;WORD COUNT REGISTER (NOT A DRIVE REG)
898      000004      RMBA=4      ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
899      000006      RMDA=6      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
900      000010      RMCS2=10     ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
901      000012      RMDI=12     ;DRIVE STATUS REGISTER (DRIVE REG 01)
902      000014      RMR1=14     ;ERROR REGISTER #1 (DRIVE REG. 02)
903      000016      RMAI=16     ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
904      000020      RMLA=20     ;LOOK AHEAD REGISTER (DRIVE REG. 07)
905      000022      RMDB=22     ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
906      000024      RMR1=24     ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
907      000026      RMDT=26     ;DRIVE TYPE REGISTER (DRIVE REG. 06)
908      000030      RMSN=30     ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
909      000032      RMOF=32     ;OFFSET REGISTER (DRIVE REG. 11)
910      000034      RMDC=34     ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
911      000040      RMR2=40     ;MAINTENANCE REGISTER #2 (DRIVE REG. 14)
912      000042      RMR2=42     ;ERROR REGISTER #2 (DRIVE REG. 15)
913      000044      RMEC1=44    ;ECC POSITION REGISTER (DRIVE REG. 16)
914      000046      RMEC2=46    ;ECC PATTERN REGISTER (DRIVE REG. 17)

```

```

915
916      .SBTTL TRAP CATCHER
917
918      000000      .=0
919      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A "+2,HALT"
920      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
921      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
922
923      000174      000174      .=174
924      000176      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
925      000176      000000      SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
926
927      .SBTTL ACT11 HOOKS
928
929      ;*****
930      ;HOOKS REQUIRED BY ACT11
931      $SVPC=.      ;SAVE PC
932      000046      000046      .=46
933      000046      013206      SENDAD      ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
934      000052      020000      .=52
935      000200      000200      .WORD 20000      ;;2)SET LOC.52 TO 20000
936      000200      000200      .=$SVPC      ;; RESTORE PC
937
938      .SBTTL STARTING ADDRESS = 200
939      000200      000137      001706      JMP START      ;START THE PROGRAM
940
941      .SBTTL START THE PROGRAM AND CHANGE THE RH11 ADDRESS = 204
942      000204      000137      001714      JMP START1     ;START AND CHANGE THE RH11 ADDRESS
943
944
945
    
```

COMMON TAGS

.SBTTL COMMON TAGS

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

946									
947									
948									
949									
950									
951									
952		001100							
953	001100			SCMTAG:					START OF COMMON TAGS
954	001100	000000		SPASS:	.WORD	0			CONTAINS PASS COUNT
955	001102	000		STSTNM:	.BYTE	0			CONTAINS THE TEST NUMBER
956	001103	000		SERFLG:	.BYTE	0			CONTAINS ERROR FLAG
957	001104	000000		SICNT:	.WORD	0			CONTAINS SUBTEST ITERATION COUNT
958	001106	000000		SLPADR:	.WORD	0			CONTAINS SCOPE LOOP ADDRESS
959	001110	000000		SLPERR:	.WORD	0			CONTAINS SCOPE RETURN FOR ERRORS
960	001112	000000		SERTTL:	.WORD	0			CONTAINS TOTAL ERRORS DETECTED
961	001114	000		SITEMB:	.BYTE	0			CONTAINS ITEM CONTROL BYTE
962	001115	001		SERMAX:	.BYTE	1			CONTAINS MAX. ERRORS PER TEST
963	001116	000000		SERRPC:	.WORD	0			CONTAINS PC OF LAST ERROR INSTRUCTION
964	001120	000000		\$GADDR:	.WORD	0			CONTAINS ADDRESS OF 'GOOD' DATA
965	001122	000000		\$BADDR:	.WORD	0			CONTAINS ADDRESS OF 'BAD' DATA
966	001124	000000		\$GDDAT:	.WORD	0			CONTAINS 'GOOD' DATA
967	001126	000000		\$BDDAT:	.WORD	0			CONTAINS 'BAD' DATA
968	001130	000000			.WORD	0			RESERVED--NOT TO BE USED
969	001132	000000			.WORD	0			
970	001134	000		SAUTOB:	.BYTE	0			AUTOMATIC MODE INDICATOR
971	001135	000		SINTAG:	.BYTE	0			INTERRUPT MODE INDICATOR
972	001136	000000			.WORD	0			
973	001140	177570		SWR:	.WORD	DSWR			ADDRESS OF SWITCH REGISTER
974	001142	177570		DISPLAY:	.WORD	DLISP			ADDRESS OF DISPLAY REGISTER
975	001144	177560		\$TKS:	177560				TTY KBD STATUS
976	001146	177562		\$TKB:	177562				TTY KBD BUFFER
977	001150	177564		\$TPS:	177564				TTY PRINTER STATUS REG. ADDRESS
978	001152	177566		\$TPB:	177566				TTY PRINTER BUFFER REG. ADDRESS
979	001154	000		\$NULL:	.BYTE	0			CONTAINS NULL CHARACTER FOR FILLS
980	001155	002		\$FILLS:	.BYTE	2			CONTAINS # OF FILLER CHARACTERS REQUIRED
981	001156	012		\$FILLC:	.BYTE	12			INSERT FILL CHARS. AFTER A "LINE FEED"
982	001157	000		\$TPFLG:	.BYTE	0			"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
983	001160	000000		\$REGAD:	.WORD	0			CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
984									CONTAINS ((\$REGAD)+0)
985	001162	000000		\$REGO:	.WORD	0			USER DEFINED
986	001164	000000		\$TMP0:	.WORD	0			USER DEFINED
987	001166	000000		\$TMP1:	.WORD	0			USER DEFINED
988	001170	000000		\$TMP2:	.WORD	0			USER DEFINED
989	001172	000000		\$TMP3:	.WORD	0			USER DEFINED
990	001174	000000		\$TMP4:	.WORD	0			USER DEFINED
991	001176	000000		\$TIMES:	0				MAX. NUMBER OF ITERATIONS
992	001200	000000		\$ESCAPE:	0				ESCAPE ON ERROR ADDRESS
993	001202	177607	000377	\$BELL:	.ASCIZ	<27><377><377>			CODE FOR BELL
994	001206	077		\$QUES:	.ASCII	??			QUESTION MARK
995	001207	015		\$CRLF:	.ASCII	<15>			CARRIAGE RETURN
996	001210	000012		\$LF:	.ASCIZ	<12>			LINE FEED
997									*****
998		000015		CR	=	15			
999		000012		LF	=	12			
1000	001212	172540		\$LKCSR:	.WORD	172540			ADDR OF KW11-P STATUS REGISTER
1001	001214	172542		\$LKCSB:	.WORD	172542			ADDR OF KW11-P COUNTER BUFFER

```

1002 001216 000104 $LPVEC: .WORD 104 ;ADDR OF KW11-P VECTOR
1003 001220 177546 $LKS: .WORD 177546 ;ADDR OF KW11-L STATUS REGISTER
1004 001222 000100 $LLVEC: .WORD 100 ;ADDR OF KW11-L VECTOR
1005 001224 000000 PORTA: .WORD 0 ;ADDRESS OF PORT A
1006 001226 000000 PORTB: .WORD 0 ;ADDRESS OF PORT B
1007 001230 000000 PORTC: .WORD 0 ;ADDRESS OF DIFFERENT DRIVE
1008 001232 000000 ASR1: .WORD 0 ;ATA-A OR ATA-B = 1
1009 001234 000000 PTNBR: .WORD 0 ;CONTAINS THE PORT ADDRESS FOR ERROR TYPEOUTS
1010 001236 000000 SEIZPT: .WORD 0 ;CONTAINS THE ADDRESS OF THE SEIZING PORT
1011 001240 000000 OPPRT: .WORD 0 ;CONTAINS THE ADDRESSES OF THE 'OPPOSITE' PORT
1012 001242 000000 TSTNUM: .WORD 0 ;NUMBER OF THE CURRENT TEST
1013 001244 000000 CKERR: .WORD 0 ;IF -1, A REGISTER MISCOMPARISON OCCURRED
1014 001246 000000 NOSEIZ: .WORD 0 ;IF -1, THE PORT IN 'SEIZPT' DID NOT SEIZE THE DRIVE
1015 001250 000000 RELERR: .WORD 0 ;IF -1, THE PORT IN 'SEIZPT' DID NOT RELEASE THE DRIVE
1016 001252 000000 TIME: .WORD 0 ;ELAPSED TIME COUNTER
1017 001254 000000 WATCH: .WORD 0 ;WATCH DOG TIMER LOCATION
1018 001256 000000 TIMEA: .WORD 0 ;THE TIMEOUT ONE-SHOT VALUE MEASURED THROUGH PORT A
1019 001260 000000 TIMEAP: .WORD 0 ;PORT A TIMEOUT VALUE + 25%
1020 001262 000000 TIMEB: .WORD 0 ;THE TIMEOUT ONE-SHOT VALUE MEASURED THROUGH PORT B
1021 001264 000000 TIMEBP: .WORD 0 ;PORT B TIMEOUT VALUE + 25%
1022 001266 000000 KYBCTL: .WORD 0 ;SINGLE TEST INDICATOR
1023 001270 000000 CHGADR: .WORD 0 ;CHANGE THE RH11 ADDRESS INDICATOR
1024
1025 ;*****
1026
1027 .SBTTL RH11/RM03 UNIBUS AND VECTOR ADDRESSES
1028
1029 ;*****
1030
1031 001272 176700 $RMADR: .WORD 176700 ;RH11/RM03 UNIBUS ADDRESS
1032 001274 000254 $RMVEC: .WORD 254 ;RH11 INTERRUPT VECTOR ADDRESS
1033

```

```

1034 .SBTTL ERROR POINTER TABLE
1035
1036 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1037 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1038 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1039 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1040 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1041
1042 ;*      EM      ;;POINTS TO THE ERROR MESSAGE
1043 ;*      DH      ;;POINTS TO THE DATA HEADER
1044 ;*      DT      ;;POINTS TO THE DATA
1045 ;*      DF      ;;POINTS TO THE DATA FORMAT
1046
1047
1048 $ERRTB:
1049
1050 ;ERROR 1
1051
1052
1053 001276 017735      EM1      ;DRIVE IS NON-EXISTENT ('MED' BIT SET)
1054 001300 022511      DH1
1055 001302 024072      DT1
1056 001304 024312      DF1
1057
1058 ;ERROR 2
1059
1060 001306 020003      EM2      ;WRONG DRIVE TYPE
1061 001310 022562      DH2
1062 001312 024106      DT2
1063 001314 024317      DF2
1064
1065 ;ERROR 3
1066
1067 001316 020024      EM3      ;CONTROLLER SELECT SWITCH ON DRIVE NOT IN 'A/B'
1068 001320 022511      DH1
1069 001322 024072      DT1
1070 001324 024312      DF1
1071
1072 ;ERROR 4
1073
1074 001326 020103      EM4      ;DRIVE NOT ON LINE
1075 001330 022562      DH2
1076 001332 024106      DT2
1077 001334 024317      DF2
1078
1079 ;ERROR 5
1080
1081 001336 020125      EM5      ;SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME
1082 001340 022636      DH5
1083 001342 024124      DT5
1084 001344 024325      DF5
1085
1086 ;ERROR 6
1087
1088 001346 020207      EM6      ;TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS
1089 001350 022705      DH6
    
```

1090	001352	024140	DT6	
1091	001354	024332	DF6	
1092				
1093				;ERROR 7
1094				
1095	001356	020261	EM7	;TIMEOUT ONE-SHOT IS LESS THAN 500 MS
1096	001360	022734	DH7	
1097	001362	024150	DT7	
1098	001364	024335	DF7	
1099				
1100				;ERROR 10
1101				
1102	001366	020326	EM10	;READIN PRESET DOES NOT SET VOLUME VALID FOR THE PORT
1103	001370	022511	DH1	
1104	001372	024072	DT1	
1105	001374	024312	DF1	
1106				
1107				;ERROR 11
1108				
1109	001376	020413	EM11	; 'GO' BIT RESET DURING UNLOAD COMMAND
1110	001400	022511	DH1	
1111	001402	024072	DT1	
1112	001404	024312	DF1	
1113				
1114				;ERROR 12
1115				
1116	001406	020460	EM12	;INCORRECT STATUS DURING UNLOAD COMMAND
1117	001410	022511	DH1	
1118	001412	024072	DT1	
1119	001414	024312	DF1	
1120				
1121				;ERROR 13
1122				
1123	001416	020527	EM13	;DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND
1124	001420	023001	DH13	
1125	001422	024162	DT13	
1126	001424	024332	DF6	
1127				
1128				;ERROR 14
1129				
1130	001426	020614	EM14	;ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD
1131	001430	023057	DH14	
1132	001432	024172	DT14	
1133	001434	024341	DF14	
1134				
1135				;ERROR 15
1136				
1137	001436	020676	EM15	;ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD'
1138	001440	022511	DH1	
1139	001442	024072	DT1	
1140	001444	024312	DF1	
1141				
1142				;ERROR 16
1143				
1144	001446	020762	EM16	;DRIVE NOT IN NEUTRAL AFTER UNLOAD WITH 'CONTROLLER ;SELECT' SWITCH MOVED FROM 'A/B'
1145				

1146	001450	023001	DH13	
1147	001452	024162	DT13	
1148	001454	024332	DF6	
1149				;ERROR 17
1150				
1151	001456	021106	EM17	;DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP
1152	001460	023177	DH17	
1153	001462	024210	DT17	
1154	001464	024347	DF17	
1155				
1156				;ERROR 20
1157				
1158	001466	021171	EM20	;DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP
1159	001470	023177	DH17	
1160	001472	024210	DT17	
1161	001474	024347	DF17	
1162				
1163				;ERROR 21
1164				
1165	001476	021254	EM21	;STATUS INCORRECT FOR PORT AFTER CYCLE UP
1166	001500	022511	DH1	
1167	001502	024072	DT1	
1168	001504	024312	DF1	
1169				
1170				;ERROR 22
1171				
1172	001506	021325	EM22	;REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT
1173	001510	022511	DH1	
1174	001512	024072	DT1	
1175	001514	024312	DF1	
1176				
1177				;ERROR 23
1178				
1179	001516	021423	EM23	; 'NED' SET WHEN RMDS1 ACCESSED THROUGH PORT NOT SWITCHED
1180	001520	022511	DH1	
1181	001522	024072	DT1	
1182	001524	024312	DF1	
1183				
1184				;ERROR 24
1185				
1186	001526	021517	EM24	;DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/
1187	001530	023216	DH24	
1188	001532	024216	DT24	
1189	001534	024335	DF7	
1190				
1191				;ERROR 25
1192				
1193	001536	021577	EM25	;RH11 DIDN'T RESPOND TO ADDRESSING
1194	001540	023322	DH25	
1195	001542	024230	DT25	
1196	001544	024351	DF25	
1197				
1198				;ERROR 26
1199				
1200	001546	000000	0	;UNUSED ERROR MESSAGES
1201	001550	000000	0	

1202	001552	000000	0	
1203	001554	000000	0	
1204				
1205				;ERROR 27
1206				
1207	001556	000000	0	;UNUSED ERROR MESSAGES
1208	001560	000000	0	
1209	001562	000000	0	
1210	001564	000000	0	
1211				
1212				;ERROR 30
1213				
1214	001566	021641	EM30	;DRIVE NOT SEIZED BY PORT 'N'
1215	001570	023331	DH30	
1216	001572	024234	DT30	
1217	001574	024352	DF30	
1218				
1219				;ERROR 31
1220				
1221	001576	021672	EM31	;WRONG STATUS SEEN BY THE SEIZING PORT
1222	001600	022562	DH2	
1223	001602	024106	DT2	
1224	001604	024317	DF2	
1225				
1226				;ERROR 32
1227				
1228	001606	021740	EM32	;REGISTER CONTENTS INCORRECT
1229	001610	022562	DH2	
1230	001612	024106	DT2	
1231	001614	024317	DF2	
1232				
1233				;ERROR 33
1234				
1235	001616	021770	EM33	;CONTROL BUS PARITY ERROR WHILE READING REGISTER
1236	001620	022511	DH1	
1237	001622	024072	DT1	
1238	001624	024312	DF1	
1239				
1240				;ERROR 34
1241				
1242	001626	022054	EM34	;CAN'T ACCESS DRIVE THROUGH EITHER PORT
1243	001630	023454	DH34	
1244	001632	024254	DT34	
1245	001634	024361	DF34	
1246				
1247				;ERROR 35
1248				
1249	001636	022123	EM35	;DRIVE NOT IN NEUTRAL AFTER RELEASE, REQUEST NOT SET
1250	001640	023552	DH35	
1251	001642	024266	DT35	
1252	001644	024335	DF7	
1253				
1254				;ERROR 36
1255				
1256	001646	022210	EM36	;DRIVE NOT IN NEUTRAL AFTER TIMEOUT, REQUEST NOT SET
1257	001650	023552	DH35	

```

1258 001652 024266          DI35
1259 001654 024335          DF7
1260
1261          ;ERROR 37
1262
1263 001656 022275          EM37          ;REGISTER CONTENTS INCORRECT AFTER RELEASE/TIMEOUT
1264 001660 023650          DM37
1265 001662 024234          DT30
1266 001664 024352          DF30
1267
1268          ;ERROR 40
1269
1270 001666 022356          EM40          ;DRIVE NOT SEIZED BY PORT AFTER RELEASE WITH REQUEST SET
1271 001670 023773          DM40
1272 001672 024300          DT40
1273 001674 024335          DF7
1274
1275          ;ERROR 41
1276
1277 001676 022433          EM41          ;REGISTER WRONG AFTER RELEASE WITH REQUEST SET
1278 001700 023331          DM30
1279 001702 024234          DT30
1280 001704 024352          DF30
1281
1282
1283
1284
1285          ;*****
1286          .SBTTL  STARTUP AND INITIALIZATION ROUTINES
1287
1288          ;*****
1289
1290
1291 001706 005037 001270  START:  CLR      CHGADR          ;CLEAR THE 'CHANGE RH11 ADDRESS' INDICATOR
1292 001712 000403          BR      START2          ;GO TO THE START
1293 001714 012737 177777 001270  START1: MOV      #-1,CHGADR      ;SET THE 'CHANGE RH11 ADDRESS' INDICATOR
1294 001722 000005          START2: RESET          ;CLEAR THE BUS
1295          .SBTTL  INITIALIZE THE COMMON TAGS
1296          ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
1297 001724 012706 001100  MOV      #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1298 001730 005026          CLR      (R6)+          ;;CLEAR MEMORY LOCATION
1299 001732 022706 001140  CMP      #SWR,R6 ;;DONE?
1300 001736 001374          BNE     .-6             ;;LOOP BACK IF NO
1301 001740 012706 001100  MOV      #STACK_SP      ;;SETUP THE STACK POINTER
1302          ;;INITIALIZE A FEW VECTORS
1303 001744 012737 013440 000020  MOV      #SCOPE,@IOTVEC  ;;IOT VECTOR FOR SCOPE ROUTINE
1304 001752 012737 000340 000022  MOV      #340,@IOTVEC+2 ;;LEVEL 7
1305 001760 012737 013622 000030  MOV      #ERROR,@EMTVEC  ;;EMT VECTOR FOR ERROR ROUTINE
1306 001766 012737 000340 000032  MOV      #340,@EMTVEC+2 ;;LEVEL 7
1307 001774 012737 016422 000034  MOV      #TRAP,@TRAPVEC  ;;TRAP VECTOR FOR TRAP CALLS
1308 002002 012737 000340 000036  MOV      #340,@TRAPVEC+2;LEVEL 7
1309 002010 013737 013052 013044  MOV      #FNDCI,SEOPCT   ;;SETUP END-OF-PROGRAM COUNTER
1310 002016 005037 001176          CLR      $TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
1311 002022 005037 001200          CLR      $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1312 002026 112737 000001 001115  MOV     #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
1313 002034 012737 002034 001106  MOV     #,$SLPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE

```



```

1370 002324 104401 001207          TYPE      $CRLF          ; ANOTHER CR-LF
1371 002330 013737 001224 001230      MOV      PORTA,PORTC ; GENERATE ADDRESS OF DRIVE NOT TESTED
1372 002336 062737 000006 001230      ADD      #6,PORTC    ; COMPLEMENT SOME BITS
1373 002344 042737 177770 001230      BIC      #1C7,PORTC  ; SAVE ONLY LOWER BITS
1374 002352 013701 001224          MOV      PORTA,R1    ; USE PORT A ADDRESS AS INDEX
1375 002356 116137 024406 001232      MOV     ATABIT(R1),ASR1 ; GET ATTENTION BIT FOR DRIVE
1376 002364 005037 001256      CLR     TIMEA        ; CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
1377 002370 005037 001260      CLR     TIMEAP       ; CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
1378 002374 005037 001262      CLR     TIMEB        ; CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
1379 002400 005037 001264      CLR     TIMEBP       ; CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
1380 002404 004737 013226      JSR     PC,CKCLK     ; SETUP CLOCK
1381 002410 000137 002424      JMP     EXEC         ; CLOCK HAS BEEN STARTED
1382 002414 104401 016741          TYPE      ,NOCLOCK   ; NO CLOCK ON SYSTEM
1383 002420 000000          HALT     ; FATAL ERROR
1384 002422 000776          BR      3$          ; INTERLOCK THE HALT
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
    002424 000005          EXEC:   RESET        ; CLEAR EVERYTHING
    002426 005037 177776      CLR     PS           ; CLEAR THE PROCESSOR STATUS WORD
    002432 104401 001207      TYPE      $CRLF      ; CR-LF
    002436 013700 001272      MOV     $RMADR,RO    ; RH11 ADDRESS FOR INDEXING
    002442 012706 001100      MOV     #STACK,SP   ; LOAD STACK POINTER
    002446 004737 013226      JSR     PC,CKCLK     ; START THE CLOCK
    002452 000240      NOP                ; RETURN IF NO CLOCK
    002454 004737 015032      JSR     PC,STKINT    ; INITIALIZE THE KEYBOARD
    002460 005037 001266      CLR     KYBCTL       ; CLEAR SINGLE TEST INDICATOR
    002464 005037 001100      CLR     $PASS        ; CLEAR THE PASS COUNT
    002470 112737 000001 001115      MOV     #1,$ERMAX    ; SET ERROR MAX TO 1
    002476 012737 002476 001106      MOV     #,$SLPADR    ; INITIAL SETTING FOR LOOP ADDRESS
    002504 012737 002504 001110      MOV     #,$SLPERR    ; INITIAL SETTING FOR LOOP ON ERROR ADDRESS
    002512 104401 017010      1$:    TYPE      ,TESTNO ; ASK FOR TEST NUMBER
    002516 104412          RDOCT             ; GET THE NUMBER
    002520 012601      MOV     (SP)+,R1    ; PUT ENTRY INTO R1
    002522 001002      BNE     2$          ; BR IF NOT ZERO
    002524 000137 002720      JMP     TST1         ; ENTER ZERO - PERFORM ALL TESTS
    002530 020137 024416      2$:    CMP     R1,MAXTN   ; SEE IF NUMBER GREATER THAN MAXIMUM
    002534 003403      BLE     3$          ; BR IF LESS OR EQUAL
    002536 104401 017030      TYPE      ,BADNO    ; BAD ENTRY
    002542 000763      BR      1$          ; TRY AGAIN
    002544 005301      3$:    DEC     R1        ; DECREMENT ENTRY
    002546 006301      ASL     R1          ; SHIFT IT LEFT
    002550 016137 024366 002574      MOV     TSTADR(R1),4$ ; GET THE TEST ADDRESS
    002556 005237 001266      INC     KYBCTL       ; SET SINGLE TEST INDICATOR
    002562 012737 000001 001104      MOV     #1,$ICNT    ; PRESET ITERATION COUNT
    002570 000177 000000      JMP     24$         ; GO TO THE SELECTED TEST
    002574 000000      4$:    .WORD     0    ; TEST ADDRESS GOES HERE
1418
1419
1420
1421
1422
1423
1424
1425
    002576 005737 001270      CHANGE: TST     CHGADR ; CHANGE THE ADDRESS ?
    002602 001421      BEQ     3$          ; BR IF NOT
    002604 005037 001270      CLR     CHGADR       ; CLEAR THE INDICATOR
    002610 104401 017056      1$:    TYPE      ,ADDRIS ; TYPE OUT WHAT THE PRESENT ADDRESS IS
    002614 013746 001272      MOV     $RMADR,-(SP) ; PUT THE ADDRESS ON THE STACK
    002620 104402      TYPOC             ; TYPE THE ACTUAL ADDRESS
    
```

```

1426 002622 104401 001207 TYPE ,SCRLF ;CR-LF
1427 002626 104401 017136 TYPE ,NTRH11 ;ASK FOR NEW ADDRESS
1428 002632 104412 RDOCT
1429 002634 005716 TST (SP) ;D OR 'CR' ENTERED ?
1430 002636 001402 BEQ 2$ ;BR IF EITHER ENTERED (NO ADDRESS CHANGE)
1431 002640 011637 001272 MOV (SP), $RMAADR ;NEW RH11 ADDRESS
1432 002644 005726 2$: TST (SP) ;CORRECT THE STACK POINTER
1433 002646 012737 002666 000004 3$: MOV #4$ ,#4 ;LOAD TRAP ADDRESS
1434 002654 013700 001272 MOV $RMAADR, RO ;RH11 ADDRESS
1435 002660 005760 000002 TST RMAC(RO) ;SEE IF RH11 RESPONDS AT THAT ADDRESS
1436 002664 000404 BR 5$ ;BR, RH11 ALIVE AT PRESENT ADDRESS
1437 002666 104025 4$: ERROR 2$ ;NO RESPONSE TO ADDRESS
1438 002670 062706 000004 ADD #4, SP ;RESET THE STACK POINTER
1439 002674 000745 BR 1$ ;GET ADDRESS AGAIN
1440 002676 012737 000006 000004 5$: MOV #6, #4 ;RESTORE THE VECTOR
1441 002704 000207 RTS PC ;RETURN

```

;;*****

.SBTTL *** TESTS ***

;;*****

```

1450 002706 013700 001272 000010 TST1AA: MOV $RMAADR, RO ;RESTORE RO AFTER END OF PASS
1451 002712 012760 000040 000010 MOV #CLR, RACS2(RO) ;CLEAR MASSBUS

```

;;*****

```

1452 *TEST 1 DRIVE ACCESS TEST
1453 *
1454 *VERIFY THAT THE DRIVE CAN BE ACCESSED THROUGH BOTH PORTS
1455 *
1456 * A. SELECT DRIVE, VERIFY THAT THE DRIVE IS PRESENT, THAT THE
1457 * DRIVE IS A DUAL PORT RMD3, THAT THE DRIVE IS ONLINE (RMD31 HAS
1458 * 'MOL', 'PGM', 'DPR', & 'DRY' BITS SET), AND THE THE DRIVE SERIAL
1459 * NUMBER READ THROUGH BOTH PORTS IS THE SAME.
1460 *
1461 * B. THE TEST IS REPEATED THROUGH BOTH PORTS.
1462 *
1463 *
1464 *

```

;;*****

```

1465 †ST1:
1466 002720 TST KYBCTL ;PERFORMING ONLY SINGLE TESTS ?
1467 002720 005737 001266 BEQ 2$ ;BR IF NOT
1468 002724 001406 BPL 1$ ;BR IF JUST ENTERED TEST
1469 002726 100002 JMP EXEC ;RETURN & GET NEXT TEST NUMBER
1470 002730 000137 002424 1$: MOV #-1, KYBCTL ;SET SINGLE TEST INDICATOR
1471 002734 012737 177777 001266 2$: MOV #1, $STNM ;TEST NUMBER
1472 002742 112737 000001 001102 MOV #TEST1, $LPADR ;LOAD LOOP ON TEST ADDRESS
1473 002750 012737 002772 001106 MOV #TEST1, $LPERR ;LOAD LOOP ON ERROR ADDRESS
1474 002756 012737 002772 001110 MOV #1, $TIMES ;DO 1 ITERATION
1475 002764 012737 000001 001176 TEST1: MOV #STACK, SP ;SETUP THE STACK POINTER
1476 002772 012706 001100

```

;;*****

```

1477 ;VERIFY THAT DRIVE IS PRESENT THROUGH PORTS A & B
1478
1479
1480
1481 MOVB PORTA, RACS2(RO) ;SELECT PORT A

```

```

1482 003004 013737 001224 001234 MOV PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1483 003012 005760 000012 TST RMD51(R0) ;SEE IF DRIVE (PORT A) PRESENT
1484 003016 005037 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
1485 003022 016037 000010 001126 MOV RMCS2(R0),SBCDAT ;GET CONTENTS OF RMCS2
1486 003030 012737 000010 001122 MOV #RMCS2,SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1487 003036 060037 001122 ADD R0,SBOADR ;ADD RH11 BASE ADDRESS
1488 003042 005037 001124 CLR $GDDAT ;WHAT REGISTER SHOULD BE
1489 003046 013737 001126 001164 MOV SBODAT,$TMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
1490 003054 042737 167777 001164 BIC #1CNED,$TMP0 ;SAVE SPECIFIED BITS
1491 003052 023737 001124 001164 CMP $GDDAT,$TMP0 ;COMPARE THE BITS
1492 003070 001414 BEQ 64$ ;BR IF OK
1493 003072 013737 001126 001174 MOV SBODAT,$TMP4 ;COPY 'BAD DATA'
1494 003100 042737 010000 001174 BIC #NED,$TMP4 ;CLEAR THE MASKED BITS
1495 003106 053737 001174 001124 BIS $TMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
1496 003114 104001 ERROR 1 ;TYPE MESSAGE 1
1497 003116 005137 001244 COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
1498 003122 000240 64$: NOP
1499 003124 005737 001244 TST CKERR ;WAS 'NED' SET ?
1500 003130 001403 BEQ .+10 ;BR IF NOT
1501 003132 012760 000040 000010 MOV #CLR,RMCS2(R0) ;ISSUE MASSBUS INIT TO CLEAR 'NED'
1502 003140 113760 001226 000010 MOV#B PORTB,RMCS2(R0) ;SELECT PORT B
1503 003146 013737 001226 001234 MOV PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1504 003154 005760 000012 TST RMD51(R0) ;SEE IF DRIVE (PORT B) PRESENT
1505 003160 005037 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
1506 003164 016037 000010 001126 MOV RMCS2(R0),SBODAT ;GET CONTENTS OF RMCS2
1507 003172 012737 000010 001122 MOV #RMCS2,SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1508 003200 060037 001122 ADD R0,SBOADR ;ADD RH11 BASE ADDRESS
1509 003204 005037 001124 CLR $GDDAT ;WHAT REGISTER SHOULD BE
1510 003210 013737 001126 001164 MOV SBODAT,$TMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
1511 003216 042737 167777 001164 BIC #1CNED,$TMP0 ;SAVE SPECIFIED BITS
1512 003224 023737 001124 001164 CMP $GDDAT,$TMP0 ;COMPARE THE BITS
1513 003232 001414 BEQ 66$ ;BR IF OK
1514 003234 013737 001126 001174 MOV SBODAT,$TMP4 ;COPY 'BAD DATA'
1515 003242 042737 010000 001174 BIC #NED,$TMP4 ;CLEAR THE MASKED BITS
1516 003250 053737 001174 001124 BIS $TMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
1517 003256 104001 ERROR 1 ;TYPE MESSAGE 1
1518 003260 005137 001244 COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
1519 003264 000240 66$: NOP
1520 003266 005737 001244 TST CKERR ;WAS 'NED' SET ?
1521 003272 001403 BEQ .+10 ;BR IF NOT
1522 003274 012760 000040 000010 MOV #CLR,RMCS2(R0) ;ISSUE MASSBUS INIT TO CLEAR 'NED'
1523
1524 ;*****
1525 ;CONFIRM THAT DRIVE IS AN RMD3 AND IS DUAL PORT
1526
1527 003302 113760 001224 000010 MOV#B PORTA,RMCS2(R0) ;SELECT PORT A
1528 003310 013737 001224 001234 MOV PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1529 003316 005037 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
1530 003322 016037 000026 001126 MOV RMDT(R0),SBODAT ;GET CONTENTS OF RMDT
1531 003330 012737 000026 001122 MOV #RMDT,SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1532 003336 060037 001122 ADD R0,SBOADR ;ADD RH11 BASE ADDRESS
1533 003342 012737 024024 001124 MOV #024024,$GDDAT ;WHAT REGISTER SHOULD BE
1534 003350 013737 001126 001164 MOV SBODAT,$TMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
1535 003356 042737 000003 001164 BIC #1C177774,$TMP0 ;SAVE SPECIFIED BITS
1536 003364 023737 001124 001164 CMP $GDDAT,$TMP0 ;COMPARE THE BITS
1537 003372 001414 BEQ 68$ ;BR IF OK

```

E03

```

1538 003374 013737 001126 001174      MOV      $BDDAT,$TMP4      ;COPY 'BAD DATA'
1539 003402 042737 177774 001174      BIC      #177774,$TMP4     ;CLEAR THE MASKED BITS
1540 003410 053737 001174 001124      BIS      $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
1541 003416 104002                ERROR    2                ;TYPE MESSAGE 2
1542 003420 005137 001244                COM      CKERR            ;SET THE REGISTER COMPARE ERROR INDICATOR
1543 003424 000240                NOP
1544 003426 113760 001226 000010      MOVB     PORTB,RMCS2(RO)   ;SELECT PORT B
1545 003434 013737 001226 001234      MOV      PORTB,PTNBR      ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1546 003442 005037 001244                CLR      CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
1547 003446 016037 000026 001126      MOV      RMDT(RO),$BDDAT  ;GET CONTENTS OF RMDT
1548 003454 012737 000026 001122      MOV      #RMDT,$BADR      ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1549 003462 060037 001122                ADD      RO,$BADR         ;ADD RHI1 BASE ADDRESS
1550 003466 012737 024024 001124      MOV      #024024,$GDDAT  ;WHAT REGISTER SHOULD BE
1551 003474 013737 001126 001164      MOV      $GDDAT,$TMP0     ;MOVE REGISTER CONTENTS TO 'TMP0'
1552 003502 042737 000003 001164      BIC      #1C177774,$TMP0  ;SAVE SPECIFIED BITS
1553 003510 023737 001124 001164      CMP      $GDDAT,$TMP0     ;COMPARE THE BITS
1554 003516 001414                BEQ      70$              ;BR IF OK
1555 003520 013737 001126 001174      MOV      $BDDAT,$TMP4     ;COPY 'BAD DATA'
1556 003526 042737 177774 001174      BIC      #177774,$TMP4     ;CLEAR THE MASKED BITS
1557 003534 053737 001174 001124      BIS      $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
1558 003542 104002                ERROR    2                ;TYPE MESSAGE 2
1559 003544 005137 001244                COM      CKERR            ;SET THE REGISTER COMPARE ERROR INDICATOR
1560 003550 000240                NOP
    
```

68\$:

70\$:

 ;VERIFY THROUGH BOTH PORTS THAT THE DRIVE IS ON LINE AND IN NEUTRAL

```

1561
1562
1563
1564
1565 003552 113760 001224 000010      MOVB     PORTA,RMCS2(RO)   ;SELECT PORT A
1566 003560 013737 001224 001234      MOV      PORTA,PTNBR      ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1567 003566 005037 001244                CLR      CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
1568 003572 016037 000012 001126      MOV      RMD1(RO),$BDDAT  ;GET CONTENTS OF RMD1
1569 003600 012737 000012 001122      MOV      #RMD1,$BADR      ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1570 003606 060037 001122                ADD      RO,$BADR         ;ADD RHI1 BASE ADDRESS
1571 003612 012737 001000 001124      MOV      #PGM,$GDDAT     ;WHAT REGISTER SHOULD BE
1572 003620 013737 001126 001164      MOV      $BDDAT,$TMP0     ;MOVE REGISTER CONTENTS TO 'TMP0'
1573 003626 042737 176777 001164      BIC      #1CPGM,$TMP0     ;SAVE SPECIFIED BITS
1574 003634 023737 001124 001164      CMP      $GDDAT,$TMP0     ;COMPARE THE BITS
1575 003642 001414                BEQ      72$              ;BR IF OK
1576 003644 013737 001126 001174      MOV      $BDDAT,$TMP4     ;COPY 'BAD DATA'
1577 003652 042737 001000 001174      BIC      #PGM,$TMP4       ;CLEAR THE MASKED BITS
1578 003660 053737 001174 001124      BIS      $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
1579 003666 104003                ERROR    3                ;TYPE MESSAGE 3
1580 003670 005137 001244                COM      CKERR            ;SET THE REGISTER COMPARE ERROR INDICATOR
1581 003674 000240                NOP
1582 003676 005037 001244                CLR      CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
1583 003702 016037 000012 001126      MOV      RMD1(RO),$BDDAT  ;GET CONTENTS OF RMD1
1584 003710 012737 000012 001122      MOV      #RMD1,$BADR      ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1585 003716 060037 001122                ADD      RO,$BADR         ;ADD RHI1 BASE ADDRESS
1586 003722 012737 010600 001124      MOV      #MOL!DPR!DRY,$GDDAT ;WHAT REGISTER SHOULD BE
1587 003730 013737 001126 001164      MOV      $BDDAT,$TMP0     ;MOVE REGISTER CONTENTS TO 'TMP0'
1588 003736 042737 167177 001164      BIC      #1C10600,$TMP0   ;SAVE SPECIFIED BITS
1589 003744 023737 001124 001164      CMP      $GDDAT,$TMP0     ;COMPARE THE BITS
1590 003752 001414                BEQ      74$              ;BR IF OK
1591 003754 013737 001126 001174      MOV      $BDDAT,$TMP4     ;COPY 'BAD DATA'
1592 003762 042737 010600 001174      BIC      #10600,$TMP4     ;CLEAR THE MASKED BITS
1593 003770 053737 001174 001124      BIS      $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
    
```

72\$:

74\$:

F03

MD-11-DZRMH-A, RMD3 DUAL CONTRLR LOGIC TEST - PART 2
DZRMH.A.P11 01-AUG-77 11:02

T1 DRIVE ACCESS TEST

MACY11 30(1046) 01-AUG-77 11:14 PAGE 31

SEQ 0032

```

1594 003776 104004          ERROR 4          ;TYPE MESSAGE 4
1595 004000 005137 001244    COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
1596 004004 000240          NOP
1597 004006 113760 001226 000010 74$:  MOVB  PORTB,RMCS2(RO) ;SELECT PORT B
1598 004014 013737 001226 001234    MOV  PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1599 004022 005037 001244    CLR  CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
1600 004026 016037 000012 001126    MOV  RMD51(RO),SBDDAT ;GET CONTENTS OF RMD51
1601 004034 012737 000012 001122    MOV  #RMD51,SBADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1602 004042 060037 001122    ADD  RO,SBADR      ;ADD RHI1 BASE ADDRESS
1603 004046 012737 001000 001124    MOV  #PGM,$GDDAT   ;WHAT REGISTER SHOULD BE
1604 004054 013737 001126 001164    MOV  SBDDAT,$TMP0  ;MOVE REGISTER CONTENTS TO 'STMP0'
1605 004062 042737 176777 001164    BIC  #1CPGM,$TMP0  ;SAVE SPECIFIED BITS
1606 004070 023737 001124 001164    CMP  $GDDAT,$TMP0 ;COMPARE THE BITS
1607 004076 001414          BEQ  76$         ;BR IF OK
1608 004100 013737 001126 001174    MOV  SBDDAT,$TMP4  ;COPY 'BAD DATA'
1609 004106 042737 001000 001174    BIC  #PGM,$TMP4    ;CLEAR THE MASKED BITS
1610 004114 053737 001174 001124    BIS  $TMP4,$GDDAT  ;'OR' WITH GOOD DATA FOR TYPEOUT
1611 004122 104003          ERROR 3          ;TYPE MESSAGE 3
1612 004124 005137 001244    COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
1613 004130 000240          NOP
1614 004132 005037 001244    CLR  CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
1615 004136 016037 000012 001126    MOV  RMD51(RO),SBDDAT ;GET CONTENTS OF RMD51
1616 004144 012737 000012 001122    MOV  #RMD51,SBADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
1617 004152 060037 001122    ADD  RO,SBADR      ;ADD RHI1 BASE ADDRESS
1618 004156 012737 010600 001124    MOV  #MOL!DPR!DRY,$GDDAT ;WHAT REGISTER SHOULD BE
1619 004164 013737 001126 001164    MOV  SBDDAT,$TMP0  ;MOVE REGISTER CONTENTS TO 'STMP0'
1620 004172 042737 167177 001164    BIC  #1C10600,$TMP0 ;SAVE SPECIFIED BITS
1621 004200 023737 001124 001164    CMP  $GDDAT,$TMP0 ;COMPARE THE BITS
1622 004206 001414          BEQ  78$         ;BR IF OK
1623 004210 013737 001126 001174    MOV  SBDDAT,$TMP4  ;COPY 'BAD DATA'
1624 004216 042737 010600 001174    BIC  #10600,$TMP4  ;CLEAR THE MASKED BITS
1625 004224 053737 001174 001124    BIS  $TMP4,$GDDAT  ;'OR' WITH GOOD DATA FOR TYPEOUT
1626 004232 104004          ERROR 4          ;TYPE MESSAGE 4
1627 004234 005137 001244    COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
1628 004240 000240          NOP
1629
1630
1631 ;*****
1632 ;VERIFY THAT DRIVE SERIAL NUMBER SEEN THROUGH BOTH PORTS IS THE SAME
1633 004242 113760 001224 000010    MOVB  PORTA,RMCS2(RO) ;SELECT PORT A
1634 004250 016037 000030 001124    MOV  RMSN(RO),$GDDAT ;STORE THE PORT A SERIAL NUMBER
1635 004256 113760 001226 000010    MOVB  PORTB,RMCS2(RO) ;SELECT PORT B
1636 004264 016037 000030 001126    MOV  RMSN(RO),SBDDAT ;STORE THE PORT B SERIAL NUMBER
1637 004272 023737 001124 001126    CMP  $GDDAT,SBDDAT ;ARE THEY THE SAME ?
1638 004300 001406          BEQ  1$         ;BR IF THEY ARE
1639 004302 104005          ERROR 5          ;REPORT THE ERROR
1640 004304 032777 100000 174626    BIT  #SW15,$SWR    ;HALT ON ERROR ?
1641 004312 001001          BNE  1$         ;BR IF SET - PROGRAM HAS ALREADY HALTED
1642 004314 000000          HALT
1643 004316 000004          1$: SCOPE ;HALT, POSSIBLE CABLE CONNECTION PROBLEM
1644
1645
1646
1647 ;*****
1648 ;*TEST 2          SET 'VV' FOR PORT A
1649 ;*

```


1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662 004320
1663 004320 005737 001266
1664 004324 001406
1665 004326 100002
1666 004330 000137 002424
1667 004334 012737 177777 001266
1668 004342 112737 000002 001102
1669 004350 012737 004372 001106
1670 004356 012737 004372 001110
1671 004364 012737 000001 001176
1672 004372 012706 001100
1673 004376 113760 001224 000010
1674 004404 013737 001224 001234
1675
1676
1677
1678
1679 004412 012760 000011 000000
1680 004420 012760 000021 000000
1681 004426 012760 010000 000032
1682
1683
1684
1685
1686 004434 005037 001244
1687 004440 016037 000012 001126
1688 004446 012737 000012 001122
1689 004454 060037 001122
1690 004460 012737 011700 001124
1691 004466 013737 001126 001164
1692 004474 042737 106077 001164
1693 004502 023737 001124 001164
1694 004510 001414
1695 004512 013737 001126 001174
1696 004520 042737 071700 001174
1697 004526 053737 001174 001124
1698 004534 104010
1699 004536 005137 001244
1700 004542 000240
1701
1702
1703
1704
1705

```
;*SET VOLUME VALID
;*
;* A. ISSUE A DRIVE CLEAR COMMAND THROUGH PORT A.
;*
;* B. ISSUE A READIN PRESET COMMAND THROUGH PORT A. VERIFY
;* THAT THE 'VV' BIT IS SET FOR PORT A.
;*
;* C. ISSUE A RELEASE COMMAND THROUGH PORT A. VERIFY THAT
;* THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION
;* BIT IS SET.
*****
†ST2:
TST KYBCTL ;PERFORMING ONLY SINGLE TESTS ?
BEQ 25 ;BR IF NOT
BPL 15 ;BR IF JUST ENTERED TEST
JMP EXEC ;RETURN & GET NEXT TEST NUMBER
15: MOV #1,KYBCTL ;SET SINGLE TEST INDICATOR
25: MOVB #2,$STNM ;TEST NUMBER
MOV #TEST2,$LPADR ;LOAD LOOP ON TEST ADDRESS
MOV #TEST2,$LPERR ;LOAD LOOP ON ERROR ADDRESS
MOV #1,$TIMES ;DO 1 ITERATION
TEST2: MOV #STACK,$SP ;SETUP THE STACK POINTER
MOVB PORTA,$RMC52($R0) ;SELECT PORT A
MOV PORTA,$PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
*****
;SET VOLUME VALUE FOR PORT
MOV #11,$RMC51($R0) ;ISSUE A DRIVE CLEAR
MOV #21,$RMC51($R0) ;ISSUE A READIN PRESET
MOV #FMT16,$RMOF($R0) ;SET FMT16
*****
;VERIFY THAT THE DRIVE STATUS IS CORRECT
CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
MOV $RMC51($R0),$SBODAT ;GET CONTENTS OF $RMC51
MOV #RMC51,$SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
ADD $R0,$SBOADR ;ADD RHI1 BASE ADDRESS
MOV #MOL!PGM!DPR!DRY!VV,$GDOAT ;WHAT REGISTER SHOULD BE
MOV $SBODAT,$STMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
BIC #C71700,$STMP0 ;SAVE SPECIFIED BITS
CMP $GDOAT,$STMP0 ;COMPARE THE BITS
BEQ 645 ;BR IF OK
MOV $SBODAT,$STMP4 ;COPY 'BAD DATA'
BIC #71700,$STMP4 ;CLEAR THE MASKED BITS
BIS $STMP4,$GDOAT ;'OR' WITH GOOD DATA FOR TYPEOUT
ERROR 10 ;TYPE MESSAGE 10
COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
645: NOP
*****
;RELEASE THE DRIVE FROM PORT A
```

H03

```

1706 004544 113760 001224 000010      MOVB   PORTA,RMCS2(RO) ;SELECT PORT A
1707 004552 013737 001224 001234      MOV    PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1708 004560 012760 000013 000000      MOV    #13,RMCS1(RO) ;ISSUE RELEASE THROUGH PORT A
1709
1710                                     ;VERIFY THAT THE DRIVE IS IN NEUTRAL
1711
1712 004566 005037 001250      CLR    RELERR ;CLEAR THE 'RELEASE ERROR' INDICATOR
1713 004572 012737 000012 001122      MOV    #RMS1,SBDADR ;FORM THE ADDRESS OF RMS1 FOR TYPEOUT
1714 004600 060037 001122      ADD    RO,SBDADR ;ADD THE I/O BASE ADDRESS
1715 004604 012737 011600 001124      MOV    #MOL!PGH!DPR!DRY,SGDAT ;COMPARISON CONSTANT
1716 004612 113760 001224 000010      MOVB   PORTA,RMCS2(RO) ;SELECT PORT A.
1717 004620 016037 000012 001170      MOV    RMS1(RO),STMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
1718 004626 013737 001170 001164      MOV    STMP2,STMP0 ;COPY IT INTO 'STMP0'
1719 004634 042737 100100 001164      BIC    #ATA!VV,STMP0 ;CLEAR PORT DEPENDENT BITS FROM THE COPY
1720 004642 113760 001226 000010      MOVB   PORTB,RMCS2(RO) ;SELECT PORT B.
1721 004650 016037 000012 001172      MOV    RMS1(RO),STMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
1722 004656 013737 001172 001166      MOV    STMP3,STMP1 ;COPY IT INTO 'STMP1'
1723 004664 042737 100100 001166      BIC    #ATA!VV,STMP1 ;CLEAR PORT DEPENDENT BITS FROM THE COPY
1724 004672 023737 001164 001166      CMP    STMP0,STMP1 ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
1725 004700 001006      BNE    66$ ;BR IF NOT
1726 004702 005737 001164      TST    STMP0 ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
1727 004706 001037      BNE    68$ ;BR IF NOT
1728 004710 104034      ERROR  34 ;REPORT DRIVE NOT IN NEUTRAL OR NOT SEIZED
1729 004712 000137 005076      JMP    70$ ;BYPASS THE REST OF THE CHECKS
1730 004716 013737 001170 001126 66$:      MOV    STMP2,SBDAT ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
1731 004724 013737 001226 001234      MOV    PORTB,PTNBR ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
1732 004732 113760 001226 000010      MOVB   PORTB,RMCS2(RO) ;SELECT PORT B.
1733 004740 005737 001164      TST    STMP0 ;SEE IF STATUS EQ 0 FROM PORT A.
1734 004744 001414      BEQ    67$ ;BR IF ZERO
1735 004746 013737 001224 001234      MOV    PORTA,PTNBR ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
1736 004754 013737 001172 001126      MOV    STMP3,SBDAT ;'BAD DATA' FOR ERROR TYPE OUT
1737 004762 113760 001224 000010      MOVB   PORTA,RMCS2(RO) ;SELECT PORT A.
1738 004770 005737 001166      TST    STMP1 ;SEE IF STATUS EQ ZERO FROM PORT B.
1739 004774 001004      BNE    68$ ;BR IF NOT
1740 004776 012737 177777 001250 67$:      MOV    #-1,RELERR ;SET 'RELEASE ERROR' INDICATOR
1741 005004 104036      ERROR  36 ;TYPE ERROR MESSAGE 36
1742 005006 013737 001170 001126 68$:      MOV    STMP2,SBDAT ;LOOK FOR BIT FAILURES WHEN RMS1 READ
1743 005014 013737 001224 001234      MOV    PORTA,PTNBR ;CHANGE PORT NUMBER
1744 005022 042737 100100 001170      BIC    #ATA!VV,STMP2 ;DON'T CHECK ATTN BIT OR VV BIT
1745 005030 023737 001124 001170      CMP    SGDAT,STMP2 ;ALL BITS OK ?
1746 005036 001401      BEQ    69$ ;BR IF OK FROM PORT A.
1747 005040 104037      ERROR  37 ;REPORT ERROR
1748 005042 013737 001172 001126 69$:      MOV    STMP3,SBDAT ;CHECK RMS1 FOR BIT FAILURES - FROM PORT B.
1749 005050 013737 001226 001234      MOV    PORTB,PTNBR ;CHANGE PORT NUMBER
1750 005056 042737 100100 001172      BIC    #ATA!VV,STMP3 ;DON'T CHECK ATTN BIT OR VV BIT
1751 005064 023737 001124 001172      CMP    SGDAT,STMP3 ;SEE IF READ OK FROM PORT B.
1752 005072 001401      BEQ    70$ ;BR IF OK
1753 005074 104037      ERROR  37 ;REPORT THE ERROR
1754 005076 000240      NOP
1755 005100 000004      SCOPE ;LOOP ?
1756
1757 ;*****
1758 ;*TEST 3 SET 'VV' FOR PORT B
1759 ;*
1760 ;*SET VOLUME VALID
1761 ;*
  
```

T3 SET 'VV' FOR PORT B

- * A. ISSUE A DRIVE CLEAR COMMAND THROUGH PORT B.
- * B. ISSUE A READIN PRESET COMMAND THROUGH PORT B. VERIFY THAT THE 'VV' BIT IS SET FOR PORT B.
- * C. ISSUE A RELEASE COMMAND THROUGH PORT B. VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION BIT IS SET.

TEST3:

```

TST KYBCTL ;PERFORMING ONLY SINGLE TESTS ?
BEQ 25 ;BR IF NOT
BPL 15 ;BR IF JUST ENTERED TEST
JMP EXEC ;RETURN & GET NEXT TEST NUMBER
MOV #1, KYBCTL ;SET SINGLE TEST INDICATOR
MOV #3, $STNM ;TEST NUMBER
MOV #TEST3, $LPADR ;LOAD LOOP ON TEST ADDRESS
MOV #TEST3, $LPERR ;LOAD LOOP ON ERROR ADDRESS
MOV #1, $STAES ;DO 1 ITERATION
TEST3: MOV #STACK, SP ;SETUP THE STACK POINTER
MOV #PORTB, RMCS2(RO) ;SELECT PORT B
MOV #PORTB, PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

```

;SET VOLUME VALUE FOR PORT

```

MOV #11, RMCS1(RO) ;ISSUE A DRIVE CLEAR
MOV #21, RMCS1(RO) ;ISSUE A READIN PRESET
MOV #FMT16, RMCF(RO) ;SET FMT16

```

;VERIFY THAT THE DRIVE STATUS IS CORRECT

```

CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
MOV #RMS1(RO), $PDDAT ;GET CONTENTS OF RMS1
MOV #RMS1, $B0ADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
ADD #R0, $B0ADR ;ADD FULL BASE ADDRESS
MOV #MOL!PGM!DPR!DRY!VV, $GDDAT ;WHAT REGISTER SHOULD BE
MOV $B0DAT, $TMP0 ;MOVE REGISTER CONTENTS TO 'TMP0'
BIC #1C71700, $TMP0 ;SAVE SPECIFIED BITS
CMP $GDDAT, $TMP0 ;COMPARE THE BITS
BEQ 645 ;BR IF OK
MOV $B0DAT, $TMP4 ;COPY 'BAD DATA'
BIC #71700, $TMP4 ;CLEAR THE MASKED BITS
BIS $TMP4, $GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
ERROR 10 ;TYPE MESSAGE 10
COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
645: NOP

```

;RELEASE THE DRIVE FROM PORT B

```

MOV #PORTB, RMCS2(RO) ;SELECT PORT B
MOV #PORTB, PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

```

1762				
1763				
1764				
1765				
1766				
1767				
1768				
1769				
1770				
1771				
1772	005102			
1773	005102	005737	001266	
1774	005106	001406		
1775	005110	100002		
1776	005112	000137	002424	
1777	005116	012737	177777	001266
1778	005124	112737	000003	001102
1779	005132	012737	005154	001106
1780	005140	012737	005154	001110
1781	005146	012737	000001	001176
1782	005154	012706	001100	
1783	005160	113760	001226	000010
1784	005166	013737	001226	001234
1785				
1786				
1787				
1788				
1789	005174	012760	000011	000000
1790	005202	012760	000021	000000
1791	005210	012760	010000	000032
1792				
1793				
1794				
1795				
1796	005216	005037	001244	
1797	005222	016037	000012	001126
1798	005230	012737	000012	001122
1799	005236	060037	001122	
1800	005242	012737	011700	001124
1801	005250	013737	001126	001164
1802	005256	042737	106077	001164
1803	005264	023737	001124	001164
1804	005272	001414		
1805	005274	013737	001126	001174
1806	005302	042737	071700	001174
1807	005310	053737	001174	001124
1808	005316	104010		
1809	005320	005137	001244	
1810	005324	000240		
1811				
1812				
1813				
1814				
1815				
1816	005326	113760	001226	000010
1817	005334	013737	001226	001234

```

1818 005342 012760 000013 000000      MOV      #13,RMCS1(RO) ;ISSUE RELEASE THROUGH PORT B
1819
1820                                     ;VERIFY THAT THE DRIVE IS IN NEUTRAL
1821
1822 005350 005037 001250      CLR      RELERR        ;CLEAR THE 'RELEASE ERROR' INDICATOR
1823 005354 012737 000012 001122      MOV      #RMS1,$BODAT ;FORM THE ADDRESS OF RMS1 FOR TYPEOUT
1824 005352 060037 001122      ADD     RO,$BODADR    ;ADD THE I/O BASE ADDRESS
1825 005356 012737 011600 001124      MOV      #MOL!PGH!DPR!DRY,$GDAT ;COMPARISON CONSTANT
1826 005374 113760 001224 000010      MOVVB   PORTA,RMCS2(RO) ;SELECT PORT A.
1827 005402 016037 000012 001170      MOV      RMS1(RO),$TMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
1828 005410 013737 001170 001164      MOV      $TMP2,$TMP0   ;COPY IT INTO '$TMP0'
1829 005416 042737 100100 001164      BIC     #ATA!VV,$TMP0  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
1830 005424 113760 001226 000010      MOVVB   PORTB,RMCS2(RO) ;SELECT PORT B.
1831 005432 016037 000012 001172      MOV      RMS1(RO),$TMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
1832 005440 013737 001172 001166      MOV      $TMP3,$TMP1   ;COPY IT INTO '$TMP1'
1833 005446 042737 100100 001166      BIC     #ATA!VV,$TMP1  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
1834 005454 023737 001164 001166      CMP     $TMP0,$TMP1    ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
1835 005462 001006      BNE     66$            ;BR IF NOT
1836 005464 005737 001164      TST     $TMP0          ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
1837 005470 001037      BNE     68$            ;BR IF NOT
1838 005472 104034      ERROR   34            ;REPORT DRIVE NOT IN NEUTRAL OR NOT SEIZED
1839 005474 000137 005660      JMP     70$            ;BYPASS THE REST OF THE CHECKS
1840 005500 013737 001170 001126 66$:      MOV     $TMP2,$BODAT   ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
1841 005506 013737 001226 001234      MOV     PORTB,PTNBR    ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
1842 005514 113760 001226 000010      MOVVB   PORTB,RMCS2(RO) ;SELECT PORT B.
1843 005522 005737 001164      TST     $TMP0          ;SEE IF STATUS EQ 0 FROM PORT A.
1844 005526 001414      BEQ     67$            ;BR IF ZERO
1845 005530 013737 001224 001234      MOV     PORTA,PTNBR    ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
1846 005536 013737 001172 001126      MOV     $TMP3,$BODAT   ;'BAD DATA' FOR ERROR TYPE OUT
1847 005544 113760 001224 000010      MOVVB   PORTA,RMCS2(RO) ;SELECT PORT A.
1848 005552 005737 001166      TST     $TMP1          ;SEE IF STATUS EQ ZERO FROM PORT B.
1849 005556 001004      BNE     68$            ;BR IF NOT
1850 005560 012737 177777 001250 67$:      MOV     #-1,RELERR     ;SET 'RELEASE ERROR' INDICATOR
1851 005566 104036      ERROR   36            ;TYPE ERROR MESSAGE 36
1852 005570 013737 001170 001126 68$:      MOV     $TMP2,$BODAT   ;LOOK FOR BIT FAILURES WHEN RMS1 READ
1853 005576 013737 001224 001234      MOV     PORTA,PTNBR    ;CHANGE PORT NUMBER
1854 005584 042737 100100 001170      BIC     #ATA!VV,$TMP2  ;DON'T CHECK ATTN BIT OR VV BIT
1855 005592 023737 001124 001170      CMP     $GDAT,$TMP2    ;ALL BITS OK ?
1856 005596 001401      BEQ     69$            ;BR IF OK FROM PORT A.
1857 005600 104037      ERROR   37            ;REPORT ERROR
1858 005604 013737 001172 001126 69$:      MOV     $TMP3,$BODAT   ;CHECK RMS1 FOR BIT FAILURES - FROM PORT B.
1859 005608 013737 001226 001234      MOV     PORTB,PTNBR    ;CHANGE PORT NUMBER
1860 005612 042737 100100 001172      BIC     #ATA!VV,$TMP3  ;DON'T CHECK ATTN BIT OR VV BIT
1861 005616 023737 001124 001172      CMP     $GDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
1862 005620 001401      BEQ     70$            ;BR IF OK
1863 005624 104037      ERROR   37            ;REPORT THE ERROR
1864 005628 000240 70$:      NOP
1865 005632 000004      SCOPE
1866
1867
1868
1869
1870
1871                                     ;*****
1872                                     ;*TEST 4      MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A
1873                                     ;*
1874                                     ;*MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT A

```

```

1874
1875
1876
1877
1878
1879
1880
1881
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

```

```

;*
;* A. WRITE 0'S INTO RMD51 THROUGH PORT A AND VERIFY THAT THE
;* DRIVE HAS BEEN SEIZED.
;*
;* B. WAIT FOR TIMEOUT TO OCCUR. MEASURE THE DURATION OF THE TIMEOUT
;* ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
;*
;* C. VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS
;* TO NEUTRAL
;*
;*****
TST4:
TST KYBCTL ;PERFORMING ONLY SINGLE TESTS ?
BEQ 25 ;BR IF NOT
BPL 15 ;BR IF JUST ENTERED TEST
JMP EXEC ;RETURN & GET NEXT TEST NUMBER
15: MOV #1,KYBCTL ;SET SINGLE TEST INDICATOR
25: MOVB #4,$TSTNM ;TEST NUMBER
MOV #TEST4,$LPADR ;LOAD LOOP ON TEST ADDRESS
MOV #TEST4,$LPERR ;LOAD LOOP ON ERROR ADDRESS
MOV #1,$TIMES ;DO 1 ITERATION
TEST4: MOV #STACK,$SP ;SETUP THE STACK POINTER
CLR TIMEA ;CLEAR THE TIMEOUT VALUE STORAGE LOCATION
CLR TIMEAP ;CLEAR THE + 25% TOLERANCE LOCATION
;*****
;START THE TIMER
CLR TIME ;CLEAR THE ELAPSED TIME COUNTER
MOV #2000.,WATCH ;SET WATCH TO 2000 MS
;*****
;SEIZE THE DRIVE THROUGH PORT A
MOVB PORTA,RMCS2(RO) ;SELECT PORT A
MOV PORTA,SEIZPT ;STORE SEIZING PORT'S ADDRESS
CLR RMD51(RO) ;WRITE RMD51
MOVB PORTB,RMCS2(RO) ;SELECT PORT B
MOV PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
MOV PORTB,OPPRT ;'OPPOSITE' PORT ADDRESS
MOV RMD51(RO),$BDDAT ;SEE IF DRIVE SEIZED BY PORT A
MOV RO,$BDAOR ;R11 BASE ADDRESS
ADD #RMD51,$BDAOR ;GENERATE BAD REGISTER ADDRESS
CLR $GDDAT ;REGISTER SHOULD BE ZERO
CMP $GDDAT,$BDDAT ;IS THE REGISTER ZERO
BEQ 645 ;BR IF IT IS
ERROR 30 ;REPORT THE ERROR
JMP 45 ;BYPASS REST OF THE SUBTEST
645: MOVB PORTA,RMCS2(RO) ;SELECT PORT A
MOV PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
MOV RMD51(RO),$BDDAT ;SEE IF SEIZING PORT SEES CORRECT STATUS
MOV #MOL!PGM!DPR!DRY!VV,$GDDAT ;EXPECTED STATUS
MOV $GDDAT,$TMP1 ;USE GOOD DATA AS A MASK
COM $TMP1 ;COMPLEMENT THE EXPECTED STATUS

```

L03

MD-11-DZRMH-A, RM03 DUAL CONTRLR LOGIC TEST - PART 2
DZRMH.A.P11 01-AUG-77 11:02

MACY11 30(1046) 01-AUG-77 11:14 PAGE 37
T4 MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A

SEQ 0038

```

1930 006132 013737 001126 001164      MOV      $BDDAT,$TMP0      ;SAVE THE ACTUAL STATUS
1931 006140 043737 001166 001164      BIC      $TMP1,$TMP0      ;CLEAR UNWANTED BITS
1932 006146 023737 001124 001164      CMP      $GDDAT,$TMP0     ;ARE THE EXPECTED STATUS BITS SET ?
1933 006154 001401                BEQ      65$              ;BR IF THEY ARE
1934 006156 104031                ERROR   31              ;REPORT THE ERROR
1935 006160 000240      65$:      NOP
1936
1937
1938      ;*****
1939      ;WAIT FOR PORT A TO TIMEOUT
1940 006162 113760 001226 000010      MOV      PORTB,RMCS2(RO)  ;SELECT PORT B
1941 006170 013737 001226 001234      MOV      PORTB,PINBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
1942 006176 005760 000012      1$:      TST      RMDS1(RO)      ;WAIT FOR THE DRIVE TO TIMEOUT
1943 006202 001006                BNE      2$              ;BR WHEN TIMEOUT OCCURS
1944 006204 005737 001254      TST      WATCH           ;CHECK WATCH
1945 006210 001372                BNE      1$              ;BR IF NOT ZERO
1946 006212 104006                ERROR   6              ;NO TIMEOUT WITHIN 2 SECONDS
1947 006214 000137 006252      JMP      3$              ;BYPASS THE REST OF THE TEST
1948 006220 013737 001252 001256      2$:      MOV      TIME,TIMEA     ;SAVE THE ELAPSED TIME FOR PORT A
1949 006226 004537 013412      JSR      RS,TOLER        ;CALCULATE THE TOLERANCE
1950 006232 001256                .WORD   TIMEA           ;TIMEOUT VALUE FOR PORT A
1951 006234 012637 001260      MOV      (SP)+,TIMEAP    ;+25% TOLERANCE
1952
1953      ;*****
1954      ;VERIFY THAT THE TIMEOUT ONE-SHOT VALUE IS AT LEAST 500 MS
1955
1956 006240 023727 001256 000764      CMP      TIMEA,#500.     ;IS TIMEOUT VALUE AT LEAST 500 MS ?
1957 006246 103001                BHS     3$              ;BR IF IT IS
1958 006250 104007                ERROR   7              ;TIMEOUT LESS THAN 500 MS
1959
1960      ;*****
1961      ;VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AFTER PORT A TIMED OUT
1962
1963 006252      3$:
1964
1965      ;VERIFY THAT THE DRIVE IS IN NEUTRAL
1966
1967 006252 005037 001250      CLR      RELERR          ;CLEAR THE 'RELEASE ERROR' INDICATOR
1968 006256 012737 000012 001122      MOV      #RMDS1,$BDDADR  ;FORM THE ADDRESS OF RMDS1 FOR TYPEOUT
1969 006264 060037 001122      ADD      RO,$DDADR       ;ADD THE I/O LINE ADDRESS
1970 006270 012737 011700 001124      MOV      #MOL!PGH!DPR!DRY!VV,$GDDAT ;COMPARISON CONSTANT
1971 006276 113760 001224 000010      MOV      PORTA,RMCS2(RO) ;SELECT PORT A.
1972 006304 016037 000012 001170      MOV      RMDS1(RO),$TMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
1973 006312 013737 001170 001164      MOV      $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
1974 006320 042737 100100 001164      BIC      #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
1975 006326 113760 001226 000010      MOV      PORTB,RMCS2(RO) ;SELECT PORT B.
1976 006334 016037 000012 001172      MOV      RMDS1(RO),$TMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
1977 006342 013737 001172 001166      MOV      $TMP3,$TMP1    ;COPY IT INTO '$TMP1'
1978 006350 042737 100100 001166      BIC      #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
1979 006356 023737 001164 001166      CMP      $TMP0,$TMP1    ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
1980 006364 001006                BNE     66$            ;BR IF NOT
1981 006366 005737 001164      TST      $TMP0          ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
1982 006372 001045                BNE     68$            ;BR IF NOT
1983 006374 104034                ERROR   34            ;REPORT DRIVE NOT IN NEUTRAL OR NOT SEIZED
1984 006376 000137 006576      JMP      70$            ;BYPASS THE REST OF THE CHECKS
1985 006402 013737 001170 001126      66$:      MOV      $TMP2,$BDDAT   ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE

```

M03

MD-11-DZRMH-A, RM03 DUAL CONTRLR LOGIC TEST - PART 2
DZRMH.A.P11 01-AUG-77 11:02

T4 MEASURE

MACY11 30(1046) 01-AUG-77 11:14 PAGE 38
THE TIMEOUT ONE-SHOT THROUGH PORT A

SEQ 0039

1986	006410	013737	001226	001234	MOV	PORTB,PTNBR	;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
1987	006416	113760	001226	000010	MOV	PORTB,RMCS2(RO)	;SELECT PORT B.
1988	006424	005737	001164		TST	\$TMP0	;SEE IF STATUS EQ 0 FROM PORT A.
1989	006430	001414			BEQ	67\$;BR IF ZERO
1990	006432	013737	001224	001234	MOV	PORTA,PTNBR	;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
1991	006440	013737	001172	001126	MOV	\$TMP3,\$BDDAT	; 'BAD DATA' FOR ERROR TYPE OUT
1992	006446	113760	001224	000010	MOV	PORTA,RMCS2(RO)	;SELECT PORT A.
1993	006454	005737	001166		TST	\$TMP1	;SEE IF STATUS EQ ZERO FROM PORT B.
1994	006460	001012			BNE	68\$;BR IF NOT
1995	006462	012737	177777	001250	MOV	#-1,RELEA	;SET 'RELEASE ERROR' INDICATOR
1996	006470	012760	000011	000000	MOV	#11,RMCS1(RO)	;CLEAR THE DRIVE
1997	006476	012760	000013	000000	MOV	#13,RMCS1(RO)	;RELEASE THE DRIVE
1998	006504	104035			ERROR	35	;TYPE ERROR MESSAGE 35
1999	006506	013737	001170	001126	MOV	\$TMP2,\$BDDAT	;LOOK FOR BIT FAILURES WHEN RMDS1 READ
2000	006514	013737	001224	001234	MOV	PORTA,PTNBR	;CHANGE PORT NUMBER
2001	006522	042737	100000	001170	BIC	#ATA,\$TMP2	;DON'T CHECK THE ATTN BIT
2002	006530	023737	001124	001170	CMP	\$GDDAT,\$TMP2	;ALL BITS OK ?
2003	006536	001401			BEQ	69\$;BR IF OK FROM PORT A.
2004	006540	104037			ERROR	37	;REPORT ERROR
2005	006542	013737	001172	001126	MOV	\$TMP3,\$BDDAT	;CHECK RMDS1 FOR BIT FAILURES - FROM PORT B.
2006	006550	013737	001226	001234	MOV	PORTB,PTNBR	;CHANGE PORT NUMBER
2007	006556	042737	100000	001172	BIC	#ATA,\$TMP3	;DON'T CHECK THE ATTN BIT
2008	006564	023737	001124	001172	CMP	\$GDDAT,\$TMP3	;SEE IF READ OK FROM PORT B.
2009	006572	001401			BEQ	70\$;BR IF OK
2010	006574	104037			ERROR	37	;REPORT THE ERROR
2011	006576	000240			NOP		
2012	006600	000004			4\$:	SCOPE	;LOOP ?

2013							
2014							
2015							
2016							
2017							
2018							
2019							
2020							
2021							
2022							
2023							
2024							
2025							
2026							
2027							
2028							
2029							

```

*****
*TEST 5      MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT B
*
*MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT B
*
*  A.  WRITE 0'S INTO RMDS1 THROUGH PORT B AND VERIFY THAT THE
*       DRIVE HAS BEEN SEIZED.
*
*  B.  WAIT FOR TIMEOUT TO OCCUR.  MEASURE THE DURATION OF THE TIMEOUT
*       ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
*
*  C.  VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS
*       TO NEUTRAL
*
*****

```

2030	006602				TST5:		
2031	006602	005737	001266		TST	KYBCTL	;PERFORMING ONLY SINGLE TESTS ?
2032	006606	001406			BEQ	2\$;BR IF NOT
2033	006610	100002			BPL	1\$;BR IF JUST ENTERED TEST
2034	006612	000137	002424		JMP	EXEC	;RETURN & GET NEXT TEST NUMBER
2035	006616	012737	177777	001266	MOV	#-1,KYBCTL	;SET SINGLE TEST INDICATOR
2036	006624	112737	000005	001102	MOV	#5,\$STNM	;TEST NUMBER
2037	006632	012737	006654	001106	MOV	#TESTS,\$LPADR	;LOAD LOOP ON TEST ADDRESS
2038	006640	012737	006654	001110	MOV	#TESTS,\$LPERR	;LOAD LOOP ON ERROR ADDRESS
2039	006646	012737	000001	001176	MOV	#1,\$TIMES	;DO 1 ITERATION
2040	006654	012706	001100		MOV	#STACK,\$P	;SETUP THE STACK POINTER
2041	006660	005037	001262		CLR	TIMEB	;CLEAR THE TIMEOUT VALUE STORAGE LOCATION

N03

```

2042 006664 005037 001264          CLR    TIMEBP ;CLEAR THE + 25% TOLERANCE LOCATION
2043
2044                               ;*****
2045                               ;START THE TIMER
2046
2047 006670 005037 001252          CLR    TIME ;CLEAR THE ELAPSED TIME COUNTER
2048 006674 012737 003720 001254  MOV    #2000.,WATCH ;SET WATCH TO 2000 MS
2049
2050                               ;*****
2051
2052                               ;SEIZE THE DRIVE THROUGH PORT B
2053
2054 006702 113760 001226 000010  MOVB   PORTB,RMCS2(RO) ;SELECT PORT B
2055 006710 013737 001226 001236  MOV    PORTB,SEIZPT ;STORE SEIZING PORT'S ADDRESS
2056 006716 005060 000012          CLR    RMD51(RO) ;WRITE RMD51
2057 006722 113760 001224 000010  MOVB   PORTA,RMCS2(RO) ;SELECT PORT A
2058 006730 013737 001224 001234  MOV    PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2059 006736 013737 001224 001240  MOV    PORTA,OPPRT ;'OPPOSITE' PORT ADDRESS
2060 006744 016037 000012 001126  MOV    RMD51(RO),SBD0AT ;SEE IF DRIVE SEIZED BY PORT B
2061 006752 010037 001122          MOV    RO,SBDADR ;RHLI BASE ADDRESS
2062 006756 012737 000012 001122  ADD    #RMD51,SBDADR ;GENERATE BAD REGISTER ADDRESS
2063 006764 010037 001124          CLR    $GDDAT ;REGISTER SHOULD BE ZERO
2064 006770 023737 001124 001126  CMP    $GDDAT,SBD0AT ;IS THE REGISTER ZERO
2065 006776 001403          BEQ    64$ ;BR IF IT IS
2066 007000 104030          ERROR  30 ;REPORT THE ERROR
2067 007002 000137 007516          JMP    4$ ;BYPASS REST OF THE SUBTEST
2068 007006
2069 007006 113760 001226 000010 64$: MOVB   PORTB,RMCS2(RO) ;SELECT PORT B
2070 007014 013737 001226 001234  MOV    PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2071 007022 016037 000012 001126  MOV    RMD51(RO),SBD0AT ;SEE IF SEIZING PORT SEES CORRECT STATUS
2072 007030 012737 011700 001124  MOV    #MOL!PGM!DPR!DRY!VW,$GDDAT ;EXPECTED STATUS
2073 007036 013737 001124 001166  MOV    $GDDAT,$TMP1 ;USE GOOD DATA AS A MASK
2074 007044 005137 001166          COM    $TMP1 ;COMPLEMENT THE EXPECTED STATUS
2075 007050 013737 001126 001164  MOV    SBD0AT,$TMP0 ;SAVE THE ACTUAL STATUS
2076 007056 043737 001166 001164  BIC    $TMP1,$TMP0 ;CLEAR UNWANTED BITS
2077 007064 023737 001124 001164  CMP    $GDDAT,$TMP0 ;ARE THE EXPECTED STATUS BITS SET ?
2078 007072 001401          BEQ    65$ ;BR IF THEY ARE
2079 007074 104031          ERROR  31 ;REPORT THE ERROR
2080 007076 000240          65$: NOP
2081
2082                               ;*****
2083                               ;WAIT FOR PORT B TO TIMEOUT
2084
2085 007100 113760 001224 000010  MOVB   PORTA,RMCS2(RO) ;SELECT PORT A
2086 007106 013737 001224 001234  MOV    PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2087 007114 005760 000012          1$: TST    RMD51(RO) ;WAIT FOR THE DRIVE TO TIMEOUT
2088 007120 001006          BNE    2$ ;BR WHEN TIMEOUT OCCURS
2089 007122 005737 001254          TST    WATCH ;CHECK WATCH
2090 007126 001372          BNE    1$ ;BR IF NOT ZERO
2091 007130 104006          ERROR  6 ;NO TIMEOUT WITHIN 2 SECONDS
2092 007132 000137 007170          JMP    3$ ;BYPASS THE REST OF THE TEST
2093 007136 013737 001252 001262 2$: MOV    TIME,TIMEB ;SAVE THE ELAPSED TIME FOR PORT B
2094 007144 004537 013412          JSR    RS,TOLER ;CALCULATE THE TOLERANCE
2095 007150 001262          .WORD TIMEB ;TIMEOUT VALUE FOR PORT B
2096 007152 012637 001264          MOV    (SP)+,TIMEBP ;+25% TOLERANCE
2097

```



```

2098      ;:*****
2099      ;VERIFY THAT THE TIMEOUT ONE-SHOT VALUE IS AT LEAST 500 MS
2100
2101      007156 023727 001262 000764      CMP      TIMEB,#500.      ;IS TIMEOUT VALUE AT LEAST 500 MS ?
2102      007164 103001                      BHS      35              ;BR IF IT IS
2103      007166 104007                      ERROR    7              ;TIMEOUT LESS THAN 500 MS
2104
2105      ;:*****
2106      ;VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AFTER PORT B TIMED OUT
2107
2108      007170      35:
2109
2110      ;VERIFY THAT THE DRIVE IS IN NEUTRAL
2111
2112      007170 005037 001250      CLR      RELERR          ;CLEAR THE 'RELEASE ERROR' INDICATOR
2113      007174 012737 000012 001122      MOV      #RMS1,$BDDADR  ;FORM THE ADDRESS OF RMS1 FOR TYPEOUT
2114      007202 060037 001122                      ADD      R0,$BDDADR     ;ADD THE I/O BASE ADDRESS
2115      007206 012737 011700 001124      MOV      #M0L!PGM!DPR!DRY!VV,$GDDAT ;COMPARISON CONSTANT
2116      007214 113760 001224 000010      MOV      PORTA,RMCS2(R0) ;SELECT PORT A.
2117      007222 016037 000012 001170      MOV      RMS1(R0),$TMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
2118      007230 013737 001170 001164      MOV      $TMP2,$TMP0    ;COPY IT INTO '$TMP0'
2119      007236 042737 100100 001164      BIC      #ATA!VV,$TMP0  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
2120      007244 113760 001226 000010      MOV      PORTB,RMCS2(R0) ;SELECT PORT B.
2121      007252 016037 000012 001172      MOV      RMS1(R0),$TMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
2122      007260 013737 001172 001166      MOV      $TMP3,$TMP1    ;COPY IT INTO '$TMP1'
2123      007266 042737 100100 001166      BIC      #ATA!VV,$TMP1  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
2124      007274 023737 001164 001166      CMP      $TMP0,$TMP1    ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
2125      007302 001006                      BNE      66$           ;BR IF NOT
2126      007304 005737 001164                      TST      $TMP0         ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
2127      007310 001045                      BNE      68$           ;BR IF NOT
2128      007312 104034                      ERROR    34           ;REPORT DRIVE NOT IN NEUTRAL OR NOT SEIZED
2129      007314 000137 007514                      JMP      70$           ;BYPASS THE REST OF THE CHECKS
2130      007320 013737 001170 001126 66$:      MOV      $TMP2,$BDDAT   ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
2131      007326 013737 001226 001234      MOV      PORTB,PTNBR    ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
2132      007334 113760 001226 000010      MOV      PORTB,RMCS2(R0) ;SELECT PORT B.
2133      007342 005737 001164                      TST      $TMP0         ;SEE IF STATUS EQ 0 FROM PORT A.
2134      007346 001414                      BEQ      67$           ;BR IF ZERO
2135      007350 013737 001224 001234      MOV      PORTA,PTNBR    ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
2136      007356 013737 001172 001126      MOV      $TMP3,$BDDAT   ;'BAD DATA' FOR ERROR TYPE OUT
2137      007364 113760 001224 000010      MOV      PORTA,RMCS2(R0) ;SELECT PORT A.
2138      007372 005737 001166                      TST      $TMP1         ;SEE IF STATUS EQ ZERO FROM PORT B.
2139      007376 001012                      BNE      68$           ;BR IF NOT
2140      007400 012737 177777 001250 67$:      MOV      #-1,RELERR     ;SET 'RELEASE ERROR' INDICATOR
2141      007406 012760 000011 000000      MOV      #11,RMCS1(R0)  ;CLEAR THE DRIVE
2142      007414 012760 000013 000000      MOV      #13,RMCS1(R0)  ;RELEASE THE DRIVE
2143      007422 104035                      ERROR    35           ;TYPE ERROR MESSAGE 35
2144      007424 013737 001170 001126 68$:      MOV      $TMP2,$BDDAT   ;LOOK FOR BIT FAILURES WHEN RMS1 READ
2145      007432 013737 001224 001234      MOV      PORTA,PTNBR    ;CHANGE PORT NUMBER
2146      007440 042737 100000 001170      BIC      #ATA,$TMP2     ;DON'T CHECK THE ATTN BIT
2147      007446 023737 001124 001170      CMP      $GDDAT,$TMP2  ;ALL BITS OK ?
2148      007454 001401                      BEQ      69$           ;BR IF OK FROM PORT A.
2149      007456 104037                      ERROR    37           ;REPORT ERROR
2150      007460 013737 001172 001126 69$:      MOV      $TMP3,$BDDAT   ;CHECK RMS1 FOR BIT FAILURES - FROM PORT B.
2151      007466 013737 001226 001234      MOV      PORTB,PTNBR    ;CHANGE PORT NUMBER
2152      007474 042737 100000 001172      BIC      #ATA,$TMP3     ;DON'T CHECK THE ATTN BIT
2153      007502 023737 001124 001172      CMP      $GDDAT,$TMP3  ;SEE IF READ OK FROM PORT B.

```

2154 007510 001401
2155 007512 104037
2156 007511 000240
2157 007516 000004

BEQ 70S ;BR IF OK
ERROR 37 ;REPORT THE ERROR
70S: NOP
4S: SCOPE ;LOOP ?

2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177

```
*****
*TEST 6 TEST 'CONTROLLER SELECT' SWITCH, DRIVE CYCLED UP
*
*TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED UP).
*
* A. SWITCH TO CONTROLLER 'A' POSITION. VERIFY THAT THE DRIVE IS IN
* NEUTRAL AND THAT THE STATUS BITS IN RMD51, AS READ THROUGH BOTH
* PORTS, ARE CORRECT.
*
* B. SWITCH TO CONTROLLER 'B' POSITION. VERIFY THAT THE DRIVE IS IN
* NEUTRAL AND THAT THE STATUS BITS IN RMD51, AS READ THROUGH BOTH
* PORTS, ARE CORRECT.
*
* C. RETURN THE 'CONTROLLER SELECT' SWITCH TO THE 'A/B' POSITION. VERIFY
* THE DRIVE STATE.
*****
```

2178 007520
2179 007520 005737 001266
2180 007524 001406
2181 007526 100002
2182 007530 000137 002424
2183 007534 012737 177777 001266
2184 007542 112737 000006 001102
2185 007550 012737 007572 001106
2186 007556 012737 007572 001110
2187 007564 012737 000001 001176
2188 007572 012706 001100

```
TST KYBCTL ;PERFORMING ONLY SINGLE TESTS ?
BEQ 2S ;BR IF NOT
BPL 1S ;BR IF JUST ENTERED TEST
JMP EXEC ;RETURN & GET NEXT TEST NUMBER
1S: MOV #-1,KYBCTL ;SET SINGLE TEST INDICATOR
2S: MOVB #6,$STNM ;TEST NUMBER
MOV #TEST6,$LPADR ;LOAD LOOP ON TEST ADDRESS
MOV #TEST6,$LPERR ;LOAD LOOP ON ERROR ADDRESS
MOV #1,$TIMES ;DO 1 ITERATION
TEST6: MOV #STACK,$SP ;SETUP THE STACK POINTER
```

2189
2190
2191
2192 007576 113760 001224 000010
2193 007604 005060 000012
2194 007610 012760 000011 000000
2195 007616 012760 000013 000000
2196 007624 113760 001226 000010
2197 007632 005060 000012
2198 007636 012760 000011 000000
2199 007644 012760 000013 000000
2200 007652 104401 017402
2201 007656 104401 017550
2202 007662 000000

```
;CLEAR ATTENTION BITS FOR BOTH PORTS
MOVB PORTA,RMCS2(RO) ;SELECT PORT #A
CLR RMD51(RO) ;SEIZE THE DRIVE
MOV #11,RMCS1(RO) ;ISSUE DRIVE CLEAR
MOV #13,RMCS1(RO) ;RELEASE THE DRIVE
MOVB PORTB,RMCS2(RO) ;SELECT PORT #B
CLR RMD51(RO) ;SEIZE THE DRIVE THROUGH PORT 'B'
MOV #11,RMCS1(RO) ;ISSUE DRIVE CLEAR
MOV #13,RMCS1(RO) ;RELEASE THE DRIVE
TYPE ,SWTCHA ;SWITCH TO 'A'
TYPE ,CONTUE ;PRESS 'CONTINUE'
HALT
```

2203
2204
2205
2206 007664 005037 001250
2207 007670 012737 000012 001122
2208 007676 060037 001122
2209 007702 012737 011700 001124

```
;VERIFY THAT THE DRIVE IS IN NEUTRAL
CLR RELERR ;CLEAR THE 'RELEASE ERROR' INDICATOR
MOV #RMD51,$BOADR ;FORM THE ADDRESS OF RMD51 FOR TYPEOUT
ADD RO,$BOADR ;ADD THE I/O BASE ADDRESS
MOV #MOL!PGM!DPR!DRY!VV,$GDDAT ;COMPARISON CONSTANT
```

2210	007710	113760	001224	000010		MOV	PORTA, RMCS2(RO)	; SELECT PORT A.
2211	007716	016037	000012	001170		MOV	RMD51(RO), \$TMP2	; GET THE DRIVE STATUS REGISTER FROM PORT A.
2212	007724	013737	001170	001164		MOV	\$TMP2, \$TMP0	; COPY IT INTO '\$TMP0'
2213	007732	042737	100100	001164		BIC	#ATA!VV, \$TMP0	; CLEAR PORT DEPENDENT BITS FROM THE COPY
2214	007740	113760	001226	000010		MOV	PORTB, RMCS2(RO)	; SELECT PORT B.
2215	007746	016037	000012	001172		MOV	RMD51(RO), \$TMP3	; GET THE DRIVE STATUS REGISTER FROM PORT B.
2216	007754	013737	001172	001166		MOV	\$TMP3, \$TMP1	; COPY IT INTO '\$TMP1'
2217	007762	042737	100100	001166		BIC	#ATA!VV, \$TMP1	; CLEAR PORT DEPENDENT BITS FROM THE COPY
2218	007770	023737	001164	001166		CMP	\$TMP0, \$TMP1	; IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
2219	007776	001006				BNE	64\$; BR IF NOT
2220	010000	005737	001164			TST	\$TMP0	; REGISTERS ARE THE SAME: ARE THEY ZERO ?
2221	010004	001045				BNE	66\$; BR IF NOT
2222	010006	104034				ERROR	34	; REPORT DRIVE NOT IN NEUTRAL OR NOT SEIZED
2223	010010	000137	010210			JMP	68\$; BYPASS THE REST OF THE CHECKS
2224	010014	013737	001170	001126	64\$:	MOV	\$TMP2, \$BDDAT	; SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
2225	010022	013737	001226	001234		MOV	PORTB, PTNBR	; SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
2226	010030	113760	001226	000010		MOV	PORTB, RMCS2(RO)	; SELECT PORT B.
2227	010036	005737	001164			TST	\$TMP0	; SEE IF STATUS EQ 0 FROM PORT A.
2228	010042	001414				BEQ	65\$; BR IF ZERO
2229	010044	013737	001224	001234		MOV	PORTA, PTNBR	; SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
2230	010052	013737	001172	001126		MOV	\$TMP3, \$BDDAT	; 'BAD DATA' FOR ERROR TYPE OUT
2231	010060	113760	001224	000010		MOV	PORTA, RMCS2(RO)	; SELECT PORT A.
2232	010066	005737	001166			TST	\$TMP1	; SEE IF STATUS EQ ZERO FROM PORT B.
2233	010072	001012				BNE	66\$; BR IF NOT
2234	010074	012737	177777	001250	65\$:	MOV	#-1, RELERR	; SET 'RELEASE ERROR' INDICATOR
2235	010102	012760	000011	000000		MOV	#11, RMCS1(RO)	; CLEAR THE DRIVE
2236	010110	012760	000013	000000		MOV	#13, RMCS1(RO)	; RELEASE THE DRIVE
2237	010116	104017				ERROR	17	; TYPE ERROR MESSAGE 17
2238	010120	013737	001170	001126	66\$:	MOV	\$TMP2, \$BDDAT	; LOOK FOR BIT FAILURES WHEN RMD51 READ
2239	010126	013737	001224	001234		MOV	PORTA, PTNBR	; CHANGE PORT NUMBER
2240	010134	042737	100000	001170		BIC	#ATA, \$TMP2	; DON'T CHECK THE ATTN BIT
2241	010142	023737	001124	001170		CMP	\$GDDAT, \$TMP2	; ALL BITS OK ?
2242	010150	001401				BEQ	67\$; BR IF OK FROM PORT A.
2243	010152	104037				ERROR	37	; REPORT ERROR
2244	010154	013737	001172	001126	67\$:	MOV	\$TMP3, \$BDDAT	; CHECK RMD51 FOR BIT FAILURES - FROM PORT B.
2245	010162	013737	001226	001234		MOV	PORTB, PTNBR	; CHANGE PORT NUMBER
2246	010170	042737	100000	001172		BIC	#ATA, \$TMP3	; DON'T CHECK THE ATTN BIT
2247	010176	023737	001124	001172		CMP	\$GDDAT, \$TMP3	; SEE IF READ OK FROM PORT B.
2248	010204	001401				BEQ	68\$; BR IF OK
2249	010206	104037				ERROR	37	; REPORT THE ERROR
2250	010210	000240			68\$:	NOP		
2251	010212	104401	017465			TYPE	, SWTCHB	; SWITCH TO 'B'
2252	010216	104401	017550			TYPE	, CONTUE	; PRESS 'CONTINUE'
2253	010222	000000				HALT		
2254								
2255								; VERIFY THAT THE DRIVE IS IN NEUTRAL
2256								
2257	010224	005037	001250			CLR	RELERR	; CLEAR THE 'RELEASE ERROR' INDICATOR
2258	010230	012737	000012	001122		MOV	#RMD51, \$BDDAR	; FORM THE ADDRESS OF RMD51 FOR TYPEOUT
2259	010236	060037	001122			ADD	RO, \$BDDAR	; ADD THE I/O BASE ADDRESS
2260	010242	012737	011700	001124		MOV	#MOL!PGM!DPR!DRY!VV, \$GDDAT	; COMPARISON CONSTANT
2261	010250	113760	001224	000010		MOV	PORTA, RMCS2(RO)	; SELECT PORT A.
2262	010256	016037	000012	001170		MOV	RMD51(RO), \$TMP2	; GET THE DRIVE STATUS REGISTER FROM PORT A.
2263	010264	013737	001170	001164		MOV	\$TMP2, \$TMP0	; COPY IT INTO '\$TMP0'
2264	010272	042737	100100	001164		BIC	#ATA!VV, \$TMP0	; CLEAR PORT DEPENDENT BITS FROM THE COPY
2265	010300	113760	001226	000010		MOV	PORTB, RMCS2(RO)	; SELECT PORT B.

2266	010306	016037	000012	001172		MOV	RMS1(RO), \$TMP3	: GET THE DRIVE STATUS REGISTER FROM PORT B.
2267	010314	013737	001172	001166		MOV	\$TMP3, \$TMP1	: COPY IT INTO 'TMP1'
2268	010322	042737	100100	001166		BIC	#ATA, \$TMP1	: CLEAR PORT DEPENDENT BITS FROM THE COPY
2269	010330	023737	001164	001166		CMP	\$TMP0, \$TMP1	: IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
2270	010336	001006				BNE	69\$: BR IF NOT
2271	010340	005737	001164			TST	\$TMP0	: REGISTERS ARE THE SAME: ARE THEY ZERO ?
2272	010344	001045				BNE	71\$: BR IF NOT
2273	010346	104034				ERROR	34	: REPORT DRIVE NOT IN NEUTRAL OR NOT SEIZED
2274	010350	000137	010550			JMP	73\$: BYPASS THE REST OF THE CHECKS
2275	010354	013737	001170	001126	69\$:	MOV	\$TMP2, \$BDDAT	: SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
2276	010362	013737	001226	001234		MOV	PORTB, PTNBR	: SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
2277	010370	113760	001226	000010		MOV	PORTB, RMS2(RO)	: SELECT PORT B.
2278	010376	005737	001164			TST	\$TMP0	: SEE IF STATUS EQ 0 FROM PORT A.
2279	010402	001414				BEQ	70\$: BR IF ZERO
2280	010404	013737	001224	001234		MOV	PORTA, PTNBR	: SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
2281	010412	013737	001172	001126		MOV	\$TMP3, \$BDDAT	: 'BAD DATA' FOR ERROR TYPE OUT
2282	010420	113760	001224	000010		MOV	PORTA, RMS2(RO)	: SELECT PORT A.
2283	010426	005737	001166			TST	\$TMP1	: SEE IF STATUS EQ ZERO FROM PORT B.
2284	010432	001012				BNE	71\$: BR IF NOT
2285	010434	012737	177777	001250	70\$:	MOV	#-1, RELERR	: SET 'RELEASE ERROR' INDICATOR
2286	010442	012760	000011	000000		MOV	#11, RMS1(RO)	: CLEAR THE DRIVE
2287	010450	012760	000013	000000		MOV	#13, RMS1(RO)	: RELEASE THE DRIVE
2288	010456	104020				ERROR	20	: TYPE ERROR MESSAGE 20
2289	010460	013737	001170	001126	71\$:	MOV	\$TMP2, \$BDDAT	: LOOK FOR BIT FAILURES WHEN RMS1 READ
2290	010466	013737	001224	001234		MOV	PORTA, PTNBR	: CHANGE PORT NUMBER
2291	010474	042737	100000	001170		BIC	#ATA, \$TMP2	: DON'T CHECK THE ATTN BIT
2292	010502	023737	001124	001170		CMP	\$GDDAT, \$TMP2	: ALL BITS OK ?
2293	010510	001401				BEQ	72\$: BR IF OK FROM PORT A.
2294	010512	104037				ERROR	37	: REPORT ERROR
2295	010514	013737	001172	001126	72\$:	MOV	\$TMP3, \$BDDAT	: CHECK RMS1 FOR BIT FAILURES - FROM PORT B.
2296	010522	013737	001226	001234		MOV	PORTB, PTNBR	: CHANGE PORT NUMBER
2297	010530	042737	100000	001172		BIC	#ATA, \$TMP3	: DON'T CHECK THE ATTN BIT
2298	010536	023737	001124	001172		CMP	\$GDDAT, \$TMP3	: SEE IF READ OK FROM PORT B.
2299	010544	001401				BEQ	73\$: BR IF OK
2300	010546	104037				ERROR	37	: REPORT THE ERROR
2301	010550	000240			73\$:	NOP		
2302	010552	005737	001266			TST	KYBCTL	: SINGLE TEST MODE ?
2303	010556	001402				BEQ	1\$: BR IF NOT
2304	010560	104401	017313			TYPE	, SWTCHN	: RETURN SWITCH TO 'A/B'
2305	010564	000004			1\$:	SCOPE		: LOOP ?

2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321

```

*****
*TEST 7      TEST 'CONTROLL' SELECT' SWITCH LOCKED ON PORT A
*
*TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).
*
*  A.  CYCLE THE DRIVE DOWN.
*
*  B.  SWITCH TO CONTROLLER A POSITION.  VERIFY THAT THE DRIVE IS IN
*      NEUTRAL AND THAT THE STATUS BITS IN RMS1, AS READ THROUGH BOTH
*      PORTS, ARE CORRECT.
*
*  C.  SWITCH THE 'CONTROLLER SELECT' SWITCH TO A; CYCLE THE DRIVE UP.
*
*****
    
```

F04

- 2322 : * D. WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-A IS RESET, AND
- 2323 : * THAT 'ATA-A IS SET.
- 2324 : * E. ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
- 2325 : * PORT A.
- 2326 : * F. VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT B AND
- 2327 : * 'MED' SETS WHEN ATEMPTING TO ACCESS THE DRIVE THROUGH
- 2328 : * PORT B. ATTEMPT TO SET PORT REQUEST BY WRITING 0'S
- 2329 : * INTO RMD51 THROUGH PORT B.
- 2330 : * G. ISSUE A RELEASE COMMAND THROUGH PORT A. VERIFY THAT THE
- 2331 : * DRIVE REMAINS LOCKED ON PORT A.
- 2332 : *
- 2333 : *
- 2334 : *
- 2335 : *
- 2336 : *

```

2337 010566 005737 001266
2338 010566 001406
2339 010572 001406
2340 010574 100002
2341 010576 000137 002424
2342 010602 012737 177777 001266
2343 010610 112737 000007 001102
2344 010616 012737 010640 001106
2345 010624 012737 010640 001110
2346 010632 012737 000001 001176
2347 010640 012706 001100
2348 010644 113760 001224 000010
2349 010652 013737 001224 001234
2350 010660 104401 017617
    
```

```

*****
TST7:
    TST    KYBCTL      ;PERFORMING ONLY SINGLE TESTS ?
    BEQ    2$          ;BR IF NOT
    BPL    1$          ;BR IF JUST ENTERED TEST
    JMP    EXEC        ;RETURN & GET NEXT TEST NUMBER
1$:    MOV    #-1,KYBCTL ;SET SINGLE TEST INDICATOR
2$:    MOV    #7,$STNM  ;TEST NUMBER
    MOV    #TEST7,$LPADR ;LOAD LOOP ON TEST ADDRESS
    MOV    #TEST7,$LPERR ;LOAD LOOP ON ERROR ADDRESS
    MOV    #1,$TIMES   ;DO 1 ITERATION
TEST7: MOV    #STACK,SP ;SETUP THE STACK POINTER
    MOV    PORTA,RMCS2(RO) ;SELECT PORT A
    MOV    PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
    TYPE    ,CYCLED    ;'CYCLE DOWN THE DRIVE'
    
```

```

*****
;WAIT FOR 'MOL' TO RESET
1$:    BIT    #MOL,RMDS1(RO) ;TEST 'MOL'
    BNE    1$          ;BR IF IT IS STILL SET
    TYPE    ,SWTCHA     ;SWITCH TO 'A'
    TYPE    ,CYCLEU    ;'CYCLE UP THE DRIVE'
    
```

```

*****
;WAIT FOR DRIVE TO CYCLE UP AFTER SWITCH CHANGED
2$:    BIT    #MOL,RMDS1(RO) ;TEST 'MOL' AGAIN
    BEQ    2$          ;BR IF NOT SET
    
```

```

*****
;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT A
    CLR    CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
    MOV    RMDS1(RO),SBODAT ;GET CONTENTS OF RMD51
    MOV    #RMDS1,SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
    ADD    RO,SBOADR   ;ADD RHI1 BASE ADDRESS
    MOV    #ATA!MOL!DPR!DRY,$GDDAT ;WHAT REGISTER SHOULD BE
    MOV    SBODAT,$TMP0 ;MOVE REGISTER CONTENTS TO '$TMP0'
    BIC    #1C111700,$TMP0 ;SAVE SPECIFIED BITS
    CMP    $GDDAT,$TMP0 ;COMPARE THE BITS
    BEQ    64$        ;BR IF OK
    
```

G04

MD-11-DZRMH-A, RMD3 DUAL CONTRLR LOGIC
DZRMHA.P11 01-AUG-77 11:02

TEST - PART 2
T7 TEST 'CONTROLLER SELECT'

MACY11 30(1046) 01-AUG-77 11:14 PAGE 45
SWITCH LOCKED ON PORT A

SEQ 0046

2378 010772 013737 001126 001174
2379 011000 042737 111700 001174
2380 011006 053737 001174 001124
2381 011014 104021
2382 011016 005137 001244
2383 011022 000240
2384
2385
2386
2387
2388 011024 012760 000011 000000
2389 011032 012760 000021 000000
2390 011040 012760 010000 000032
2391
2392
2393
2394
2395
2396 011046 113760 001226 000010
2397 011054 013737 001226 001234
2398 011062 005037 001244
2399 011066 016037 000012 001126
2400 011074 012737 000012 001122
2401 011102 060037 001122
2402 011106 005037 001124
2403 011112 013737 001126 001164
2404 011120 042737 000077 001164
2405 011126 023737 001124 001164
2406 011134 001414
2407 011136 013737 001126 001174
2408 011144 042737 177700 001174
2409 011152 053737 001174 001124
2410 011160 104022
2411 011162 005137 001244
2412 011166 000240
2413 011170 005037 001244
2414 011174 016037 000010 001126
2415 011202 012737 000010 001122
2416 011210 060037 001122
2417 011214 012737 010000 001124
2418 011222 013737 001126 001164
2419 011230 042737 167777 001164
2420 011236 023737 001124 001164
2421 011244 001414
2422 011246 013737 001126 001174
2423 011254 042737 010000 001174
2424 011262 053737 001174 001124
2425 011270 104023
2426 011272 005137 001244
2427 011276 000240
2428 011300 005060 000012
2429
2430
2431
2432
2433

MOV \$B00AT,STMP4 ;COPY 'BAD DATA'
BIC #111700,STMP4 ;CLEAR THE MASKED BITS
BIS STMP4,\$G00AT ;'OR' WITH GOOD DATA FOR TYPEOUT
ERROR 21 ;TYPE MESSAGE 21
COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
64\$: NOP
;*****
;SET VOLUME VALID FOR PORT A
MOV #11,RMCS1(RO) ;ISSUE A DRIVE CLEAR
MOV #21,RMCS1(RO) ;ISSUE A READIN PRESET
MOV #FMT16,RMOF(RO) ;SET FMT16
;*****
;CHECK THE DRIVE STATUS THROUGH PORT B; VERIFY THAT 'MED'
;SETS WHEN THE DRIVE IS ACCESSED THROUGH PORT B.
MOVB PORTB,RMCS2(RO) ;SELECT PORT B
MOV PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
MOV RMDS1(RO),SB00AT ;GET CONTENTS OF RMDS1
MOV #RMDS1,SB0ADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
ADD RO,SB0ADR ;ADD RH11 BASE ADDRESS
CLR \$G00AT ;WHAT REGISTER SHOULD BE
MOV SB00AT,STMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
BIC #1C17700,STMP0 ;SAVE SPECIFIED BITS
CMP \$G00AT,STMP0 ;COMPARE THE BITS
BEQ 66\$;BR IF OK
MOV SB00AT,STMP4 ;COPY 'BAD DATA'
BIC #177700,STMP4 ;CLEAR THE MASKED BITS
BIS STMP4,\$G00AT ;'OR' WITH GOOD DATA FOR TYPEOUT
ERROR 22 ;TYPE MESSAGE 22
COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
65\$: NOP
CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
MOV RMCS2(RO),SB00AT ;GET CONTENTS OF RMCS2
MOV #RMCS2,SB0ADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
ADD RO,SB0ADR ;ADD RH11 BASE ADDRESS
MOV #MED,\$G00AT ;WHAT REGISTER SHOULD BE
MOV SB00AT,STMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
BIC #1CNE0,STMP0 ;SAVE SPECIFIED BITS
CMP \$G00AT,STMP0 ;COMPARE THE BITS
BEQ 68\$;BR IF OK
MOV SB00AT,STMP4 ;COPY 'BAD DATA'
BIC #MED,STMP4 ;CLEAR THE MASKED BITS
BIS STMP4,\$G00AT ;'OR' WITH GOOD DATA FOR TYPEOUT
ERROR 23 ;TYPE MESSAGE 23
COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
68\$: NOP
CLR RMDS1(RO) ;TRY TO SET REQUEST BY WRITING THROUGH
;THE LOCKED OUT PORT (PORT 'B')

;*****
;VERIFY THAT DRIVE STAYS LOCKED ON PORT A

H04

```

2434 011304 113760 001224 000010 MOVB PORTA,RMCS2(RO) ;SELECT PORT A
2435 011312 013737 001224 001234 MOV PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2436 011320 012760 000013 000012 MOV #13,RMDS1(RO) ;ISSUE A RELEASE THROUGH PORT A
2437 011326 013737 001224 001236 MOV PORTA,SFIZPT ;ADDRESS OF 'LOCKED ON' PORT
2438 011334 113760 001226 000010 MOVB PORTB,RMCS2(RO) ;SELECT PORT B
2439 011342 013737 001226 001234 MOV PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2440 011350 005037 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
2441 011354 016037 000012 001126 MOV RMDS1(RO),SPODAT ;GET CONTENTS OF RMDS1
2442 011362 012737 000012 001122 MOV #RMDS1,SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
2443 011370 060037 001122 ADD RO,SBOADR ;ADD RHI1 BASE ADDRESS
2444 011374 005037 001124 CLR SGOADR ;WHAT REGISTER SHOULD BE
2445 011400 013737 001126 001164 MOV SBOADR,STMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
2446 011406 042737 000077 001164 BIC #1C177700,STMP0 ;SAVE SPECIFIED BITS
2447 011414 023737 001124 001164 CMP SGOADR,STMP0 ;COMPARE THE BITS
2448 011422 001414 BEQ 70S ;BR IF OK
2449 011424 013737 001126 001174 MOV SBOADR,STMP4 ;COPY 'BAD DATA'
2450 011432 042737 177700 001174 BIC #17700,STMP4 ;CLEAR THE MASKED BITS
2451 011440 053737 001174 001124 BIS STMP4,SGOADR ;'OR' WITH GOOD DATA FOR TYPEOUT
2452 011446 104024 ERROR 24 ;TYPE MESSAGE 24
2453 011450 005137 001244 COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
2454 011454 000240 70S: NOP

```

;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST

```

2458 011456 105737 001103 TSTB SERFLG ;DID AN ERROR OCCUR
2459 011462 001412 BEQ 3S ;BR IF NOT
2460 011464 032777 001000 167446 BIT #SW09,2SWR ;SEE IF LOOP ON ERROR (SWR9=1)
2461 011472 001406 BEQ 3S ;BR IF NOT
2462 011474 105037 001103 CLRB SERFLG ;CLEAR THE ERROR FLAG
2463 011500 005037 001176 CLR $TIMES ;CLEAR THE MAX ITERATION COUNT
2464 011504 000177 167400 JMP 2SLPERR ;GO TO THE LOOP ADDRESS
2465 011510 005737 001266 3S: TST KYBCTL ;IN SINGLE TEST MODE ?
2466 011514 001460 BEQ 6S ;BR IF NOT
2467 011516 032777 040000 167414 BIT #SW14,2SWR ;LOOP ON TEST ?
2468 011524 001054 BNE 6S ;BR IF LOOPING
2469 011526 104401 017617 TYPE ,CYCLED ;TYPE 'CYCLE DOWN'
2470 011532 104401 017313 TYPE ,SWTCHN ;'SWITCH TO A/B'
2471 011536 104401 017641 TYPE ,CYCLEU ;'CYCLE THE DRIVE UP'
2472 011542 113760 001224 000010 MOVB PORTA,RMCS2(RO) ;SELECT PORT A
2473 011550 013737 001224 001234 MOV PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2474 011556 032760 010000 000012 4S: BIT #MOL,RMDS1(RO) ;CHECK 'MOL'
2475 011564 001374 BNE 4S ;BR IF SET (DRIVE NOT CYCLED DOWN)
2476 011566 032760 010000 000012 5S: BIT #MOL,RMDS1(RO) ;CHECK 'MOL' AGAIN
2477 011574 001774 BEQ 5S ;BR IF NOT SET (DRIVE NOT CYCLED UP)

```

;SET VOLUME VALID FOR BOTH PORTS

```

2481 011576 012760 000011 000000 MOV #11,RMCS1(RO) ;ISSUE A DRIVE CLEAR THROUGH PORT A
2482 011604 012760 000021 000000 MOV #21,RMCS1(RO) ;ISSUE A READIN PRESET THROUGH PORT A
2483 011612 012760 000013 000000 MOV #13,RMCS1(RO) ;RELEASE PORT A
2484 011620 113760 001226 000010 MOVB PORTB,RMCS2(RO) ;SELECT PORT B
2485 011626 013737 001226 001234 MOV PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2486 011634 012760 000021 000000 MOV #21,RMCS1(RO) ;ISSUE A READIN PRESET THROUGH PORT B
2487 011642 012760 010000 000032 MOV #FMT16,RMOF(RO) ;SET FMT16
2488 011650 012760 000013 000000 MOV #13,RMCS1(RO) ;RELEASE PORT B

```

2490	011656	104401	001207	
2491	011662	012737	011610	001254
2492	011670	005737	001254	
2493	011674	001375		
2494	011676	000004		
2495	011700	000400		

```

6$: TYPE      SCRLF      ;CR-LF
MOV      #5000.,WATCH ;SPINDLE MOTOR 'COOL DOWN' DELAY
7$: TST      WATCH      ;FINISHED ?
BNE     7$          ;BR IF NOT
SCOPE   7$          ;LOOP ?
BR      TST10      ;;GO TO NEXT TEST

```

```

*****
*TEST 10      TEST 'CONTROLLER SELECT' SWITCH LOCKED ON PORT B
*
*TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).
*
* A. CYCLE THE DRIVE DOWN.
*
* B. SWITCH TO CONTROLLER B POSITION. VERIFY THAT THE DRIVE IS IN
*    NEUTRAL AND THAT THE STATUS BITS IN RMD51, AS READ THROUGH BOTH
*    PORTS, ARE CORRECT.
*
* C. SWITCH THE 'CONTROLLER SELECT' SWITCH TO B; CYCLE THE DRIVE UP.
*
* D. WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-B IS RESET, AND
*    THAT 'ATA-B IS SET.
*
* E. ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
*    PORT B.
*
* F. VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT A AND
*    'MED' SETS WHEN ATTEMPTING TO ACCESS THE DRIVE THROUGH
*    PORT A. ATTEMPT TO SET PORT REQUEST BY WRITING 0'S
*    INTO RMD51 THROUGH PORT A.
*
* G. ISSUE A RELEASE COMMAND THROUGH PORT B. VERIFY THAT THE
*    DRIVE REMAINS LOCKED ON PORT B.
*
* H. CYCLE THE DRIVE DOWN. CHANGE THE 'CONTROLLER SELECT' SWITCH TO
*    A/B; CYCLE THE DRIVE UP.
*
* I. VERIFY THAT BOTH PORTS CAN ACCESS THE DRIVE, THAT BOTH ATTENTION
*    BITS ARE SET, AND THAT BOTH 'VV' BITS ARE RESET.
*****

```

2531	011702			
2532	011702	005737	001266	
2533	011706	001406		
2534	011710	100002		
2535	011712	000137	002424	
2536	011716	012737	177777	001266
2537	011724	112737	000010	001102
2538	011732	012737	011754	001106
2539	011740	012737	011754	001110
2540	011746	012737	000001	001176
2541	011754	012706	001100	
2542	011760	113760	001226	000010
2543	011766	013737	001226	001234
2544	011774	104401	017617	
2545				

```

*****
TST10:
TST      KYBCTL      ;PERFORMING ONLY SINGLE TESTS ?
BEQ     2$          ;BR IF NOT
BPL     1$          ;BR IF JUST ENTERED TEST
JMP     EXEC        ;RETURN & GET NEXT TEST NUMBER
1$: MOV   #-1,KYBCTL ;SET SINGLE TEST INDICATOR
2$: MOVB #10,$STN#  ;TEST NUMBER
MOV     #TEST10,$LPADR ;LOAD LOOP ON TEST ADDRESS
MOV     #TEST10,$LPERR ;LOAD LOOP ON ERROR ADDRESS
MOV     #1,$TIMES   ;DO 1 ITERATION
TEST10: MOV  #STACK,$SP ;SETUP THE STACK POINTER
MOVB   PORTB,$RMC52(RO) ;SELECT PORT B
MOV    PORTB,$PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
TYPE   ,CYCLED      ;'CYCLE DOWN THE DRIVE'

```



```

2546 ;:*****
2547 ;WAIT FOR 'MOL' TO RESET
2548
2549 012000 032760 010000 000012 15: BIT #MOL,RMDS1(RO) ;TEST 'MOL'
2550 012006 001374 ;BNE 15 ;BR IF IT IS STILL SET
2551 012010 104401 017465 ;TYPE ,SWTCHB ;SWITCH TO 'B'
2552 012014 104401 017641 ;TYPE ,CYCLEU ;'CYCLE UP THE DRIVE'
2553
2554 ;:*****
2555 ;WAIT FOR DRIVE TO CYCLE UP AFTER SWITCH CHANGED
2556
2557 012020 032760 010000 000012 25: BIT #MOL,RMDS1(RO) ;TEST 'MOL' AGAIN
2558 012026 001774 ;BEQ 25 ;BR IF NOT SET
2559
2560 ;:*****
2561 ;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT B
2562
2563 012030 005037 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
2564 012034 016037 000012 001126 MOV RMDS1(RO),SBO0AT ;GET CONTENTS OF RMDS1
2565 012042 012737 000012 001122 MOV #RMDS1,SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
2566 012050 060037 001122 ADD RO,SBOADR ;ADD RH11 BASE ADDRESS
2567 012054 012737 110600 001124 MOV #ATA!MOL!DPR!DRY,SGO0AT ;WHAT REGISTER SHOULD BE
2568 012062 013737 001126 001164 MOV SBO0AT,$TMP0 ;MOVE REGISTER CONTENTS TO '$TMP0'
2569 012070 042737 066077 001164 BIC #1C111700,$TMP0 ;SAVE SPECIFIED BITS
2570 012076 023737 001124 001164 CMP SGO0AT,$TMP0 ;COMPARE THE BITS
2571 012104 001414 BEQ 64$ ;BR IF OK
2572 012106 013737 001126 001174 MOV SBO0AT,$TMP4 ;COPY 'BAD DATA'
2573 012114 042737 111700 001174 BIC #111700,$TMP4 ;CLEAR THE MASKED BITS
2574 012122 053737 001174 001124 BIS $TMP4,$SGO0AT ;'OR' WITH GOOD DATA FOR TYPEOUT
2575 012130 104021 ERROR 21 ;TYPE MESSAGE 21
2576 012132 005137 001244 COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
2577 012136 000240 64$: NOP
2578
2579 ;:*****
2580 ;SET VOLUME VALID FOR PORT B
2581
2582 012140 012760 000011 000000 MOV #11,RMCS1(RO) ;ISSUE A DRIVE CLEAR
2583 012146 012760 000021 000000 MOV #21,RMCS1(RO) ;ISSUE A READIN PRESET
2584 012154 012760 010000 000032 MOV #FMT16,RMOF(RO) ;SET FMT16
2585
2586 ;:*****
2587 ;CHECK THE DRIVE STATUS THROUGH PORT A; VERIFY THAT 'MED'
2588 ; SETS WHEN THE DRIVE IS ACCESSED THROUGH PORT A.
2589
2590 012162 113760 001224 000010 MOV#B PORTA,RMCS2(RO) ;SELECT PORT A
2591 012170 013737 001224 001234 MOV PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2592 012176 005037 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
2593 012202 016037 000012 001126 MOV RMDS1(RO),SBO0AT ;GET CONTENTS OF RMDS1
2594 012210 012737 000012 001122 MOV #RMDS1,SBOADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
2595 012216 060037 001122 ADD RO,SBOADR ;ADD RH11 BASE ADDRESS
2596 012222 005037 001124 CLR SGO0AT ;WHAT REGISTER SHOULD BE
2597 012226 013737 001126 001164 MOV SBO0AT,$TMP0 ;MOVE REGISTER CONTENTS TO '$TMP0'
2598 012234 042737 000077 001164 BIC #1C177700,$TMP0 ;SAVE SPECIFIED BITS
2599 012242 023737 001124 001164 CMP SGO0AT,$TMP0 ;COMPARE THE BITS
2600 012250 001414 BEQ 66$ ;BR IF OK
2601 012252 013737 001126 001174 MOV SBO0AT,$TMP4 ;COPY 'BAD DATA'

```

K04

MD-11-DZRMH-A, RMO3 DUAL CONTRLR LOGIC TEST - PART 2
DZRMHA.P11 01-AUG-77 11:02 T10

MACY11 30(1046) 01-AUG-77 11:14 PAGE 49
TEST 'CONTROLLER SELECT' SWITCH LOCKED ON PORT B

SEQ 0050

```

2602 012260 042737 177700 001174 BIC #177700,STMP4 ;CLEAR THE MASKED BITS
2603 012266 053737 001174 001124 BIS STMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
2604 012274 104022 ERROR 22 ;TYPE MESSAGE 22
2605 012276 005137 001244 COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
2606 012302 001240 66$: NOP
2607 012304 01337 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
2608 012310 016037 000010 001126 MOV RMCS2(RO),SBDDAT ;GET CONTENTS OF RMCS2
2609 012316 012737 000010 001122 MOV #RMCS2,$SDADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
2610 012324 060037 001122 ADD RO,R ;ADD RHI1 BASE ADDRESS
2611 012330 012737 010000 001124 MOV #MED,$GDDAT ;WHAT REGISTER SHOULD BE
2612 012336 013737 001126 001164 MOV SBDDAT,STMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
2613 012344 042737 167777 001164 BIC #1CND,STMP0 ;SAVE SPECIFIED BITS
2614 012352 023737 001124 001164 CMP $GDDAT,STMP0 ;COMPARE THE BITS
2615 012360 001414 BEQ 68$ ;BR IF OK
2616 012362 013737 001126 001174 MOV SBDDAT,STMP4 ;COPY 'BAD DATA'
2617 012370 042737 010000 001174 BIC #MED,STMP4 ;CLEAR THE MASKED BITS
2618 012376 053737 001174 001124 BIS STMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
2619 012404 104023 ERROR 23 ;TYPE MESSAGE 23
2620 012406 005137 001244 COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
2621 012412 000240 68$: NOP
2622 012414 005060 000012 CLR RMDS1(RO) ;TRY TO SET REQUEST BY WRITING THROUGH
;THE LOCKED OUT PORT (PORT 'A')
;*****
;VERIFY THAT DRIVE STAYS LOCKED ON PORT B
2628 012420 113760 001226 000010 MOVB PORTB,RMCS2(RO) ;SELECT PORT B
2629 012426 013737 001226 001234 MOV PORTB,PTNLR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2630 012434 012760 000013 000012 MOV #13,RMDS1(RO) ;ISSUE A RELEASE THROUGH PORT B
2631 012442 013737 001226 001236 MOV PORTB,SEIZPT ;ADDRESS OF 'LOCKED ON' PORT
2632 012450 113760 001224 000010 MOVB PORTA,RMCS2(RO) ;SELECT PORT A
2633 012456 013737 001224 001234 MOV PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2634 012464 005037 001244 CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
2635 012470 016037 000012 001126 MOV RMDS1(RO),SBDDAT ;GET CONTENTS OF RMDS1
2636 012476 012737 000012 001122 MOV #RMDS1,$SDADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
2637 012504 060037 001122 ADD RO,$SDADR ;ADD RHI1 BASE ADDRESS
2638 012510 005037 001124 CLR $GDDAT ;WHAT REGISTER SHOULD BE
2639 012514 013737 001126 001164 MOV SBDDAT,STMP0 ;MOVE REGISTER CONTENTS TO 'STMP0'
2640 012522 042737 000077 001164 BIC #1C177700,STMP0 ;SAVE SPECIFIED BITS
2641 012530 023737 001124 001164 CMP $GDDAT,STMP0 ;COMPARE THE BITS
2642 012536 001414 BEQ 70$ ;BR IF OK
2643 012540 013737 001126 001174 MOV SBDDAT,STMP4 ;COPY 'BAD DATA'
2644 012546 042737 177700 001174 BIC #177700,STMP4 ;CLEAR THE MASKED BITS
2645 012554 053737 001174 001124 BIS STMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
2646 012562 104024 ERROR 24 ;TYPE MESSAGE 24
2647 012564 005137 001244 COM CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
2648 012570 000240 70$: NOP
;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST
2652 012572 105737 001103 TSTB $ERFLG ;DID AN ERROR OCCUR
2653 012576 001412 BEQ 3$ ;BR IF NOT
2654 012600 032777 001000 166332 BIT #SW09,$SWR ;SEE IF LOOP ON ERROR (SWR9=1)
2655 012606 001406 BEQ 3$ ;BR IF NOT
2656 012610 105037 001103 CLR $ERFLG ;CLEAR THE ERROR FLAG
2657 012614 005037 001176 CLR $TIMES ;CLEAR THE MAX ITERATION COUNT

```

L04

MD-11-DZRMH-A, RMO3 DUAL CONTRLR LOGIC TEST - PART 2
DZRMH.A.P11 01-AUG-77 11:02

T10 TEST 'CONTROLLER SELECT'

MACY11 30(1046) 01-AUG-77 11:14 PAGE 50
SWITCH LOCKED ON PORT B

SEQ 0051

```

2658 012620 000177 166264          JMP      @SLPERR          ;GO TO THE LOOP ADDRESS
2659 012624 032777 040000 166306 3$: BIT      @SW14,@SWR      ;LOOP ON TEST ?
2660 012632 001054          BNE     6$              ;BR IF LOOPING
2661 012634 104401 017617          TYPE   ,CYCLED          ;TYPE 'CYCLE DOWN'
2662 012640 104401 017313          TYPE   ,SWTCHN          ;'SWITCH TO A/B'
2663 012644 104401 017641          TYPE   ,CYCLEU          ;'CYCLE THE DRIVE UP'
2664 012650 113760 001226 000010  MOVB   PORTB,RMCS2(RO)   ;SELECT PORT B
2665 012656 013737 001226 001234  MOV    PORTB,PTN&R      ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2666 012664 032760 010000 000012 4$: BIT      @MOL,RMDS1(RO) ;CHECK 'MOL'
2667 012672 001374          BNE     4$              ;BR IF SET (DRIVE NOT CYCLED DOWN)
2668 012674 032760 010000 000012 5$: BIT      @MOL,RMDS1(RO) ;CHECK 'MOL' AGAIN
2669 012702 001774          BEQ    5$              ;BR IF NOT SET (DRIVE NOT CYCLED UP)

```

```

;*****
;SET VOLUME VALID FOR BOTH PORTS

```

```

2674 012704 012760 000011 000000  MOV    #11,RMCS1(RO)   ;ISSUE A DRIVE CLEAR THROUGH PORT B
2675 012712 012760 000021 000000  MOV    #21,RMCS1(RO)   ;ISSUE A READIN PRESET THROUGH PORT B
2676 012720 012760 000013 000000  MOV    #13,RMCS1(RO)   ;RELEASE PORT B
2677 012726 113760 001224 000010  MOVB   PORTA,RMCS2(RO) ;SELECT PORT A
2678 012734 013737 001224 001234  MOV    PORTA,PTN&R      ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
2679 012742 012760 000021 000000  MOV    #21,RMCS1(RO)   ;ISSUE A READIN PRESET THROUGH PORT A
2680 012750 012760 010000 000032  MOV    #FMT16,RMOF(RO) ;SET FMT16
2681 012756 012760 000013 000000  MOV    #13,RMCS1(RO)   ;RELEASE PORT A
2682 012764 012737 011610 001254 6$: MOV    #5000.,WATCH  ;SPINDLE MOTOR 'COOL DOWN' DELAY
2683 012772 005737 001254 7$: TST    WATCH          ;FINISHED ?
2684 012776 001375          BNE     7$              ;BR IF NOT
2685 013000 000004          SCOPE ;LOOP ?
2686 013002 000137 013006  JMP    SEOP             ;GO TO THE END OF PASS ROUTINE

```

.SBTTL END OF PASS ROUTINE

```

;*****
;INCREMENT THE PASS NUMBER ($PASS)
;INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
;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 TST1AA

```

```

2698 013006          SEOP:
2699 013006 005737 001266  TST    KYBCTL          ;ENTERED TEST VIA KEYBOARD COMMAND ?
2700 013012 001402          BEQ    .+6             ;BR IF NOT
2701 013014 000137 002424  JMP    EXEC            ;RETURN TO KEYBOARD CONTROL
2702 013020 005037 001102  CLR    $STNM           ;ZERO THE TEST NUMBER
2703 013024 005037 001176  CLR    $TIMES          ;ZERO THE NUMBER OF ITERATIONS
2704 013030 005237 001100  INC    $PASS           ;INCREMENT THE PASS NUMBER
2705 013034 042737 100000 001100  BIC    #100000,$PASS   ;DON'T ALLOW A NEG. NUMBER
2706 013042 005327          DEC    (PC)+          ;LOOP?
2707 013044 000001          SEOPCT: .WORD 1
2708 013046 003063          BGT    $DOAGN         ;YES
2709 013050 012737          MOV    (PC)+,@(PC)+   ;RESTORE COUNTER
2710 013052 000001          SENDCT: .WORD 1
2711 013054 013044          SEOPCT
2712 013056 104401 013064  TYPE   ,65$           ;TYPE ASCIZ STRING
2713 013062 000407          BR     ,64$           ;GET OVER THE ASCIZ

```

END OF PASS ROUTINE

```

2714      ;:65$: .ASCIZ <12><15>/END PASS #/
2715      64$:
2716 013102      MOV      $PASS,-(SP)      ;:SAVE $PASS FOR TYPEOUT
2717      TYPDS      ;:TYPE PASS NUMBER
2718 013106 104405      ;:GO TYPE--DECIMAL ASCII WITH SIGN
2719 013110 104401 013116      TYPE      ,67$      ;:TYPE ASCIZ STRING
2720 013114 000421      BR        66$      ;:GET OVER THE ASCIZ
2721      ;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
2722      66$:
2723 013160      MOV      $ERTTL,-(SP)      ;:SAVE $ERTTL FOR TYPEOUT
2724      TYPDS      ;:TOTAL NUMBER OF ERRORS
2725 013164 104405      ;:GO TYPE--DECIMAL ASCII WITH SIGN
2726 013166 104401 001207      TYPE      ,SCLF      ;:TYPE CARRIAGE RETURN, LINE FEED
2727 013172 005037 001112      CLR      $ERTTL      ;:CLEAR ERROR TOTAL
2728 013176 013700 000042      $GET42: MOV      2*42,RO      ;:GET MONITOR ADDRESS
2729 013202 001405      BEQ      $DOAGN      ;:BRANCH IF NO MONITOR
2730 013204 000005      RESET      ;:CLEAR THE WORLD
2731 013206 004710      $ENDAD: JSR      PC,(RO)      ;:GO TO MONITOR
2732 013210 000240      NOP      ;:SAVE ROOM
2733 013212 000240      NOP      ;:FOR
2734 013214 000240      NOP      ;:ACT11
2735 013216      $DOAGN:
2736 013216 000137      JMP      2(PC)+      ;:RETURN
2737 013220 002706      $RTNAD: .WORD      TST1AA
2738 013222      377      377      000      $ENULL: .BYTE      -1,-1,0      ;:NULL CHARACTER STRING
2739      013226      .EVEN
2740
2741      ;:*****
2742
2743      .SBTTL  *** SUBROUTINES ***
2744
2745      ;:*****
2746
2747
2748      ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
2749
2750 013226 012737 013276 000004      CKCLK:  MOV      #CKCLK1,2*ERRVEC      ;:SET UP VECTOR FOR CLOCK CHECK
2751 013234 005037 000006      CLR      2*ERRVEC+2      ;:NEW PSW
2752 013240 005777 165746      TST      2$ILKCSR      ;:CHECK FOR KW11-P
2753 013244 013701 001216      MOV      $LPVEC,R1      ;:KW11-P VECTOR ADDRESS
2754 013250 012721 013360      MOV      #CLOCK,(R1)+      ;:SET UP KW11-P VECTOR
2755 013254 012711 000300      MOV      #300,(R1)      ;:PSW - PRI 6
2756 013260 012777 177777 165726      MOV      #-1,2$ILKCSB      ;:LOAD COUNTER BUFFER WITH 1'S
2757 013266 012777 000135 165716      MOV      #135,2$ILKCSR      ;:SET CLOCK - CNT UP, 16MS, CONT INT
2758 013274 000425      BR        CKCLK3
2759 013276 062706 000004      CKCLK1: ADD      #4,SP      ;:RESTORE THE STACK POINTER
2760 013302 012737 013340 000004      MOV      #CKCLK2,2*ERRVEC      ;:CHANGE ERROR VECTOR TO CHECK FOR KW11-L
2761 013310 005777 165704      TST      2$ILKS      ;:LOOK FOR KW11-L
2762 013314 013701 001222      MOV      $LLVEC,R1      ;:KW11-L VECTOR ADDRESS
2763 013320 012721 013360      MOV      #CLOCK,(R1)+      ;:SET UP KW11-L VECTOR
2764 013324 012711 000300      MOV      #300,(R1)      ;:PSW - PRI 6
2765 013330 012777 000100 165662      MOV      #100,2$ILKS      ;:SET KW11-L INTERRUPT
2766 013336 000404      BR        CKCLK3
2767 013340 062706 000004      CKCLK2: ADD      #4,SP      ;:RESTORE THE STACK POINTER
2768 013344 062716 000002      ADD      #2,(SP)      ;:INCREMENT RETURN, NO CLOCK
2769 013350 012737 000006 000004      CKCLK3: MOV      #6,2*ERRVEC      ;:RESTORE THE ERROR VECTOR

```

```

2770 013356 000207          RTS      PC
2771
2772          ;ROUTINE TO COUNT CLOCK TICKS
2773
2774 013360 062737 000021 001252 CLOCK:  ADD     #17, TIME      ;ADD 17 MS TO ELAPSED TIME COUNTER
2775 013356 005737 001254          TST     WATCH      ;IS WATCH ALREADY ZERO?
2776 013372 001406          BEQ     1$          ;BR IF IT IS
2777 013374 162737 000021 001254          SUB     #17, WATCH  ;SUBTRACT 17 MS FROM WATCH DOG COUNTER
2778 013402 100002          BPL     1$          ;BR IF NOT MINUS
2779 013404 005037 001254          CLR     WATCH      ;CLEAR WATCH LOG COUNTER
2780 013410 000002          1$:    RTI          ;RETURN
2781
2782          ;ROUTINE TO CALCULATE + 25% TIME TOLERANCE VALUES
2783
2784 013412 005746          TOLER: TST     -(SP)      ;MAKE ROOM ON THE STACK
2785 013414 016616 000002          MOV     2(SP), (SP)   ;SAVE STACK
2786 013420 013546          MOV     @2(RS)+, -(SP) ;GET TIME VALUE
2787 013422 011666 000004          MOV     (SP), 4(SP)  ;MOVE TIME VALUE
2788 013426 006216          ASR     (SP)          ;DIVIDE BY 2
2789 013430 006216          ASR     (SP)          ;DIVIDE BY 2 AGAIN (FOR A TOTAL OF 4)
2790 013432 062666 000002          ADD     (SP)+, 2(SP) ;CALCULATE UPPER LIMIT FOR TIMEOUT
2791 013436 000205          RTS      RS         ;RETURN WITH TOLERANCES ON THE STACK
2792
2793          ;;*****
2794
2795          .SBTTL 'SYSMAC' UTILITY ROUTINES
2796
2797          .SBTTL SCOPE HANDLER ROUTINE
2798
2799          ;;*****
2800          ;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2801          ;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG. (DISPLAY<7:0>)
2802          ;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2803          ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2804          ;$SW14=1      LOOP ON TEST
2805          ;$SW11=1      INHIBIT ITERATIONS
2806          ;$CALL
2807          ;$      SCOPE          ;;SCOPE=IOT
2808
2809          $$SCOPE:
2810 013440 104407          CKSWR
2811 013442 032777 040000 165470 1$:    BIT     @BIT14, @SWR  ;:TEST FOR CHANGE IN SOFT-SWR
2812 013450 001055          BNE     $OVER      ;:LOOP ON PRESENT TEST?
2813          ;*****START OF CODE FOR THE XOR TESTER***** ;:YES IF SW14=1
2814 013452 000416          $XTSTR: BR     6$   ;:IF RUNNING ON THE "XOR" TESTER CHANGE
2815          ;THIS INSTRUCTION TO A "NOP" (NOP=240)
2816 013454 013746 000004          MOV     @ERRVEC, -(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
2817 013460 012737 013500 000004          MOV     $$, @ERRVEC  ;:SET FOR TIMEOUT
2818 013466 005737 177060          TST     @177060     ;:TIME OUT ON XOR?
2819 013472 012637 000004          MOV     (SP)+, @ERRVEC ;:RESTORE THE ERROR VECTOR
2820 013476 000436          BR     $$VLAD      ;:GO TO THE NEXT TEST
2821 013500 022626          $$:    CMP     (SP)+, (SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
2822 013502 012637 000004          MOV     (SP)+, @ERRVEC ;:RESTORE THE ERROR VECTOR
2823 013506 000436          BR     $OVER      ;:LOOP ON THE PRESENT TEST
2824 013510          6$: ;*****END OF CODE FOR THE XOR TESTER*****
2825 013510 105737 001103          2$:    TST     $ERFLG ;:HAS AN ERROR OCCURRED?

```

```

2826 013514 001404          BEQ      3$          ;; BR IF NO
2827 013516 105037 001103  4$: CLR      SERFL     ;; ZERO THE ERROR FLAG
2828 013522 005037 001176          CLR      STIMES     ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
2829 013526 032777 004000 165404 3$: BIT      #BIT11, @SWR  ;; INHIBIT ITERATIONS?
2830 013534 001011          BNE      IS          ;; BR IF YES
2831 013536 005737 001100          TST      $PASS      ;; IF FIRST PASS OF PROGRAM
2832 013542 001406          BEQ      IS          ;; INHIBIT ITERATIONS
2833 013544 005237 001104          INC      $ICNT      ;; INCREMENT ITERATION COUNT
2834 013550 023737 001176 001104  CMP      STIMES, $ICNT  ;; CHECK THE NUMBER OF ITERATIONS MADE
2835 013556 002012          BGE      $OVER      ;; BR IF MORE ITERATION REQUIRED
2836 013560 012737 000001 001104 1$: MOV      #1, $ICNT  ;; REINITIALIZE THE ITERATION COUNTER
2837 013566 013737 013620 001176  MOV      $MXCNT, STIMES ;; SET NUMBER OF ITERATIONS TO DO
2838 013574 105237 001102  $SVLAD: INCB     $STNM    ;; COUNT TEST NUMBERS
2839 013600 011637 001106          MOV      (SP), $LPADR  ;; SAVE SCOPE LOOP ADDRESS
2840 013604 013777 001102 165330 $OVER: MOV      $STNM, @DISPLAY  ;; DISPLAY TEST NUMBER
2841 013612 013716 001106          MOV      $LPADR, (SP)  ;; FUDGE RETURN ADDRESS
2842 013616 000002          RTI              ;; FIXES PS
2843 013620 000004  $MXCNT: 4          ;; MAX. NUMBER OF ITERATIONS
2844          .SBTTL  ERROR HANDLER ROUTINE
2845
2846          ; *****
2847          ; THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2848          ; SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2849          ; AND GO TO $ERRTYP ON ERROR
2850          ; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2851          ; $SW15=1  HALT ON ERROR
2852          ; $SW13=1  INHIBIT ERROR TYPEOUTS
2853          ; $SW10=1  BELL ON ERROR
2854          ; *CALL
2855          ; * ERROR N ;; ERROR=EMT AND N=ERROR ITEM NUMBER
2856
2857 013622          $ERROR:
2858 013622 104407          CKSWR          ;; TEST FOR CHANGE IN SOFT-SWR
2859 013624 113737 001102 001242  MOVB     $STNM, TSTNUM
2860 013632 105237 001103  7$: INCB     $ERFLG     ;; SET THE ERROR FLAG
2861 013636 001775          BEQ      7$          ;; DON'T LET THE FLAG GO TO ZERO
2862 013640 013777 001102 165274  MOV      $STNM, @DISPLAY  ;; DISPLAY TEST NUMBER AND ERROR FLAG
2863 013646 032777 002000 165264  BIT      #BIT10, @SWR  ;; BELL ON ERROR?
2864 013654 001402          BEQ      IS          ;; NO - SKIP
2865 013656 104401 001202          TYPE     $BELL      ;; RING BELL
2866 013662 005237 001112  1$: INC      $ERTTL    ;; COUNT THE NUMBER OF ERRORS
2867 013666 011637 001116          MOV      (SP), $ERRPC  ;; GET ADDRESS OF ERROR INSTRUCTION
2868 013672 162737 000002 001116  SUB      #2, $ERRPC
2869 013700 117737 165212 001114  MOVB     @ERRPC, $ITEMB  ;; STRIP AND SAVE THE ERROR ITEM CODE
2870 013706 032777 020000 165224  BIT      #BIT13, @SWR  ;; SKIP TYPEOUT IF SET
2871 013714 001004          BNE      20$        ;; SKIP TYPEOUTS
2872 013716 004737 013754          JSR     PC, $ERRTYP  ;; GO TO USER ERROR ROUTINE
2873 013722 104401 001207          TYPE     , $CRLF
2874 013726          20$:
2875 013726 005777 165206  2$: TST      @SWR      ;; HALT ON ERROR
2876 013732 100002          BPL      3$          ;; SKIP IF CONTINUE
2877 013734 000000          HALT          ;; HALT ON ERROR!
2878 013736 104407          CKSWR          ;; TEST FOR CHANGE IN SOFT-SWR
2879 013740          3$:
2880 013740 022737 013206 000042  CMP      #ENDAD, @#42  ;; ACT-11 AUTO-ACCEPT?
2881 013746 001001          BNE      6$          ;; BRANCH IF NO

```

2882 013750 000000
 2883 013752
 2884 013752 000002
 2885
 2886
 2887
 2888
 2889
 2890
 2891
 2892 013754
 2893 013754 104401 001207
 2894 013760 010046
 2895 013762 005000
 2896 013764 153700 001114
 2897 013770 001004
 2898
 2899 013772 013746 001116
 2900
 2901 013776 104402
 2902 014000 000445
 2903 014002 005300
 2904 014004 006300
 2905 014006 006300
 2906 014010 006300
 2907 014012 062700 001276
 2908 014016 012037 014026
 2909 014022 001404
 2910 014024 104401
 2911 014026 000000
 2912 014030 104401 001207
 2913 014034 012037 014044
 2914 014040 001404
 2915 014042 104401
 2916 014044 000000
 2917 014046 104401 001207
 2918 014052 010146
 2919 014054 012001
 2920 014056 001415
 2921 014060 012000
 2922 014062 105720
 2923 014064 001003
 2924 014066 013146
 2925 014070 104402
 2926 014072 000402
 2927 014074
 2928 014074 013146
 2929 014076 104405
 2930 014100 005711
 2931 014102 001403
 2932 014104 104401 014124
 2933 014110 000764
 2934
 2935 014112 012601
 2936 014114 012600
 2937 014116 104401 001207

```

    HALT                ;; YES
6$:
    RTI                 ;; RETURN
.SBTTL  ERROR MESSAGE TIMEOUT ROUTINE

;*****
; *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
; *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
; *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
;*****
$ERRTYP:
    TYPE      $SCRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
    MOV       R0,-(SP)    ;; SAVE R0
    CLR       R0          ;; PICKUP THE ITEM INDEX
    BISB     @($ITEMB,R0
    BNE      1$          ;; IF ITEM NUMBER IS ZERO, JUST
                        ;; TYPE THE PC OF THE ERROR
    MOV       $ERRPC,-(SP) ;; SAVE $ERRPC FOR TIMEOUT
                        ;; ERROR ADDRESS
    TYPDC    BR          10$  ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
                        ;; GET OUT
    1$:      DEC         R0  ;; ADJUST THE INDEX SO THAT IT WILL
                        ;; WORK FOR THE ERROR TABLE
    ASL      R0
    ASL      R0
    ASL      R0
    ADD      @($ERRTB,R0  ;; FORM TABLE POINTER
    MOV      (R0)+,$     ;; PICKUP "ERROR MESSAGE" POINTER
    BEQ     3$          ;; SKIP TIMEOUT IF NO POINTER
    TYPE    "ERROR MESSAGE"
                        ;; TYPE THE "ERROR MESSAGE"
    2$:      .WORD      0   ;; "ERROR MESSAGE" POINTER GOES HERE
    TYPE    $SCRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
    3$:      MOV       (R0)+,$  ;; PICKUP "DATA HEADER" POINTER
    BEQ     5$          ;; SKIP TIMEOUT IF 0
    TYPE    "DATA HEADER"
                        ;; TYPE THE "DATA HEADER"
    4$:      .WORD      0   ;; "DATA HEADER" POINTER GOES HERE
    TYPE    $SCRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
    5$:      MOV       R1,-(SP)  ;; SAVE R1
    MOV      (R0)+,R1   ;; PICKUP "DATA TABLE" POINTER
    BEQ     9$          ;; BR IF NO DATA TO BE TYPED
    MOV      (R0)+,R0   ;; PICKUP "DATA FORMAT" POINTER
    6$:      TSTB     (R0)+     ;; "OCTAL" OR "DECIMAL"
    BNE     7$          ;; BR IF DECIMAL
    MOV      @ (R1)+,-(SP) ;; SAVE @ (R1)+ FOR TIMEOUT
    TYPDC    BR          8$  ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
    7$:      MOV      @ (R1)+,-(SP) ;; SAVE @ (R1)+ FOR TIMEOUT
    TYPDC    BR          8$  ;; GO TYPE--DECIMAL ASCII WITH SIGN
    8$:      TST      (R1)     ;; IS THERE ANOTHER NUMBER?
    BEQ     9$          ;; BR IF NO
    TYPE    "  "        ;; TYPE TWO(2) SPACES
    BR      6$         ;; LOOP
    9$:      MOV      (SP)+,R1  ;; RESTORE R1
    10$:     MOV      (SP)+,R0  ;; RESTORE R0
    TYPE    $SCRLF      ;; "CARRIAGE RETURN" & "LINE FEED"

```

```

2938 014122 000207          RTS      PC          ;;RETURN
2939 014124 020040 000     11$: .ASCIZ  / /      ;;TWO(2) SPACES
2940          014130          .EVEN
2941          .SBTTL  TYPE ROUTINE
2942
2943          ;*****
2944          ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2945          ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2946          ;NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2947          ;NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2948          ;NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2949          ;
2950          ;CALL:
2951          ;*1) USING A TRAP INSTRUCTION
2952          ;*      TYPE      ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2953          ;*OR
2954          ;*      TYPE
2955          ;*      MESADR
2956          ;*
2957
2958 014130 105737 001157     $TYPE:  TSTB     $TFPLG          ;; IS THERE A TERMINAL?
2959 014134 100002          BPL      1$          ;; BR IF YES
2960 014136 000000          HALT          ;; HALT HERE IF NO TERMINAL
2961 014140 000407          BR      3$          ;; LEAVE
2962 014142 010046     1$:  MOV      RO, -(SP)          ;; SAVE RO
2963 014144 017600 000002     2$:  MOV      22(SP), RO          ;; GET ADDRESS OF ASCIZ STRING
2964 014150 112046     2$:  MOVVB   (RO)+, -(SP)          ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2965 014152 001005          BNE     4$          ;; BR IF IT ISN'T THE TERMINATOR
2966 014154 005726          TST     (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
2967 014156 012600     6$:  MOV      (SP)+, RO          ;; RESTORE RO
2968 014160 062716 000002     3$:  ADD      #2, (SP)          ;; ADJUST RETURN PC
2969 014164 000002          RTI          ;; RETURN
2970 014166 122716 000011     4$:  CMPB    #HT, (SP)          ;; BRANCH IF <HT>
2971 014172 001430          BEQ     8$          ;;
2972 014174 122716 000200          CMPB    #CRLF, (SP)          ;; BRANCH IF NOT <CRLF>
2973 014200 001006          BNE     5$          ;;
2974 014202 005726          TST     (SP)+          ;; POP <CR><LF> EQUIV
2975 014204 104401          TYPE          ;; TYPE A CR AND LF
2976 014206 001207          $CRLF
2977 014210 105037 014344          CLRB    $CHARCNT          ;; CLEAR CHARACTER COUNT
2978 014214 000755          BR      2$          ;; GET NEXT CHARACTER
2979 014216 004737 014300     5$:  JSR     PC, $TYPEC          ;; GO TYPE THIS CHARACTER
2980 014222 123726 001156     6$:  CMPB    $FILLC, (SP)+          ;; IS IT TIME FOR FILLER CHARS.?
2981 014226 001350          BNE     2$          ;; IF NO GO GET NEXT CHAR.
2982 014230 013746 001154          MOV     $NULL, -(SP)          ;; GET # OF FILLER CHARS. NEEDED
2983          AND     THE NULL CHAR.
2984 014234 105366 000001     7$:  DECB    1(SP)          ;; DOES A NULL NEED TO BE TYPED?
2985 014240 002770          BLT     6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
2986 014242 004737 014300          JSR     PC, $TYPEC          ;; GO TYPE A NULL
2987 014246 105337 014344          DECB    $CHARCNT          ;; DO NOT COUNT AS A COUNT
2988 014252 000770          BR      7$          ;; LOOP
2989
2990          ;HORIZONTAL TAB PROCESSOR
2991
2992 014254 112716 000040     8$:  MOVVB   #' , (SP)          ;; REPLACE TAB WITH SPACE
2993 014260 004737 014300     9$:  JSR     PC, $TYPEC          ;; TYPE A SPACE

```



```

2994 014264 132737 000007 014344 BITB #7, $CHARCNT ;; BRANCH IF NOT AT
2995 014272 001372 BNE 95 ;; TAB STOP
2996 014274 005726 TST (SP)+ ;; POP SPACE OFF STACK
2997 014276 000724 BR 25 ;; GET NEXT CHARACTER
2998 014300 105777 164644 $TYPEC: TSTB #STPB ;; WAIT UNTIL PRINTER IS READY
2999 014304 100375 BPL $TYPEC
3000 014306 116677 000002 164636 MOVB 2(SP), #STPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
3001 014314 122766 000015 000002 CMPB #CR, 2(SP) ;; IS CHARACTER A CARRIAGE RETURN?
3002 014322 001003 BNE 15 ;; BRANCH IF NO
3003 014324 105037 014344 CLRB $CHARCNT ;; YES--CLEAR CHARACTER COUNT
3004 014330 000406 BR $TYPEX ;; EXIT
3005 014332 122766 000012 000002 15: CMPB #LF, 2(SP) ;; IS CHARACTER A LINE FEED?
3006 014340 001402 BEQ $TYPEX ;; BRANCH IF YES
3007 014342 105227 INCB (PC)+ ;; COUNT THE CHARACTER
3008 014344 000000 $CHARCNT: WORD 0 ;; CHARACTER COUNT STORAGE
3009 014346 000207 $TYPEX: RTS PC
    
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

3010
3011
3012
3013 *****
3014 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3015 *OCTAL (ASCII) NUMBER AND TYPE IT.
3016 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3017 *CALL:
3018 * MOV NUM, -(SP) ;; NUMBER TO BE TYPED
3019 * TYPOS ;; CALL FOR TYPEOUT
3020 * .BYTE N ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3021 * .BYTE M ;; M=1 OR 0
3022 * ;; 1=TYPE LEADING ZEROS
3023 * ;; 0=SUPPRESS LEADING ZEROS
3024
3025 *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3026 *$TYPOS OR $TYPOC
3027 *CALL:
3028 * MOV NUM, -(SP) ;; NUMBER TO BE TYPED
3029 * TYPON ;; CALL FOR TYPEOUT
3030
3031 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3032 *CALL:
3033 * MOV NUM, -(SP) ;; NUMBER TO BE TYPED
3034 * TYPOC ;; CALL FOR TYPEOUT
3035
3036 014350 017646 000000 $TYPOS: MOV 2(SP), -(SP) ;; PICKUP THE MODE
3037 014354 116637 000001 014573 MOVB 1(SP), $OFILL ;; LOAD ZERO FILL SWITCH
3038 014362 112637 014575 MOVB (SP)+, $OMODE+1 ;; NUMBER OF DIGITS TO TYPE
3039 014366 062716 000002 ADD #2, (SP) ;; ADJUST RETURN ADDRESS
3040 014372 000406 BR $TYPON
3041 014374 112737 000001 014573 $TYPOC: MOVB #1, $OFILL ;; SET THE ZERO FILL SWITCH
3042 014402 112737 000006 014575 MOVB #6, $OMODE+1 ;; SET FOR SIX(6) DIGITS
3043 014410 112737 000005 014572 $TYPON: MOVB #5, $OCNT ;; SET THE ITERATION COUNT
3044 014416 010346 MOV R3, -(SP) ;; SAVE R3
3045 014420 010446 MOV R4, -(SP) ;; SAVE R4
3046 014422 010546 MOV R5, -(SP) ;; SAVE R5
3047 014424 113704 014575 MOVB $OMODE+1, R4 ;; GET THE NUMBER OF DIGITS TO TYPE
3048 014430 005404 NEG R4
3049 014432 062704 000006 ADD #6, R4 ;; SUBTRACT IT FOR MAX. ALLOWED
    
```

F05

```

3050 014436 110437 014574      MOVB   R4,$OMODE      ;;SAVE IT FOR USE
3051 014442 113704 014573      MOVB   $OFILL,R4      ;;GET THE ZERO FILL SWITCH
3052 014446 016605 000012      MOV    12(SP),R5      ;;PICKUP THE INPUT NUMBER
3053 014452 005003                CLR    R3              ;;CLEAR THE OUTPUT WORD
3054 014454 006105      1$:   ROL    R5          ;;ROTATE MSB INTO "C"
3055 014456 000404                BR     3$              ;;GO DO MSB
3056 014460 006105      2$:   ROL    R5          ;;FORM THIS DIGIT
3057 014462 006105                ROL    R5
3058 014464 006105                ROL    R5
3059 014466 010503                MOV    R5,R3
3060 014470 006103      3$:   ROL    R3          ;;GET LSB OF THIS DIGIT
3061 014472 105337 014574      DECB   $OMODE          ;;TYPE THIS DIGIT?
3062 014476 100016                BPL    7$              ;;BR IF NO
3063 014500 042703 177770      BIC    #177770,R3      ;;GET RID OF JUNK
3064 014504 001002                BNE    4$              ;;TEST FOR 0
3065 014506 005704                TST    R4              ;;SUPPRESS THIS 0?
3066 014510 001403                BEQ    5$              ;;BR IF YES
3067 014512 005204      4$:   INC    R4          ;;DON'T SUPPRESS ANYMORE 0'S
3068 014514 052703 000060      BIS    #'0,R3          ;;MAKE THIS DIGIT ASCII
3069 014520 052703 000040      BIS    #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3070 014524 110337 014570      MOVB   R3,$$          ;;SAVE FOR TYPING
3071 014530 104401 014570                TYPE   $$              ;;GO TYPE THIS DIGIT
3072 014534 105337 014572      7$:   DECB   $OCNT          ;;COUNT BY 1
3073 014540 003347                BGT    2$              ;;BR IF MORE TO DO
3074 014542 002402                BLT    6$              ;;BR IF DONE
3075 014544 005204                INC    R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3076 014546 000744                BR     2$              ;;GO DO THE LAST DIGIT
3077 014550 012605      6$:   MOV    (SP)+,R5      ;;RESTORE R5
3078 014552 012604                MOV    (SP)+,R4      ;;RESTORE R4
3079 014554 012603                MOV    (SP)+,R3      ;;RESTORE R3
3080 014556 016666 000002 000004      MOV    2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
3081 014564 012616                MOV    (SP)+,(SP)
3082 014566 000002                RTI                    ;;RETURN
3083 014570                8$:   .BYTE 0          ;;STORAGE FOR ASCII DIGIT
3084 014571                .BYTE 0          ;;TERMINATOR FOR TYPE ROUTINE
3085 014572                .BYTE 0          ;;OCTAL DIGIT COUNTER
3086 014573                .BYTE 0          ;;ZERO FILL SWITCH
3087 014574 000000      $OMODE: .WORD 0      ;;NUMBER OF DIGITS TO TYPE
3088                .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3089
3090                ;;*****
3091                ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
3092                ;;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
3093                ;;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
3094                ;;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
3095                ;;REPLACED WITH SPACES.
3096                ;;CALL:
3097                ;;      MOV    NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
3098                ;;      TYPDS                ;;GO TO THE ROUTINE
3099
3100                $TYPDS:
3101                MOV    R0,-(SP)      ;;PUSH R0 ON STACK
3102                MOV    R1,-(SP)      ;;PUSH R1 ON STACK
3103                MOV    R2,-(SP)      ;;PUSH R2 ON STACK
3104                MOV    R3,-(SP)      ;;PUSH R3 ON STACK
3105                MOV    R5,-(SP)      ;;PUSH R5 ON STACK
    
```

G05

MD-11-DZRMH-A, RM03 DUAL CONTRLR LOGIC TEST - PART 2
 DZRMH.A.P11 01-AUG-77 11:02

MACY11 30(1046) 01-AUG-77 11:14 PAGE 58
 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0059

```

3106 014610 012746 020200      MOV      #20200,-(SP)      ;; SET BLANK SWITCH AND SIGN
3107 014614 016605 000020      MOV      20(SP),R5      ;; GET THE INPUT NUMBER
3108 014620 100004      BPL      1$             ;; BR IF INPUT IS POS.
3109 014622 005405      NEG      R5             ;; MAKE THE BINARY NUMBER POS.
3110 014624 112766 000055 000001      MOVVB   #'-,1(SP)      ;; MAKE THE ASCII NUMBER NEG.
3111 014632 005000      CLR      R0             ;; ZERO THE CONSTANTS INDEX
3112 014634 012703 015012      MOV      #SDBLK,R3      ;; SETUP THE OUTPUT POINTER
3113 014640 112723 000040      MOVVB   #' ,(R3)+      ;; SET THE FIRST CHARACTER TO A BLANK
3114 014644 005002      CLR      R2             ;; CLEAR THE BCD NUMBER
3115 014646 016001 015002      MOV      SOTBL(R0),R1   ;; GET THE CONSTANT
3116 014652 160105      SUB      R1,R5          ;; FORM THIS BCD DIGIT
3117 014654 002402      BLT      4$             ;; BR IF DONE
3118 014656 005202      INC      R2             ;; INCREASE THE BCD DIGIT BY 1
3119 014660 000774      BR       3$
3120 014662 060105      ADD      R1,R5          ;; ADD BACK THE CONSTANT
3121 014664 005702      TST      R2             ;; CHECK IF BCD DIGIT=0
3122 014666 001002      BNE      5$             ;; FALL THROUGH IF 0
3123 014670 105716      TSTB    (SP)           ;; STILL DOING LEADING 0'S?
3124 014672 100407      BMI      7$             ;; BR IF YES
3125 014674 106316      ASLB    (SP)           ;; MSD?
3126 014676 103003      BCC      6$             ;; BR IF NO
3127 014700 116663 000001 177777      MOVVB   1(SP),-1(R3)   ;; YES--SET THE SIGN
3128 014706 052702 000060      BIS      #'0,R2        ;; MAKE THE BCD DIGIT ASCII
3129 014712 052702 000040      BIS      #' ,R2        ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
3130 014716 110223      MOVVB   R2,(R3)+      ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
3131 014720 005720      TST      (R0)+        ;; JUST INCREMENTING
3132 014722 020027 000010      CMP      R0,#10       ;; CHECK THE TABLE INDEX
3133 014726 002746      BLT      2$             ;; GO DO THE NEXT DIGIT
3134 014730 003002      BGT      8$             ;; GO TO EXIT
3135 014732 010502      MOV      R5,R2        ;; GET THE LSD
3136 014734 000764      BR       6$             ;; GO CHANGE TO ASCII
3137 014736 105726      TSTB    (SP)+        ;; WAS THE LSD THE FIRST NON-ZERO?
3138 014740 100003      BPL      9$             ;; BR IF NO
3139 014742 116663 177777 177776      MOVVB   -1(SP),-2(R3)  ;; YES--SET THE SIGN FOR TYPING
3140 014750 105013      CLRB    (R3)          ;; SET THE TERMINATOR
3141 014752 012605      MOV      (SP)+,R5      ;; POP STACK INTO R5
3142 014754 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
3143 014756 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
3144 014760 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
3145 014762 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
3146 014764 104401 015012      TYPE    SDBLK         ;; NOW TYPE THE NUMBER
3147 014770 016666 000002 000004      MOV      2(SP),4(SP)   ;; ADJUST THE STACK
3148 014776 012616      MOV      (SP)+,(SP)
3149 015000 000002      RTI
3150 015002 023420      SOTBL: 10000.
3151 015004 001750      1000.
3152 015006 000144      100.
3153 015010 000012      10.
3154 015012 000004      SDBLK: .BLKW 4
3155      .SBTTL TTY INPUT ROUTINE
3156
3157      ;; *****
3158      .ENABL LSB
3159 015022 000000      STKCNT: .WORD 0      ;; NUMBER OF ITEMS IN QUEUE
3160 015024 000000      STKQIN: .WORD 0     ;; INPUT POINTER
3161 015026 000000      STKQOUT: .WORD 0    ;; OUTPUT POINTER
    
```

```

3162 015030 000001      $TKQSRT: .BLKB 1      ;; TTY KEYBOARD QUEUE
3163                015031      $TKQEND=.
3164                015032      .EVEN
3165
3166                ;*TK INITIALIZE ROUTINE
3167                ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
3168                ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
3169
3170                ;*CALL:
3171                *      JSR      PC,$TKINT
3172                *      RETURN
3173
3174 015032 005037 015022  $TKINT: CLR      $TKCNT      ;; CLEAR COUNT OF ITEMS IN QUEUE
3175 015036 012737 015030 015024  MOV      $TKQSRT,$TKQIN  ;; MOVE THE STARTING ADDRESS OF THE
3176 015044 013737 015024 015026  MOV      $TKQIN,$TKQOUT  ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
3177 015052 012737 015102 000060  MOV      $TKSRV,$TKVEC  ;; INITIALIZE THE KEYBOARD VECTOR
3178 015060 012737 000200 000062  MOV      #200,$TKVEC+2  ;; "BR" LEVEL 4
3179 015066 005777 164054      TST      $TKB          ;; CLEAR DONE FLAG
3180 015072 012777 000100 164044  MOV      #100,$TKS      ;; ENABLE TTY KEYBOARD INTERRUPT
3181 015100 000207      RTS      PC          ;; RETURN TO CALLER
3182
3183                ;*TK SERVICE ROUTINE
3184                ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
3185                ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
3186                ;*IT IN THE QUEUE.
3187
3188 015102 117746 164040  $TKSRV: MOVB     $TKB,-(SP)  ;; PICKUP THE CHARACTER
3189 015106 042716 177600      BIC      #177,(SP)      ;; STRIP THE JUNK
3190 015112 021627 000007 1$:    CMP      (SP),#7      ;; IS IT A CONTROL G?
3191 015116 001004      BNE      2$          ;; BRANCH IF NO
3192 015120 022737 000176 001140  CMP      $SWREG,$SWR    ;; IS SOFT-SWR SELECTED?
3193 015126 001500      BEQ      6$          ;; GO TO SWR CHANGE
3194
3195 015130 2$:
3196 015130 022737 000001 015022  CMP      #1,$TKCNT      ;; IS THE QUEUE FULL?
3197 015136 001004      BNE      3$          ;; BRANCH IF NO
3198 015140 104401 001202      TYPE     $BELL          ;; RING THE TTY BELL
3199 015144 005726      TST      (SP)+         ;; CLEAN CHARACTER OFF OF STACK
3200 015146 000451      BR       5$          ;; EXIT
3201 015150 021627 000023 3$:    CMP      (SP),#23      ;; IS IT A CONTROL-S?
3202 015154 001021      BNE      32$         ;; BRANCH IF NO
3203 015156 005077 163762      CLR      $TKS          ;; DISABLE TTY KEYBOARD INTERRUPTS
3204 015162 005726      TST      (SP)+         ;; CLEAN CHAR OFF STACK
3205 015164 105777 163754 31$:   TSTB     $TKS          ;; WAIT FOR A CHAR
3206 015170 100375      BPL      31$         ;; LOOP UNTIL ITS THERE
3207 015172 117746 163750      MOVB     $TKB,-(SP)    ;; GET THE CHARACTER
3208 015176 042716 177600      BIC      #177,(SP)    ;; MAKE IT 7-BIT ASCII
3209 015202 022627 000021      CMP      (SP)+,#21    ;; IS IT A CONTROL-Q?
3210 015206 001366      BNE      31$         ;; BRANCH IF NO
3211 015210 012777 000100 163726  MOV      #100,$TKS    ;; REENABLE TTY KEYBOARD INTERRUPTS
3212 015216 000002      RTI
3213 015220 005237 015022 32$:   INC      $TKCNT      ;; COUNT THIS CHARACTER
3214 015224 021627 000140      CMP      (SP),#140    ;; IS IT UPPER CASE?
3215 015230 002405      BLT
3216 015232 021627 000175      CMP      (SP),#175    ;; IS IT A SPECIAL CHAR?
3217 015236 003002      BGT      4$          ;; BRANCH IF YES

```

```

3218 015240 042716 000040          BIC      #40,(SP)      ;; MAKE IT UPPER CASE
3219 015244 112677 177554          4$:     MOVB     (SP)+,2$TKQIN  ;; AND PUT IT IN QUEUE
3220 015250 005237 015024          INC      $TKQIN      ;; UPDATE THE POINTER
3221 015254 023727 015024 015031  CMP      $TKQIN,#$TKQEND ;; GO OFF THE END?
3222 015262 001003                    BNE      5$          ;; BRANCH IF NO
3223 015264 012737 015030 015024  MOV      #$TKQSR7,$TKQIN ;; RESET THE POINTER
3224 015272 000002          5$:     RTI          ;; RETURN

```

```

3225
3226      ;;*****
3227      ;;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3228      ;;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3229      ;;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
3230      ;;CALL WHEN OPERATING IN TTY INTERRUPT MODE.
3231 015274 022737 000176 001140  $CKSWR: CMP      #SWREG,SWR      ;; IS THE SOFT-SWR SELECTED
3232 015302 001104                    BNE      15$          ;; EXIT IF NOT
3233 015304 105777 163634          TSTB    2$TKS        ;; IS A CHAR WAITING?
3234 015310 100101                    BPL      15$          ;; IF NOT, EXIT
3235 015312 117746 163630          MOVB    2$TKB,-(SP)   ;; YES
3236 015316 042716 177600          BIC     #1C177,(SP)  ;; MAKE IT 7-BIT ASCII
3237 015322 021627 000007          CMP     (SP),#7      ;; IS IT A CONTROL-G?
3238 015326 001300                    BNE      2$          ;; IF NOT, PUT IT IN THE TTY QUEUE
3239
3240
3241      ;;*****
3242      ;;CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
3243      ;;ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
3244      ;;CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
3245 015330 123727 001134 000001  6$:     CMPB    $AUTOB,#1  ;; ARE WE RUNNING IN AUTO-MODE?
3246 015336 001674                    BEQ      2$          ;; BRANCH IF YES
3247 015340 005726                    TST     (SP)+        ;; CLEAR CONTROL-G OFF STACK
3248 015342 004737 015032          JSR     PC,$TKINT    ;; FLUSH THE TTY INPUT QUEUE
3249 015346 005077 163572          CLR     2$TKS        ;; DISABLE TTY KEYBOARD INTERRUPTS
3250 015352 112737 000001 001135  MOVB    #1,$INTAG    ;; SET INTERRUPT MODE INDICATOR
3251
3252 015360 104401 016136          SGT$WR: TYPE    ,SCNTLG  ;; ECHO THE CONTROL-G (1G)
3253 015364 104401 016143          TYPE    $MSWR      ;; TYPE CURRENT CONTENTS
3254 015370 013746 000176          MOV     SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
3255 015374 104402                    TYPCC   $MNEW       ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
3256 015376 104401 016154          TYPE    ,MNEW      ;; PROMPT FOR NEW SWR
3257 015402 005046          19$:    CLR     -(SP)    ;; CLEAR COUNTER
3258 015404 005046          CLR     -(SP)    ;; THE NEW SWR
3259 015406 105777 163532          7$:     TSTB    2$TKS    ;; CHAR THERE?
3260 015412 100375                    BPL      7$          ;; IF NOT TRY AGAIN
3261
3262 015414 117746 163526          MOVB    2$TKB,-(SP) ;; PICK UP CHAR
3263 015420 042716 177600          BIC     #1C177,(SP) ;; MAKE IT 7-BIT ASCII
3264
3265
3266
3267 015424 021627 000025          9$:     CMP     (SP),#25  ;; IS IT A CONTROL-U?
3268 015430 001005                    BNE     10$         ;; BRANCH IF NOT
3269 015432 104401 016131          TYPE    ,SCNTLU    ;; YES, ECHO CONTROL-U (1U)
3270 015436 062706 000006          20$:    ADD     #6,SP   ;; IGNORE PREVIOUS INPUT
3271 015442 000757                    BR      19$        ;; LET'S TRY IT AGAIN
3272
3273

```

```

3274 015444 021627 000015      10$:  CMP      (SP),#15      ;; IS IT A <CR>?
3275 015450 001022              BNE      16$              ;; BRANCH IF NO
3276 015452 005766 000004      TST      4(SP)           ;; YES, IS IT THE FIRST CHAR?
3277 015456 001403              BEQ      11$              ;; BRANCH IF YES
3278 015460 016677 000002 163452  MOV      2(SP),2SWR      ;; SAVE NEW SWR
3279 015466 062706 000006      11$:  ADD      #6,SP           ;; CLEAR UP STACK
3280 015472 104401 001207      14$:  TYPE      $CRLF        ;; ECHO <CR> AND <LF>
3281 015476 123727 001135 000001  CMPB     $INTAG,#1       ;; RE-ENABLE TTY KBD INTERRUPTS?
3282 015504 001003              BNE      15$              ;; BRANCH IF NOT
3283 015506 012777 000100 163430  MOV      #100,2STKS      ;; RE-ENABLE TTY KBD INTERRUPTS
3284 015514 000002      15$:  RTI                    ;; RETURN
3285 015516 004737 014300      16$:  JSR      PC,$TYPEPC     ;; ECHO CHAR
3286 015522 021627 000060      CMP      (SP),#60        ;; CHAR < 0?
3287 015530 002420              BLT      18$              ;; BRANCH IF YES
3288 015534 021627 000067      CMP      (SP),#67        ;; CHAR > 7?
3289 015534 003015              BGT      18$              ;; BRANCH IF YES
3290 015536 042726 000060      BIC      #60,(SP)+        ;; STRIP-OFF ASCII
3291 015542 005766 000002      TST      2(SP)           ;; IS THIS THE FIRST CHAR
3292 015546 001403              BEQ      17$              ;; BRANCH IF YES
3293 015550 006316              ASL      (SP)             ;; NO, SHIFT PRESENT
3294 015552 006316              ASL      (SP)             ;; CHAR OVER TO MAKE
3295 015554 006316              ASL      (SP)             ;; ROOM FOR NEW ONE.
3296 015556 005266 000002      17$:  INC      2(SP)           ;; KEEP COUNT OF CHAR
3297 015562 056616 177776      BIS      -2(SP),(SP)     ;; SET IN NEW CHAR
3298 015566 000707              BR       7$               ;; GET THE NEXT ONE
3299 015570 104401 001206      18$:  TYPE      $QUES        ;; TYPE ?<CR><LF>
3300 015574 000720              BR       20$             ;; SIMULATE CONTROL-U
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311

```

```

*****
; 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

```

```

3312 015576 011646 000004 000002 $RDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC AND
3313 015600 016666 000004              MOV      4(SP),2(SP)     ;; THE PS
3314 015606 005066 000004              CLR      4(SP)           ;; GET READY FOR A CHARACTER
3315 015612 005046              CLR      -(SP)           ;; PUT NEW PS ON STACK
3316 015614 012746 015622              MOV      #64$,-(SP)     ;; PUT NEW PC ON STACK
3317 015620 000002              RTI                    ;; POP NEW PC AND PS
3318 015622
3319 015622 005737 015022      64$:  TST      $STKCNT        ;; WAIT ON A CHARACTER
3320 015626 001775      1$:  BEQ      1$              ;;
3321 015630 005337 015022      DEC      $STKCNT        ;; DECREMENT THE COUNTER
3322 015634 117766 177166 000004      MOVB     2$STKQOUT,4(SP) ;; GET ONE CHARACTER
3323 015642 005237 015026              INC      $STKQOUT        ;; UPDATE THE POINTER
3324 015646 023727 015026 015031      CMP      $STKQOUT,$STKQEND ;; DID IT GO OFF OF THE END?
3325 015654 001003              BNE      2$              ;; BRANCH IF NO
3326 015656 012737 015030 015026      MOV      #2,$STKQRT,$STKQOUT ;; RESET THE POINTER
3327 015664 000002      2$:  RTI                    ;; RETURN
3328
3329

```

```

*****
; THIS ROUTINE WILL INPUT A STRING FROM THE TTY

```

```

3330 ;*CALL:
3331 ;* RDLIN
3332 ;* RETURN HERE
3333 ;*
3334 ;*
3335 015666 010346 SRDLIN: MOV R3, -(SP) ;: SAVE R3
3336 015670 005046 CLR -(SP) ;: CLEAR THE RUBOUT KEY
3337 015672 012703 016122 1$: MOV #STTYIN, R3 ;: GET ADDRESS
3338 015676 022703 016131 2$: CMP #STTYIN+7, R3 ;: BUFFER FULL?
3339 015702 101456 BLOS 4$ ;: BR IF YES
3340 015704 104410 RDCHR ;: GO READ ONE CHARACTER FROM THE TTY
3341 015706 112613 MOV (SP)+, (R3) ;: GET CHARACTER
3342 015710 122713 000177 10$: CMPB #177, (R3) ;: IS IT A RUBOUT
3343 015714 001022 BNE 5$ ;: BR IF NO
3344 015716 005716 TST (SP) ;: IS THIS THE FIRST RUBOUT?
3345 015720 001007 BNE 6$ ;: BR IF NO
3346 015722 112737 000134 016120 MOVB #' \, 9$ ;: TYPE A BACK SLASH
3347 015730 104401 016120 TYPE 9$
3348 015734 012716 177777 MOV #-1, (SP) ;: SET THE RUBOUT KEY
3349 015740 005303 6$: DEC R3 ;: BACKUP BY ONE
3350 015742 020327 016122 CMP R3, #STTYIN ;: STACK EMPTY?
3351 015746 103434 BLO 4$ ;: BR IF YES
3352 015750 111337 016120 MOV (R3), 9$ ;: SETUP TO TYPEOUT THE DELETED CHAR.
3353 015754 104401 016120 TYPE 9$ ;: GO TYPE
3354 015760 000746 2$: BR 2$ ;: GO READ ANOTHER CHAR.
3355 015762 005716 5$: TST (SP) ;: RUBOUT KEY SET?
3356 015764 001406 BEQ 7$ ;: BR IF NO
3357 015766 112737 000134 016120 MOVB #' \, 9$ ;: TYPE A BACK SLASH
3358 015774 104401 016120 TYPE 9$
3359 016000 005016 CLR (SP) ;: CLEAR THE RUBOUT KEY
3360 016002 122713 000025 7$: CMPB #25, (R3) ;: IS CHARACTER A CTRL U?
3361 016006 001003 BNE 8$ ;: BR IF NO
3362 016010 104401 016131 TYPE %CNTLU ;: TYPE A CONTROL "U"
3363 016014 000726 BR 1$ ;: GO START OVER
3364 016016 122713 000022 8$: CMPB #22, (R3) ;: IS CHARACTER A "↑"?
3365 016022 001011 BNE 3$ ;: BRANCH IF NO
3366 016024 105013 CLR (R3) ;: CLEAR THE CHARACTER
3367 016026 104401 001207 TYPE %SCRLF ;: TYPE A "CR" & "LF"
3368 016032 104401 016122 TYPE #STTYIN ;: TYPE THE INPUT STRING
3369 016036 000717 BR 2$ ;: GO PICKUP ANOTHER CHARACTER
3370 016040 104401 001206 4$: TYPE %SQUES ;: TYPE A ' '
3371 016044 000712 BR 1$ ;: CLEAR THE BUFFER AND LOOP
3372 016046 111337 016120 3$: MOV (R3), 9$ ;: ECHO THE CHARACTER
3373 016052 104401 016120 TYPE 9$
3374 016056 122723 000015 CMPB #15, (R3)+ ;: CHECK FOR RETURN
3375 016062 001305 BNE 2$ ;: LOOP IF NOT RETURN
3376 016064 105063 177777 CLR -1(R3) ;: CLEAR RETURN (THE 15)
3377 016070 104401 001210 TYPE %SLF ;: TYPE A LINE FEED
3378 016074 005726 TST (SP)+ ;: CLEAR RUBOUT KEY FROM THE STACK
3379 016076 012603 MOV (SP)+, R3 ;: RESTORE R3
3380 016100 011646 MOV (SP), -(SP) ;: ADJUST THE STACK AND PUT ADDRESS OF THE
3381 016102 016666 000004 000002 MOV 4(SP), 2(SP) ;: FIRST ASCII CHARACTER ON IT
3382 016110 012766 016122 000004 MOV #STTYIN, 4(SP)
3383 016116 000002 RTI ;: RETURN
3384 016120 000 9$: .BYTE 0 ;: STORAGE FOR ASCII CHAR. TO TYPE
3385 016121 000 .BYTE 0 ;: TERMINATOR

```

```

3386 016122 000007 $TTYIN: .BLKB 7 ;;RESERVE 7 BYTES FOR TTY INPUT
3387 016131 136 00525 000012 $CNTLU: .ASCIZ /1U<15><12> ;;CONTROL "U"
3388 016136 043536 005015 000 $CNTLG: .ASCIZ /1G<15><12> ;;CONTROL "G"
3389 016143 015 051412 051127 $MSWR: .ASCIZ <15><12>/SWR = /
3390 016150 036440 000040
3391 016154 020040 042516 020127 $MNEW: .ASCIZ / NEW = /
3392 016162 020075 000
3393
3394 .EVEN
3395 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
3396
3397 *****
3398 *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
3399 *CHANGE IT TO BINARY.
3400 *THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
3401 *OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
3402 *FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
3403 *THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
3404 *CALL:
3405 * RDOCT ;;READ AN OCTAL NUMBER
3406 * RETURN HERE ;;LOW ORDER BITS ARE ON TOP OF THE STACK
3407 * ;;HIGH ORDER BITS ARE IN $HI OCT
3408 016166 011646 $RDOCT: MOV (SP),-(SP) ;;PROVIDE SPACE FOR THE
3409 016170 016666 000004 000002 MOV 4(SP),2(SP) ;;INPUT NUMBER
3410 016176 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
3411 016200 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
3412 016202 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
3413 016204 104411 1$: RDLIN ;;READ AN ASCIZ LINE
3414 016206 012600 MOV (SP)+,RO ;;GET ADDRESS OF 1ST CHARACTER
3415 016210 010037 016314 MOV RO,$$ ;;AND SAVE IT
3416 016214 005001 CLR R1 ;;CLEAR DATA WORD
3417 016216 005002 CLR R2
3418 016220 112046 2$: MOVB (RO)+,-(SP) ;;PICKUP THIS CHARACTER
3419 016222 001420 BEQ 3$ ;;IF ZERO GET OUT
3420 016224 122716 000060 CMPB #'0,(SP) ;;MAKE SURE THIS CHARACTER
3421 016230 003026 BGT 4$ ;;IS AN OCTAL DIGIT
3422 016232 122716 000067 CMPB #'7,(SP)
3423 016236 002423 BLT 4$
3424 016240 006301 ASL R1 ;;*2
3425 016242 006102 ROL R2
3426 016244 006301 ASL R1 ;;*4
3427 016246 006102 ROL R2
3428 016250 006301 ASL R1 ;;*8
3429 016 2 006102 ROL R2
3430 016 34 042716 177770 BIC #'C7,(SP) ;;STRIP THE ASCII JUNK
3431 016 50 062601 ADD (SP)+,R1 ;;ADD IN THIS DIGIT
3432 016 52 000756 BR 2$ ;;LOOP
3433 016 54 005726 3$: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
3434 016 56 010166 000012 MOV R1,12(SP) ;;SAVE THE RESULT
3435 016272 010237 016324 MOV R2,$HI OCT
3436 016276 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
3437 016300 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
3438 016302 012600 MOV (SP)+,RO ;;POP STACK INTO RO
3439 016304 000002 RTI ;;RETURN
3440 016306 005726 4$: TST (SP)+ ;;CLEAN PARTIAL FROM STACK
3441 016310 105010 CLRB (RO) ;;SET A TERMINATOR

```


M05

3142 016312 104401
3143 016314 000000
3144 016316 104401 001206
3145 016322 000730
3146 016324 000000

;;TYPE UP THRU THE BAD CHAR.
SS: .WORD 0
TYPE \$QUES
BR 1S
SHIOCT: .WORD 0
.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

*SAVE RO-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

3164 016326
3165 016326 010046
3166 016330 010146
3167 016332 010246
3168 016334 010346
3169 016336 010446
3170 016340 010546
3171 016342 016646 000022
3172 016346 016646 000022
3173 016352 016646 000022
3174 016356 016646 000022
3175 016362 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

3177
3178
3179
3180 016364
3181 016364 012666 000022
3182 016370 012666 000022
3183 016374 012666 000022
3184 016400 012666 000022
3185 016404 012605
3186 016406 012604
3187 016410 012603
3188 016412 012602
3189 016414 012601
3190 016416 012600
3191 016420 000002

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

.SBTTL TRAP DECODER

3192
3193
3194
3195
3196
3197

*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

TRAP DECODER
; *GO TO THAT ROUTINE.

3498
3499
3500 016422 010046
3501 016424 016600 000002
3502 016430 005740
3503 016432 111000
3504 016434 006300
3505 016436 016000 016456
3506 016442 000200

\$TRAP: MOV RO, -(SP) ;: SAVE RO
MOV 2(SP), RO ;: GET TRAP ADDRESS
TST -(RO) ;: BACKUP BY 2
M'VB (RO), RO ;: GET RIGHT BYTE OF TRAP
ASL RO ;: POSITION FOR INDEXING
MOV \$TRPAD(RO), RO ;: INDEX TO TABLE
RTS RO ;: GO TO ROUTINE

;: THIS IS USE TO HANDLE THE "GETPRI" MACRO

3507
3508
3509
3510
3511 016444 011646
3512 016446 016666 000004 000002
3513 016454 000002

\$TRAP2: MOV (SP), -(SP) ;: MOVE THE PC DOWN
MOV 4(SP), 2(SP) ;: MOVE THE PSW DOWN
RTI ;: RESTORE THE PSW

.SBTTL TRAP TABLE

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

3514
3515
3516
3517
3518
3519
3520
3521
3522 016456 016444
3523 016460 014130
3524 016462 014374
3525 016464 014350
3526 016466 014410
3527 016470 014576
3528
3529 016472 015364
3530
3531 016474 015274
3532 016476 015576
3533 016500 015666
3534 016502 016166
3535 016504 016326
3536 016506 016364

: ROUTINE
:-----
\$TRPAD: .WORD \$TRAP2
\$TYPE ;: CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
\$TYPOC ;: CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS ;: CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON ;: CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS ;: CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

\$GTSWR ;: CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING

\$CKSWR ;: CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
\$RDCHR ;: CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
\$RDLIN ;: CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
\$RDOCT ;: CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
\$SAVREG ;: CALL=SAVREG TRAP+13(104413) SAVE RO-R5 ROUTINE
\$RESREG ;: CALL=RESREG TRAP+14(104414) RESTORE RO-R5 ROUTINE

;: *****

.SBTTL TELETYPE MESSAGES

;: *****

3544 016510 005015 046412 044501
3545 016516 042116 041505 030455
3546 016524 026461 055104 045122
3547 016532 026506 006501 012
3548 016537 122 030115 020063
3549 016544 052504 046101 041440
3550 016552 047117 051124 046117
3551 016560 042514 020122 047514
3552 016566 044507 020103 042524
3553 016574 052123 026440 050040

TITLE: .ASCII <CR><LF><LF>/MAINDEC-11-DZRJF-A/<CR><LF>

.ASCIIZ RMO3 DUAL CONTROLLER LOGIC TEST - PART 2<CR><LF><LF>

3554	016602	051101	020124	006462	
3555	016610	005012	000		
3556	016613	015	042412	052116	ENTERA: .ASCIZ <CR><LF>/ENTER DRIVE ADDRESS: /
3557	016620	051105	042040	044522	
3558	016626	042526	040440	042104	
3559	016634	042522	051523	020072	
3560	016642	000			
3561	016643	111	053116	046101	ADRERR: .ASCIZ /INVALID ADDRESS/<CR><LF>
3562	016650	042111	040440	042104	
3563	016656	042522	051523	005015	
3564	016664	000			
3565	016665	015	050012	051117	PORTAIS: .ASCIZ <CR><LF>/PORT A ADDRESS IS: /
3566	016672	020124	020101	042101	
3567	016700	051104	051505	020123	
3568	016706	051511	020072	000	
3569	016713	015	050012	051117	PORTBIS: .ASCIZ <CR><LF>/PORT B ADDRESS IS: /
3570	016720	020124	020102	042101	
3571	016726	051104	051505	020123	
3572	016734	051511	020072	000	
3573	016741	015	051412	051531	NOCLOCK: .ASCIZ <CR><LF>/SYSTEM MUST HAVE 'L' OR 'P' CLOCK/<CR><LF><LF>
3574	016746	042524	020115	052515	
3575	016754	052123	044040	053101	
3576	016762	020105	046047	020047	
3577	016770	051117	023440	023520	
3578	016776	041440	047514	045503	
3579	017004	005015	000012		
3580	017010	042412	052116	051105	TESTNO: .ASCIZ <LF>/ENTER TEST #: /
3581	017016	052040	051505	020124	
3582	017024	035043	000040		
3583	017030	047111	040526	044514	BADNO: .ASCIZ /INVALID TEST NUMBER/<CR><LF>
3584	017036	020104	042524	052123	
3585	017044	047040	046525	042502	
3586	017052	006522	000012		
3587	017056	005015	052012	042510	ADDRIS: .ASCIZ <CR><LF><LF>/THE PRESENT ADDRESS OF THE RH11 (RMCS1) IS: /
3588	017064	050040	042522	042523	
3589	017072	052116	040440	042104	
3590	017100	042522	051523	047440	
3591	017106	020106	044124	020105	
3592	017114	044122	030461	024040	
3593	017122	046522	051503	024461	
3594	017130	044440	035123	000040	
3595	017136	042412	052116	051105	NTRH11: .ASCIZ <LF>/ENTER NEW RH11 ADDRESS: /
3596	017144	047040	053505	051040	
3597	017152	030510	020061	042101	
3598	017160	051104	051505	035123	
3599	017166	000040			
3600	017170	005015	050012	042522	STANDBY: .ASCII <CR><LF><LF>/PRESS 'STANDBY' ON DRIVE/
3601	017176	051523	023440	052123	
3602	017204	047101	041104	023531	
3603	017212	047440	020116	051104	
3604	017220	053111	105		
3605	017223	015	050012	047522	.ASCIZ <CR><LF>/PROGRAM WILL LOOP WAITING FOR 'MOL' TO SET FROM PORT /
3606	017230	051107	046501	053440	
3607	017236	046111	020114	047514	
3608	017244	050117	053440	044501	
3609	017252	044524	043516	043040	

3610	017260	051117	023440	047515	
3611	017266	023514	052040	020117	
3612	017274	042523	020124	051106	
3613	017302	046517	050040	051117	
3614	017310	020124	000		
3615	017313	015	005012	042522	SWTCHN: .ASCIZ <CR><LF><LF>@RETURN 'CONTROLLER SELECT' SWITCH ON DRIVE TO 'A/B'@
3616	017320	052524	047122	023440	
3617	017335	047503	052116	047522	
3618	017334	046114	051105	051440	
3619	017342	046105	041505	023524	
3620	017350	051440	044527	041524	
3621	017356	020110	047117	042040	
3622	017364	044522	042526	052040	
3623	017372	020117	040447	041057	
3624	017400	000047			
3625	017402	05015	052012	051125	SWTCHA: .ASCIZ <CR><LF><LF>/TURN 'CONTROLLER SELECT' SWITCH ON DRIVE TO 'A'/'
3626	017410	020116	041447	047117	
3627	017416	051124	046117	042514	
3628	017424	020122	042523	042514	
3629	017432	052103	020047	053523	
3630	017440	052111	044103	047440	
3631	017446	020116	051104	053111	
3632	017454	020105	047524	023440	
3633	017462	023501	000		
3634	017465	015	005012	052524	SWTCHB: .ASCIZ <CR><LF><LF>/TURN 'CONTROLLER SELECT' SWITCH ON DRIVE TO 'B'/'
3635	017472	047122	023440	047503	
3636	017500	052116	047522	046114	
3637	017506	051105	051440	046105	
3638	017514	041505	023524	051440	
3639	017522	044527	041524	020110	
3640	017530	047117	042040	044522	
3641	017536	042526	052040	020117	
3642	017544	041047	000047		
3643	017550	005015	044124	047105	CONTUE: .ASCIZ <CR><LF>/THEN PRESS 'CONT' ON THE PROCESSOR/<CR><LF>
3644	017556	050040	042522	051523	
3645	017564	023440	047503	052116	
3646	017572	020047	047117	052040	
3647	017600	042510	050040	047522	
3648	017606	042503	051523	051117	
3649	017614	005015	000		
3650	017617	015	005012	052123	CYCLED: .ASCIZ <CR><LF><LF>/STOP THE DRIVE/'
3651	017624	050117	052040	042510	
3652	017632	042040	044522	042526	
3653	017640	000			
3654	017641	015	005012	052123	CYCLEU: .ASCIZ <CR><LF><LF>/START THE DRIVE - THE PROGRAM WILL WAIT FOR 'MOL' TO SET/'
3655	017646	051101	020124	044124	
3656	017654	020105	051104	053111	
3657	017662	020105	020055	044124	
3658	017670	020105	051120	043517	
3659	017676	040522	020115	044527	
3660	017704	046114	053440	044501	
3661	017712	020124	047506	020122	
3662	017720	046447	046117	020047	
3663	017726	047524	051440	052105	
3664	017734	000			
3665					

```

3666 ;*****
3667
3668 .SBTTL TEST ERROR MESSAGES
3669 ;*****
3670
3671
3672 017735 104 044522 042526 EM1: .ASCIZ /DRIVE IS NON-EXISTENT ('NED' BIT SET)/
3673 017742 044440 020123 047516
3674 017750 026516 054105 051511
3675 017756 042524 052116 024040
3676 017764 047047 042105 020047
3677 017772 044502 020124 042523
3678 020000 024524 000
3679
3680 020003 127 047522 043516 EM2: .ASCIZ /WRONG DRIVE TYPE/
3681 020010 042040 044522 042526
3682 020016 052040 050131 000105
3683
3684 020024 047503 052116 047522 EM3: .ASCIZ @CONTROLLER SELECT SWITCH ON DRIVE NOT IN 'A/B'@
3685 020032 046114 051105 051440
3686 020040 046105 041505 020124
3687 020046 053523 052111 044103
3688 020054 047440 020116 051104
3689 020062 053111 020105 047516
3690 020070 020124 047111 023440
3691 020076 027501 023502 000
3692
3693 020103 104 044522 042526 EM4: .ASCIZ /DRIVE NOT ON LINE/
3694 020110 047040 052117 047440
3695 020116 020116 044514 042516
3696 020124 000
3697
3698 020125 123 051105 040511 EMS: .ASCIZ /SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME/
3699 020132 020114 052516 041115
3700 020140 051105 051040 040505
3701 020146 020104 044124 047522
3702 020154 043525 020110 040505
3703 020162 044103 050040 051117
3704 020170 020124 047516 020124
3705 020176 044124 020105 040523
3706 020204 042515 000
3707
3708 020207 124 046511 047505 EM6: .ASCIZ /TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS/
3709 020214 052125 044040 051501
3710 020222 047040 052117 047440
3711 020230 041503 051125 042522
3712 020236 020104 044527 044124
3713 020244 047111 031040 051440
3714 020252 041505 047117 051504
3715 020260 000
3716
3717 020261 124 046511 047505 EM7: .ASCIZ /TIMEOUT ONE-SHOT IS LESS THAN 500 MS/
3718 020266 052125 047440 042516
3719 020274 051455 047510 020124
3720 020302 051511 046040 051505
3721 020310 020123 044124 047101

```

3722	020316	032440	030060	046440	
3723	020324	000123			
3724					
3725	020326	042032	042101	047111	EM10: .ASCIZ /READIN PRESET DOES NOT SET VOLUME VALID FOR THE PORT/
3726	020327	042032	042101	042523	
3727	020328	042032	042101	051505	
3728	020329	047040	052117	051440	
3729	020330	052105	053040	046117	
3730	020331	046525	020105	040526	
3731	020332	044514	020104	047506	
3732	020333	020122	044124	020105	
3733	020406	047520	052122	000	
3734					
3735	020413	047	047507	020047	EM11: .ASCIZ /'GO' BIT RESET DURING UNLOAD COMMAND/
3736	020420	044502	020124	042522	
3737	020426	042523	020124	052504	
3738	020434	044522	043516	052440	
3739	020442	046116	040517	020104	
3740	020450	047503	046515	047101	
3741	020456	000104			
3742					
3743	020460	047111	047503	051122	EM12: .ASCIZ /INCORRECT STATUS DURING UNLOAD COMMAND/
3744	020466	041505	020124	052123	
3745	020474	052101	051525	042040	
3746	020502	051125	047111	020107	
3747	020510	047125	047514	042101	
3748	020516	041440	046517	040515	
3749	020524	042116	000		
3750					
3751	020527	104	044522	042526	EM13: .ASCIZ /DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND/
3752	020534	042040	042111	047040	
3753	020542	052117	051040	052105	
3754	020550	051125	020116	047524	
3755	020556	047040	052505	051124	
3756	020564	046101	040440	052106	
3757	020572	051105	052440	046116	
3758	020600	040517	020104	047503	
3759	020606	046515	047101	000104	
3760					
3761	020614	052101	042524	052116	EM14: .ASCIZ /ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD/
3762	020622	047511	020116	044502	
3763	020630	020124	042523	020124	
3764	020636	047117	023440	050117	
3765	020644	047520	044523	042524	
3766	020652	050040	051117	023524	
3767	020660	040440	052106	051105	
3768	020666	052440	046116	040517	
3769	020674	000104			
3770					
3771	020676	052101	042524	052116	EM15: .ASCIZ /ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD'/
3772	020704	047511	020116	044502	
3773	020712	020124	047516	020124	
3774	020720	042523	020124	047117	
3775	020726	050040	051117	020124	
3776	020734	044127	041511	020110	
3777	020742	051511	052523	042105	

3778	020750	023440	047125	047514	
3779	020756	042101	000047		
3780					
3781	020762	051104	053111	020105	EM16: .ASCII /DRIVE NOT IN NEUTRAL AFTER UNLOAD WITH 'CONTROLLER/<CR><LF>
3782	020770	047516	020124	047111	
3783	020776	047040	052505	051124	
3784	021004	046101	040440	052106	
3785	021012	051105	052440	046116	
3786	021020	040517	020104	044527	
3787	021026	044124	023440	047503	
3788	021034	052116	047522	046114	
3789	021042	051105	005015		
3790	021046	042523	042514	052103	.ASCIZ 'SELECT' SWITCH MOVED FROM 'A/B'
3791	021054	020047	053523	052111	
3792	021062	044103	046440	053117	
3793	021070	042105	043040	047522	
3794	021076	020115	040447	041057	
3795	021104	000047			
3796					
3797	021106	051104	053111	020105	EM17: .ASCIZ /DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP/
3798	021114	047514	045503	042105	
3799	021122	047440	020116	047520	
3800	021130	052122	023440	023501	
3801	021136	041040	020131	053523	
3802	021144	052111	044103	053440	
3803	021152	044510	042514	041440	
3804	021160	041531	042514	020104	
3805	021166	050125	000		
3806					
3807	021171	104	044522	042526	EM20: .ASCIZ /DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP/
3808	021176	046040	041517	042513	
3809	021204	020104	047117	050040	
3810	021212	051117	020124	041047	
3811	021220	020047	054502	051440	
3812	021226	044527	041524	020110	
3813	021234	044127	046111	020105	
3814	021242	054503	046103	052105	
3815	021250	052440	000120		
3816					
3817	021254	052123	052101	051525	EM21: .ASCIZ /STATUS INCORRECT FOR PORT AFTER CYCLE UP/
3818	021262	044440	041516	051117	
3819	021270	042522	052103	043040	
3820	021276	051117	050040	051117	
3821	021304	020124	043101	042524	
3822	021312	020122	054503	046103	
3823	021320	020105	050125	000	
3824					
3825	021325	122	043505	051511	EM22: .ASCIZ /REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT/
3826	021332	042524	020122	047503	
3827	021340	052116	047105	051524	
3828	021346	051440	042505	020116	
3829	021354	044127	047105	042040	
3830	021362	044522	042526	051440	
3831	021370	044527	041524	042510	
3832	021376	020104	047117	023440	
3833	021404	050117	047520	044523	

3834	021412	042524	020047	047520	
3835	021420	052122	000		
3836					
3837	021423	047	042516	023504	EM23: .ASCIZ /'NED' NOT SET WHEN RMO31 ACCESSED THROUGH PORT NOT SWITCHED/
3838	021430	047040	052117	051440	
3839	021436	052105	053440	042510	
3840	021444	020116	046522	051504	
3841	021452	020061	041501	042503	
3842	021460	051523	042105	052040	
3843	021466	051110	052517	044107	
3844	021474	050040	051117	020124	
3845	021502	047516	020124	053523	
3846	021510	052111	044103	042105	
3847	021516	000			
3848					
3849	021517	104	044522	042526	EM24: .ASCIZ /DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/
3850	021524	051440	044527	041524	
3851	021532	042510	020104	047524	
3852	021540	046040	041517	042513	
3853	021546	020104	052517	020124	
3854	021554	047520	052122	053440	
3855	021552	042510	020116	042522	
3856	021570	042514	051501	042105	
3857	021576	000			
3858					
3859	021577	122	030510	020061	EM25: .ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/
3860	021604	044504	047104	052047	
3861	021612	051040	051505	047520	
3862	021620	042116	052040	020117	
3863	021626	042101	051104	051505	
3864	021634	044523	043516	000	
3865					
3866	021641	104	044522	042526	EM30: .ASCIZ /DRIVE NOT SEIZED BY PORT/
3867	021646	047040	052117	051440	
3868	021654	044505	042532	020104	
3869	021662	054502	050040	051117	
3870	021670	000124			
3871					
3872	021672	051127	047117	020107	EM31: .ASCIZ /WRONG STATUS SEEN BY THE SEIZING PORT/
3873	021700	052123	052101	051525	
3874	021706	051440	042505	020116	
3875	021714	054502	052040	042510	
3876	021722	051440	044505	044532	
3877	021730	043516	050040	051117	
3878	021736	000124			
3879					
3880	021740	042522	044507	052123	EM32: .ASCIZ /REGISTER CONTENTS WRONG/
3881	021746	051105	041440	047117	
3882	021754	042524	052116	020123	
3883	021762	051127	047117	000107	
3884					
3885	021770	047503	052116	047522	EM33: .ASCIZ /CONTROL BUS PARITY ERROR READING INDICATED REGISTER/
3886	021776	020114	052502	020123	
3887	022004	040520	044522	054524	
3888	022012	042440	051122	051117	
3889	022020	051040	040505	044504	

3890	022026	043516	044440	042116	
3891	022034	041511	052101	042105	
3892	022042	051040	043505	051511	
3893	022050	042524	000122		
3894					
3895	022054	040503	023516	020124	EM34: .ASCIZ /CAN'T ACCESS DRIVE THROUGH EITHER PORT/
3896	022062	041501	042503	051523	
3897	022070	042040	044522	042526	
3898	022076	052040	051110	052517	
3899	022104	044107	042440	052111	
3900	022112	042510	020122	047520	
3901	022120	052122	000		
3902					
3903	022123	104	044522	042526	EM35: .ASCIZ /DRIVE NOT IN NEUTRAL AFTER RELEASE - REQUEST NOT SET/
3904	022130	047040	052117	044440	
3905	022136	020116	042516	052125	
3906	022144	040F22	020114	043101	
3907	022152	042524	020122	042522	
3908	022160	042514	051501	020105	
3909	022166	020055	042522	052521	
3910	022174	051505	020124	047516	
3911	022202	020124	042523	000124	
3912					
3913	022210	051104	053111	020105	EM36: .ASCIZ /DRIVE NOT IN NEUTRAL AFTER TIMEOUT - REQUEST NOT SET/
3914	022216	047516	020124	047111	
3915	022224	047040	052505	051124	
3916	022232	046101	040440	052106	
3917	022240	051105	052040	046511	
3918	022246	047505	052125	026440	
3919	022254	051040	050505	042525	
3920	022262	052123	047040	052117	
3921	022270	051440	052105	000	
3922					
3923	022275	122	043505	051511	EM37: .ASCIZ /REGISTER CONTENTS WRONG AFTER RELEASE OR TIMEOUT/
3924	022302	042524	020122	047503	
3925	022310	052116	047105	051524	
3926	022316	053440	047522	043516	
3927	022324	040440	052106	051105	
3928	022332	051040	046105	040505	
3929	022340	042523	047440	020122	
3930	022346	044524	042515	052517	
3931	022354	000124			
3932					
3933	022356	051104	053111	020105	EM40: .ASCIZ /DRIVE IN NEUTRAL AFTER RELEASE - REQUEST SET/
3934	022364	047111	047040	052505	
3935	022372	051124	046101	040440	
3936	022400	052106	051105	051040	
3937	022406	046105	040505	042523	
3938	022414	026440	051040	050505	
3939	022422	042525	052123	051440	
3940	022430	052105	000		
3941					
3942	022433	122	043505	051511	EM41: .ASCIZ /REGISTER WRONG AFTER RELEASE WITH REQUEST SET/
3943	022440	042524	020122	051127	
3944	022446	047117	020107	043101	
3945	022454	042524	020122	042522	

3946	022462	042514	051501	020105
3947	022470	044527	044124	051040
3948	022476	050505	042525	052123
3949	022504	051440	052105	000
3950				
3951	022511	042514	051505	020124
3952	022516	020043	042440	051122
3953	022522	020040	020103	050040
3954	022528	051117	020124	020043
3955	022534	051040	043505	040440
3956	022540	051104	041440	047117
3957	022546	042524	052116	000123
3958	022552	042524	052123	021440
3959	022558	020040	051105	020122
3960	022604	041520	020040	047520
3961	022610	052122	021440	020040
3962	022616	020122	020107	042101
3963	022622	020040	047507	042117
3964	022628	020040	020040	040502
3965	022634	000104		
3966	022640	020040	052123	021440
3967	022646	020040	051105	020122
3968	022652	041520	020040	042522
3969	022658	020107	042101	020122
3970	022704	047520	052122	040440
3971	022710	020040	047520	052122
3972	022716	041040	000	
3973	022722	041040	000	
3974	022728	020043	051505	020124
3975	022734	020040	042440	051122
3976	022740	020040	020103	050040
3977	022746	051117	020124	000043
3978	022752	042524	052123	021440
3979	022758	020040	051105	020122
3980	022804	041520	020040	047520
3981	022810	052122	021440	020040
3982	022816	044524	042515	024040
3983	022822	047111	046440	024523
3984	022828	000		
3985	022834	020040	020040	020040
3986	022840	020040	020040	020040
3987	022846	020040	020040	020040
3988	022852	044505	042532	005015
3989	022858	042524	052123	021440
3990	022904	020040	051105	020122
3991	022910	041520	020040	047520
3992	022916	052122	021440	000
3993	022922	040	020040	020040
3994	022928	020040	020040	020040
3995	022934	020040	020040	051440
3996	022940	044505	042532	020040
3997	022946	042440	051122	051117
3998	022952	005015		
3999	022958	042524	052123	021440
4000	023004	020040	051105	020122
4001	023010	041520	020040	047520

DH1: .ASCIZ /TEST # ERR PC PORT # REG ADR CONTENTS/

DH2: .ASCIZ /TEST # ERR PC PORT # REG ADR GOOD BAD/

DH5: .ASCIZ /TEST # ERR FC REG ADR PORT A PORT B/

DH6: .ASCIZ /TEST # ERR PC PORT #/

DH7: .ASCIZ /TEST # ERR PC PORT # TIME (IN MS)/

DH13: .ASCII / SEIZE/(CR)<LF>

.ASCIZ /TEST # ERR PC PORT #/

DH14: .ASCII / SEIZE ERROR/(CR)<LF>

.ASCIZ /TEST # ERR PC PORT # PORT # REG ADR CONTENTS/

4058	023611	124	051505	020124		.ASCIZ	/TEST #	ERR PC	PORT #	PORT #/	
4059	023616	020043	042440	051122							
4060	023624	050040	020103	050040							
4061	023632	051117	020124	020043							
4062	023640	050040	051117	020124							
4063	023646	003043									
4064	023650	023040	020040	020040	DH37:	.ASCII	/		RELSNG	ERROR<<CR><LF>	
4065	023656	023040	023040	020040							
4066	023664	020040	023040	042522							
4067	023672	051514	043516	020040							
4068	023700	051105	047522	006522							
4069	023706	012									
4070	023707	124	051505	020124		.ASCIZ	/TEST #	ERR PC	PORT #	PORT #	REG ADR GOOD BAD/
4071	023714	020043	042440	051122							
4072	023722	050040	020103	050040							
4073	023730	051117	020124	020043							
4074	023736	050040	051117	020124							
4075	023744	020043	051040	043505							
4076	023752	040440	051104	043440							
4077	023760	047517	020104	020040							
4078	023766	041040	042101	000							
4079	023773	040	020040	020040	DH40:	.ASCII	/		RELSNG	RQSTNG<<CR><LF>	
4080	024000	020040	020040	020040							
4081	024006	020040	020040	051040							
4082	024014	046105	047123	020107							
4083	024022	051040	051521	047124							
4084	024030	006507	012								
4085	024033	124	051505	020124		.ASCIZ	/TEST #	ERR PC	PORT #	PORT #/	
4086	024040	020043	042440	051122							
4087	024046	050040	020103	050040							
4088	024054	051117	020124	020043							
4089	024062	050040	051117	020124							
4090	024070	000043									
4091											
4092						.EVEN					
4093											
4094	024072	001242	001116	001234	DT1:	.WORD	TSTNUM,	SERRPC,	PTNBR,	\$BDAOR,	\$BDDAT,0
4095	024100	001122	001126	000000							
4096	024106	001242	001116	001234	DT2:	.WORD	TSTNUM,	SERRPC,	PTNBR,	\$BDAOR,	\$GDDAT,\$BDDAT,0
4097	024114	001122	001124	001126							
4098	024122	000000									
4099	024124	001242	001116	001122	DT5:	.WORD	TSTNUM,	SERRPC,	\$BDAOR,	\$GDDAT,\$BDDAT,0	
4100	024132	001124	001126	000000							
4101	024140	001242	001116	001234	DT6:	.WORD	TSTNUM,	SERRPC,	PTNBR,0		
4102	024146	000000									
4103	024150	001242	001116	001234	DT7:	.WORD	TSTNUM,	SERRPC,	PTNBR,	TIME,0	
4104	024156	001252	000000								
4105	024162	001242	001116	001236	DT13:	.WORD	TSTNUM,	SERRPC,	SEIZPT,0		
4106	024170	000000									
4107	024172	001242	001116	001236	DT14:	.WORD	TSTNUM,	SERRPC,	SEIZPT,	PTNBR,\$BDAOR,\$BDDAT,0	
4108	024200	001234	001122	001126							
4109	024206	000000									
4110	024210	001242	001116	000000	DT17:	.WORD	TSTNUM,	SERRPC,0			
4111	024216	001242	001116	001236	DT24:	.WORD	TSTNUM,	SERRPC,	SEIZPT,	PTNBR,0	
4112	024224	001234	000000								
4113	024230	001272	000000		DT25:	.WORD	\$RMAOR,0				

```

4114 024234 001242 001116 001236 DT30: .WORD TSTNUM, $ERRPC, SEIZPT, PTNBR, $B0ADR, $GDDAT, $BDDAT, 0
4115 024242 001234 001122 001124
4116 024250 001126 000000
4117 024254 001242 001116 001170 DT34: .WORD TSTNUM, $ERRPC, $TMP2, $TMP3, 0
4118 024262 001112 000000
4119 024266 001242 001116 001236 DT35: .WORD TSTNUM, $ERRPC, SEIZPT, PTNBR, 0
4120 024274 001234 000000
4121 024300 001242 001116 001236 DT40: .WORD TSTNUM, $ERRPC, SEIZPT, OPPRT, 0
4122 024306 001240 000000
4123
4124 024312 000 000 001 DF1: .BYTE 0,0,1,0,0
4125 024315 000 000
4126 024317 000 000 001 DF2: .BYTE 0,0,1,0,0,0
4127 024322 000 000
4128 024325 000 000 000 DF5: .BYTE 0,0,0,0,0
4129 024330 000 000
4130 024332 000 000 001 DF6: .BYTE 0,0,1
4131 024335 000 000 001 DF7: .BYTE 0,0,1,1
4132 024340 001
4133 024341 000 000 001 DF14: .BYTE 0,0,1,1,0,0
4134 024344 001 000
4135 024347 000 000 DF17: .BYTE 0,0
4136 024351 000 DF25: .BYTE 0
4137 024352 000 000 001 DF30: .BYTE 0,0,1,1,0,0,0
4138 024355 001 000
4139 024360 000
4140 024361 000 000 000 DF34: .BYTE 0,0,0,0
4141 024364 000
4142
4143 024366 .EVEN
4144
4145
4146 ;*****
4147 ;*****
4148 .SBTTL CONSTANTS, TABLES, ETC
4149 ;*****
4150 ;*****
4151 ;*****
4152 ;TABLE OF TEST STARTING ADDRESSES
4153
4154 TSTADR: .WORD TST1+2 ;STARTING ADDRESS OF TEST 1
4155 .WORD TST2+2 ;STARTING ADDRESS OF TEST 2
4156 .WORD TST3+2 ;STARTING ADDRESS OF TEST 3
4157 .WORD TST4+2 ;STARTING ADDRESS OF TEST 4
4158 .WORD TST5+2 ;STARTING ADDRESS OF TEST 5
4159 .WORD TST6+2 ;STARTING ADDRESS OF TEST 6
4160 .WORD TST7+2 ;STARTING ADDRESS OF TEST 7
4161 .WORD TST10+2 ;STARTING ADDRESS OF TEST 10
4162
4163 ;ATTENTION BIT TABLE
4164
4165 ATABIT: .BYTE 1 ;ATTENTION BIT FOR DRIVE 0
4166 .BYTE 2 ;ATTENTION BIT FOR DRIVE 1
4167 .BYTE 4 ;ATTENTION BIT FOR DRIVE 2
4168 .BYTE 10 ;ATTENTION BIT FOR DRIVE 3
4169 .BYTE 20 ;ATTENTION BIT FOR DRIVE 4

```

M06

MD-11-DZRMH-A, RMO3 DUAL CONTRLR LOGIC TEST - PART 2 MACY11 30(1046) 01-AUG-77 11:14 PAGE 77
DZRMHA.P11 01-AUG-77 11:02 CONSTANTS, TABLES, ETC

SEQ 0078

4170	024413	040	.BYTE	40	:ATTENTION BIT FOR DRIVE 5
4171	024414	100	.BYTE	100	:ATTENTION BIT FOR DRIVE 6
4172	024415	200	.BYTE	200	:ATTENTION BIT FOR DRIVE 7
4173					
4174	024416	000010	MAXTN: .WORD	STN-1	;MAXIMUM TEST NUMBER
4175					
4176		000001	.END		

ADDBIS	017056	1423	3587#																	
ADREAR	016643	1350	3561#																	
AOE	= 001000	811#																		
ASR1	001232	1008#	1375#																	
ATA	= 100000	798#	1719	1723	1744	1750	1829	1833	1854	1860	1974	1978	2001	2007						
		2119	2123	2146	2152	2213	2217	2240	2246	2264	2268	2291	2297	2373						
		2567																		
ATABIT	024406	1375	4165#																	
ATO	= 000001	825#																		
AT1	= 000002	826#																		
AT2	= 000004	827#																		
AT3	= 000010	828#																		
AT4	= 000020	829#																		
AT5	= 000040	830#																		
AT6	= 000100	831#																		
AT7	= 000200	832#																		
A16	= 000400	729#																		
A17	= 001000	730#																		
BAONO	017030	1408	3583#																	
BAI	= 000010	747#																		
BIT0	= 000001	701#																		
BIT00	= 000001	691#	701																	
BIT01	= 000002	690#	700																	
BIT02	= 000004	689#	699																	
BIT03	= 000010	688#	698																	
BIT04	= 000020	687#	697																	
BIT05	= 000040	686#	696																	
BIT06	= 000100	685#	695																	
BIT07	= 000200	684#	694																	
BIT08	= 000400	683#	693																	
BIT09	= 001000	682#	692																	
BIT1	= 000002	700#																		
BIT10	= 000200	681#	2863																	
BIT11	= 000400	680#	2829																	
BIT12	= 010000	679#																		
BIT13	= 020000	678#	2870																	
BIT14	= 040000	677#	2811																	
BIT15	= 100000	676#																		
BIT2	= 000004	699#																		
BIT3	= 000010	698#																		
BIT4	= 000020	697#																		
BIT5	= 000040	696#																		
BIT6	= 000100	695#																		
BIT7	= 000200	694#																		
BIT8	= 000400	693#																		
BIT9	= 001000	692#																		
BPTVEC	= 000014	708#																		
CHANGE	002576	1344	1420#																	
CHGAOR	001270	1023#	1291#	1293#	1420	1422#														
CKCLK	013226	1380	1393	2750#																
CKCLK1	013276	2750	2759#																	
CKCLK2	013340	2760	2767#																	
CKCLK3	013350	2758	2766	2769#																
CKERR	001244	1013#	1484#	1497#	1499	1505#	1518#	1520	1529#	1542#	1546#	1559#	1567#	1580#						
		1582#	1595#	1599#	1612#	1614#	1627#	1686#	1699#	1796#	1809#	2369#	2382#	2398#						
		2411#	2413#	2426#	2440#	2453#	2563#	2576#	2592#	2605#	2607#	2620#	2634#	2647#						

\$ERTTL	001112	960#	2723	2727*	2866*	2885									
\$ESCAP	001200	992#	1311*												
\$FILLC	001156	981#	2980	3011											
\$FILLS	001155	980#	3011												
\$GDADR	001120	964#													
\$GDOAT	001124	966#	1488*	1491	1495*	1509*	1512	1516*	1533*	1536	1540*	1550*	1553	1557*	
		1571#	1574	1578*	1586*	1589	1593*	1603*	1606	1610*	1618*	1621	1625*	1634*	
		1637	1690*	1693	1697*	1715*	1745	1751	1800*	1803	1807*	1825*	1855	1861	
		1918#	1919	1927*	1928	1932	1970*	2002	2008	2063*	2064	2072*	2073	2077	
		2115#	2147	2153	2209*	2241	2247	2260*	2292	2298	2373*	2376	2380*	2402*	
		2405	2409*	2417*	2420	2424*	2444*	2447	2451*	2567*	2570	2574*	2596*	2599	
		2603#	2611*	2614	2618*	2638*	2641	2645*	4096	4099	4114				
\$GET42	013176	2728#													
\$GTSWR	015364	3253#	3529												
\$HD =	000000	593													
\$HIOCT	016324	3435#	3446#												
\$ICNT	001104	957#	1414*	2833*	2834	2836*	2843								
\$INTAG	001135	971#	3250*	3281	3393										
\$ITEMB	001114	961#	2869*	2885	2896										
\$LF	001210	996#	2885	3011	3377	3387	3447								
\$LKCSB	001214	1001#	2756*												
\$LKCSR	001212	1000#	2752	2757*											
\$LKS	001220	1003#	2761	2765*											
\$LLVEC	001222	1004#	2762												
\$LPADR	001106	958#	1313*	1399*	1473*	1669*	1779*	1892*	2037*	2185*	2344*	2538*	2839*	2841	
		2843													
\$LPERR	001110	959#	1314*	1400*	1474*	1670*	1780*	1893*	2038*	2186*	2345*	2464	2539*	2658	
\$LPVEC	001216	1002#	2753												
\$MAIL =	***** U	1331	1338	2839	2875	2964									
\$MNEW	016154	3256	3391#												
\$MSWR	016143	3253	3389#												
\$MXCNT	013620	2837	2843#												
\$NULL	001154	979#	2982	3011											
\$NWTST=	000001	1453#	1455	1647#	1649	1757#	1759	1870#	1872	2015#	2017	2161#	2163	2309#	
		2311	2497#	2499											
\$OCNT	014572	3043#	3072*	3085#											
\$OMODE	014574	3038#	3042*	3047	3050*	3061*	3087#								
\$OVER	013604	2812	2823	2835	2840#										
\$PASS	001100	954#	1397*	2704*	2705*	2716	2738	2831	2844						
\$QUES	001206	994#	2885	3011	3299	3370	3387	3444	3447						
\$RDCHR	015576	3312#	3532												
\$RDOEC=	***** U	3535													
\$ROLIN	015666	3335#	3533												
\$ROOCT	016166	3408#	3534												
\$ROSZ =	000007	3328#													
\$REGAD	001160	983#													
\$REGO	001162	985#													
\$RESRE	016364	3480#	3536												
\$RMAOR	001272	1031#	1391	1424	1431*	1434	1450	4113							
\$RMVEC	001274	1032#													
\$RTNAD	013220	2737#													
\$R2A =	***** U	3537													
\$SAVRE	016326	3464#	3535												
\$SCOPE	013440	1303	2809#												
\$SETUP=	000127	1284#	1302	1303	1305	1307	1309	1310	1311	1313	1335	2702	2810	2858	
		2878	2880	3190	3195	3196	3226	3393							

USREQ	2688	2699										
.TCRE	946	985										
.SCTM	946	986	987	988	989	990						
.SESCA	716											
.NEMT	716	1453	1647	1757	1870	2015	2161	2309	2497			
.JET	3515	3524	3525	3526	3527	3529	3531	3532	3533	3534	3535	3536
.SXIP	716	2495										
.EQUAT	582	606										
.HEADE	582	583										
.SETUP	582	1284										
.SWHT	582	594										
.SWLO	582	604										
.SACT1	582	926										
.SCATC	582	916										
.SCTA	582	946										
.SEOP	582	2688										
.SERRO	582	2844										
.SERRT	582	2885										
.SROOC	582	3394										
.SREAD	582	3155										
.SSAVE	582	3447										
.SSCOP	582	2797										
.STRAP	582	3492										
.STYPO	582	3088										
.STYPE	582	2941										
.STYPO	582	3011										

. ABS. 024420 000

ERRORS DETECTED: 0

DSKZ:DZRMHA.BIN, DSKZ:DZRMHA.SEQ/DOC/SOL/CRF=DSKM:DZRMHA.P11

RUN-TIME: 20 13 1 SECONDS

RUN-TIME RATIO: 177/35=4.9

CORE USED: 28K (55 PAGES)

DOCUMENT PAGES: 91

EOF1DZRMHASEQ

00010000

770804

PDP10 411