

RM80

DU PORT TST PT 2
CZRNIAO

AH-T121A-MC
FICHE 1 OF 1

JUL 1982
COPYRIGHT© 1982
MADE IN USA



A microfiche grid containing approximately 10 columns and 15 rows of frames. Each frame contains a small, high-contrast image of a document page, likely a technical drawing or test report. The frames are arranged in a regular grid pattern on the left side of the microfiche card.



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

.REM ^

IDENTIFICATION

PRODUCT CODE: AC-T120A-MC
PRODUCT NAME: CZRNIAO RM80 DUAL PORT TEST, PT 2
PRODUCT DATE: APRIL 1, 1982
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1982 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

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

1. ABSTRACT

THE RM80 DUAL PORT LOGIC TEST PERFORMS A SERIES OF TESTS WHICH VERIFY THAT THE RM80 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 AND RH70.

THIS PROGRAM IS THE SECOND PART OF THE RM80 DUAL PORT OPTION LOGIC TEST, AND IS USED TO TEST THE 'PORT SELECT' SWITCH.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11/70 PROCESSOR
8K MEMORY
KW11-L OR KW11-P CLOCK
TERMINAL
RH70 CONTROLLER
1 - RM80 DISK DRIVE
RM DUAL PORT TEST CABLE (P/N: 7010507-02)

2.2 PREREQUISITE PROGRAMS

RM80 DISKLESS TEST, PART 1 & 2
RM80 FUNCTIONAL TEST, PART 1, 2 & 3
RM80 DUAL PORT LOGIC TEST, PART 1

THE PRELIMINARY PROGRAMS MUST BE RUN TWICE: ONCE FROM EACH PORT (A & B).

2.3 OTHER PROGRAMS

DYNAMIC OPERATION OF THE DUAL PORT OPTION IS TESTED BY THE RM80 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 'XXDP' LOADER. THE PROGRAM MAY NOT

58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

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 ADDRESS.
- C. THE PROGRAM CAN BE STARTED AT LOCATION 210(8) TO ALLOW THE 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 FROM LOCATION 204.

115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
1715. OPERATING PROCEDURES

5.1 'SOFTWARE' SWITCH REGISTER

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR 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 RM80 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 1'S IN THE SWITCHES. ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

5.2 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 |

5.3 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<14>, MUST BE SET TO ALL CONTINUOUS EXECUTION OF THE SELECTED TEST.

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

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

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 RM80 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 RM80 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, RH CONTROLLER IS TO TEST THE DRIVE. IF THE DRIVE IS TO BE TESTED BY THE PROCESSOR ON PORT A, CONNECT THE MASSBUS CABLE FROM THE RH CONTROLLER TO J3 OF THE R1 ADAPTER BACK PANEL, THEN CONNECT THE TEST CABLE (P/N: 7010507-02) FROM J2 TO J7 OF THE BACK PANEL AND TERMINATE THE PORT B AT J6.

WHEN THE DUAL PORT TEST CABLE IS CONNECTED, THE ATTENTION BITS FOR PORTS A & B ARE ASSERTED IN THE SAME BIT POSITION WHEN 'RMAS' (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 'RMDS' (DRIVE STATUS REGISTER) TO DETERMINE THE STATE OF THE SELECTED PORT'S ATTENTION BIT.

6. ERRORS

WHEN THE PROGRAM ENCOUNTERS AN ERROR, THE ERROR ROUTINE IS CALLED AND IF SW<13> IS NOT SET, THE ERROR MESSAGE PERTAINING

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

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 RM80 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 7 MINUTES 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

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

286 IF SW<09> IS SET, THE PROGRAM WILL LOOP ON A FAILING TEST
287 UNTIL EITHER THE SWITCH IS RESET OR THE ERROR STOPS OCCURING.
288 BECAUSE THE PROGRAM MUST RESET THE RM80 TO A KNOWN STATE
289 BEFORE LOOPING ON THE ERROR, THE TEST FOR SW<09> IS PERFORMED
290 AT THE END OF THE TEST - NOT AT THE POINT WHERE THE ERROR
291 WAS DETECTED.
292
293

294 8. TEST DESCRIPTIONS
295 -----

296 8.1 METHOD USED TO VERIFY THAT DRIVE IS IN NEUTRAL
297

298 THE PROGRAM DETERMINES THE THE DRIVE IS IN NEUTRAL BY CHECKING
299 THE CONTENTS OF THE DRIVE STATUS REGISTER (RMDS) THROUGH
300 BOTH PORTS. THE PROGRAM MASKS OUT THE PORT DEPENDENT BITS
301 ('ATA' & 'VV') AND VERIFIES THAT CORRECT STATUS IS READ
302 THROUGH BOTH PORTS. (THE CORRECT STATUS IS 'MOL', 'PGM',
303 'DPR', & 'DRY'.) IF NEITHER PORT SEES ALL ZEROS FROM
304 RMDS, THE PROGRAM CONCLUDES THAT THE DRIVE IS IN NEUTRAL
305 AND THAT ANY BIT DISCREPANCY BETWEEN PORTS INDICATES A
306 FAILURE IN THE PATH FOR THAT BIT.
307

308 8.2 METHOD USED TO VERIFY THAT THE DRIVE HAS BEEN SEIZED
309

310 THE PROGRAM VERIFIES THAT THE DRIVE HAS BEEN SEIZED BY
311 CHECKING THE DRIVE STATUS REGISTER (RMDS) THROUGH
312 THE SEIZING PORT AND VERIFYING THAT CORRECT STATUS IS
313 SEEN. WHEN RMDS IS READ THROUGH THE OPPOSITE PORT,
314 ZEROS SHOULD BE SEEN. IF BOTH CONDITIONS EXIST, (I.E.
315 THE OPPOSITE PORT), THE PROGRAM CONCLUDES THAT THE DRIVE
316 HAS BEEN SEIZED BY THE SPECIFIED PORT.
317

318 TEST 1 DRIVE ACCESS TEST
319

320 VERIFY THAT THE DRIVE CAN BE ACCESSED THROUGH BOTH PORTS
321

- 322 A. SELECT DRIVE, VERIFY THAT THE DRIVE IS PRESENT, THAT THE
323 DRIVE IS A DUAL PORT RM80, THAT THE DRIVE IS ONLINE (RMDS HAS
324 'MOL', 'PGM', 'DPR', & 'DRY' BITS SET), AND THE THE DRIVE SERIAL
325 NUMBER READ THROUGH BOTH PORTS IS THE SAME.
326
327 B. THE TEST IS REPEATED THROUGH BOTH PORTS.
328

329 TEST 2 SET 'VV' FOR PORT A
330

331 SET VOLUME VALID
332

- 333 A. ISSUE A DRIVE CLEAR COMMAND THROUGH PORT A.
334
335 B. ISSUE A READIN PRESET COMMAND THROUGH PORT A. VERIFY
336 THAT THE 'VV' BIT IS SET FOR PORT A.
337
338 C. ISSUE A RELEASE COMMAND THROUGH PORT A. VERIFY THAT
339 THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION
340
341
342

343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399

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 RMDS 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 RMDS 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

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

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

- A. SWITCH TO PORT 'A' POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH PORTS, ARE CORRECT.
- B. SWITCH TO PORT 'B' POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH PORTS, ARE CORRECT.

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

C. RETURN THE 'PORT SELECT' SWITCH TO THE 'A/B' POSITION. VERIFY THE DRIVE STATE.

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

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

- A. CYCLE THE DRIVE DOWN.
- B. SWITCH TO PORT 'A' POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH PORTS, ARE CORRECT.
- C. SWITCH THE 'PORT 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 RMDS THROUGH PORT B.
- G. ISSUE A RELEASE COMMAND THROUGH PORT A. VERIFY THAT THE DRIVE REMAINS LOCKED ON PORT A.

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

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

- A. CYCLE THE DRIVE DOWN.
- B. SWITCH TO PORT 'B' POSITION. VERIFY THAT THE DRIVE IS IN NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH PORTS, ARE CORRECT.
- C. SWITCH THE 'PORT 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 RMDS THROUGH PORT A.
- G. ISSUE A RELEASE COMMAND THROUGH PORT B. VERIFY THAT THE DRIVE REMAINS LOCKED ON PORT B.

457
458
459
460
461
462
463
464

- H. CYCLE THE DRIVE DOWN. CHANGE THE 'PORT 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.

1
509
510

:*LAST REVISION 04-AUG-81

.TITLE CZRNIAO RM80 DUAL PORT PT2
:*COPYRIGHT (C) 1982
:*DIGITAL EQUIPMENT CORPORATION
:*COLORADO SPGS., CO. 80919

:*PROGRAM BY MIKE LEAVITT

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

511

.SBTTL OPERATIONAL SWITCH SETTINGS

| SWITCH | USE |
|--------|------------------------|
| 15 | HALT ON ERROR |
| 14 | LOOP ON TEST |
| 13 | INHIBIT ERROR TYPEOUTS |
| 11 | INHIBIT ITERATIO'S |
| 10 | BELL ON ERROR |
| 9 | LOOP ON ERROR |

512
513

.SBTTL BASIC DEFINITIONS

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

001100
104000
000004

STACK = 1100

ERROR = EMT

SCOPE = IOT

::BASIC DEFINITION OF ERROR CALL

::BASIC DEFINITION OF SCOPE CALL

:*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

HT = 11

LF = 12

CR = 15

CRLF = 200

PS = 177776

PSW=PS

STKLMT = 177774

PIRQ = 177772

DSWR = 177570

DDISP = 177570

::CODE FOR HORIZONTAL TAB

::CODE FOR LINE FEED

::CODE FOR CARRIAGE RETURN

::CODE FOR CARRIAGE RETURN-LINE FEED

::PROCESSOR STATUS WORD

::STACK LIMIT REGISTER

::PROGRAM INTERRUPT REQUEST REGISTER

::HARDWARE SWITCH REGISTER

::HARDWARE DISPLAY REGISTER

:*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

R0 = %0

R1 = %1

R2 = %2

R3 = %3

R4 = %4

R5 = %5

R6 = %6

R7 = %7

SP = %6

PC = %7

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::GENERAL REGISTER

::STACK POINTER

::PROGRAM COUNTER

:*PRIORITY LEVEL DEFINITIONS

000000
000040

PRO = 0

PR1 = 40

::PRIORITY LEVEL 0

::PRIORITY LEVEL 1

| | | | | |
|--------|-----|---|-----|--------------------|
| 000100 | PR2 | = | 100 | ::PRIORITY LEVEL 2 |
| 000140 | PR3 | = | 140 | ::PRIORITY LEVEL 3 |
| 000200 | PR4 | = | 200 | ::PRIORITY LEVEL 4 |
| 000240 | PR5 | = | 240 | ::PRIORITY LEVEL 5 |
| 000300 | PR6 | = | 300 | ::PRIORITY LEVEL 6 |
| 000340 | PR7 | = | 340 | ::PRIORITY LEVEL 7 |

;*SWITCH REGISTER SWITCH DEFINITIONS

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

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

| | | | |
|--------|------------|---|--------|
| 100000 | BIT15 | = | 100000 |
| 040000 | BIT14 | = | 40000 |
| 020000 | BIT13 | = | 20000 |
| 010000 | BIT12 | = | 10000 |
| 004000 | BIT11 | = | 4000 |
| 002000 | BIT10 | = | 2000 |
| 001000 | BIT09 | = | 1000 |
| 000400 | BIT08 | = | 400 |
| 000200 | BIT07 | = | 200 |
| 000100 | BIT06 | = | 100 |
| 000040 | BIT05 | = | 40 |
| 000020 | BIT04 | = | 20 |
| 000010 | BIT03 | = | 10 |
| 000004 | BIT02 | = | 4 |
| 000002 | BIT01 | = | 2 |
| 000001 | BIT00 | = | 1 |
| 001000 | BIT9=BIT09 | | |
| 000400 | BIT8=BIT08 | | |
| 000200 | BIT7=BIT07 | | |
| 000100 | BIT6=BIT06 | | |
| 000040 | BIT5=BIT05 | | |

000020 BIT4=BIT04
 000010 BIT3=BIT03
 000004 BIT2=BIT02
 000002 BIT1=BIT01
 000001 BIT0=BIT00

```

.*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC = 4           ;: TIME OUT AND OTHER ERRORS
000010 RESVEC = 10        ;: RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC = 14       ;: "T" BIT
000014 TRTVEC = 14       ;: TRACE TRAP
000014 BPTVEC = 14       ;: BREAKPOINT TRAP (BPT)
000020 IOTVEC = 20        ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC = 24        ;: POWER FAIL
000030 EMTVEC = 30        ;: EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC = 34       ;: "TRAP" TRAP
000060 TKVEC = 60         ;: TTY KEYBOARD VECTOR
000064 TPVEC = 64         ;: TTY PRINTER VECTOR
000240 PIRQVEC = 240      ;: PROGRAM INTERRUPT REQUEST VECTOR
    
```

514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551

.SBTTL RH CONTROLLER REGISTERS

:CONTROL AND STATUS REGISTER 1 (RMCS1)

```

000100 IE = 100           ;: INTERRUPT ENABLE (BIT #6)
000200 RDY = 200         ;: READY (BIT #7)
000400 A16 = 400         ;: HIGH ORDER BUS ADDRESS BIT (BIT #8)
001000 A17 = 1000        ;: HIGH ORDER BUS ADDRESS BIT (BIT #9)
002000 PSEL = 2000       ;: PORT SELECT (BIT #10)
020000 MCPE = 20000      ;: MASSBUS PARITY ERROR (BIT #13)
040000 TRE = 40000       ;: TRANSFER ERROR (BIT #14)
100000 SC = 100000      ;: SPECIAL CONDITION (BIT #15)
    
```

:WORD COUNT REGISTER (RMWC)
 :(EACH BIT IS CALLED BY BIT NUMBER)

:BUS ADDRESS REGISTER (RMBA)
 :(EACH BIT IS CALLED BY BIT NUMBER)

:CONTROL AND STATUS REGISTER 2 (RMCS2)

```

000001 U0 = 1           ;: UNIT SELECT (BIT # )
000002 U1 = 2           ;: UNIT SELECT (BIT # )
000004 U3 = 4           ;: UNIT SELECT (BIT # )
000010 BAI = 10         ;: BUS ADDRESS INCREMENT IN BIT (BIT #3)
000020 PAT = 20         ;: MASSBUS PARITY TEST (BIT #4)
000040 CLR = 40         ;: CLEAR (BIT #5)
000100 IR = 100         ;: INPUT READY (BIT #6)
000200 OR = 200         ;: OUTPUT READY (BIT #7)
000400 MDPE = 400       ;: MASS BUS PARITY ERROR (BIT #8)
001000 MXF = 1000       ;: MISSED TRANSFER ERROR (BIT #9)
002000 PGE = 2000       ;: PROGRAM ERROR (BIT #10)
004000 NEM = 4000       ;: NON EXISTENT MEMORY (BIT #11)
010000 NED = 10000      ;: NON EXISTENT DRIVE (BIT #12)
020000 UPE = 20000      ;: UNIBUS PARITY ERROR (BIT #13)
040000 WCE = 40000      ;: WRITE CHECK ERROR (BIT #14)
100000 DLT = 100000     ;: DATA LATE (BIT #15)
    
```

```

552
553      ;DATA BUFFER REGISTER (RMDB)
554      ;(EACH BIT IS CALLED BY BIT NUMBER)
555
556      .SBTTL RM80 REGISTERS
557
558      ;CONTROL AND STATUS 1 REGISTER. (#00)
559
560      000001      GO      = 1      ;GO BIT (BIT #0)
561      000002      FO      = 2      ;FUNCTION CODE BIT #1
562      000004      F1      = 4      ;FUNCTION CODE BIT #2
563      000010      F2      = 10     ;FUNCTION CODE BIT #3
564      000020      F3      = 20     ;FUNCTION CODE BIT #4
565      000040      F4      = 40     ;FUNCTION CODE BIT #5
566      004000      DVA      = 4000   ;DEVICE AVAILABLE (BIT #11)
567
568      ;DRIVE STATUS REGISTER (RMD5) (#01)
569
570      ;DFS      = 1      DRIVE FORWARD 5''/SEC. (BIT #0)
571      000002      DFF20   = 2      ;DRIVE FORWARD 20''/SEC. (BIT #1)
572      000004      DIGB    = 4      ;DRIVE TO INNER GUARD BAND (BIT #2)
573      000010      GRV     = 10     ;GO REVERSE (BIT #3)
574      000020      DL64    = 20     ;DIFFERENCE LESS THAN 64 (BIT #4)
575      000040      DE1     = 40     ;DIFFERENCE EQUALS 1 (BIT #5)
576      000100      VV      = 100    ;VOLUME VALID (BIT #6)
577      000200      DRY     = 200    ;DRIVE READY (BIT #7)
578      000400      DPR     = 400    ;DRIVE PRESENT (BIT #8)
579      001000      PGM     = 1000   ;PROGRAMABLE (BIT #9)
580      002000      LBT     = 2000   ;LAST SECTOR TRANSFERRED (BIT #10)
581      004000      WRL     = 4000   ;WRITE LOCK (BIT #11)
582      010000      MOL     = 10000  ;MEDIUM ON-LINE (BIT #12)
583      020000      PIP     = 20000  ;POSITIONING OPERATION IN PROGRESS (BIT #13)
584      040000      ERR     = 40000  ;COMPOSITE ERROR (BIT #14)
585      100000      ATA     = 100000 ;ATTENTION ACTIVE (BIT #15)
586
587      ;ERROR REGISTER #01 (RMER1) (#02)
588
589      000001      ILF     = 1      ;ILLEGAL FUNCTION (BIT #0)
590      000002      ILR     = 2      ;ILLEGAL REGISTER (BIT #1)
591      000004      RMR     = 4      ;REGISTER MODIFICATION REFUSED (BIT #2)
592      000010      PAR     = 10     ;PARITY ERROR (BIT #3)
593      000020      FER     = 20     ;FORMAT ERROR (BIT #4)
594      000040      WCF     = 40     ;WRITE CLOCK FAIL (BIT #5)
595      000100      ECH     = 100    ;ECC HARD ERROR (BIT #6)
596      000200      HCE     = 200    ;HEADER COMPARE ERROR (BIT #7)
597      000400      HCRC    = 400    ;HEADER CRC ERROR (BIT #8)
598      001000      AOE     = 1000   ;ADDRESS OVERFLOW ERROR (BIT #9)
599      002000      IAE     = 2000   ;INVALID ADDRESS ERROR (BIT #10)
600      004000      WLE     = 4000   ;WRITE LOCK ERROR (BIT #11)
601      010000      DTE     = 10000  ;DRIVE TIMING ERROR (BIT #12)
602      020000      OPI     = 20000  ;OPERATION INCOMPLETE (BIT #13)
603      040000      UNS     = 40000  ;DRIVE UNSAFE (BIT #14)
604      100000      DCK     = 100000 ;DATA CHECK ERROR (BIT 15)
605
606      ;MAINTAINABILITY REGISTER (RMMR1) (#03)
607
608      000001      DMD     = 1      ;DIAGINOSTIC MODE (BIT #0)
  
```


| | | | |
|-----|--------|---|--|
| 609 | | | |
| 610 | | :ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04) | |
| 611 | | | |
| 612 | 000001 | AT0 = 1 | :DEVICE 0 (BIT #0) |
| 613 | 000002 | AT1 = 2 | :DEVICE 1 (BIT #1) |
| 614 | 000004 | AT2 = 4 | :DEVICE 2 (BIT #2) |
| 615 | 000010 | AT3 = 10 | :DEVICE 3 (BIT #3) |
| 616 | 000020 | AT4 = 20 | :DEVICE 4 (BIT #4) |
| 617 | 000040 | AT5 = 40 | :DEVICE 5 (BIT #5) |
| 618 | 000100 | AT6 = 100 | :DEVICE 6 (BIT #6) |
| 619 | 000200 | AT7 = 200 | :DEVICE 7 (BIT #7) |
| 620 | | | |
| 621 | | :DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05) | |
| 622 | | :(EACH BIT IS CALLED BY BIT NUMBER) | |
| 623 | | | |
| 624 | | :DRIVE TYPE REGISTER (RMDT) (#06) | |
| 625 | | | |
| 626 | 000001 | DT00 = 1 | :DRIVE TYPE NUMBER BIT 1 |
| 627 | 000002 | DT01 = 2 | :DRIVE TYPE NUMBER BIT 2 |
| 628 | 000004 | DT02 = 4 | :DRIVE TYPE NUMBER BIT 3 |
| 629 | 000010 | DT03 = 10 | :DRIVE TYPE NUMBER BIT 4 |
| 630 | 000020 | DT04 = 20 | :DRIVE TYPE NUMBER BIT 5 |
| 631 | 000040 | DT05 = 40 | :DRIVE TYPE NUMBER BIT 6 |
| 632 | 000100 | DT06 = 100 | :DRIVE TYPE NUMBER BIT 7 |
| 633 | 000200 | DT07 = 200 | :DRIVE TYPE NUMBER BIT 8 |
| 634 | 000400 | DT08 = 400 | :DRIVE TYPE NUMBER BIT 9 |
| 635 | 004000 | DRQ = 4000 | :DRIVE REQUEST REQUIRED (BIT #11) |
| 636 | 020000 | MOH = 20000 | :MOVING HEAD (BIT #12) |
| 637 | 040000 | TAP = 40000 | :TAPE DRIVE (BIT #14) |
| 638 | 100000 | NBA = 100000 | :NOT BLOCK ADDRESSED (BIT #15) |
| 639 | | | |
| 640 | | :LOOK-AHEAD REGISTER (RMLA) (#07) | |
| 641 | | | |
| 642 | 000100 | SC0 = 100 | :SECTOR COUNT FIELD 0 (BIT #6) |
| 643 | 000200 | SC1 = 200 | :SECTOR COUNT FIELD 1 (BIT #7) |
| 644 | 000400 | SC2 = 400 | :SECTOR COUNT FIELD 2 (BIT #8) |
| 645 | 001000 | SC3 = 1000 | :SECTOR COUNT FIELD 3 (BIT #9) |
| 646 | 002000 | SC4 = 2000 | :SECTOR COUNT FIELD 4 (BIT #10) |
| 647 | | | |
| 648 | | :RM ERROR REGISTER #2 (RMER2) (#10) | |
| 649 | | | |
| 650 | 000010 | DPE = 10 | :DATA PARITY ERROR (BIT #3) |
| 651 | 000200 | DVC = 200 | :DEVICE CHECK (BIT #7) |
| 652 | 002000 | LBC = 2000 | :LOSS OF BIT CLOCK (BIT #10) |
| 653 | 004000 | LSC = 4000 | :LOSS OF SYSTEM CLOCK (BIT #11) |
| 654 | 010000 | IVC = 10000 | :INVALID COMMAND (BIT #12) |
| 655 | 020000 | OPE = 20000 | :OPERATOR ERROR (BIT #13) |
| 656 | 100000 | SKI = 100000 | :SEEK INCOMPLETE (BIT #14) |
| 657 | | | |
| 658 | | :OFFSET REGISTER (RMOF) (#11) | |
| 659 | | | |
| 660 | 000200 | OFD = 200 | :OFFSET FORWARD (BIT #5) |
| 661 | 002000 | HCI = 2000 | :HEADER COMPARE INHIBIT (BIT #10) |
| 662 | 004000 | ECI = 4000 | :ERROR CORRECTION CODE INHIBIT (BIT #11) |
| 663 | 010000 | FMT16 = 10000 | :FORMAT BIT (BIT #12) |
| 664 | | | |
| 665 | | :DESIRED CYLINDER ADDRESS (RMDC) (#12) | |

RM80 REGISTERS

666 : (EACH BIT IS CALLED BY BIT NUMBER)
 667
 668 : SERIAL NUMBER REGISTER (RMSN) (#14)
 669 : (EACH IS CALLED BY BIT NUMBER)
 670
 671 : ECC POSITION REGISTER (RMEC1) (#16)
 672 : (EACH BIT IS CALLED BY BIT NUMBER)
 673
 674 : ECC PATTERN REGISTER (RMEC2) (#17)
 675 : (EACH BIT IS CALLED BY BIT NUMBER)
 676

.SBTTL DEFINITIONS OF THE RH/RM ADDRESS INDEXES

| | | | |
|-----|--------|------------|---|
| 678 | | | |
| 679 | 000000 | RMCS1 = 0 | : CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00) |
| 680 | 000002 | RMWC = 2 | : WORD COUNT REGISTER (NOT A DRIVE REG) |
| 681 | 000004 | RMBA = 4 | : UNIBUS ADDRESS REGISTER (NOT A DRIVE REG) |
| 682 | 000006 | RMDA = 6 | : DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05) |
| 683 | 000010 | RMCS2 = 10 | : CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG) |
| 684 | 000012 | RMDS = 12 | : DRIVE STATUS REGISTER (DRIVE REG 01) |
| 685 | 000014 | RMER1 = 14 | : ERROR REGISTER #1 (DRIVE REG. 02) |
| 686 | 000016 | RMAS = 16 | : ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04) |
| 687 | 000020 | RMLA = 20 | : LOOK AHEAD REGISTER (DRIVE REG. 07) |
| 688 | 000022 | RMDB = 22 | : DATA BUFFER REGISTER (NOT A DRIVE REG.) |
| 689 | 000024 | RMMR1 = 24 | : MAINTAINABILITY REGISTER (DRIVE REG. 03) |
| 690 | 000026 | RMDT = 26 | : DRIVE TYPE REGISTER (DRIVE REG. 06) |
| 691 | 000030 | RMSN = 30 | : SERIAL NUMBER REGISTER (DRIVE REG. 10) |
| 692 | 000032 | RMOF = 32 | : OFFSET REGISTER (DRIVE REG. 11) |
| 693 | 000034 | RMDC = 34 | : DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12) |
| 694 | 000040 | RMMR2 = 40 | : MAINTENANCE REGISTER #2 (DRIVE REG. 14) |
| 695 | 000042 | RMER2 = 42 | : ERROR REGISTER #2 (DRIVE REG. 15) |
| 696 | 000044 | RMEC1 = 44 | : ECC POSITION REGISTER (DRIVE REG. 16) |
| 697 | 000046 | RMEC2 = 46 | : ECC PATTERN REGISTER (DRIVE REG. 17) |
| 698 | | | |

1
2
3
4
5
6

000000
000174 000174
000176 000000
000176 000000

000200 000137 001766
000204 000137 001776

000210
000046 013316
000052 020000
000210

```
.SBTTL TRAP CATCHER
      .=0
      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
DISPREG: .=174
SWREG:  .WORD 0          ;;SOFTWARE DISPLAY REGISTER
        .WORD 0          ;;SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)
      JMP @#START        ;;JUMP TO STARTING ADDRESS OF PROGRAM
      JMP @#START1       ;START AND CHANGE THE RH/RM ADDRESS

.SBTTL ACT11 HOOKS
      ;*****
      ;HOOKS REQUIRED BY ACT11
      $SVPC=.            ;SAVE PC
      .=46               ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
      $ENDAD
      .=52               ;;2)SET LOC.52 TO 20000
      .WORD 20000        ;; RESTORE PC
      .=$SVPC
```

0

.SBTTL COMMON TAGS

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

001100 001100
001100 000000
001102 000
001103 000
001104 000000
001106 000000
001110 000000
001112 000000
001114 000
001115 001
001116 000000
001120 000000
001122 000000
001124 000000
001126 000000
001130 000000
001132 000000
001134 000
001135 000
001136 000000
001140 177570
001142 177570
001144 177560
001146 177562
001150 177564
001152 177566
001154 000
001155 002
001156 012
001157 000
001160 000000

001162 000000
001164 000000
001166 000000
001170 000000
001172 000000
001174 000000
001176 000000
001200 000000
001202 207
001206 077
001207 015
001210 012

377 377
000

.=1100
\$CMTAG: .WORD 0
\$PASS: .WORD 0
\$STNM: .BYTE 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
\$SAU`JB: .BYTE 0
\$INTAG: .BYTE 0
\$SWR: .WORD DSWR
\$DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$REGAD: .WORD 0

\$REGO: .WORD 0
\$TMP0: .WORD 0
\$TMP1: .WORD 0
\$TMP2: .WORD 0
\$TMP3: .WORD 0
\$TMP4: .WORD 0
\$TIMES: 0
\$ESCAPE: 0
\$BELL: .ASCIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

:: START OF COMMON TAGS
:: CONTAINS PASS COUNT
:: CONTAINS THE TEST NUMBER
:: CONTAINS ERROR FLAG
:: CONTAINS SUBTEST ITERATION COUNT
:: CONTAINS SCOPE LOOP ADDRESS
:: CONTAINS SCOPE RETURN FOR ERRORS
:: CONTAINS TOTAL ERRORS DETECTED
:: CONTAINS ITEM CONTROL BYTE
:: CONTAINS MAX. ERRORS PER TEST
:: CONTAINS PC OF LAST ERROR INSTRUCTION
:: CONTAINS ADDRESS OF 'GOOD' DATA
:: CONTAINS ADDRESS OF 'BAD' DATA
:: CONTAINS 'GOOD' DATA
:: CONTAINS 'BAD' DATA
:: RESERVED--NOT TO BE USED

:: AUTOMATIC MODE INDICATOR
:: INTERRUPT MODE INDICATOR

:: ADDRESS OF SWITCH REGISTER
:: ADDRESS OF DISPLAY REGISTER
:: TTY KBD STATUS
:: TTY KBD BUFFER
:: TTY PRINTER STATUS REG. ADDRESS
:: TTY PRINTER BUFFER REG. ADDRESS
:: CONTAINS NULL CHARACTER FOR FILLS
:: CONTAINS # OF FILLER CHARACTERS REQUIRED
:: INSERT FILL CHARS. AFTER A 'LINE FEED'
:: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
:: CONTAINS THE ADDRESS FROM
:: WHICH (\$REGO) WAS OBTAINED
:: CONTAINS ((\$REGAD)+0)
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: USER DEFINED
:: MAX. NUMBER OF ITERATIONS
:: ESCAPE ON ERROR ADDRESS
:: CODE FOR BELL
:: QUESTION MARK
:: CARRIAGE RETURN
:: LINE FEED
::*****

.SBTTL USER DEFINED TAGS

| | | | | |
|--------|--------|----------------|--------|--|
| 001212 | 172540 | \$LKCSR: .WORD | 172540 | :ADDR OF KW11-P STATUS REGISTER |
| 001214 | 172542 | \$LKCSB: .WORD | 172542 | :ADDR OF KW11-P COUNTER BUFFER |
| 001216 | 000104 | \$LPVEC: .WORD | 104 | :ADDR OF KW11-P VECTOR |
| 001220 | 177546 | \$LKS: .WORD | 177546 | :ADDR OF KW11-L STATUS REGISTER |
| 001222 | 000100 | \$LLVEC: .WORD | 100 | :ADDR OF KW11-L VECTOR |
| 001224 | 000000 | PORTA: .WORD | 0 | :ADDRESS OF PORT A |
| 001226 | 000000 | PORTB: .WORD | 0 | :ADDRESS OF PORT B |
| 001230 | 000000 | PORTC: .WORD | 0 | :ADDRESS OF DIFFERENT DRIVE |
| 001232 | 000000 | ASR1: .WORD | 0 | :ATA-A OR ATA-B = 1 |
| 001234 | 000000 | PTNBR: .WORD | 0 | :CONTAINS THE PORT ADDRESS FOR ERROR TYPEOUTS |
| 001236 | 000000 | SEIZPT: .WORD | 0 | :CONTAINS THE ADDRESS OF THE SEIZING PORT |
| 001240 | 000000 | OPPR: .WORD | 0 | :CONTAINS THE ADDRESS OF THE 'OPPOSITE' PORT |
| 001242 | 000000 | TSTNUM: .WORD | 0 | :NUMBER OF THE CURRENT TEST |
| 001244 | 000000 | CKERR: .WORD | 0 | :IF -1, A REGISTER MISCOMPARISON OCCURRED |
| 001246 | 000000 | NOSEIZ: .WORD | 0 | :IF -1, THE PORT IN 'SEIZPT' DID NOT SEIZE THE DRIVE |
| 001250 | 000000 | RELERR: .WORD | 0 | :IF -1, THE PORT IN 'SEIZPT' DID NOT RELEASE THE DRIVE |
| 001252 | 000000 | TIME: .WORD | 0 | :ELAPSED TIME COUNTER |
| 001254 | 000000 | WATCH: .WORD | 0 | :WATCH DC'S TIMER LOCATION |
| 001256 | 000000 | TIMEA: .WORD | 0 | :THE TIMEOUT ONE-SHOT VALUE MEASURED THROUGH PORT A |
| 001260 | 000000 | TIMEAP: .WORD | 0 | :PORT A TIMEOUT VALUE + 25% |
| 001262 | 000000 | TIMEB: .WORD | 0 | :THE TIMEOUT ONE-SHOT VALUE MEASURED THROUGH PORT B |
| 001264 | 000000 | TIMEBP: .WORD | 0 | :PORT B TIMEOUT VALUE + 25% |
| 001266 | 000000 | KYBCTL: .WORD | 0 | :SINGLE TEST INDICATOR |
| 001270 | 000000 | CHGADR: .WORD | 0 | :CHANGE THE RH/RM ADDRESS INDICATOR |

.SBTTL RH/RM UNIBUS AND VECTOR ADDRESSES

| | | | | |
|--------|--------|----------------|--------|---------------------------|
| 001272 | 176700 | \$RMADR: .WORD | 176700 | :RH/RM UNIBUS ADDRESS |
| 001274 | 000254 | \$RMVEC: .WORD | 254 | :INTERRUPT VECTOR ADDRESS |

.SBTTL ERROR POINTER TABLE

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

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

| | | | |
|------------------|----------|-----|--|
| 001276 | \$ERRTB: | | |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 001276 020431 | | EM1 | ;DRIVE IS NON-EXISTENT ('NED' BIT SET) |
| 5 001300 023174 | | DH1 | |
| 6 001302 024544 | | DT1 | |
| 7 001304 024764 | | DF1 | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 001306 020477 | | EM2 | ;WRONG DRIVE TYPE |
| 12 001310 023245 | | DH2 | |
| 13 001312 024560 | | DT2 | |
| 14 001314 024771 | | DF2 | |
| 15 | | | |
| 16 | | | |
| 17 | | | |
| 18 001316 020520 | | EM3 | ;PORT SELECT SWITCH ON DRIVE NOT IN 'A/B' |
| 19 001320 023174 | | DH1 | |
| 20 001322 024544 | | DT1 | |
| 21 001324 024764 | | DF1 | |
| 22 | | | |
| 23 | | | |
| 24 | | | |
| 25 001326 020571 | | EM4 | ;DRIVE NOT ON LINE |
| 26 001330 023245 | | DH2 | |
| 27 001332 024560 | | DT2 | |
| 28 001334 024771 | | DF2 | |
| 29 | | | |
| 30 | | | |
| 31 | | | |
| 32 001336 020613 | | EM5 | ;SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME |
| 33 001340 023321 | | DH5 | |
| 34 001342 024576 | | DT5 | |
| 35 001344 024777 | | DF5 | |
| 36 | | | |
| 37 | | | |
| 38 | | | |
| 39 001346 020675 | | EM6 | ;TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS |
| 40 001350 023370 | | DH6 | |
| 41 001352 024612 | | DT6 | |
| 42 001354 025004 | | DF6 | |

| | | | |
|----|--------|--------|--|
| 43 | | | |
| 44 | | | ;ERROR 7 |
| 45 | | | |
| 46 | 001356 | 020747 | EM7 ;TIMEOUT ONE-SHOT IS LESS THAN 500 MS |
| 47 | 001360 | 023417 | DH7 |
| 48 | 001362 | 024622 | DT7 |
| 49 | 001364 | 025007 | DF7 |
| 50 | | | |
| 51 | | | ;ERROR 10 |
| 52 | | | |
| 53 | 001366 | 021014 | EM10 ;READIN PRESET DOES NOT SET VOLUME VALID FOR THE PORT |
| 54 | 001370 | 023174 | DH1 |
| 55 | 001372 | 024544 | DT1 |
| 56 | 001374 | 024764 | DF1 |
| 57 | | | |
| 58 | | | ;ERROR 11 |
| 59 | | | |
| 60 | 001376 | 021102 | EM11 ;'GO' BIT RESET DURING UNLOAD COMMAND |
| 61 | 001400 | 023174 | DH1 |
| 62 | 001402 | 024544 | DT1 |
| 63 | 001404 | 024764 | DF1 |
| 64 | | | |
| 65 | | | ;ERROR 12 |
| 66 | | | |
| 67 | 001406 | 021147 | EM12 ;INCORRECT STATUS DURING UNLOAD COMMAND |
| 68 | 001410 | 023174 | DH1 |
| 69 | 001412 | 024544 | DT1 |
| 70 | 001414 | 024764 | DF1 |
| 71 | | | |
| 72 | | | ;ERROR 13 |
| 73 | | | |
| 74 | 001416 | 021216 | EM13 ;DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND |
| 75 | 001420 | 023464 | DH13 |
| 76 | 001422 | 024634 | DT13 |
| 77 | 001424 | 025004 | DF6 |
| 78 | | | |
| 79 | | | ;ERROR 14 |
| 80 | | | |
| 81 | 001426 | 021303 | EM14 ;ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD |
| 82 | 001430 | 023541 | DH14 |
| 83 | 001432 | 024644 | DT14 |
| 84 | 001434 | 025013 | DF14 |
| 85 | | | |
| 86 | | | ;ERROR 15 |
| 87 | | | |
| 88 | 001436 | 021365 | EM15 ;ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD' |
| 89 | 001440 | 023174 | DH1 |
| 90 | 001442 | 024544 | DT1 |
| 91 | 001444 | 024764 | DF1 |
| 92 | | | |
| 93 | | | ;ERROR 16 |
| 94 | | | |
| 95 | 001446 | 021451 | EM16 ;DRIVE NOT IN NEUTRAL AFTER UNLOAD WITH 'PORT |
| 96 | | | ;SELECT' SWITCH MOVED FROM 'A/B' |
| 97 | 001450 | 023464 | DH13 |
| 98 | 001452 | 024634 | DT13 |
| 99 | 001454 | 025004 | DF6 |

| | | | | |
|-----|--------|--------|-----------|--|
| 100 | | | | |
| 101 | | | :ERROR 17 | |
| 102 | | | | |
| 103 | 001456 | 021571 | EM17 | :DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP |
| 104 | 001460 | 023660 | DH17 | |
| 105 | 001462 | 024662 | DT17 | |
| 106 | 001464 | 025021 | DF17 | |
| 107 | | | | |
| 108 | | | :ERROR 20 | |
| 109 | | | | |
| 110 | 001466 | 021654 | EM20 | :DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP |
| 111 | 001470 | 023660 | DH17 | |
| 112 | 001472 | 024662 | DT17 | |
| 113 | 001474 | 025021 | DF17 | |
| 114 | | | | |
| 115 | | | :ERROR 21 | |
| 116 | | | | |
| 117 | 001476 | 021737 | EM21 | :STATUS INCORRECT FOR PORT AFTER CYCLE UP |
| 118 | 001500 | 023174 | DH1 | |
| 119 | 001502 | 024544 | DT1 | |
| 120 | 001504 | 024764 | DF1 | |
| 121 | | | | |
| 122 | | | :ERROR 22 | |
| 123 | | | | |
| 124 | 001506 | 022010 | EM22 | :REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT |
| 125 | 001510 | 023174 | DH1 | |
| 126 | 001512 | 024544 | DT1 | |
| 127 | 001514 | 024764 | DF1 | |
| 128 | | | | |
| 129 | | | :ERROR 23 | |
| 130 | | | | |
| 131 | 001516 | 022106 | EM23 | : 'NED' SET WHEN RMDS ACCESSED THROUGH PORT NOT SWITCHED |
| 132 | 001520 | 023174 | DH1 | |
| 133 | 001522 | 024544 | DT1 | |
| 134 | 001524 | 024764 | DF1 | |
| 135 | | | | |
| 136 | | | :ERROR 24 | |
| 137 | | | | |
| 138 | 001526 | 022201 | EM24 | :DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/ |
| 139 | 001530 | 023677 | DH24 | |
| 140 | 001532 | 024670 | DT24 | |
| 141 | 001534 | 025007 | DF7 | |
| 142 | | | | |
| 143 | | | :ERROR 25 | |
| 144 | | | | |
| 145 | 001536 | 022261 | EM25 | :RH/RM DIDN'T RESPOND TO ADDRESSING |
| 146 | 001540 | 024002 | DH25 | |
| 147 | 001542 | 024702 | DT25 | |
| 148 | 001544 | 025023 | DF25 | |
| 149 | | | | |
| 158 | | | :ERROR 26 | |
| | 001546 | 000000 | 0 | :UNUSED ERROR MESSAGES |
| | 001550 | 000000 | 0 | |
| | 001552 | 000000 | 0 | |
| | 001554 | 000000 | 0 | |

| | | | | |
|-----|--------|--------|-----------|--|
| | | | :ERROR 27 | |
| | 001556 | 000000 | 0 | :UNUSED ERROR MESSAGES |
| | 001560 | 000000 | 0 | |
| | 001562 | 000000 | 0 | |
| | 001564 | 000000 | 0 | |
| 159 | | | :ERROR 30 | |
| 160 | | | | |
| 161 | 001566 | 022324 | EM30 | :DRIVE NOT SEIZED BY PORT 'N' |
| 162 | 001570 | 024011 | DH30 | |
| 163 | 001572 | 024706 | DT30 | |
| 164 | 001574 | 025024 | DF30 | |
| 165 | | | | |
| 166 | | | :ERROR 31 | |
| 167 | | | | |
| 168 | 001576 | 022355 | EM31 | :WRONG STATUS SEEN BY THE SEIZING PORT |
| 169 | 001600 | 023245 | DH2 | |
| 170 | 001602 | 024560 | DT2 | |
| 171 | 001604 | 024771 | DF2 | |
| 172 | | | | |
| 173 | | | :ERROR 32 | |
| 174 | | | | |
| 175 | 001606 | 022423 | EM32 | :REGISTER CONTENTS INCORRECT |
| 176 | 001610 | 023245 | DH2 | |
| 177 | 001612 | 024560 | DT2 | |
| 178 | 001614 | 024771 | DF2 | |
| 179 | | | | |
| 180 | | | :ERROR 33 | |
| 181 | | | | |
| 182 | 001616 | 022453 | EM33 | :CONTROL BUS PARITY ERROR WHILE READING REGISTER |
| 183 | 001620 | 023174 | DH1 | |
| 184 | 001622 | 024544 | DT1 | |
| 185 | 001624 | 024764 | DF1 | |
| 186 | | | | |
| 187 | | | :ERROR 34 | |
| 188 | | | | |
| 189 | 001626 | 022537 | EM34 | :CAN'T ACCESS DRIVE THROUGH EITHER PORT |
| 190 | 001630 | 024133 | DH34 | |
| 191 | 001632 | 024726 | DT34 | |
| 192 | 001634 | 025033 | DF34 | |
| 193 | | | | |
| 194 | | | :ERROR 35 | |
| 195 | | | | |
| 196 | 001636 | 022606 | EM35 | :DRIVE NOT IN NEUTRAL AFTER RELEASE, REQUEST NOT SET |
| 197 | 001640 | 024227 | DH35 | |
| 198 | 001642 | 024740 | DT35 | |
| 199 | 001644 | 025007 | DF7 | |
| 200 | | | | |
| 201 | | | :ERROR 36 | |
| 202 | | | | |
| 203 | 001646 | 022673 | EM36 | :DRIVE NOT IN NEUTRAL AFTER TIMEOUT, REQUEST NOT SET |
| 204 | 001650 | 024227 | DH35 | |
| 205 | 001652 | 024740 | DT35 | |
| 206 | 001654 | 025007 | DF7 | |
| 207 | | | | |
| 208 | | | :ERROR 37 | |

| | | | | |
|-----|--------|--------|------|--|
| 209 | | | | |
| 210 | 001656 | 022760 | EM37 | ;REGISTER CONTENTS INCORRECT AFTER RELEASE/TIMEOUT |
| 211 | 001660 | 024324 | DH37 | |
| 212 | 001662 | 024706 | DT30 | |
| 213 | 001664 | 025024 | DF30 | |
| 214 | | | | |
| 215 | | | | ;ERROR 40 |
| 216 | | | | |
| 217 | 001666 | 023041 | EM40 | ;DRIVE NOT SEIZED BY PORT AFTER RELEASE WITH REQUEST SET |
| 218 | 001670 | 024446 | DH40 | |
| 219 | 001672 | 024752 | DT40 | |
| 220 | 001674 | 025007 | DF7 | |
| 221 | | | | |
| 222 | | | | ;ERROR 41 |
| 223 | | | | |
| 224 | 001676 | 023116 | EM41 | ;REGISTER WRONG AFTER RELEASE WITH REQUEST SET |
| 225 | 001700 | 024011 | DH30 | |
| 226 | 001702 | 024706 | DT30 | |
| 227 | 001704 | 025024 | DF30 | |
| 228 | | | | |

ERROR POINTER TABLE

```

1          ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 001706 011600 BADTMO: MOV      (SP),R0          ;SAVE PC WHERE THE TIME OUT OCCURED
4 001710 005740      TST      -(R0)          ;ADJUST PC -2
5 001712 022626      CMP      (SP)+,(SP)+    ;RESTORE STACK POINTER
6 001714 104401 001722      TYPE    ,65$      ;:TYPE ASCIZ STRING
   001720 000417      BR       64$          ;:GET OVER THE ASCIZ
   ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
   64$:
7 001760 010046      MOV      R0,-(SP)      ;SETUP FOR TYPING OUT PC
8 001762 104402      TYPOC
9 001764 000240      NOP
   ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
   ;TO STOP ON UNEXPECTED TIMEOUT.
10
11
12          .SBTTL  START OF PROGRAM
13
14 001766 000240      START:  NOP
15 001770 0C5037 001270      CLR      CHGADR          ;CLEAR THE 'CHANGE RH/RM ADDRESS' INDICATOR
16 001774 000403      BR       START2          ;GO TO THE START
17 001776 012737 177777 001270  START1: MOV     #-1,CHGADR      ;SET THE 'CHANGE RH/RM ADDRESS' INDICATOR
18
19 002004 005227 000000      START2: INC     #0          ;TTY LOOP, WAIT FOR INCREMENT
20 002010 001375      BNE     -4          ;OF WORD
21 002012 000005      RESET
   ;CLEAR THE WORLD
22
23          .SBTTL  INITIALIZE THE COMMON TAGS
   ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
   MOV     #SCMTAG,R6      ;:FIRST LOCATION TO BE CLEARED
   CLR     (R6)+          ;:CLEAR MEMORY LOCATION
   CMP     #SWR,R6        ;:DONE?
   BNE     -6            ;:LOOP BACK IF NO
   MOV     #STACK,SP     ;:SE 'JP THE STACK POINTER
   ;:INITIALIZE A FEW VECTORS
   MOV     #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
   MOV     #340,@IOTVEC+2 ;:LEVEL 7
   MOV     #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
   MOV     #340,@EMTVEC+2 ;:LEVEL 7
   MOV     #STRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
   MOV     #340,@TRAPVEC+2 ;:LEVEL 7
   MOV     SENDCT,SEOPCT ;:SETUP END-OF-PROGRAM COUNTER
   CLR     $TIMES        ;:INITIALIZE NUMBER OF ITERATIONS
   CLR     $ESCAPE       ;:CLEAR THE ESCAPE ON ERROR ADDRESS
   MOV     #1,$ERMAX     ;:ALLOW ONE ERROR PER TEST
   MOV     #,$SLPADR     ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
   MOV     #,$SLPERR     ;:SETUP THE ERROR LOOP ADDRESS
   ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
   MOV     @ERRVEC,-(SP) ;:SAVE ERROR VECTOR
   MOV     #64$,@ERRVEC  ;:SET UP ERROR VECTOR
   MOV     #DSWR,SWR     ;:SETUP FOR A HARDWARE SWICH REGISTER
   MOV     #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
   CMP     #-1,@SWR      ;:TRY TO REFERENCE HARDWARE SWR
   BNE     66$          ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
   ;:AND THE HARDWARE SWR IS NOT = -1
   BR     65$          ;:BRANCH IF NO TIMEOUT
   64$:  MOV     #65$,(SP) ;:SET UP FOR TRAP RETURN
   RTI

```

```

002206 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
002214 012737 000174 001142 MOV #DISPREG,DISPLAY
002222 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR

24 ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 002226 012737 001706 000004 MOV #BADTMO,ERRVEC ;SETUP FOR UNEXPECTED TIMEOUT
26 002234 012737 000300 000006 MOV #PR6,ERRVEC+2 ;LEVEL 6
27
28 .SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002242 005227 177777 INC #-1 ;;FIRST TIME?
002246 001032 BNE 67$ ;;BRANCH IF NO
002250 022737 013316 000042 CMP #SENDAD,@#42 ;;ACT-11?
002256 001426 BEQ 67$ ;;BRANCH IF YES
002260 104401 002266 TYPE ,68$ ;;TYPE ASCIZ STRING
002264 000423 BR 67$ ;;GET OVER THE ASCIZ
;;68$: .ASCIZ <CRLF>@CZRNIAO - RM80 DUAL PORT TEST, PT 2@<CRLF>
67$:
;.BTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
002334 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
002340 001006 BNE 69$ ;;BRANCH IF YES
002342 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
002350 001005 BNE 70$ ;;BRANCH IF NO
002352 104406 GTSWR ;;GET SOFT-SWR SETTINGS
002354 000403 BR 70$
002356 112737 000001 001134 69$: MOV #1,$AUTOB ;;SET AUTO-MODE INDICATOR
70$:
1$: JSR PC,$TKINT ;SETUP THE TTY KEYBOARD
JSR PC,CHANGE ;CHECK/CHANGE THE RH/RM ADDRESS
TYPE ,ENTERA ;ENTER DRIVE ADDRESS
RDOCT ;GET THE ADDRESS
MOV (SP)+,PORTA ;STORE THE ADDRESS
34 002406 023727 001224 000007 CMP PORTA,#7 ;SEE IF ADDRESS TOO LARGE
35 002414 101403 BLOS 2$ ;BR IF NOT
36 002416 104401 017605 TYPE ,ADRERR ;TYPE ADDRESS ERROR MESSAGE
37 002422 000760 BR 1$ ;TRY AGAIN
38 002424 013737 001224 001226 2$: MOV PORTA,PORTB ;GENERATE THE PORT B ADDRESS
39 002432 005237 001226 INC PORTB ;INCREMENT THE ADDRESS
40 002436 042737 000016 001226 BIC #16,PORTB ;LEAVE BIT 0
41 002444 013746 001224 MOV PORTA,-(SP) ;PUT PORT A ADDRESS ON THE STACK
42 002450 042716 177771 BIC #^C6,(SP) ;SAVE BITS 1 & 2
43 002454 052637 001226 BIS (SP)+,PORTB ;SET BITS 1 & 2 IN PORT B ADDRESS
44 002460 104401 017630 TYPE ,PORTAIS ;'PORT A ADDRESS IS '
45 002464 013746 001224 MOV PORTA,-(SP) ;;SAVE PORTA FOR TYPEOUT
;;TYPE PORT A ADDRESS
;;GO TYPE--OCTAL ASCII
002470 104403 TYPOS
002472 001 .BYTE 1 ;;TYPE 1 DIGIT(S)
002473 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
46 002474 104401 017657 TYPE ,PORTBIS ;'PORT B ADDRESS IS '
47 002500 013746 001226 MOV PORTB,-(SP) ;;SAVE PORTB FOR TYPEOUT
;;TYPE PORT B ADDRESS
;;GO TYPE--OCTAL ASCII
002504 104403 TYPOS
002506 001 .BYTE 1 ;;TYPE 1 DIGIT(S)
002507 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
48 002510 104401 001207 TYPE ,$CRLF ;ANOTHER CR-LF
49 002514 013737 001224 001230 MOV PORTA,PORTC ;GENERATE ADDRESS OF DRIVE NOT TESTED
50 002522 062737 000006 001230 ADD #6,PORTC ;COMPLEMENT SOME BITS

```

```

51 002530 042737 177770 001230      BIC      #*C7,PORTC      ;SAVE ONLY LOWER BITS
52 002536 013701 001224                MOV      PORTA,R1        ;USE PORT A ADDRESS AS INDEX
53 002542 116137 025060 001232      MOVVB   ATABIT(R1),ASR1 ;GET ATTENTION BIT FOR DRIVE
56 002550 005037 001256                CLR      TIMEA           ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
    002554 005037 001260                CLR      TIMEAP          ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
    002560 005037 001262                CLR      TIMEB           ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
    002564 005037 001264                CLR      TIMEBP          ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
57 002570 004737 013336                JSR      PC,CKCLK        ;SETUP CLOCK
58 002574 000137 002610                JMP      EXEC            ;CLOCK HAS BEEN STARTED
59 002600 104401 017706                TYPE    ,NOCLOCK        ;NO CLOCK ON SYSTEM
60 002604 000000 3$: HALT                ;FATAL ERROR
61 002606 000776                BR      3$              ;INTERLOCK THE HALT
62
63                                     ;ROUTINE TO GET THE TEST NUMBER FROM THE OPERATOR
64
65 002610 000005                EXEC:   RESET           ;CLEAR EVERYTHING
66 002612 005037 177776                CLR      PS             ;CLEAR THE PROCESSOR STATUS WORD
67 002616 104401 001207                TYPE    ,$CRLF         ;CR-LF
68 002622 013700 001272                MOV      $RMADR,R0      ;RH/RM ADDRESS FOR INDEXING
69 002626 012706 001100                MOV      #STACK,SP     ;LOAD STACK POINTER
70 002632 004737 013336                JSR      PC,CKCLK        ;START THE CLOCK
71 002636 000240                NOP                    ;RETURN IF NO CLOCK
72 002640 004737 015774                JSR      PC,$TKINT      ;INITIALIZE THE KEYBOARD
73 002644 005037 001266                CLR      KYBCTL         ;CLEAR SINGLE TEST INDICATOR
74 002650 005037 001100                CLR      $PASS         ;CLEAR THE PASS COUNT
75 002654 112737 000001 001115      MOVVB   #1,$ERMAX      ;SET ERROR MAX TO 1
76 002662 012737 002662 001106      MOV      #,$SLPADR     ;INITIAL SETTING FOR LOOP ADDRESS
77 002670 012737 002670 001110      MOV      #,$SLPERR     ;INITIAL SETTING FOR LOOP ON ERROR ADDRESS
78
79 002676 104401 017753                1$:    TYPE    ,TESTNO  ;ASK FOR TEST NUMBER
80 002702 104412                RDOCT                ;GET THE NUMBER
81 002704 012601                MOV      (SP)+,R1      ;PUT ENTRY INTO R1
82 002706 001002                BNE     2$             ;BR IF NOT ZERO
83 002710 000137 003074                JMP      TST1AA        ;ENTER ZERO - PERFORM ALL TESTS
84
85 002714 020137 025070                2$:    CMP      R1,MAXTN ;SEE IF NUMBER GREATER THAN MAXIMUM
86 002720 003403                BLE     3$             ;BR IF LESS OR EQUAL
87 002722 104401 017773                TYPE    ,BADNO        ;BAD ENTRY
88 002726 000763                BR      1$             ;TRY AGAIN
89 002730 005301                3$:    DEC      R1       ;DECREMENT ENTRY
90 002732 006301                ASL     R1             ;SHIFT IT LEFT
91 002734 005237 001266                INC     KYBCTL         ;SET SINGLE TEST INDICATOR
92 002740 012737 000001 001104      MOV      #1,$ICNT     ;PRESET ITERATION COUNT
93 002746 012746 000240                MOV      #PR5,-(SP)   ;:PUT NEW PS ON STACK
    002752 012746 002760                MOV      #64$,-(SP)  ;:PUT NEW PC ON STACK
    002756 000002                RTI                    ;:POP NEW PC AND PS
94 002760 000171 025040                64$:   JMP      @TSTADR(R1) ;JUMP TO TEST
95
96                                     ;CHANGE THE RH/RM UNIBUS ADDRESS USED BY THE PROGRAM
97
98 002764 005737 001270                CHANGE: TST   CHGADR    ;CHANGE THE ADDRESS ?
99 002770 001421                BEQ     3$             ;BR IF NOT
100 002772 005037 001270                CLR     CHGADR        ;CLEAR THE INDICATOR
101 002776 104401 020022                1$:    TYPE    ,ADDRIS  ;TYPE OUT WHAT THE PRESENT ADDRESS IS
102 003002 013746 001272                MOV     $RMADR,-(SP)  ;PUT THE ADDRESS ON THE STACK
103 003006 104402                TYPOC                ;TYPE THE ACTUAL ADDRESS
  
```

```

104 003010 104401 001207          TYPE      ,SCLF          :CR-LF
105 003014 104401 020057          TYPE      ,NTRH          :ASK FOR NEW ADDRESS
106 003020 104412                    RDOCT
107 003022 005716                    TST      (SP)           :0 OR 'CR' ENTERED ?
108 003024 001402                    BEQ      2$             :BR IF EITHER ENTERED (NO ADDRESS CHANGE)
109 003026 011637 001272          MOV      (SP), $RMADR   :NEW RH/RM ADDRESS
110 003032 005726                    TST      (SP)+         :CORRECT THE STACK POINTER
111 003034 012737 003054 000004 2$:  MOV      #4$, @#4      :LOAD TRAP ADDRESS
112 003042 013700 001272          MOV      $RMADR, R0    :GET RH/RM ADDRESS
113 003046 005760 000002          TST      RMWC(R0)     :RESPONDS AT THAT ADDRESS ?
114 003052 000404                    BR       5$            :BR IF YES
115 003054                    4$:
116 003054 104025                    EMT      25
117 003056 062706 000004          ADD      #4, SP        :RESET THE STACK POINTER
118 003062 000745                    BR       1$            :GET ADDRESS AGAIN
119 003064 012737 000006 000004 5$:  MOV      #6, @#4      :RESTORE THE VECTOR
119 003072 000207                    RTS      PC            :RETURN

```

```

120
133
134 003074 013700 001272          TST1AA: MOV      $RMADR, R0    ;;RESTORE R0 AFTER END OF PASS
135 003100 012746 000240          MOV      #PR5, -(SP)   ;;PUT NEW PS ON STACK
135 003104 012746 003112          MOV      #64$, -(SP)  ;;PUT NEW PC ON STACK
135 003110 000002                    RTI                    ;;POP NEW PC AND PS
136
137
64$:

```

```

*****
*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 RM80, THAT THE DRIVE IS ONLINE (RMDS 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.
*****

```

```

003112
003112 005737 001266          TST1:  TST      KYBCTL          :PERFORMING ONLY SINGLE TESTS ?
003116 001406                    BEQ      2$             :BR IF NOT
003120 100002                    BPL      1$            :BR IF JUST ENTERED TEST
003122 000137 002610          JMP      EXEC          :RETURN & GET NEXT TEST NUMBER
003126 012737 177777 001266 1$:  MOV      #-1, KYBCTL   :SET SINGLE TEST INDICATOR
003130 012737 003150 001106 2$:  MOV      #TEST1, $LPADR :SETUP SCOPE LOOP ADDRESS
003142 012737 003150 001110          MOV      #TEST1, $LPERR :SETUP ERROR LOOP ADDRESS
003150
003150 112737 000001 001102          TEST1: MOVVB   #1, $STNM       :MOVE #1 TEST NUMBER
003156 012706 001100          MOV      #STACK, SP   :SETUP THE STACK POINTER
003162 012737 000001 001176          MOV      #1, $TIMES   ;;DO 1 ITERATION
138
139 003170 012760 000040 000010          MOV      #CLR, RMCS2(R0) :CLEAR MASSBUS
140
141
142
150 003176 113760 001224 000010          MOVVB   PORTA, RMCS2(R0) :SELECT PORT A
150 003204 013737 001224 001234          MOV      PORTA, PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

```

:VERIFY THAT DRIVE IS PRESENT THROUGH PORTS A & B

```

003212 005760 000012          TST  RMD5(R0)          ;SEE IF DRIVE (PORT A) PRESENT
003216 005037 001244          CLR  CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
003222 016037 000010 001126  MOV  RMCS2(R0), $BDDAT ;GET CONTENTS OF RMCS2
003230 012737 000010 001122  MOV  #RMCS2, $BDADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003236 060037 001122          ADD  R0, $BDADR      ;ADD RH/RM BASE ADDRESS
003242 005037 001124          CLR  $GDDAT          ;WHAT REGISTER SHOULD BE
003246 013737 001126 001164  MOV  $BDDAT, $TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
003254 042737 167777 001164  BIC  #^CNED, $TMP0   ;SAVE SPECIFIED BITS
003262 023737 001124 001164  CMP  $GDDAT, $TMP0   ;COMPARE THE BITS
003270 001414          BEQ  64$             ;BR IF OK
003272 013737 001126 001174  MOV  $BDDAT, $TMP4    ;COPY 'BAD DATA'
003300 042737 010000 001174  BIC  #NED, $TMP4     ;CLEAR THE MASKED BITS
003306 053737 001174 001124  BIS  $TMP4, $GDDAT   ;'OR' WITH GOOD DATA FOR TYPEOUT
003314 104001          EMT  1
003316 005137 001244          COM  CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003322 000240          NOP
003324 005737 001244          TST  CKERR           ;WAS 'NED' SET ?
003330 001403          BEQ  .+10           ;BR IF NOT
003332 012760 000040 000010  MOV  #CLR, RMCS2(R0) ;ISSUE MASSBUS INIT TO CLEAR 'NED'
003340 113760 001226 000010  MOV  PORTB, RMCS2(R0) ;SELECT PORT B
003346 013737 001226 001234  MOV  PORTB, PTNBR    ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003354 005760 000012          TST  RMD5(R0)          ;SEE IF DRIVE (PORT B) PRESENT
003360 005037 001244          CLR  CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
003364 016037 000010 001126  MOV  RMCS2(R0), $BDDAT ;GET CONTENTS OF RMCS2
003372 012737 000010 001122  MOV  #RMCS2, $BDADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003400 060037 001122          ADD  R0, $BDADR      ;ADD RH/RM BASE ADDRESS
003404 005037 001124          CLR  $GDDAT          ;WHAT REGISTER SHOULD BE
003410 013737 001126 001164  MOV  $BDDAT, $TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
003416 042737 167777 001164  BIC  #^CNED, $TMP0   ;SAVE SPECIFIED BITS
003424 023737 001124 001164  CMP  $GDDAT, $TMP0   ;COMPARE THE BITS
003432 001414          BEQ  66$             ;BR IF OK
003434 013737 001126 001174  MOV  $BDDAT, $TMP4    ;COPY 'BAD DATA'
003442 042737 010000 001174  BIC  #NED, $TMP4     ;CLEAR THE MASKED BITS
003450 053737 001174 001124  BIS  $TMP4, $GDDAT   ;'OR' WITH GOOD DATA FOR TYPEOUT
003456 104001          EMT  1
003460 005137 001244          COM  CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003464 000240          NOP
003466 005737 001244          TST  CKERR           ;WAS 'NED' SET ?
003472 001403          BEQ  .+10           ;BR IF NOT
003474 012760 000040 000010  MOV  #CLR, RMCS2(R0) ;ISSUE MASSBUS INIT TO CLEAR 'NED'

```

64\$:

66\$:

;CONFIRM THAT DRIVE IS AN RM80 AND IS DUAL PORT

151
152
153
157

```

003502 113760 001224 000010  MOV  PORTA, RMCS2(R0) ;SELECT PORT A
003510 013737 001224 001234  MOV  PORTA, PTNBR    ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003516 005037 001244          CLR  CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
003522 016037 000026 001126  MOV  RMDT(R0), $BDDAT ;GET CONTENTS OF RMDT
003530 012737 000026 001122  MOV  #RMDT, $BDADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003536 060037 001122          ADD  R0, $BDADR      ;ADD RH/RM BASE ADDRESS
003542 012737 024026 001124  MOV  #024026, $GDDAT ;WHAT REGISTER SHOULD BE
003550 023737 001124 001126  CMP  $GDDAT, $BDDAT  ;IS THE REGISTER OK ?
003556 001403          BEQ  68$             ;BR IF OK
003560 104002          EMT  2
003562 005137 001244          COM  CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003566 000240          NOP
003570 113760 001226 000010  MOV  PORTB, RMCS2(R0) ;SELECT PORT B
003576 013737 001226 001234  MOV  PORTB, PTNBR    ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

```

68\$:

```

003604 005037 001244          CLR      CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
003610 016037 000026 001126    MOV      RMDT(RO), $BDDAT ;GET CONTENTS OF RMDT
003616 012737 000026 001122    MOV      #RMDT, $BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003624 060037 001122          ADD      RO, $BDADR     ;ADD RH/RM BASE ADDRESS
003630 012737 024026 001124    MOV      #024026, $GDDAT ;WHAT REGISTER SHOULD BE
003636 023737 001124 001126    CMP      $GDDAT, $BDDAT  ;IS THE REGISTER OK ?
003644 001403          BEQ      70$          ;BR IF OK
003646 104002          EMT
003650 005137 001244          COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
003654 000240          NOP

158
159
160
165
003656 113760 001224 000010    MOVB     PORTA, RMCS2(RO) ;SELECT PORT A
003664 013737 001224 001234    MOV      PORTA, PTNBR  ;MOVE PORT ADDRESS TO LOCAT!ON FOR TYPEOUT
003672 005037 001244          CLR      CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
003676 016037 000012 001126    MOV      RMDT(RO), $BDDAT ;GET CONTENTS OF RMDT
003704 012737 000012 001122    MOV      #RMDT, $BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003712 060037 001122          ADD      RO, $BDADR     ;ADD RH/RM BASE ADDRESS
003716 012737 001000 001124    MOV      #PGM, $GDDAT   ;WHAT REGISTER SHOULD BE
003724 013737 001126 001164    MOV      $BDDAT, $TMP0  ;MOVE REGISTER CONTENTS TO '$TMP0'
003732 042737 176777 001164    BIC      #^CPGM, $TMP0  ;SAVE SPECIFIED BITS
003740 023737 001124 001164    CMP      $GDDAT, $TMP0  ;COMPARE THE BITS
003746 001414          BEQ      72$          ;BR IF OK
003750 013737 001126 001174    MOV      $BDDAT, $TMP4  ;COPY 'BAD DATA'
003756 042737 001000 001174    BIC      #PGM, $TMP4    ;CLEAR THE MASKED BITS
003764 053737 001174 001124    BIS      $TMP4, $GDDAT  ;'OR' WITH GOOD DATA FOR TYPEOUT
003772 104003          EMT
003774 005137 001244          COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
004000 000240          NOP
004002 005037 001244          CLR      CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
004006 016037 000012 001126    MOV      RMDT(RO), $BDDAT ;GET CONTENTS OF RMDT
004014 012737 000012 001122    MOV      #RMDT, $BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004022 060037 001122          ADD      RO, $BDADR     ;ADD RH/RM BASE ADDRESS
004026 012737 010600 001124    MOV      #MOL!DPR!DRY, $GDDAT ;WHAT REGISTER SHOULD BE
004034 013737 001126 001164    MOV      $BDDAT, $TMP0  ;MOVE REGISTER CONTENTS TO '$TMP0'
004042 042737 167177 001164    BIC      #^C10600, $TMP0 ;SAVE SPECIFIED BITS
004050 023737 001124 001164    CMP      $GDDAT, $TMP0  ;COMPARE THE BITS
004056 001414          BEQ      74$          ;BR IF OK
004060 013737 001126 001174    MOV      $BDDAT, $TMP4  ;COPY 'BAD DATA'
004066 042737 010600 001174    BIC      #10600, $TMP4  ;CLEAR THE MASKED BITS
004074 053737 001174 001124    BIS      $TMP4, $GDDAT  ;'OR' WITH GOOD DATA FOR TYPEOUT
004102 104004          EMT
004104 005137 001244          COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
004110 000240          NOP
004112 113760 001226 000010    MOVB     PORTB, RMCS2(RO) ;SELECT PORT B
004120 013737 001226 001234    MOV      PORTB, PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
004126 005037 001244          CLR      CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
004132 016037 000012 001126    MOV      RMDT(RO), $BDDAT ;GET CONTENTS OF RMDT
004140 012737 000012 001122    MOV      #RMDT, $BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004146 060037 001122          ADD      RO, $BDADR     ;ADD RH/RM BASE ADDRESS
004152 012737 001000 001124    MOV      #PGM, $GDDAT   ;WHAT REGISTER SHOULD BE
004160 013737 001126 001164    MOV      $BDDAT, $TMP0  ;MOVE REGISTER CONTENTS TO '$TMP0'
004166 042737 176777 001164    BIC      #^CPGM, $TMP0  ;SAVE SPECIFIED BITS
004174 023737 001124 001164    CMP      $GDDAT, $TMP0  ;COMPARE THE BITS
004202 001414          BEQ      76$          ;BR IF OK
004204 013737 001126 001174    MOV      $BDDAT, $TMP4  ;COPY 'BAD DATA'

```

70\$:

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

72\$:

74\$:


```

004212 042737 001000 001174      BIC    #PGM,$TMP4      ;CLEAR THE MASKED BITS
004220 053737 001174 001124      BIS    $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
004226 104003                    EMT    3
004230 051137 001244                    COM    CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
004234 000240                    NOP
004236 005037 001244                    CLR    CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
004242 016037 000012 001126      MOV    RMD5(R0),$BDDAT ;GET CONTENTS OF RMD5
004250 012737 000012 001122      MOV    #RMD5,$BADDR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004256 060037 001122                    ADD    R0,$BADDR      ;ADD RH/RM BASE ADDRESS
004262 012737 010600 001124      MOV    #MOL!DPR!DRY,$GDDAT ;WHAT REGISTER SHOULD BE
004270 013737 001126 001164      MOV    $BDDAT,$TMP0   ;MOVE REGISTER CONTENTS TO '$TMP0'
004276 042737 167177 001164      BIC    #^C10600,$TMP0 ;SAVE SPECIFIED BITS
004304 023737 001124 001164      CMP    $GDDAT,$TMP0   ;COMPARE THE BITS
004312 001414                    BEQ    78$            ;BR IF OK
004314 013737 001126 001174      MOV    $BDDAT,$TMP4   ;COPY 'BAD DATA'
004322 042737 010600 001174      BIC    #10600,$TMP4   ;CLEAR THE MASKED BITS
004330 053737 001174 001124      BIS    $TMP4,$GDDAT   ;'OR' WITH GOOD DATA FOR TYPEOUT
004336 104004                    EMT    4
004340 005137 001244                    COM    CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
004344 000240                    NOP

```

76\$:

78\$:

:VERIFY THAT DRIVE SERIAL NUMBER SEEN THROUGH BOTH PORTS IS THE SAME

166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
214
215

```

004346 113760 001224 000010      MOV    PORTA,RMCS2(R0) ;SELECT PORT A
004354 016037 000030 001124      MOV    RMSN(R0),$GDDAT ;STORE THE PORT A SERIAL NUMBER
004362 113760 001226 000010      MOV    PORTB,RMCS2(R0) ;SELECT PORT B
004370 016037 000030 001126      MOV    RMSN(R0),$BDDAT ;STORE THE PORT B SERIAL NUMBER
004376 023737 001124 001126      CMP    $GDDAT,$BDDAT  ;ARE THEY THE SAME ?
004404 001406                    BEQ    1$            ;BR IF THEY ARE
004406 104005                    EMT    5
004410 032777 100000 174522      BIT    #SW15,$SWR     ;HALT ON ERROR ?
004416 001001                    BNE    1$            ;BR IF SET - PROGRAM HAS ALREADY HALTED
004420 000000                    HALT
004422 000004                    1$:    SCOPE         ;HALT, POSSIBLE CABLE CONNECTION PROBLEM
                                ;LOOP ?

```

```

*****
*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.
*****

```

```

004424                    TST2:
004424 005737 001266      TST    KYBCTL         ;PERFORMING ONLY SINGLE TESTS ?
004430 001406                    BEQ    2$            ;BR IF NOT
004432 100002                    BPL    1$            ;BR IF JUST ENTERED TEST
004434 000137 002610      JMP    EXEC           ;RETURN & GET NEXT TEST NUMBER

```

```

004440 012737 177777 001266 1$:   MOV   #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
004446 012737 004462 001106 2$:   MOV   #TEST2,$LPADR  ;SETUP SCOPE LOOP ADDRESS
004454 012737 004462 001110       MOV   #TEST2,$LPERR  ;SETUP ERROR LOOP ADDRESS
004462
004462 112737 000002 001102 TEST2: MOVVB  #2,$STSTM   ;MOVE #2 TEST NUMBER
004470 012706 001100       MOV   #STACK,SP     ;SETUP THE STACK POINTER
004474 012737 000001 001176       MOV   #1,$TIMES     ;:DO 1 ITERATION

004502 113760 001224 000010       MOVVB  PORTA, RMCS2(R0) ;SELECT PORT A
004510 013737 001224 001234       MOV   PORTA,PTNBR   ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

```

;SET VOLUME VALUE FOR PORT

```

004516 012760 000011 000000       MOV   #11, RMCS1(R0) ;ISSUE A DRIVE CLEAR
004524 012760 000021 000000       MOV   #21, RMCS1(R0) ;ISSUE A READIN PRESET
004532 012760 010000 000032       MOV   #FMT16, RMOF(R0) ;SET FMT16

```

;VERIFY THAT THE DRIVE STATUS IS CORRECT

```

004540 005037 001244       CLR   CKERR          ;CLEAR THE 'CHECK ERROR' INDICATOR
004544 016037 000012 001126       MOV   RMDS(R0), $BDDAT ;GET CONTENTS OF RMDS
004552 012737 000012 001122       MOV   #RMDS, $BDADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004560 060037 001122       ADD   R0, $BDADR     ;ADD RH/RM BASE ADDRESS
004564 012737 011700 001124       MOV   #MOL!PGM!DPR!DRY!VV, $GDDAT ;WHAT REGISTER SHOULD BE
004572 013737 001126 001164       MOV   $BDDAT, $STMP0 ;MOVE REGISTER CONTENTS TO '$STMP0'
004600 042737 106077 001164       BIC   #^C71700, $STMP0 ;SAVE SPECIFIED BITS
004606 023737 001124 001164       CMP   $GDDAT, $STMP0 ;COMPARE THE BITS
004614 001414       BEQ   64$           ;BR IF OK
004616 013737 001126 001174       MOV   $BDDAT, $STMP4 ;COPY 'BAD DATA'
004624 042737 071700 001174       BIC   #71700, $STMP4 ;CLEAR THE MASKED BITS
004632 053737 001174 001124       BIS   $STMP4, $GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
004640 104010       EMT   10
004642 005137 001244       COM   CKERR          ;SET THE REGISTER COMPARE ERROR INDICATOR
004646 000240 64$:   NOP

```

;RELEASE THE DRIVE FROM PORT A

```

004650 113760 001224 000010       MOVVB  PORTA, RMCS2(R0) ;SELECT PORT A
004656 013737 001224 001234       MOV   PORTA,PTNBR   ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
004664 012760 000013 000000       MOV   #13, RMCS1(R0) ;ISSUE RELEASE THROUGH PORT A

```

;VERIFY THAT THE DRIVE IS IN NEUTRAL

```

004672 005037 001250       CLR   RELERR         ;CLEAR THE 'RELEASE ERROR' INDICATOR
004676 012737 000012 001122       MOV   #RMDS, $BDADR  ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
004704 060037 001122       ADD   R0, $BDADR     ;ADD THE I/O BASE ADDRESS
004710 012737 011600 001124       MOV   #MOL!PGM!DPR!DRY, $GDDAT ;COMPARISON CONSTANT
004716 113760 001224 000010       MOVVB  PORTA, RMCS2(R0) ;SELECT PORT A.
004724 016037 000012 001170       MOV   RMDS(R0), $STMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
004732 013737 001170 001164       MOV   $STMP2, $STMP0 ;COPY IT INTO '$STMP0'
004740 042737 100100 001164       BIC   #ATA!VV, $STMP0 ;CLEAR PORT DEPENDENT BITS FROM THE COPY
004746 113760 001226 000010       MOVVB  PORTB, RMCS2(R0) ;SELECT PORT B.
004754 016037 000012 001172       MOV   RMDS(R0), $STMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
004762 013737 001172 001166       MOV   $STMP3, $STMP1 ;COPY IT INTO '$STMP1'
004770 042737 100100 001166       BIC   #ATA!VV, $STMP1 ;CLEAR PORT DEPENDENT BITS FROM THE COPY
004776 023737 001164 001166       CMP   $STMP0, $STMP1 ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
005004 001006       BNE   66$           ;BR IF NOT

```

```

005006 005737 001164      TST      $TMP0      ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
005012 001037      BNE      68$      ;BR IF NOT
005014 104034      EMT      34
005016 000137 005202      JMP      70$      ;BYPASS THE REST OF THE CHECKS
005022 013737 001170 001126 66$:  MOV     $TMP2,$BDDAT ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
005030 013737 001226 001234      MOV     PORTB,PTNBR ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005036 113760 001226 000010      MOV     PORTB,RMCS2(R0) ;SELECT PORT B.
005044 005737 001164      TST     $TMP0      ;SEE IF STATUS EQ 0 FROM PORT A.
005050 001414      BEQ     67$      ;BR IF ZERO
005052 013737 001224 001234      MOV     PORTA,PTNBR ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005060 013737 001172 001126      MOV     $TMP3,$BDDAT ;'BAD DATA' FOR ERROR TYPE OUT
005066 113760 001224 000010      MOV     PORTA,RMCS2(R0) ;SELECT PORT A.
005074 005737 001166      TST     $TMP1      ;SEE IF STATUS EQ ZERO FROM PORT B.
005100 001004      BNE     68$      ;BR IF NOT
005102 012737 177777 001250 67$:  MOV     #-1,RELERR ;SET 'RELEASE ERROR' INDICATOR
005110 104036      EMT     36
005112 013737 001170 001126 68$:  MOV     $TMP2,$BDDAT ;LOOK FOR BIT FAILURES WHEN RMDS READ
005120 013737 001224 001234      MOV     PORTA,PTNBR ;CHANGE PORT NUMBER
005126 042737 100100 001170      BIC     #ATA!VV,$TMP2 ;DON'T CHECK ATTN BIT OR VV BIT
005134 023737 001124 001170      CMP     $GDDAT,$TMP2 ;ALL BITS OK ?
005142 001401      BEQ     69$      ;BR IF OK FROM PORT A.
005144 104037      EMT     37
005146 013737 001172 001126 69$:  MOV     $TMP3,$BDDAT ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
005154 013737 001226 001234      MOV     PORTB,PTNBR ;CHANGE PORT NUMBER
005162 042737 100100 001172      BIC     #ATA!VV,$TMP3 ;DON'T CHECK ATTN BIT OR VV BIT
005170 023737 001124 001172      CMP     $GDDAT,$TMP3 ;SEE IF READ OK FROM PORT B.
005176 001401      BEQ     70$      ;BR IF OK
005200 104037      EMT     37
005202 000240      NOP
005204 000004      SCOPE      ;LOOP ?
  
```

216

```

*****
*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.
*****
  
```

```

005206 005737 001266      TST3:  TST      KYBCTL      ;PERFORMING ONLY SINGLE TESTS ?
005206 001406      BEQ     2$      ;BR IF NOT
005212 100002      BPL     1$      ;BR IF JUST ENTERED TEST
005216 000137 002610      JMP     EXEC     ;RETURN & GET NEXT TEST NUMBER
005222 012737 177777 001266 1$:  MOV     #-1,KYBCTL ;SET SINGLE TEST INDICATOR
005230 012737 005244 001106 2$:  MOV     #TEST3,$LPADR ;SETUP SCOPE LOOP ADDRESS
005236 012737 005244 001110      MOV     #TEST3,$LPERR ;SETUP ERROR LOOP ADDRESS
005244      TEST3:
005244 112737 000003 001102      MOV     #3,$STNM   ;MOVE #3 TEST NUMBER
005252 012706 001100      MOV     #STACK,SP ;SETUP THE STACK POINTER
  
```

```
005256 012737 000001 001176      MOV      #1,$TIMES      ;;DO 1 ITERATION
005264 113760 001226 000010      MOV      PORTB, RMCS2(R0) ;SELECT PORT B
005272 013737 001226 001234      MOV      PORTB, PTNBR   ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
```

;SET VOLUME VALUE FOR PORT

```
005300 012760 000011 000000      MOV      #11, RMCS1(R0) ;ISSUE A DRIVE CLEAR
005306 012760 000021 000000      MOV      #21, RMCS1(R0) ;ISSUE A READIN PRESET
005314 012760 010000 000032      MOV      #FMT16, RMOF(R0) ;SET FMT16
```

;VERIFY THAT THE DRIVE STATUS IS CORRECT

```
005322 005037 001244      CLR      CKERR          ;CLEAR THE 'CHECK ERROR' INDICATOR
005326 016037 000012 001126      MOV      RMDS(R0), $BDDAT ;GET CONTENTS OF RMDS
005334 012737 000012 001122      MOV      #RMDS, $BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
005342 060037 001122      ADD      R0, $BDADR     ;ADD RH/RM BASE ADDRESS
005346 012737 011700 001124      MOV      #MOL!PGM!DPR!DRY!VV, $GDDAT ;WHAT REGISTER SHOULD BE
005354 013737 001126 001164      MOV      $BDDAT, $TMP0  ;MOVE REGISTER CONTENTS TO '$TMP0'
005362 042737 106077 001164      BIC      #^C71700, $TMP0 ;SAVE SPECIFIED BITS
005370 023737 001124 001164      CMP      $GDDAT, $TMP0  ;COMPARE THE BITS
005376 001414      BEQ      64$           ;BR IF OK
005400 013737 001126 001174      MOV      $BDDAT, $TMP4  ;COPY 'BAD DATA'
005406 042737 071700 001174      BIC      #71700, $TMP4  ;CLEAR THE MASKED BITS
005414 053737 001174 001124      BIS      $TMP4, $GDDAT  ;'OR' WITH GOOD DATA FOR TYPEOUT
005422 104010      EMT      10
005424 005137 001244      COM      CKERR          ;SET THE REGISTER COMPARE ERROR INDICATOR
005430 000240      NOP
```

64\$: ;RELEASE THE DRIVE FROM PORT B

```
005432 113760 001226 000010      MOV      PORTB, RMCS2(R0) ;SELECT PORT B
005440 013737 001226 001234      MOV      PORTB, PTNBR   ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
005446 012760 000013 000000      MOV      #13, RMCS1(R0) ;ISSUE RELEASE THROUGH PORT B
```

;VERIFY THAT THE DRIVE IS IN NEUTRAL

```
005454 005037 001250      CLR      RELERR        ;CLEAR THE 'RELEASE ERROR' INDICATOR
005460 012737 000012 001122      MOV      #RMDS, $BDADR  ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
005466 060037 001122      ADD      R0, $BDADR     ;ADD THE I/O BASE ADDRESS
005472 012737 011600 001124      MOV      #MOL!PGM!DPR!DRY, $GDDAT ;COMPARISON CONSTANT
005500 113760 001224 000010      MOV      PORTA, RMCS2(R0) ;SELECT PORT A.
005506 016037 000012 001170      MOV      RMDS(R0), $TMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
005514 013737 001170 001164      MOV      $TMP2, $TMP0   ;COPY IT INTO '$TMP0'
005522 042737 100100 001164      BIC      #ATA!VV, $TMP0  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005530 113760 001226 000010      MOV      PORTB, RMCS2(R0) ;SELECT PORT B.
005536 016037 000012 001172      MOV      RMDS(R0), $TMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
005544 013737 001172 001166      MOV      $TMP3, $TMP1   ;COPY IT INTO '$TMP1'
005552 042737 100100 001166      BIC      #ATA!VV, $TMP1  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005560 023737 001164 001166      CMP      $TMP0, $TMP1   ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
005566 001006      BNE      66$           ;BR IF NOT
005570 005737 001164      TST      $TMP0          ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
005574 001037      BNE      68$           ;BR IF NOT
005576 104034      EMT      34
005600 000137 005764      JMP      70$           ;BYPASS THE REST OF THE CHECKS
005604 013737 001170 001126      MOV      $TMP2, $BDDAT  ;SET UP POSSIBLE 'BAD DATA' FOR ERROR MESSAGE
005612 013737 001226 001234      MOV      PORTB, PTNBR   ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
```

66\$:

68\$:

```

005620 113760 001226 000010      MOVB   PORTB, RMCS2(R0)  ;SELECT PORT B.
005626 005737 001164              TST    $TMP0             ;SEE IF STATUS EQ 0 FROM PORT A.
005632 001414                      BEQ    67$               ;BR IF ZERO
005634 013737 001224 001234      MOV    PORTA, PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005642 013737 001172 001126      MOV    $TMP3, $BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
005650 113760 001224 000010      MOVB   PORTA, RMCS2(R0)  ;SELECT PORT A.
005656 005737 001164              TST    $TMP1             ;SEE IF STATUS EQ ZERO FROM PORT B.
005662 001004                      BNE    68$               ;BR IF NOT
005664 012737 177777 001250 67$:  MOV    #-1, RELERR     ;SET 'RELEASE ERROR' INDICATOR
005672 104036                      EMT    36
005674 013737 001170 001126 68$:  MOV    $TMP2, $BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
005702 013737 001224 001234      MOV    PORTA, PTNBR     ;CHANGE PORT NUMBER
005710 042737 100100 001170      BIC    #ATA!VV, $TMP2   ;DON'T CHECK ATTN BIT OR VV BIT
005716 023737 001124 001170      CMP    $GDDAT, $TMP2    ;ALL BITS OK ?
005724 001401                      BEQ    69$               ;BR IF OK FROM PORT A.
005726 104037                      EMT    37
005730 013737 001172 001126 69$:  MOV    $TMP3, $BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
005736 013737 001226 001234      MOV    PORTB, PTNBR     ;CHANGE PORT NUMBER
005744 042737 100100 001172      BIC    #ATA!VV, $TMP3   ;DON'T CHECK ATTN BIT OR VV BIT
005752 023737 001124 001172      CMP    $GDDAT, $TMP3    ;SEE IF READ OK FROM PORT B.
005760 001401                      BEQ    70$               ;BR IF OK
005762 104037                      EMT    37
005764 000240 70$:                NOP
005766 000004                      SCOPE                    ;LOOP ?

```

217
222
274

```

:*****
:*TEST 4          MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A
:*
:*MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT A
:*
:*  A.  WRITE 0'S INTO RMDS 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 TIMFOUT OCCURRED AND THAT THE DRIVE RETURNS
:*       TO NEUTRAL
:*
:*****

```

```

005770
005770 005737 001266      TST4:  TST    KYBCTL      ;PERFORMING ONLY SINGLE TESTS ?
005774 001406              BEQ    2$               ;BR IF NOT
005776 100002              BPL    1$               ;BR IF JUST ENTERED TEST
006000 000137 002610      JMP    EXEC             ;RETURN & GET NEXT TEST NUMBER
006004 012737 177777 001266 1$:  MOV    #-1, KYBCTL     ;SET SINGLE TEST INDICATOR
006012 012737 006026 001106 2$:  MOV    #TEST4, $LPADR  ;SETUP SCOPE LOOP ADDRESS
006020 012737 006026 001110      MOV    #TEST4, $LPERR  ;SETUP ERROR LOOP ADDRESS
006026
006026 112737 000004 001102  TEST4:  MOVB   #4, $STNM       ;MOVE #4 TEST NUMBER
006034 012706 001100      MOV    #STACK, SP     ;SETUP THE STACK POINTER
006040 012737 000001 001176      MOV    #1, $TIMES     ;;DO 1 ITERATION
006046 005037 001256      CLR    TIMEA          ;CLEAR THE TIMEOUT VALUE STORAGE LOCATION

```

```

006052 005037 001260          CLR    TIMEAP ;CLEAR THE + 25% TOLERANCE LOCATION
                                ;START THE TIMER

006056 005037 001252          CLR    TIME          ;CLEAR THE ELAPSED TIME COUNTER
006062 012737 003720 001254  MOV    #2000.,WATCH ;SET WATCH TO 2000. MS
                                ;SEIZE THE DRIVE THROUGH PORT A

006070 113760 001224 000010  MOVB   PORTA, RMCS2(R0) ;SELECT PORT A
006076 013737 001224 001236  MOV    PORTA, SEIZPT ;STORE SEIZING PORT'S ADDRESS
006104 005060 000012          CLR    RMDS(R0) ;WRITE RMDS
006110 113760 001226 000010  MOVB   PORTB, RMCS2(R0) ;SELECT PORT B
006116 013737 001226 001234  MOV    PORTB, PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006124 013737 001226 001240  MOV    PORTB, OPPRT ;'OPPOSITE' PORT ADDRESS
006132 016037 000012 001126  MOV    RMDS(R0), $BDDAT ;SEE IF DRIVE SEIZED BY PORT A
006140 010037 001122          MOV    R0, $BDADR ;RH/RM BASE ADDRESS
006144 062737 000012 001122  ADD    #RMDS, $BDADR ;GENERATE BAD REGISTER ADDRESS
006152 005037 001124          CLR    $GDDAT ;REGISTER SHOULD BE ZERO
006156 023737 001124 001126  CMP    $GDDAT, $BDDAT ;IS THE REGISTER ZERO
006164 001403          BEQ   64$ ;BR IF IT IS
006166 104030          EMT   30
006170 000137 006704          JMP   4$ ;BYPASS REST OF THE SUBTEST
006174          64$:
006174 113760 001224 000010  MOVB   PORTA, RMCS2(R0) ;SELECT PORT A
006202 013737 001224 001234  MOV    PORTA, PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006210 016037 000012 001126  MOV    RMDS(R0), $BDDAT ;SEE IF SEIZING PORT SEES CORRECT STATUS
006216 012737 011700 001124  MOV    #MOL!PGM!DPR!DRY!VV, $GDDAT ;EXPECTED STATUS
006224 013737 001124 001166  MOV    $GDDAT, $TMP1 ;USE GOOD DATA AS A MASK
006232 005137 001166          COM   $TMP1 ;COMPLEMENT THE EXPECTED STATUS
006236 013737 001126 001164  MOV    $BDDAT, $TMP0 ;SAVE THE ACTUAL STATUS
006244 043737 001166 001164  BIC   $TMP1, $TMP0 ;CLEAR UNWANTED BITS
006252 023737 001124 001164  CMP    $GDDAT, $TMP0 ;ARE THE EXPECTED STATUS BITS SET ?
006260 001401          BEQ   65$ ;BR IF THEY ARE
006262 104031          EMT   31
006264 000240          65$: NOP

                                ;WAIT FOR PORT A TO TIMEOUT

006266 113760 001226 000010  MOVB   PORTB, RMCS2(R0) ;SELECT PORT B
006274 013737 001226 001234  MOV    PORTB, PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006302 005760 000012          1$: TST   RMDS(R0) ;WAIT FOR THE DRIVE TO TIMEOUT
006306 001006          BNE   2$ ;BR WHEN TIMEOUT OCCURS
006310 005737 001254          TST   WATCH ;CHECK WATCH
006314 001372          BNE   1$ ;BR IF NOT ZERO
006316 104006          EMT   6
006320 000137 006356          JMP   3$ ;BYPASS THE REST OF THE TEST
006324 013737 001252 001256  2$: MOV    TIME, TIMEA ;SAVE THE ELAPSED TIME FOR PORT A
006332 004537 013522          JSR   R5, TOLER ;CALCULATE THE TOLERANCE
006336 001256          .WORD TIMEA ;TIMEOUT VALUE FOR PORT A
006340 012637 001260          MOV    (SP)+, TIMEAP ;+25% TOLERANCE

                                ;VERIFY THAT THE TIMEOUT ONE-SHOT VALUE IS AT LEAST 500 MS

006344 023727 001256 000764  CMP    TIMEA, #500. ;IS TIMEOUT VALUE AT LEAST 500 MS ?
006352 103001          BHIS  3$ ;BR IF IT IS
006354 104007          EMT   7
    
```

;VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AFTER PORT A TIMED OUT

006356

3\$:
 ;VERIFY THAT THE DRIVE IS IN NEUTRAL

```

006356 005037 001250          CLR      RELERR      ;CLEAR THE 'RELEASE ERROR ' INDICATOR
006362 012737 000012 001122  MOV      #RMDS,$BDADR ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
006370 060037 001122          ADD      R0,$BDADR   ;ADD THE I/O BASE ADDRESS
006374 012737 011700 001124  MOV      #MOL!PGM!DPR!DRY!VV,$GDDAT ;COMPARISON CONSTANT
006402 113760 001224 000010  MOV      PORTA, RMCS2(R0) ;SELECT PORT A.
006410 016037 000012 001170  MOV      RMDS(R0), $TMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
006416 013737 001170 001164  MOV      $TMP2, $TMP0   ;COPY IT INTO '$TMP0'
006424 042737 100100 001164  BIC      #ATA!VV, $TMP0 ;CLEAR PORT DEPENDENT BITS FROM THE COPY
006432 113760 001226 000010  MOV      PORTB, RMCS2(R0) ;SELECT PORT B.
006440 016037 000012 001172  MOV      RMDS(R0), $TMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
006446 013737 001172 001166  MOV      $TMP3, $TMP1   ;COPY IT INTO '$TMP1'
006454 042737 100100 001166  BIC      #ATA!VV, $TMP1 ;CLEAR PORT DEPENDENT BITS FROM THE COPY
006462 023737 001164 001166  CMP      $TMP0, $TMP1   ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
006470 001006          BNE      66$          ;BR IF NOT
006472 005737 001164          TST      $TMP0         ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
006476 001045          BNE      68$          ;BR IF NOT
006500 104034          EMT      34
006502 000137 006702          JMP      70$          ;BYPASS THE REST OF THE CHECKS
006506 013737 001170 001126 66$:  MOV      $TMP2, $BDDAT ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
006514 013737 001226 001234  MOV      PORTB, PTNBR   ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
006522 113760 001226 000010  MOV      PORTB, RMCS2(R0) ;SELECT PORT B.
006530 005737 001164          TST      $TMP0         ;SEE IF STATUS EQ 0 FROM PORT A.
006534 001414          BEQ      67$          ;BR IF ZERO
006536 013737 001224 001234  MOV      PORTA, PTNBR   ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
006544 013737 001172 001126  MOV      $TMP3, $BDDAT ;'BAD DATA' FOR ERROR TYPE OUT
006552 113760 001224 000010  MOV      PORTA, RMCS2(R0) ;SELECT PORT A.
006560 005737 001166          TST      $TMP1         ;SEE IF STATUS EQ ZERO FROM PORT B.
006564 001012          BNE      68$          ;BR IF NOT
006566 012737 177777 001250 67$:  MOV      #-1, RELERR   ;SET 'RELEASE ERROR' INDICATOR
006574 012760 000011 000000  MOV      #11, RMCS1(R0) ;CLEAR THE DRIVE
006602 012760 000013 000000  MOV      #13, RMCS1(R0) ;RELEASE THE DRIVE
006610 104035          EMT      35
006612 013737 001170 001126 68$:  MOV      $TMP2, $BDDAT ;LOOK FOR BIT FAILURES WHEN RMDS READ
006620 013737 001224 001234  MOV      PORTA, PTNBR   ;CHANGE PORT NUMBER
006626 042737 100000 001170  BIC      #ATA, $TMP2   ;DON'T CHECK THE ATTN BIT
006634 023737 001124 001170  CMP      $GDDAT, $TMP2 ;ALL BITS OK ?
006642 001401          BEQ      69$          ;BR IF OK FROM PORT A.
006644 104037          EMT      37
006646 013737 001172 001126 69$:  MOV      $TMP3, $BDDAT ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
006654 013737 001226 001234  MOV      PORTB, PTNBR   ;CHANGE PORT NUMBER
006662 042737 100000 001172  BIC      #ATA, $TMP3   ;DON'T CHECK THE ATTN BIT
006670 023737 001124 001172  CMP      $GDDAT, $TMP3 ;SEE IF READ OK FROM PORT B.
006676 001401          BEQ      70$          ;BR IF OK
006700 104037          EMT      37
006702 000240          NOP
006704 000004          4$:  SCOPE          ;LOOP ?
  
```

275

 ;*TEST 5 MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT B
 ;*

- :*MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT B
- :*
- :* A. WRITE 0'S INTO RMD5 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
- :*

 TST5:

```

006706 005737 001266      TST      KYBCTL      ;PERFORMING ONLY SINGLE TESTS ?
006706 001406             BEQ      2$          ;BR IF NOT
006714 100002             BPL      1$          ;BR IF JUST ENTERED TEST
006716 000137 002610     JMP      EXEC        ;RETURN & GET NEXT TEST NUMBER
006722 012737 177777 001266 1$: MOV      #-1,KYBCTL  ;SET SINGLE TEST INDICATOR
006730 012737 006744 001106 2$: MOV      #TEST5,$LPADR ;SETUP SCOPE LOOP ADDRESS
006736 012737 006744 001110 MCV      #TEST5,$LPERR ;SETUP ERROR LOOP ADDRESS
006744
006744 112737 000005 001102 TEST5:  MOVB     #5,$STSTNM ;MOVE #5 TEST NUMBER
006752 012706 001100      MOV      #STACK,$SP ;SETUP THE STACK POINTER
006756 012737 000001 001176      MOV      #1,$TIMES  ;;DO 1 ITERATION

006764 005037 001262      CLR      TIMEB      ;CLEAR THE TIMEOUT VALUE STORAGE LOCATION
006770 005037 001264      CLR      TIMEBP    ;CLEAR THE + 25% TOLERANCE LOCATION
  
```

;START THE TIMER

```

006774 005037 001252      CLR      TIME        ;CLEAR THE ELAPSED TIME COUNTER
007000 012737 003720 001254      MOV      #2000.,WATCH ;SET WATCH TO 2000. MS
  
```

;SEIZE THE DRIVE THROUGH PORT B

```

007006 113760 001226 000010      MOVB     PORTB,$RMCS2(R0) ;SELECT PORT B
007014 013737 001226 001236      MOV      PORTB,$SEIZPT ;STORE SEIZING PORT'S ADDRESS
007022 005060 000012             CLR      RMD5(R0)      ;WRITE RMD5
007026 113760 001224 000010      MOVB     PORTA,$RMCS2(R0) ;SELECT PORT A
007034 013737 001224 001234      MOV      PORTA,$PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007042 013737 001224 001240      MOV      PORTA,$OPFRT  ;'OPPOSITE' PORT ADDRESS
007050 016037 000012 001126      MOV      RMD5(R0),$BDDAT ;SEE IF DRIVE SEIZED BY PORT B
007056 010037 001122             MOV      R0,$BADDR    ;RH/RM BASE ADDRESS
007062 062737 000012 001122      ADD      #RMD5,$BADDR ;GENERATE BAD REGISTER ADDRESS
007070 005037 001124             CLR      $GDDAT      ;REGISTER SHOULD BE ZERO
007074 023737 001124 001126      CMP      $GDDAT,$BDDAT ;IS THE REGISTER ZERO
007107 001403             BEQ      64$         ;BR IF IT IS
007104 104030             EMT      30
007106 000137 007622             JMP      4$          ;BYPASS REST OF THE SUBTEST
007112
007112 113760 001226 000010      MOVB     PORTB,$RMCS2(R0) ;SELECT PORT B
007120 013737 001226 001234      MOV      PORTB,$PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007126 016037 000012 001126      MOV      RMD5(R0),$BDDAT ;SEE IF SEIZING PORT SEES CORRECT STATUS
007134 012737 011700 001124      MOV      #MOL!PGM!DPR!DRY!VV,$GDDAT ;EXPECTED STATUS
007142 013737 001124 001166      MOV      $GDDAT,$STMP1 ;USE GOOD DATA AS A MASK
007150 005137 001166             COM      $STMP1      ;COMPLEMENT THE EXPECTED STATUS
007154 013737 001126 001164      MOV      $BDDAT,$STMP0 ;SAVE THE ACTUAL STATUS
  
```



```

007162 043737 001166 001164      BIC      $TMP1,$TMP0      ;CLEAR UNWANTED BITS
007170 023737 001124 001164      CMP      $GDDAT,$TMP0    ;ARE THE EXPECTED STATUS BITS SET ?
007176 001401                      BEQ      65$             ;BR IF THEY ARE
007200 104031                      EMT      31
007202 000247                      65$:  NOP

;WAIT FOR PORT B TO TIMEOUT

007204 113760 001224 000010      MOV      PORTA,RMCS2(R0) ;SELECT PORT A
007212 013737 001224 001234      MOV      PORTA,PTNBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007220 005760 000012 1$:      TST      RMD5(R0)       ;WAIT FOR THE DRIVE TO TIMEOUT
007224 001006                      BNE      2$             ;BR WHEN TIMEOUT OCCURS
007226 005737 001254      TST      WATCH          ;CHECK WATCH
007232 001372                      BNE      1$             ;BR IF NOT ZERO
007234 104006                      EMT      6
007236 000137 007274      JMP      3$             ;BYPASS THE REST OF THE TEST
007242 013737 001252 001262 2$:  MOV      TIME,TIMEB     ;SAVE THE ELAPSED TIME FOR PORT B
007250 004537 013522      JSR      R5,TOLER       ;CALCULATE THE TOLERANCE
007254 001262                      .WORD   TIMEB           ;TIMEOUT VALUE FOR PORT B
007256 012637 001264      MOV      (SP)+,TIMEBP   ;+25% TOLERANCE

;VERIFY THAT THE TIMEOUT ONE-SHOT VALUE IS AT LEAST 500 MS

007262 023727 001262 000764      CMP      TIMEB,#500.    ;IS TIMEOUT VALUE AT LEAST 500 MS ?
007270 103001                      BHS      3$             ;BR IF IT IS
007272 104007                      EMT      7

;VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AFTER PORT B TIMED OUT

007274 3$:      ;VERIFY THAT THE DRIVE IS IN NEUTRAL

007274 005037 001250      CLR      RELERR         ;CLEAR THE 'RELEASE ERROR ' INDICATOR
007300 012737 000012 001122      MOV      #RMD5,$BDADR   ;FORM THE ADDRESS OF RMD5 FOR TYPEOUT
007306 060037 001122      ADD      R0,$BDADR      ;ADD THE I/O BASE ADDRESS
007312 012737 011700 001124      MOV      #MOL!PGM!DPR!DRY!VV,$GDDAT ;COMPARISON CONSTANT
007320 113760 001224 000010      MOV      PORTA,RMCS2(R0) ;SELECT PORT A.
007326 016037 000012 001170      MOV      RMD5(R0),$TMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
007334 013737 001170 001164      MOV      $TMP2,$TMP0    ;COPY IT INTO 'TMP0'
007342 042737 100100 001164      BIC      #ATA!VV,$TMP0  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
007350 113760 001226 000010      MOV      PORTB,RMCS2(R0) ;SELECT PORT B.
007356 016037 000012 001172      MOV      RMD5(R0),$TMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
007364 013737 001172 001166      MOV      $TMP3,$TMP1    ;COPY IT INTO 'TMP1'
007372 042737 100100 001166      BIC      #ATA!VV,$TMP1  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
007400 023737 001164 001166      CMP      $TMP0,$TMP1    ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
007406 001006                      BNE      66$           ;BR IF NOT
007410 005737 001164      TST      $TMP0          ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
007414 001045                      BNE      68$           ;BR IF NOT
007416 104034                      EMT      34
007420 000137 007620      JMP      70$           ;BYPASS THE REST OF THE CHECKS
007424 013737 001170 001126 66$:  MOV      $TMP2,$BDADR   ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
007432 013737 001226 001234      MOV      PORTB,PTNBR    ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
007440 113760 001226 000010 3:  MOV      PORTB,RMCS2(R0) ;SELECT PORT B.
007446 005737 001164      TST      $TMP0          ;SEE IF STATUS EQ 0 FROM PORT A.
007452 001414                      BEQ      67$           ;BR IF ZERO
007454 013737 001224 001234      MOV      PORTA,PTNBR    ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
007462 013737 001172 001126      MOV      $TMP3,$BDADR   ;'BAD DATA' FOR ERROR TYPE OUT
  
```

```

007470 113760 001224 000010      MOVB  PORTA, RMCS2(R0) ;SELECT PORT A.
007476 005737 001166              TST   $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
007502 001012                      BNE   68$            ;BR IF NOT
007504 012737 177777 J01250 67$:  MOV   #-1, RELERR      ;SET 'RELEASE ERROR' INDICATOR
007512 012760 000011 000000      MOV   #11, RMCS1(R0) ;CLEAR THE DRIVE
007520 012760 000013 000000      MOV   #13, RMCS1(R0) ;RELEASE THE DRIVE
007526 104035                      EMT   35
007530 013737 001170 001126 68$:  MOV   $TMP2, $BDDAT   ;LOOK FOR BIT FAILURES WHEN RMDs READ
007536 013737 001224 001234      MOV   PORTA, PTNBR   ;CHANGE PORT NUMBER
007544 042737 100000 001170      BIC   #ATA, $TMP2    ;DON'T CHECK THE ATTN BIT
007552 023737 001124 001170      CMP   $GDDAT, $TMP2  ;ALL BITS OK ?
007560 001401                      BEQ   69$            ;BR IF OK FROM PORT A.
007562 104037                      EMT   37
007564 013737 001172 001126 69$:  MOV   $TMP3, $BDDAT   ;CHECK RMDs FOR BIT FAILURES - FROM PORT B.
007572 013737 001226 001234      MOV   PORTB, PTNBR   ;CHANGE PORT NUMBER
007600 042737 100000 001172      BIC   #ATA, $TMP3    ;DON'T CHECK THE ATTN BIT
007606 023737 001124 001172      CMP   $GDDAT, $TMP3  ;SEE IF READ OK FROM PORT B.
007614 001401                      BEQ   70$            ;BR IF OK
007616 104037                      EMT   37
007620 000240 70$:  NOP
007622 000004 4$:  SCOPE ;LOOP ?
  
```

276
277
294
295

```

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

```

007624 005737 001266
007630 001406
007632 100002
007634 000137 002610
007640 012737 177777 001266 1$:  MOV   #-1, KYBCTL
007646 012737 007662 001106 2$:  MOV   #TEST6, $LPADR
007654 012737 007662 001110      MOV   #TEST6, $LPERR
007662
007662 112737 000006 001102      MOVB  #6, $STNM      ;MOVE #6 TEST NUMBER
007670 012706 001100              MOV   #STACK, SP    ;SETUP THE STACK POINTER
007674 012737 000001 001176      MOV   #1, $TIMES    ;DO 1 ITERATION
  
```

```

TST6:
      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   #TEST6, $LPADR ;SETUP SCOPE LOOP ADDRESS
      MOV   #TEST6, $LPERR ;SETUP ERROR LOOP ADDRESS
TEST6:
      MOVB  #6, $STNM   ;MOVE #6 TEST NUMBER
      MOV   #STACK, SP ;SETUP THE STACK POINTER
      MOV   #1, $TIMES ;DO 1 ITERATION
  
```

296
297

```

: CLEAR ATTENTION BITS FOR BOTH PORTS
007702 113760 001224 000010      MOVB  PORTA, RMCS2(R0) ;SELECT PORT #A
  
```

```

007710 005060 000012          CLR    RMDS(R0)      ;SEIZE THE DRIVE
007714 012760 000011 000000  MOV    #11,RMCS1(R0) ;ISSUE DRIVE CLEAR
007722 012760 000013 000000  MOV    #13,RMCS1(R0) ;RELEASE THE DRIVE
007730 113760 001226 000010  MOVB  PORTB,RMCS2(R0) ;SELECT PORT #B
007736 005060 000012          CLR    RMDS(R0)      ;SEIZE THE DRIVE THROUGH PORT 'B'
007742 012760 000011 000000  MOV    #11,RMCS1(R0) ;ISSUE DRIVE CLEAR
007750 012760 000013 000000  MOV    #13,RMCS1(R0) ;RELEASE THE DRIVE
298 007756 104401 020154      TYPE  ,SWTCHA        ;SWITCH TO 'A'
299 007762 104401 020246      TYPE  ,CONTUE       ;PRESS 'CONTINUE'
300 007766 000000          HALT
301
302          ;VERIFY THAT THE DRIVE IS IN NEUTRAL

007770 005037 001250          CLR    RELERR        ;CLEAR THE 'RELEASE ERROR ' INDICATOR
007774 012737 000012 001122  MOV    #RMDS,$BDADR  ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
010002 060037 001122          ADD    R0,$BDADR     ;ADD THE I/O BASE ADDRESS
010006 012737 011700 001124  MOV    #MOL!PGM!DPR!DRY!VV,$GDDAT ;COMPARISON CONSTANT
010014 113760 001224 000010  MOVB  PORTA,RMCS2(R0) ;SELECT PORT A.
010022 016037 000012 001170  MOV    RMDS(R0),$TMP2 ;GET THE DRIVE STATUS REGISTER FROM PORT A.
010030 013737 001170 001164  MOV    $TMP2,$TMP0   ;COPY IT INTO '$TMP0'
010036 042737 100100 001164  BIC   #ATA!VV,$TMP0  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010044 113760 001226 000010  MOVB  PORTB,RMCS2(R0) ;SELECT PORT B.
010052 016037 000012 001172  MOV    RMDS(R0),$TMP3 ;GET THE DRIVE STATUS REGISTER FROM PORT B.
010060 013737 001172 001166  MOV    $TMP3,$TMP1   ;COPY IT INTO '$TMP1'
010066 042737 100100 001166  BIC   #ATA!VV,$TMP1  ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010074 023737 001164 001166  CMP   $TMP0,$TMP1   ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
010102 001006          BNE   64$           ;BR IF NOT
010104 005737 001164          TST   $TMP0         ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
010110 001045          BNE   66$           ;BR IF NOT
010112 104034          EMT   34
010114 000137 010314          JMP   68$           ;BYPASS THE REST OF THE CHECKS
010120 013737 001170 001126 64$: MOV    $TMP2,$BDDAT  ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
010126 013737 001226 001234  MOV    PORTB,PTNBR   ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010134 113760 001226 000010  MOVB  PORTB,RMCS2(R0) ;SELECT PORT B.
010142 005737 001164          TST   $TMP0         ;SEE IF STATUS EQ 0 FROM PORT A.
010146 001414          BEQ   65$           ;BR IF ZERO
010150 013737 001224 001234  MOV    PORTA,PTNBR   ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010156 013737 001172 001126  MOV    $TMP3,$BDDAT  ;'BAD DATA' FOR ERROR TYPE OUT
010164 113760 001224 000010  MOVB  PORTA,RMCS2(R0) ;SELECT PORT A.
010172 005737 001166          TST   $TMP1         ;SEE IF STATUS EQ ZERO FROM PORT B.
010176 001012          BNE   66$           ;BR IF NOT
010200 012737 177777 001250 65$: MOV    #-1,RELERR    ;SET 'RELEASE ERROR' INDICATOR
010206 012760 000011 000000  MOV    #11,RMCS1(R0) ;CLEAR THE DRIVE
010214 012760 000013 000000  MOV    #13,RMCS1(R0) ;RELEASE THE DRIVE
010222 104017          EMT   17
010224 013737 001170 001126 66$: MOV    $TMP2,$BDDAT  ;LOOK FOR BIT FAILURES WHEN RMDS READ
010232 013737 001224 001234  MOV    PORTA,PTNBR   ;CHANGE PORT NUMBER
010240 042737 100000 001170  BIC   #ATA,$TMP2    ;DON'T CHECK THE ATTN BIT
010246 023737 001124 001170  CMP   $GDDAT,$TMP2  ;ALL BITS OK ?
010254 001401          BEQ   67$           ;BR IF OK FROM PORT A.
010256 104037          EMT   37
010260 013737 001172 001126 67$: MOV    $TMP3,$BDDAT  ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
010266 013737 001226 001234  MOV    PORTB,PTNBR   ;CHANGE PORT NUMBER
010274 042737 100000 001172  BIC   #ATA,$TMP3    ;DON'T CHECK THE ATTN BIT
010302 023737 001124 001172  CMP   $GDDAT,$TMP3  ;SEE IF READ OK FROM PORT B.
010310 001401          BEQ   68$           ;BR IF OK
010312 104037          EMT   37
  
```

```

303 010314 000240          68$:  NOP
304 010316 104401 020211    TYPE      ,SWTCHB      :SWITCH TO 'B'
305 010322 104401 020246    TYPE      ,CONTUE      :PRESS 'CONTINUE'
306 010326 000000          HALT
307
      :VERIFY THAT THE DRIVE IS IN NEUTRAL

010330 005037 001250          CLR      RELERR      :CLEAR THE 'RELEASE ERROR ' INDICATOR
010334 012737 000012 001122  MOV      #RMDS,$BDADR :FORM THE ADDRESS OF RMDS FOR TYPEOUT
010342 060037 001122          ADD      RO,$BDADR   :ADD THE I/O BASE ADDRESS
010346 012737 011700 001124  MOV      #MOL!PGM!DPR!DRY!VV,$GDDAT :COMPARISON CONSTANT
010354 113760 001224 000010  MOVVB   PORTA, RMCS2(RO) :SELECT PORT A.
010362 016037 000012 001170  MOV      RMDS(RO), $TMP2 :GET THE DRIVE STATUS REGISTER FROM PORT A.
010370 013737 001170 001164  MOV      $TMP2, $TMP0  :COPY IT INTO 'TMP0'
010376 042737 100100 001164  BIC      #ATA!VV, $TMP0 :CLEAR PORT DEPENDENT BITS FROM THE COPY
010404 113760 001226 000010  MOVVB   PORTB, RMCS2(RO) :SELECT PORT B.
010412 016037 000012 001172  MOV      RMDS(RO), $TMP3 :GET THE DRIVE STATUS REGISTER FROM PORT B.
010420 013737 001172 001166  MOV      $TMP3, $TMP1  :COPY IT INTO 'TMP1'
010426 042737 100100 001166  BIC      #ATA!VV, $TMP1 :CLEAR PORT DEPENDENT BITS FROM THE COPY
010434 023737 001164 001166  CMP      $TMP0, $TMP1  :IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
010442 001006          BNE      69$         :BR IF NOT
010444 005737 001164          TST      $TMP0        :REGISTERS ARE THE SAME: ARE THEY ZERO ?
010450 001045          BNE      71$         :BR IF NOT
010452 104034          EMT      34
010454 000137 010654          J'IP     73$         :BYPASS THE REST OF THE CHECKS
010460 013737 001170 001126 69$:  MOV      $TMP2, $BDDAT :SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
010466 013737 001226 001234  MOV      PORTB, PTNBR  :SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010474 113760 001226 000010  MOVVB   PORTB, RMCS2(RO) :SELECT PORT B.
010502 005737 001164          TST      $TMP0        :SEE IF STATUS EQ 0 FROM PORT A.
010506 001414          BEQ      70$         :BR IF ZERO
010510 013737 001224 001234  MOV      PORTA, PTNBR  :SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010516 013737 001172 001126  MOV      $TMP3, $BDDAT :'BAD DATA' FOR ERROR TYPE OUT
010524 113760 001224 000010  MOVVB   PORTA, RMCS2(RO) :SELECT PORT A.
010532 005737 001166          TST      $TMP1        :SEE IF STATUS EQ ZERO FROM PORT B.
010536 001012          BNE      71$         :BR IF NOT
010540 012737 177777 001250 70$:  MOV      #-1, RELERR   :SET 'RELEASE ERROR' INDICATOR
010546 012760 000011 000000  MOV      #11, RMCS1(RO) :CLEAR THE DRIVE
010554 012760 000013 000000  MOV      #13, RMCS1(RO) :RELEASE THE DRIVE
010562 104020          EMT      20
010564 013737 001170 001126 71$:  MOV      $TMP2, $BDDAT :LOOK FOR BIT FAILURES WHEN RMDS READ
010572 013737 001224 001234  MOV      PORTA, PTNBR  :CHANGE PORT NUMBER
010600 042737 100000 001170  BIC      #ATA, $TMP2   :DON'T CHECK THE ATTN BIT
010606 023737 001124 001170  CMP      $GDDAT, $TMP2 :ALL BITS OK ?
010614 001401          BEQ      72$         :BR IF OK FROM PORT A.
010616 104037          EMT      37
010620 013737 001172 001126 72$:  MOV      $TMP3, $BDDAT :CHECK RMDS FOR BIT FAILURES - FROM PORT B.
010626 013737 001226 001234  MOV      PORTB, PTNBR  :CHANGE PORT NUMBER
010634 042737 100000 001172  BIC      #ATA, $TMP3   :DON'T CHECK THE ATTN BIT
010642 023737 001124 001172  CMP      $GDDAT, $TMP3 :SEE IF READ OK FROM PORT B.
010650 001401          BEQ      73$         :BR IF OK
010652 104037          EMT      37
010654 000240          NOP
308 010656 005737 001266 73$:  TST      KYBCTL      :SINGLE TEST MODE ?
309 010662 001402          BEQ      1$         :BR IF NOT
310 010664 104401 020106    TYPE      ,SWTCHN
311 010670 000004          SCOPE
312
  
```

438
 439

```

*****
*TEST 7          TEST 'PORT SELECT' SWITCH ON PORT A
*
*TEST THE OPERATION OF THE 'PORT SELECT' SWITCH (DRIVE CYCLED DOWN).
*
*  A.  CYCLE THE DRIVE DOWN.
*
*  B.  SWITCH TO PORT A POSITION.  VERIFY THAT THE DRIVE IS IN
*      NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
*      PORTS, ARE CORRECT.
*
*  C.  SWITCH THE 'PORT 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 RMDS THROUGH PORT B.
*
*  G.  ISSUE A RELEASE COMMAND THROUGH PORT A.  VERIFY THAT THE
*      DRIVE REMAINS LOCKED ON PORT A.
*
*****
  
```

```

010672
010672 005737 001266
010676 001406
010700 100002
010702 000137 002610
010706 012737 177777 001266
010714 012737 010730 001106
010722 012737 010730 001110
010730 112737 000007 001102
010736 012706 001100
010742 012737 000001 001176

010750 113760 001224 000010
010756 013737 001224 001234
010764 104401 020317
010770 104401 020154
010774 104401 020337

011000 032760 010000 000012
011006 001374
011010 032760 010000 000012
011016 001774

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      #TEST7,$LPADR ;SETUP SCOPE LOOP ADDRESS
          MOV      #TEST7,$LPERR ;SETUP ERROR LOOP ADDRESS

TEST7:
          MOVB    #7,$STSTNM ;MOVE #7 TEST NUMBER
          MOV     #STACK,SP  ;SETUP THE STACK POINTER
          MOV     #1,$TIMES  ;;DO 1 ITERATION

          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A
          MOV     PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
          TYPE    ,CYCLED      ;'CYCLE DOWN THE DRIVE'
          TYPE    ,SWTCHA      ;SWITCH TO 'A'
          TYPE    ,CYCLEU      ;'CYCLE UP THE DRIVE'

1$:      BIT     #MOL,RMDS(R0) ;IS 'MOL' RESET ?
          BNE    1$          ;BR IF NO (DRIVE NOT CYCLED DOWN)
2$:      BIT     #MOL,RMDS(R0) ;IS 'MOL' SET ?
          BEQ    2$          ;BR IF NO (DRIVE NOT CYCLED UP)

;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT A

011020 005037 001244          CLR     CKERR          ;CLEAR THE 'CHECK ERROR' INDICATOR
  
```

```

011024 016037 000012 001126      MOV      RMDS(R0),SBDDAT      ;GET CONTENTS OF RMDS
011032 012737 000012 001122      MOV      #RMDS,SBADR        ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011040 060037 001122 001122      ADD      R0,SBADR           ;ADD RH/RM BASE ADDRESS
011044 012737 110600 001124      MOV      #ATA!MOL!DPR!DRY,$GDDAT ;WHAT REGISTER SHOULD BE
011052 013737 001126 001164      MOV      SBDDAT,$TMP0       ;MOVE REGISTER CONTENTS TO '$TMP0'
011060 042737 066077 001164      BIC      #^C111700,$TMP0    ;SAVE SPECIFIED BITS
011066 023737 001124 001164      CMP      $GDDAT,$TMP0      ;COMPARE THE BITS
011074 001414 001124 001164      BEQ      64$                ;BR IF OK
011076 013737 001126 001174      MOV      SBDDAT,$TMP4       ;COPY 'BAD DATA'
011104 042737 111700 001174      BIC      #111700,$TMP4      ;CLEAR THE MASKED BITS
011112 053737 001174 001124      BIS      $TMP4,$GDDAT      ;'OR' WITH GOOD DATA FOR TYPEOUT
011120 104021 001174 001124      EMT      21
011122 005137 001244 001124      COM      CKERR              ;SET THE REGISTER COMPARE ERROR INDICATOR
011126 000240 001244 001124      64$:   NOP
    
```

:SET VOLUME VALID FOR PORT A

```

011130 012760 000011 000000      MOV      #11,RMCS1(R0)     ;ISSUE A DRIVE CLEAR
011136 012760 000021 000000      MOV      #21,RMCS1(R0)     ;ISSUE A READIN PRESET
011144 012760 010000 000032      MOV      #FMT16,RMOF(R0)   ;SET FMT16
    
```

:CHECK THE DRIVE STATUS THROUGH PORT B; VERIFY THAT 'NED'
 :SETS WHEN THE DRIVE IS ACCESSED THROUGH PORT B.

```

011152 113760 001226 000010      MOV      PORTB,RMCS2(R0)   ;SELECT PORT B
011160 013737 001226 001234      MOV      PORTB,PTNBR       ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011166 005037 001244 001234      CLR      CKERR              ;CLEAR THE 'CHECK ERROR' INDICATOR
011172 016037 000012 001126      MOV      RMDS(R0),SBDDAT   ;GET CONTENTS OF RMDS
011200 012737 000012 001122      MOV      #RMDS,SBADR       ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011206 060037 001122 001122      ADD      R0,SBADR          ;ADD RH/RM BASE ADDRESS
011212 005037 001124 001124      CLR      $GDDAT            ;WHAT REGISTER SHOULD BE
011216 013737 001126 001164      MOV      SBDDAT,$TMP0      ;MOVE REGISTER CONTENTS TO '$TMP0'
011224 042737 000077 001164      BIC      #^C177700,$TMP0   ;SAVE SPECIFIED BITS
011232 023737 001124 001164      CMP      $GDDAT,$TMP0     ;COMPARE THE BITS
011240 001414 001124 001164      BEQ      66$                ;BR IF OK
011242 013737 001126 001174      MOV      SBDDAT,$TMP4      ;COPY 'BAD DATA'
011250 042737 177700 001174      BIC      #177700,$TMP4     ;CLEAR THE MASKED BITS
011256 053737 001174 001124      BIS      $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
011264 104022 001174 001124      EMT      22
011266 005137 001244 001124      COM      CKERR              ;SET THE REGISTER COMPARE ERROR INDICATOR
011272 000240 001244 001124      66$:   NOP
    
```

66\$: CLR CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR

```

011274 005037 001244 001126      MOV      RMCS2(R0),SBDDAT  ;GET CONTENTS OF RMCS2
011300 016037 000010 001126      MOV      #RMCS2,SBADR     ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011306 012737 000010 001122      MOV      R0,SBADR         ;ADD RH/RM BASE ADDRESS
011314 060037 001122 001122      ADD      R0,SBADR         ;ADD RH/RM BASE ADDRESS
011320 012737 010000 001124      MOV      #NED,$GDDAT      ;WHAT REGISTER SHOULD BE
011326 013737 001126 001164      MOV      SBDDAT,$TMP0     ;MOVE REGISTER CONTENTS TO '$TMP0'
011334 042737 167777 001164      BIC      #^CNED,$TMP0     ;SAVE SPECIFIED BITS
011342 023737 001124 001164      CMP      $GDDAT,$TMP0     ;COMPARE THE BITS
011350 001414 001124 001164      BEQ      68$                ;BR IF OK
011352 013737 001126 001174      MOV      SBDDAT,$TMP4     ;COPY 'BAD DATA'
011360 042737 010000 001174      BIC      #NED,$TMP4       ;CLEAR THE MASKED BITS
011366 053737 001174 001124      BIS      $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
011374 104023 001174 001124      EMT      23
011376 005137 001244 001124      COM      CKERR              ;SET THE REGISTER COMPARE ERROR INDICATOR
011402 000240 001244 001124      68$:   NOP
011404 005060 000012 001124      CLR      RMDS(R0)         ;TRY TO SET REQUEST BY WRITING THROUGH
    
```

68\$: CLR RMDS(R0) ;TRY TO SET REQUEST BY WRITING THROUGH

;THE LOCKED OUT PORT (PORT 'B')

;VERIFY THAT DRIVE STAYS LOCKED ON PORT A

```

011410 113760 001224 000010      MOV      PORTA, RMCS2(R0)  ;SELECT PORT A
011416 013737 001224 001234      MOV      PORTA, PTNBR    ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011424 012760 000013 000012      MOV      #13, RMDS(R0)   ;ISSUE A RELEASE THROUGH PORT A
011432 013737 001224 001236      MOV      PORTA, SEIZPT   ;ADDRESS OF 'LOCKED ON' PORT
011440 113760 001226 000010      MOV      PORTB, RMCS2(R0) ;SELECT PORT B
011446 013737 001226 001234      MOV      PORTB, PTNBR    ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011454 005037 001244                CLR      CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011460 016037 000012 001126      MOV      RMDS(R0), $BDDAT ;GET CONTENTS OF RMDS
011466 012737 000012 001122      MOV      #RMDS, $BADDR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011474 060037 001122                ADD      R0, $BADDR      ;ADD RH/RM BASE ADDRESS
011500 005037 001124                CLR      $GDDAT          ;WHAT REGISTER SHOULD BE
011504 013737 001126 001164      MOV      $BDDAT, $TMP0   ;MOVE REGISTER CONTENTS TO '$TMP0'
011512 042737 000077 001164      BIC      #^C177700, $TMP0 ;SAVE SPECIFIED BITS
011520 023737 001124 001164      CMP      $GDDAT, $TMP0   ;COMPARE THE BITS
011526 001414                BEQ      70$             ;BR IF OK
011530 013737 001126 001174      MOV      $BDDAT, $TMP4   ;COPY 'BAD DATA'
011536 042737 177700 001174      BIC      #177700, $TMP4  ;CLEAR THE MASKED BITS
011544 053737 001174 001124      BIS      $TMP4, $GDDAT   ;'OR' WITH GOOD DATA FOR TYPEOUT
011552 104024                EMT      24
011554 005137 001244                COM      CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011560 000240                NOP
  
```

70\$:

;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST

```

011562 105737 001103                TSTB     $ERFLG          ;DID AN ERROR OCCUR
011566 001412                BEQ      3$              ;BR IF NOT
011570 032777 001000 167342        BIT      #SW09, @SWR     ;SEE IF LOOP ON ERROR (SWR9 = 1)
011576 001406                BEQ      3$              ;BR IF NOT
011600 105037 001103                CLRB     $ERFLG          ;CLEAR THE ERROR FLAG
011604 005037 001176                CLR      $TIMES          ;CLEAR THE MAX ITERATION COUNT
011610 000177 167274                JMP      @SLPERR         ;GO TO THE LOOP ADDRESS
011614 005737 001266                3$:     TST      KYBCTL    ;IN SINGLE TEST MODE ?
011620 001460                BEQ      6$              ;BR IF NOT
011622 032777 040000 167310        BIT      #SW14, @SWR     ;LOOP ON TEST ?
011630 001054                BNE      6$              ;BR IF LOOPING
011632 104401 020317                TYPE     ,CYCLED         ;TYPE 'CYCLE DOWN'
011636 104401 020106                TYPE     ,SWTCHN         ;'SWITCH TO A/B'
011642 104401 020337                TYPE     ,CYCLEU         ;'CYCLE THE DRIVE UP'
011646 113760 001224 000010      MOV      PORTA, RMCS2(R0) ;SELECT PORT A
011654 013737 001224 001234      MOV      PORTA, PTNBR    ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011662 032760 010000 000012      4$:     BIT      #MOL, RMDS(R0) ;IS 'MOL' RESET ?
011670 001374                BNE      4$              ;BR IF NO (DRIVE NOT CYCLED DOWN)
011672 032760 010000 000012      5$:     BIT      #MOL, RMDS(R0) ;IS 'MOL' SET ?
011700 001774                BEQ      5$              ;BR IF NO (DRIVE NOT CYCLED UP)
  
```

;SET VOLUME VALID FOR BOTH PORTS

```

011702 012760 000011 000000      MOV      #11, RMCS1(R0)  ;ISSUE A DRIVE CLEAR THROUGH PORT A
011710 012760 000021 000000      MOV      #21, RMCS1(R0)  ;ISSUE A READIN PRESET THROUGH PORT A
011716 012760 000013 000000      MOV      #13, RMCS1(R0)  ;RELEASE PORT A
011724 113760 001226 000010      MOV      PORTB, RMCS2(R0) ;SELECT PORT B
011732 013737 001226 001234      MOV      PORTB, PTNBR    ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011740 012760 000021 000000      MOV      #21, RMCS1(R0)  ;ISSUE A READIN PRESET THROUGH PORT B
  
```

```

011746 012760 010000 000032      MOV    #FMT16,RMOF(RO)  ;SET FMT16
011754 012760 000013 000000      MOV    #13,RMCS1(RO)   ;RELEASE PORT B
011762 012737 072460 001254 6$:  MOV    #30000.,WATCH   ;SPINDLE MOTOR 'COOL DOWN' DELAY
011770 005737 001254 7$:      TST    WATCH           ;FINISHED ?
011774 001375      BNE    7$              ;BR IF NOT
011776 000004      SCOPE                   ;LOOP ?
012000 000400      BR     TST10           ;;GO TO NEXT TEST
  
```

440

```

*****
*TEST 10      TEST 'PORT SELECT' SWITCH ON PORT B
*
*TEST THE OPERATION OF THE 'PORT SELECT' SWITCH (DRIVE CYCLED DOWN).
*
*  A.  CYCLE THE DRIVE DOWN.
*
*  B.  SWITCH TO PORT B POSITION.  VERIFY THAT THE DRIVE IS IN
*      NEUTRAL AND THAT THE STATUS BITS IN RMD5, AS READ THROUGH BOTH
*      PORTS, ARE CORRECT.
*
*  C.  SWITCH THE 'PORT 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 ATEMPTING TO ACCESS THE DRIVE THROUGH
*      PORT A.  ATTEMPT TO SET PORT REQUEST BY WRITING 0'S
*      INTO RMD5 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 'PORT 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.
  
```

```

*****
TST10:
012002      TST    KYBCTL      ;PERFORMING ONLY SINGLE TESTS ?
012002 005737 001254      BEQ    2$              ;BR IF NOT
012006 001406      BPL    1$              ;BR IF JUST ENTERED TEST
012010 100002      JMP    EXEC            ;RETURN & GET NEXT TEST NUMBER
012012 000137 002610      MOV    #-1,KYBCTL     ;SET SINGLE TEST INDICATOR
012016 012737 177777 001266 1$:  MOV    #TEST10,$LPADR ;SETUP SCOPE LOOP ADDRESS
012024 012737 012040 001106 2$:  MOV    #TEST10,$LPERR ;SETUP ERROR LOOP ADDRESS
012032 012737 012040 001110      MOV    #10,$STNM     ;MOVE #10 TEST NUMBER
012040      MOV    #STACK,SP   ;SETUP THE STACK POINTER
012046 012706 001100      MOV    #1,$TIMES    ;;DO 1 ITERATION
012052 012737 000001 001176      MOV    PORTB,RMCS2(RO) ;SELECT PORT B
012060 113760 001226 000010      MOV    PORTB,PTNBR   ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012066 013737 001226 001234
  
```



```

012074 104401 020317          TYPE      ,CYCLED      ;'CYCLE DOWN THE DRIVE'
012100 104401 020211          TYPE      ,SWTCHB      ;SWITCH TO 'B'
012104 104401 020337          TYPE      ,CYCLEU      ;'CYCLE UP THE DRIVE'

012110 032760 010000 000012 1$: BIT      #MOL,RMDS(R0) ;IS 'MOL' RESET ?
012116 001374          BNE      1$          ;BR IF NO (DRIVE NOT CYCLED DOWN)
012120 032760 010000 000012 2$: BIT      #MOL,RMDS(R0) ;IS 'MOL' SET ?
012126 001774          BEQ      2$          ;BR IF NO (DRIVE NOT CYCLED UP)

```

;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT B

```

012130 005037 001244          CLR      CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
012134 016037 000012 001126  MOV      RMDS(R0),SBDDAT ;GET CONTENTS OF RMDS
012142 012737 000012 001122  MOV      #RMDS,SBADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012150 060037 001122          ADD      R0,SBADR   ;ADD RH/RM BASE ADDRESS
012154 012737 110600 001124  MOV      #ATA!MOL!DPR!DRY,$GDDAT ;WHAT REGISTER SHOULD BE
012162 013737 001126 001164  MOV      SBDDAT,$TMP0 ;MOVE REGISTER CONTENTS TO '$TMP0'
012170 042737 066077 001164  BIC      #^C111700,$TMP0 ;SAVE SPECIFIED BITS
012176 023737 001124 001164  CMP      $GDDAT,$TMP0 ;COMPARE THE BITS
012204 001414          BEQ      64$      ;BR IF OK
012206 013737 001126 001174  MOV      $BDDAT,$TMP4 ;COPY 'BAD DATA'
012214 042737 111700 001174  BIC      #111700,$TMP4 ;CLEAR THE MASKED BITS
012222 053737 001174 001124  BIS      $TMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
012230 104021          EMT      21
012232 005137 001244          COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
012236 000240          NOP
64$:

```

.SET VOLUME VALID FOR PORT B

```

012240 012760 000011 000000  MOV      #11,RMCS1(R0) ;ISSUE A DRIVE CLEAR
012246 012760 000021 000000  MOV      #21,RMCS1(R0) ;ISSUE A READIN PRESET
012254 012760 010000 000032  MOV      #FMT16,RMOF(R0) ;SET FMT16

```

;CHECK THE DRIVE STATUS THROUGH PORT A; VERIFY THAT 'NED'
;SETS WHEN THE DRIVE IS ACCESSED THROUGH PORT A.

```

012262 113760 001224 000010  MOVB     PORTA,RMCS2(R0) ;SELECT PORT A
012270 013737 001224 001234  MOV      PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012276 005037 001244          CLR      CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
012302 016037 000012 001126  MOV      RMDS(R0),SBDDAT ;GET CONTENTS OF RMDS
012310 012737 000012 001122  MOV      #RMDS,SBADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012316 060037 001122          ADD      R0,SBADR   ;ADD RH/RM BASE ADDRESS
012322 005037 001124          CLR      $GDDAT     ;WHAT REGISTER SHOULD BE
012326 013737 001126 001164  MOV      SBDDAT,$TMP0 ;MOVE REGISTER CONTENTS TO '$TMP0'
012334 042737 000077 001164  BIC      #^C177700,$TMP0 ;SAVE SPECIFIED BITS
012342 023737 001124 001164  CMP      $GDDAT,$TMP0 ;COMPARE THE BITS
012350 001414          BEQ      66$      ;BR IF OK
012352 013737 001126 001174  MOV      $BDDAT,$TMP4 ;COPY 'BAD DATA'
012360 042737 177700 001174  BIC      #177700,$TMP4 ;CLEAR THE MASKED BITS
012366 053737 001174 001124  BIS      $TMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
012374 104022          EMT      22
012376 005137 001244          COM      CKERR      ;SET THE REGISTER COMPARE ERROR INDICATOR
012402 000240          NOP
012404 005037 001244          CLR      CKERR      ;CLEAR THE 'CHECK ERROR' INDICATOR
012410 016037 000010 001126  MOV      RMCS2(R0),SBDDAT ;GET CONTENTS OF RMCS2
012416 012737 000010 001122  MOV      #RMCS2,SBADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012424 060037 001122          ADD      R0,SBADR   ;ADD RH/RM BASE ADDRESS
66$:

```

```

012430 012737 010000 001124      MOV      #NED,$GDDAT ;WHAT REGISTER SHOULD BE
012436 013737 001126 001164      MOV      $BDDAT,$TMP0 ;MOVE REGISTER CONTENTS TO '$TMP0'
012444 042737 167777 001164      BIC      #^CNED,$TMP0 ;SAVE SPECIFIED BITS
012452 023737 001124 001164      CMP      $GDDAT,$TMP0 ;COMPARE THE BITS
012460 001414                      BEQ      68$ ;BR IF OK
012462 013737 001126 001174      MOV      $BDDAT,$TMP4 ;COPY 'BAD DATA'
012470 042737 010000 001174      BIC      #NED,$TMP4 ;CLEAR THE MASKED BITS
012476 053737 001174 001124      BIS      $TMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
012504 104023                      EMT      23
012506 005137 001244                      COM      CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
012512 000240                      NOP
012514 005060 000012      68$: CLR      RMDS(R0) ;TRY TO SET REQUEST BY WRITING THROUGH
                                     ;THE LOCKED OUT PORT (PORT 'A')

```

;VERIFY THAT DRIVE STAYS LOCKED ON PORT B

```

012520 113760 001226 000010      MOV      PORTB,RMCS2(R0) ;SELECT PORT B
012526 013737 001226 001234      MOV      PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012534 012760 000013 000012      MOV      #13,RMDS(R0) ;ISSUE A RELEASE THROUGH PORT B
012542 013737 001226 001236      MOV      PORTB,SEIZPT ;ADDRESS OF 'LOCKED ON' PORT
012550 113760 001224 000010      MOV      PORTA,RMCS2(R0) ;SELECT PORT A
012556 013737 001224 001234      MOV      PORTA,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012564 005037 001244                      CLR      CKERR ;CLEAR THE 'CHECK ERROR' INDICATOR
012570 016037 000012 001126      MOV      RMDS(R0),$BDDAT ;GET CONTENTS OF RMDS
012576 012737 000012 001122      MOV      #RMDS,$BDADR ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012604 060037 001122                      ADD      R0,$BDADR ;ADD RH/RM BASE ADDRESS
012610 005037 001124                      CLR      $GDDAT ;WHAT REGISTER SHOULD BE
012614 013737 001126 001164      MOV      $BDDAT,$TMP0 ;MOVE REGISTER CONTENTS TO '$TMP0'
012622 042737 000077 001164      BIC      #^C177700,$TMP0 ;SAVE SPECIFIED BITS
012630 023737 001124 001164      CMP      $GDDAT,$TMP0 ;COMPARE THE BITS
012636 001414                      BEQ      70$ ;BR IF OK
012640 013737 001126 001174      MOV      $BDDAT,$TMP4 ;COPY 'BAD DATA'
012646 042737 177700 001174      BIC      #177700,$TMP4 ;CLEAR THE MASKED BITS
012654 053737 001174 001124      BIS      $TMP4,$GDDAT ;'OR' WITH GOOD DATA FOR TYPEOUT
012662 104024                      EMT      24
012664 005137 001244                      COM      CKERR ;SET THE REGISTER COMPARE ERROR INDICATOR
012670 000240                      NOP
70$:

```

;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST

```

012672 105737 001103                      TSTB     $ERFLG ;DID AN ERROR OCCUR
012676 001412                      BEQ      3$ ;BR IF NOT
012700 032777 001000 166232      BIT      #SW09,@SWR ;SEE IF LOOP ON ERROR (SWR9 = 1)
012706 001406                      BEQ      3$ ;BR IF NOT
012710 105037 001103                      CLRB     $ERFLG ;CLEAR THE ERROR FLAG
012714 005037 001176                      C.P.     $TIMES ;CLEAR THE MAX ITERATION COUNT
012720 000177 166164                      JMP      @SLPERR ;GO TO THE LOOP ADDRESS
012724 032777 040000 166206      3$: BIT      #SW14,@SWR ;LOOP ON TEST ?
012732 001054                      BNE     6$ ;BR IF LOOPING
012734 104401 020317                      TYPE     ,CYCLED ;TYPE 'CYCLE DOWN'
012740 104401 020106                      TYPE     ,SWTCHN ;'SWITCH TO A/B'
012744 104401 020337                      TYPE     ,CYCLEU ;'CYCLE THE DRIVE UP'
012750 113760 001226 000010      MOV      PORTB,RMCS2(R0) ;SELECT PORT B
012756 013737 001226 001234      MOV      PORTB,PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012764 032760 010000 000012      4$: BIT      #MOL,RMDS(R0) ;IS 'MOL' RESET ?
012772 001374                      BNE     4$ ;BR IF NO (DRIVE NOT CYCLED DOWN)
012774 032760 010000 000012      5$: BIT      #MOL,RMDS(R0) ;IS 'MOL' SET ?

```

013002 001774

BEQ 5\$

;BR IF NO (DRIVE NOT CYCLED UP)

;SET VOLUME VALID FOR BOTH PORTS

013004 012760 000011 000000
013012 012760 000021 000000
013020 012760 000013 000000
013026 113760 001224 000010
013034 013737 001224 001234
013042 012760 000021 000000
013050 012760 010000 000032
013056 012760 000013 000000
013064 012737 072460 001254
013072 005737 001254
013076 001375
013100 000004
013102 000137 013110

MOV #11, RMCS1(R0) ;ISSUE A DRIVE CLEAR THROUGH PORT B
MOV #21, RMCS1(R0) ;ISSUE A READIN PRESET THROUGH PORT B
MOV #13, RMCS1(R0) ;RELEASE PORT B
MOVB PORTA, RMCS2(R0) ;SELECT PORT A
MOV PORTA, PTNBR ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
MOV #21, RMCS1(R0) ;ISSUE A READIN PRESET THROUGH PORT A
MOV #FMT16, RMOF(R0) ;SET FMT16
MOV #13, RMCS1(R0) ;RELEASE PORT A
6\$: MOV #30000., WATCH ;SPINDLE MOTOR 'COOL DOWN' DELAY
7\$: TST WATCH ;FINISHED ?
BNE 7\$;BR IF NOT
SCOPE ;LOOP ?
JMP \$EOP ;GO TO THE END OF PASS ROUTINE

441
442
443
444

::*****
;PUT NEWTEST HERE
:*****
TST11: SCOPE

013106 000004

1

.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
  
```

```

013110                                $EOP:
013110 005737 001266                   TST      KYBCTL      ;ENTERED TEST VIA KEYBOARD COMMAND ?
013114 001402                          BEQ      .+6        ;BR IF NOT
013116 000137 002610                   JMP      EXEC      ;RETURN TO KEYBOARD CONTROL
013122 005037 001102                   CLR      $TSTNM    ;ZERO THE TEST NUMBER
013126 005037 001176                   CLR      $TIMES    ;ZERO THE NUMBER OF ITERATIONS
013132 005237 001100                   INC      $PASS     ;INCREMENT THE PASS NUMBER
013136 042737 100000 001100           BIC      #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
013144 005327                          DEC      (PC)+     ;LOOP?
013146 000001                          $EOPCT: .WORD    1
013150 003066                          BGT      $DOAGN    ;YES
013152 012737                          MOV      (PC)+,@(PC)+ ;RESTORE COUNTER
013154 000001                          $ENDCT: .WORD    1
013156 013146                          $EOPCT
013160 104401 013166                   TYPE    ,65$      ;:TYPE ASCIZ STRING
013164 000407                          BR       64$      ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <12><15>/END PASS #/
64$:
013204                                MOV      $PASS,-(SP) ;:SAVE $PASS FOR TYPEOJT
013204 013746 001100                                ;:TYPE PASS NUMBER
013210 104405                                TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
013212 005737 001112                   TST      $ERTTL    ;:SEE IF ANY ERRORS THIS PASS
013216 001431                          BEQ      $GT42P    ;:BR IF NO ERRORS TO REPORT
013220 104401 013226                   TYPE    ,67$      ;:TYPE ASCIZ STRING
013224 000421                          BR       66$      ;:GET OVER THE ASCIZ
;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:
013270                                MOV      $ERTTL,-(SP) ;:SAVE $ERTTL FOR TYPEOUT
013270 013746 001112                                ;:TOTAL NUMBER OF ERRORS
013274 104405                                TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
013276 005037 001112                   CLR      $ERTTL    ;:CLEAR ERROR TOTAL
013302 104401 001207                   $GT42P: TYPE    ,$CRLF ;:TYPE CARRIAGE RETURN, LINE FEED
013306 013700 000042                   $GET42: MOV     @#42,R0 ;:GET MONITOR ADDRESS
013312 001405                          BEQ      $DOAGN    ;:BRANCH IF NO MONITOR
013314 000005                          RESET    ;:CLEAR THE WORLD
013316 004710                          $ENDAD: JSR    PC,(R0) ;:GO TO MONITOR
013320 000240                                NOP      ;:SAVE ROOM
013322 000240                                NOP      ;:FOR
013324 000240                                NOP      ;:ACT11
013326                                $DOAGN:
013326 000137                          JMP     @(PC)+     ;:RETURN
013330 003074                          $RTNAD: .WORD    TST1AA
013332 377 377 000 377 000 $ENULL: .BYTE  -1,-1,0 ;:NULL CHARACTER STRING
                                .EVEN
  
```

.SBTTL CLOCK SUBROUTINES

;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS

```

2
3
4
5 013336 012737 013406 000004 CKCLK: MOV #CKCLK1,@#ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
6 013344 005037 000006 CLR @#ERRVEC+2 ;NEW PSW
7 013350 005777 165636 TST @SLKCSR ;CHECK FOR KW11-P
8 013354 013701 001216 MOV SLPVEC,R1 ;KW11-P VECTOR ADDRESS
9 013360 012721 013470 MOV #CLOCK,(R1)+ ;SET UP KW11-P VECTOR
10 013364 012711 000300 MOV #300,(R1) ;PSW - PRI 6
11 013370 012777 177777 165616 MOV #-1,@SLKCSB ;LOAD COUNTER BUFFER WITH 1'S
12 013376 012777 000135 165606 MOV #135,@SLKCSR ;SET CLOCK - CNT UP, 16MS, CONT INT
13 013404 000425 BR CKCLK3
14 013406 062706 000004 CKCLK1: ADD #4,SP ;RESTORE THE STACK POINTER
15 013412 012737 013450 000004 MOV #CKCLK2,@#ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
16 013420 005777 165574 TST @SLKS ;LOOK FOR KW11-L
17 013424 013701 001222 MOV SLLVEC,R1 ;KW11-L VECTOR ADDRESS
18 013430 012721 013470 MOV #CLOCK,(R1)+ ;SET UP KW11-L VECTOR
19 013434 012711 000300 MOV #300,(R1) ;PSW - PRI 6
20 013440 012777 000100 165552 MOV #100,@SLKS ;SET KW11-L INTERRUPT
21 013446 000404 BR CKCLK3
22 013450 062706 000004 CKCLK2: ADD #4,SP ;RESTORE THE STACK POINTER
23 013454 062716 000002 ADD #2,(SP) ;INCREMENT RETURN, NO CLOCK
24 013460 012737 000006 000004 CKCLK3: MOV #6,@#ERRVEC ;RESTORE THE ERROR VECTOR
25 013466 000207 RTS PC

```

;ROUTINE TO COUNT CLOCK TICKS

```

26
27
28
29 013470 062737 000021 001252 CLOCK: ADD #17.,TIME ;ADD 17 MS TO ELAPSED TIME COUNTER
30 013476 005737 001254 TST WATCH ;IS WATCH ALREADY ZERO ?
31 013502 001406 BEQ 1$ ;BR IF IT IS
32 013504 162737 000021 001254 SUB #17.,WATCH ;SUBTRACT 17 MS FROM WATCH DOG COUNTER
33 013512 100002 BPL 1$ ;BR IF NOT MINUS
34 013514 005037 001254 CLR WATCH ;CLEAR WATCH DOG COUNTER
35 013520 000002 1$: RTI ;RETURN

```

;ROUTINE TO CALCULATE + 25% TIME TOLERANCE VALUES

```

36
37
38
39 013522 005746 TOLER: TST -(SP) ;MAKE ROOM ON THE STACK
40 013524 016616 000002 MOV 2(SP),(SP) ;SAVE STACK
41 013530 013546 MOV @R5+,-(SP) ;GET TIME VALUE
42 013532 011666 000004 MOV (SP),4(SP) ;MOVE TIME VALUE
43 013536 006216 ASR (SP) ;DIVIDE BY 2
44 013540 006216 ASR (SP) ;DIVIDE BY 2 AGAIN (FOR A TOTAL OF 4)
45 013542 062666 000002 ADD (SP)+,2(SP) ;CALCULATE UPPER LIMIT FOR TIMEOUT
46 013546 000205 RTS R5 ;RETURN WITH TOLERANCES ON THE STACK

```



```

014000          2003$:
014000 105037 001103 4$: CLR  SERFLG      ;;ZERO THE ERROR FLAG
014004 005037 001176 CLR  $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
014010 032777 004000 165122 3$: BIT  #BIT11,@SWR    ;;INHIBIT ITERATIONS?
014016 001011 BNE  1$          ;;BR IF YES
014020 005737 001100 TST  $PASS      ;;IF FIRST PASS OF PROGRAM
014024 001406 BEQ  1$          ;;INHIBIT ITERATIONS
014026 005237 001104 INC  $ICNT      ;;INCREMENT ITERATION COUNT
014032 023737 001176 001104 CMP  $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
014040 002012 BGE  $OVER      ;;BR IF MORE ITERATION REQUIRED
014042 012737 000001 001104 1$: MOV  #1,$ICNT    ;;REINITIALIZE THE ITERATION COUNTER
014050 013737 014102 001176 MOV  $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
014056 105237 001102 $SVLAD: INCB $STNM  ;;COUNT TEST NUMBERS
014062 011637 001106 MOV  (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
014066 013777 001102 165046 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
014074 013716 001106 MOV  $LPADR,(SP) ;;FUDGE RETURN ADDRESS
014100 000002 RTI          ;;FIXES PS
014102 000005 $MXCNT: 5.      ;;MAX. NUMBER OF ITERATIONS

;DROP PRIORITY TO ALLOW CONSOLE INTERRUPT

2
3
4
5 014104 STOP:
014104 012746 000140 MOV  #PR3,-(SP)  ;;PUT NEW PS ON STACK
014110 012746 014116 MOV  #64$,-(SP) ;;PUT NEW PC ON STACK
014114 000002 RTI          ;;POP NEW PC AND PS
014116

6
7
8
9 014116 012746 000240 MOV  #PR5,-(SP)  ;;PUT NEW PS ON STACK
014122 012746 014130 MOV  #65$,-(SP) ;;PUT NEW PC ON STACK
014126 000002 RTI          ;;POP NEW PC AND PS
014130

10 014130 000207 65$: RTS  PC      ;RETURN
    
```

1

.SBTTL ERROR HANDLER ROUTINE

```

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

```

014132 105037 014456      $ERROR: CLRB      IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
014136 104407              CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
014140 113737 001102 001242      MOVB      $TSTNM,TSTNUM
014146 105237 001103      7$:      INCB      $ERFLG          ;;SET THE ERROR FLAG
014152 001775              BEQ      7$          ;;DON'T LET THE FLAG GO TO ZERO
014154 013777 001102 164760      MOV      $TSTNM,@DISPLAY      ;;DISPLAY TEST NUMBER AND ERROR FLAG
014162 032777 002000 164750      BIT      #BIT10,@SWR          ;;BELL ON ERROR?
014170 001402              BEQ      1$          ;;NO - SKIP
014172 104401 001202              TYPE      $BELL          ;;RING BELL
014176 005237 001112      1$:      INC      $ERTTL          ;;COUNT THE NUMBER OF ERRORS
014202 011637 001116      MOV      (SP),$ERRPC          ;;GET ADDRESS OF ERROR INSTRUCTION
014206 162737 000002 001116      SUB      #2,$ERRPC
014214 117737 164676 001114      MOVB     @ $ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
014222 032777 001000 164710      BIT      #BIT09,@SWR          ;;SEE IF LOOP ON ERROR IS SET
014230 001060              BNE      1004$          ;;BRANCH AROUND ROUTINE IF SO
014232 122737 000177 001114      CMPB     #177,$ITEMB          ;;SEE IF THIS IS THE POWER FAIL CALL
014240 001454              BEQ      1004$          ;;BRANCH AROUND ROUTINE IF IT IS
014242 105737 014456      TSTB     IBSAVE              ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
014246 001047              BNE      1003$          ;;BRANCH IF SO
014250 022737 177777 014454      CMP      #-1,CPSAVE          ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
014256 001445              BEQ      1004$          ;;BRANCH IF SO
014260 013746 000004      MOV      ERRVEC,-(SP)          ;;SAVE CONTENTS OF ERROR VECTOR
014264 012737 014302 000004      MOV      #1000$,ERRVEC          ;;SETUP 'TRAP' RETURN ADDRESS
014272 013737 177766 014454      MOV      177766,CPSAVE          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
014300 000406              BR       1001$
014302 012737 177777 014454 1000$:  MOV      #-1,CPSAVE          ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
014310 012716 014316              MOV      #1001$,(SP)          ;;SETUP RETURN ADDRESS
014314 000002              RTI
014316 012637 000004      1001$:  MOV      (SP)+,ERRVEC          ;;RESTORE CONTENTS OF ERROR VECTOR

014322 022737 177777 014454 1002$:  CMP      #-1,CPSAVE          ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
014330 001420              BEQ      1004$          ;;BRANCH IF SO
014332 032737 000001 014454      BIT      #BIT00,CPSAVE          ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
014340 001414              BEQ      1004$          ;;BRANCH IF OK
014342 042737 000001 177766      BIC      #BIT00,177766          ;;CLEAR THE BIT FOUND SET
014350 113737 001114 014456      MOVB     $ITEMB,IBSAVE          ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
014356 112737 000177 001114      MOVB     #177,$ITEMB          ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
014364 000402              BR       1004$          ;;BRANCH OVER IBSAVE CLEARING

014366 105037 014456      1003$:  CLRB     IBSAVE              ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
014372              1004$:
014372 032777 020000 164540      BIT      #BIT13,@SWR          ;;SKIP TYPEOUT IF SET
014400 001004              BNE      20$          ;;SKIP TYPEOUTS
014402 004737 014460              JSR      PC,$ERRTYP          ;;GO TO USER ERROR ROUTINE
    
```


| | | | | | | |
|--------|--------|--------|--------|---------------|--------------|---|
| 014406 | 104401 | 001207 | | TYPE | ,\$CRLF | |
| 014412 | | | 20\$: | | | |
| 014412 | 105737 | 014456 | 2\$: | TSTB | IBSAVE | :::SEE IF IBSAVE IS LOADED |
| 014416 | 001005 | | | BNE | 3\$ | :::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR |
| 014420 | 005777 | 164514 | | TST | @SWR | :::HALT ON ERROR |
| 014424 | 100002 | | | BPL | 3\$ | :::SKIP IF CONTINUE |
| 014426 | 000000 | | | HALT | | :::HALT ON ERROR! |
| 014430 | 104407 | | | CKSWR | | :::TEST FOR CHANGE IN SOFT-SWR |
| 014432 | | | 3\$: | | | |
| 014432 | 022737 | 013316 | 000042 | CMP | #SENDAD,@#42 | :::ACT-11 AUTO-ACCEPT? |
| 014440 | 001001 | | | BNE | 6\$ | :::BRANCH IF NO |
| 014442 | 000000 | | | HALT | | :::YES |
| 014444 | | | 6\$: | | | |
| 014444 | 105737 | 014456 | | TSTB | IBSAVE | :::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL |
| 014450 | 001236 | | | BNE | 7\$ | :::BRANCH BACK TO CALL ORIGINAL ERROR |
| 014452 | 000002 | | | RTI | | :::RETURN |
| 014454 | 000000 | | | CPSAVE: .WORD | 0 | :::LOCATION TO SAVE CPU ERROR REG CONTENTS |
| 014456 | 000000 | | | IBSAVE: .WORD | 0 | :::LOCATION TO SAVE ITEM BYTE |

.SBTTL ERROR MESSAGE TYPEOUT 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.

```

014460                                $ERRTYP:
014460 104401 001207                    TYPE      , $CRLF                ;; 'CARRIAGE RETURN' & 'LINE FEED'
014464 010046                          MOV      R0, -(SP)           ;; SAVE R0
014466 005000                          CLR      R0                ;; PICKUP THE ITEM INDEX
014470 153700 001114                    BISB    @($ITEMB, R0
014474 001004                          BNE     1$                ;; IF ITEM NUMBER IS ZERO, JUST
                                ;; TYPE THE PC OF THE ERROR
014476 013746 001116                    MOV      $ERRPC, -(SP)    ;; SAVE $ERRPC FOR TYPEOUT
                                ;; ERROR ADDRESS
014502 104402                          TYPDC   10$              ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
014504 000456                          BR      10$              ;; GET OUT
014506 122700 000177                    1$:    CMPB    #177, R0      ;; SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
014512 001006                          BNE     1000$            ;; BRANCH IF NOT
014514 113737 001102 015016            MOVB    $TSTNM, PFTSTN   ;; GET TEST NUMBER
014522 012700 014656                    MOV      #PFECH, R0      ;; MOVE POWER FAIL ERROR CALL TABLE TO R0
014526 000406                          BR      1001$           ;; BRANCH TO CALL ERROR
014530 005300                          1000$: DEC     R0          ;; ADJUST THE INDEX SO THAT IT WILL
014532 006300                          ASL     R0                ;; WORK FOR THE ERROR TABLE
014534 006300                          ASL     R0
014536 006300                          ASL     R0
014540 062700 001276                    ADD     # $ERRTB, R0     ;; FORM TABLE POINTER
014544 012037 014554                    1001$: MOV     (R0)+, 2$    ;; PICKUP 'ERROR MESSAGE' POINTER
014550 001404                          BEQ     3$                ;; SKIP TYPEOUT IF NO POINTER
014552 104401                          TYPE    'ERROR MESSAGE' ;; TYPE THE 'ERROR MESSAGE'
014554 000000                          2$:    .WORD 0            ;; 'ERROR MESSAGE' POINTER GOES HERE
014556 104401 001207                    TYPE    , $CRLF         ;; 'CARRIAGE RETURN' & 'LINE FEED'
014562 012037 014572                    3$:    MOV     (R0)+, 4$    ;; PICKUP 'DATA HEADER' POINTER
014566 001404                          BEQ     5$                ;; SKIP TYPEOUT IF 0
014570 104401                          TYPE    'DATA HEADER'   ;; TYPE THE 'DATA HEADER'
014572 000000                          4$:    .WORD 0            ;; 'DATA HEADER' POINTER GOES HERE
014574 104401 001207                    TYPE    , $CRLF         ;; 'CARRIAGE RETURN' & 'LINE FEED'
014600 010146                          5$:    MOV     R1, -(SP)    ;; SAVE R1
014602 012001                          MOV     (R0)+, R1        ;; PICKUP 'DATA TABLE' POINTER
014604 001415                          BEQ     9$                ;; BR IF NO DATA TO BE TYPED
014606 012000                          MOV     (R0)+, R0        ;; PICKUP 'DATA FORMAT' POINTER
014610 105720                          6$:    TSTB    (R0)+      ;; 'OCTAL' OR 'DECIMAL'
014612 001003                          BNE     7$                ;; BR IF DECIMAL
014614 013146                          MOV     @ (R1)+, -(SP)   ;; SAVE @ (R1)+ FOR TYPEOUT
014616 104402                          TYPDC   6$              ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
014620 000402                          BR      8$
014622                                7$:    MOV     @ (R1)+, -(SP) ;; SAVE @ (R1)+ FOR TYPEOUT
014624 104405                          TYPDC   8$              ;; GO TYPE--DECIMAL ASCII WITH SIGN
014626 005711                          8$:    TST     (R1)        ;; IS THERE ANOTHER NUMBER?
014630 001403                          BEQ     9$                ;; BR IF NO
014632 104401 014652                    TYPE    , 11$           ;; TYPE TWO(2) SPACES
014636 000764                          BR      6$              ;; LOOP
014640 012601                          9$:    MOV     (SP)+, R1   ;; RESTORE R1
014642 012600                          10$:   MOV     (SP)+, R0   ;; RESTORE R0
    
```

| | | | | | | | |
|--------|--------|--------|--------|---------|-----------------------------|---|--|
| 014644 | 104401 | 001207 | | | TYPE | \$CRLF | :::'CARRIAGE RETURN' & 'LINE FEED' |
| 014650 | 000207 | | | | RTS | PC | :::RETURN |
| 014652 | 040 | 040 | 000 | 11\$: | .ASCIZ | / / | :::TWO(2) SPACES |
| | | | | | .EVEN | | |
| 014656 | 014666 | 014750 | 015002 | PFECH: | PFECH1,PFECH2,PFECH3,PFECH4 | :::WORDS DEFINING TABLES BELOW | |
| 014666 | 120 | 117 | 127 | PFECH1: | .ASCIZ | ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET? | |
| 014750 | 124 | 105 | 123 | PFECH2: | .ASCIZ | ?TESTNO ERR PC CPUERREG? | |
| | | | | | .EVEN | | |
| 015002 | 015016 | 001116 | 014454 | PFECH3: | .WORD | PFTSTN,\$ERRPC,CPSAVE,0 | |
| 015012 | 000 | 000 | 000 | PFECH4: | .BYTE | 0,0,0,0 | |
| 015016 | 000000 | | | PFTSTN: | .WORD | 0 | :::CONTAINS TEST NUMBER FOR PF BIT ERROR |

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

```

015020 105737 001157 $TYPE: TSTB $TPFLG      ;;IS THERE A TERMINAL?
015024 100002      BPL 1$          ;;BR IF YES
015026 000000      HALT      ;;HALT HERE IF NO TERMINAL
015030 000407      BR 3$          ;;LEAVE
015032 010046      1$: MOV RO,-(SP)  ;;SAVE RO
015034 017600 000002 2$: MOV @2(SP),RO  ;;GET ADDRESS OF ASCIZ STRING
015040 112046      MOVB (RO)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
015042 001005      BNE 4$          ;;BR IF IT ISN'T THE TERMINATOR
015044 005726      TST (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
015046 012600      60$: MOV (SP)+,RO  ;;RESTORE RO
015050 062716 000002 3$: ADD #2,(SP)  ;;ADJUST RETURN PC
015054 000002      RTI          ;;RETURN
015056 122716 000011 4$: CMPB #HT,(SP)  ;;BRANCH IF <HT>
015062 001430      BEQ 8$          ;;BRANCH IF NOT <CRLF>
015064 122716 000200      CMPB #CRLF,(SP)
015070 001006      BNE 5$          ;;POP <CR><LF> EQUIV
015072 005726      TST (SP)+      ;;TYPE A CR AND LF
015074 104401      TYPE
015076 001207      $CRLF
015100 105037 015306      CLRB $CHARCNT  ;;CLEAR CHARACTER COUNT
015104 000755      BR 2$          ;;GET NEXT CHARACTER
015106 004737 015170      5$: JSR PC,$TYPEPC  ;;GO TYPE THIS CHARACTER
015112 123726 001156      6$: CMPB $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
015116 001350      BNE 2$          ;;IF NO GO GET NEXT CHAR.
015120 013746 001154      MOV $NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
                                ;;AND THE NULL CHAR.
015124 105366 000001      7$: DECB 1(SP)  ;;DOES A NULL NEED TO BE TYPED?
015130 002770      BLT 6$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
015132 004737 015170      JSR PC,$TYPEPC  ;;GO TYPE A NULL
015136 105337 015306      DECB $CHARCNT  ;;DO NOT COUNT AS A COUNT
015142 000770      BR 7$          ;;LOOP

```

:HORIZONTAL TAB PROCESSOR

```

015144 112716 000040      8$: MOVB #' (SP)  ;;REPLACE TAB WITH SPACE
015150 004737 015170      9$: JSR PC,$TYPEPC  ;;TYPE A SPACE
015154 132737 000007 015306 BITB #7,$CHARCNT  ;;BRANCH IF NOT AT
015162 001372      BNE 9$          ;;TAB STOP
015164 005726      TST (SP)+      ;;POP SPACE OFF STACK
015166 000724      BR 2$          ;;GET NEXT CHARACTER

```

| | | | | | | | |
|--------|--------|--------|--------|----------|------------|---------------|--|
| 015170 | | | | \$TYPEC: | TSTB | @\$TKS | ::CHAR IN KYBD BUFFER? |
| 015170 | 105777 | 163750 | | | BPL | 10\$ | ::BR IF NOT |
| 015174 | 100022 | | | | MOV | @\$TKB, -(SP) | ::GET CHAR |
| 015176 | 017746 | 163744 | | | BIC | #177600, (SP) | ::STRIP EXTRANEIOUS BITS |
| 015202 | 042716 | 177600 | | | CMPB | #\$XOFF, (SP) | ::WAS CHAR XOFF |
| 015206 | 122716 | 000023 | | | BNE | 102\$ | ::BR IF NOT |
| 015212 | 001012 | | | | | | |
| 015214 | | | | 101\$: | TSTB | @\$TKS | ::WAIT FOR CHAR |
| 015214 | 105777 | 163724 | | | BPL | 101\$ | |
| 015220 | 100375 | | | | MOVB | @\$TKB, (SP) | ::GET CHAR |
| 015222 | 117716 | 163720 | | | BIC | #177600, (SP) | ::STRIP IT |
| 015226 | 042716 | 177600 | | | CMPB | #\$XON, (SP) | ::WAS IT XON? |
| 015232 | 122716 | 000021 | | | BNE | 101\$ | ::BR IF NOT |
| 015236 | 001366 | | | | | | |
| 015240 | | | | 102\$: | TST | (SP)+ | ::FIX STACK |
| 015240 | 005726 | | | | | | |
| 015242 | | | | 10\$: | TSTB | @\$TPS | ::WAIT UNTIL PRINTER IS READY |
| 015242 | 105777 | 163702 | | | BPL | 10\$ | |
| 015246 | 100375 | | | | MOVB | 2(SP), @\$TPB | ::LOAD CHAR TO BE TYPED INTO DATA REG. |
| 015250 | 116677 | 000002 | 163674 | | CMPB | #CR, 2(SP) | ::IS CHARACTER A CARRIAGE RETURN? |
| 015256 | 122766 | 000015 | 000002 | | BNE | 1\$ | ::BRANCH IF NO |
| 015264 | 001003 | | | | CLRB | \$CHARCNT | ::YES--CLEAR CHARACTER COUNT |
| 015266 | 105037 | 015306 | | | BR | \$TYPEX | ::EXIT |
| 015272 | 000406 | | | | CMPB | #LF, 2(SP) | ::IS CHARACTER A LINE FEED? |
| 015274 | 122766 | 000012 | 000002 | 1\$: | BEQ | \$TYPEX | ::BRANCH IF YES |
| 015302 | 001402 | | | | INCB | (PC)+ | ::COUNT THE CHARACTER |
| 015304 | 105227 | | | | | | ::CHARACTER COUNT STORAGE |
| 015306 | 000000 | | | | \$CHARCNT: | .WORD | |
| 015310 | 000207 | | | | \$TYPEX: | RTS | |
| | | | | | | PC | |

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC

```

```

*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON    ;;CALL FOR TYPEOUT

```

```

*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    ;;CALL FOR TYPEOUT

```

```

015312 017646 000000
015316 116637 000001 015535
015324 112637 015537
015330 062716 000002
015334 000406
015336 112737 000001 015535
015344 112737 000006 015537
015352 112737 000005 015534
015360 010346
015362 010446
015364 010546
015366 113704 015537
015372 005404
015374 062704 000006
015400 110437 015536
015404 113704 015535
015410 016605 000012
015414 005003
015416 006105 1$:
015420 000404 BR 3$
015422 006105 2$:
015424 006105 ROL R5
015426 006105 ROL R5
015430 010503 MOV R5,R3
015432 006103 3$:
015434 105337 015536 DECB $OMODE
015440 100016 BPL 7$
015442 042703 177770 BIC #177770,R3
015446 001002 BNE 4$
015450 005704 TST R4
015452 001403 BEQ 5$
015454 005204 4$: INC R4

```

```

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

```

| | | | | | | |
|--------|--------|---------------|----------|-------|-------------|-----------------------------------|
| 015456 | 052703 | 000060 | | BIS | #'0,R3 | ::MAKE THIS DIGIT ASCII |
| 015462 | 052703 | 000040 | 5\$: | BIS | #' R3 | ::MAKE ASCII IF NOT ALREADY |
| 015466 | 110337 | 015532 | | MOVB | R3,8\$ | ::SAVE FOR TYPING |
| 015472 | 104401 | 015532 | | TYPE | 8\$ | ::GO TYPE THIS DIGIT |
| 015476 | 105337 | 015534 | 7\$: | DECB | \$OCNT | ::COUNT BY 1 |
| 015502 | 003347 | | | BGT | 2\$ | ::BR IF MORE TO DO |
| 015504 | 002402 | | | BLT | 6\$ | ::BR IF DONE |
| 015506 | 005204 | | | INC | R4 | ::INSURE LAST DIGIT ISN'T A BLANK |
| 015510 | 000744 | | | BR | 2\$ | ::GO DO THE LAST DIGIT |
| 015512 | 012605 | | 6\$: | MOV | (SP)+,R5 | ::RESTORE R5 |
| 015514 | 012604 | | | MOV | (SP)+,R4 | ::RESTORE R4 |
| 015516 | 012603 | | | MOV | (SP)+,R3 | ::RESTORE R3 |
| 015520 | 016666 | 000002 000004 | | MOV | 2(SP),4(SP) | ::SET THE STACK FOR RETURNING |
| 015526 | 012616 | | | MOV | (SP)+,(SP) | |
| 015530 | 000002 | | | RTI | | ::RETURN |
| 015532 | 000 | | 8\$: | .BYTE | 0 | ::STORAGE FOR ASCII DIGIT |
| 015533 | 000 | | | .BYTE | 0 | ::TERMINATOR FOR TYPE ROUTINE |
| 015534 | 000 | | \$OCNT: | .BYTE | 0 | ::OCTAL DIGIT COUNTER |
| 015535 | 000 | | \$OFILL: | .BYTE | 0 | ::ZERO FILL SWITCH |
| 015536 | 000000 | | \$OMODE: | .WORD | 0 | ::NUMBER OF DIGITS TO TYPE |

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

```

015540          $TYPDS:
015540 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
015542 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
015544 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
015546 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
015550 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
015552 012746 020200  MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
015556 016605 000020  MOV      20(SP),R5      ;;GET THE INPUT NUMBER
015562 100004      BPL      1$      ;;BR IF INPUT IS POS.
015564 005405      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
015566 112766 000055 000001  MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
015574 005000      CLR      R0      ;;ZERO THE CONSTANTS INDEX
015576 012703 015754      MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
015602 112723 000040      MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
015606 005002      CLR      R2      ;;CLEAR THE BCD NUMBER
015610 016001 015744      MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
015614 160105      3$:      SUB      R1,R5      ;;FORM THIS BCD DIGIT
015616 002402      BLT      4$      ;;BR IF DONE
015620 005202      INC      R2      ;;INCREASE THE BCD DIGIT BY 1
015622 000774      BR      3$
015624 060105      4$:      ADD      R1,R5      ;;ADD BACK THE CONSTANT
015626 005702      TST      R2      ;;CHECK IF BCD DIGIT=0
015630 001002      BNE      5$      ;;FALL THROUGH IF 0
015632 105716      TSTB     (SP)      ;;STILL DOING LEADING 0'S?
015634 100407      BMI      7$      ;;BR IF YES
015636 106316      5$:      ASLB     (SP)      ;;MSD?
015640 103003      BCC      6$      ;;BR IF NO
015642 116663 000001 177777  MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN
015650 052702 000060      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
015654 052702 000040      7$:      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
015660 110223      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
015662 005720      TST      (R0)+      ;;JUST INCREMENTING
015664 020027 000010      CMP      R0,#10      ;;CHECK THE TABLE INDEX
015670 002746      BLT      2$      ;;GO DO THE NEXT DIGIT
015672 003002      BGT      8$      ;;GO TO EXIT
015674 010502      MOV      R5,R2      ;;GET THE LSD
015676 000764      BR      6$      ;;GO CHANGE TO ASCII
015700 105726      8$:      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
015702 100003      BPL      9$      ;;BR IF NO
015704 116663 177777 177776  MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
015712 105013      9$:      CLRB     (R3)      ;;SET THE TERMINATOR
015714 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
015716 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
015720 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
015722 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
    
```


| | | | | | | |
|--------|--------|--------|---------|--------|-------------|-----------------------|
| 015724 | 012600 | | | MOV | (SP)+,R0 | ::POP STACK INTO R0 |
| 015726 | 104401 | 015754 | | TYPE | ,SDBLK | ::NOW TYPE THE NUMBER |
| 015732 | 016666 | 000002 | 000004 | MOV | 2(SP),4(SP) | ::ADJUST THE STACK |
| 015740 | 012616 | | | MOV | (SP)+,(SP) | |
| 015742 | 000002 | | | RTI | | ::RETURN TO USER |
| 015744 | 023420 | | \$DTBL: | 10000. | | |
| 015746 | 001750 | | | 1000. | | |
| 015750 | 000144 | | | 100. | | |
| 015752 | 000012 | | | 10. | | |
| 015754 | | | \$DBLK: | .BLKW | 4 | |

.SBTTL TTY INPUT ROUTINE

```

:*****
:ENABL LSB
015764 000000 $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
015766 000000 $TKQIN: .WORD 0      ;;INPUT POINTER
015770 000000 $TKQOUT: .WORD 0     ;;OUTPUT POINTER
015772      015773 $TKQSRT: .BLKB 1     ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
:
;*CALL:
:
:      JSR      PC,$TKINT
:      RETURN
:
015774 005037 015764 $TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
016000 012737 015772 015766 MOV      # $TKQSRT,$TKQIN  ;;MOVE THE STARTING ADDRESS OF THE
016006 013737 015766 015770 MOV      $TKQIN,$TKQOUT    ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
016014 012737 016044 000060 MOV      # $TKSRV,@ $TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
016022 012737 000200 000062 MOV      #200,@ $TKVEC+2    ;;'BR' LEVEL 4
016030 005777 163112 TST      @ $TKB           ;;CLEAR DONE FLAG
016034 012777 000100 163102 MOV      #100,@ $TKS       ;;ENABLE TTY KEYBOARD INTERRUPT
016042 000207      RTS      PC           ;;RETURN TO CALLER

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START)
:
$TKSRV: MOVB     @ $TKB, -(SP)      ;;PICKUP THE CHARACTER
        BIC     #^C177, (SP)      ;;STRIP THE JUNK
        CMP     (SP), # $XON      ;;IS IT A RANDOM XON?
        BNE     30$              ;;BRANCH IF NO
        TST     (SP)+            ;;CLEAN RANDOM XON OFF STACK
        RTI
30$:
        CMP     (SP), #3          ;;IS IT A CONTROL C?
        BNE     1$              ;;BRANCH IF NO
        TYPE    ,SCNTLC          ;;TYPE A CONTROL-C (^C)
        JSR     PC,$TKINT        ;;INIT THE KEYBOARD
        TST     (SP)+            ;;CLEAN UP STACK
        JMP     START            ;;CONTROL C RESTART
1$:
        CMP     (SP), #7          ;;IS IT A CONTROL G?
        BNE     2$              ;;BRANCH IF NO
        CMP     #SWREG,SWR       ;;IS SOFT-SWR SELECTED?
        BEQ     6$              ;;GO TO SWR CHANGE
2$:
        CMP     #1,$TKCNT        ;;IS THE QUEUE FULL?
        BNE     3$              ;;BRANCH IF NO
        TYPE    ,SBELL           ;;RING THE TTY BELL
016044 117746 163076 $TKSRV: MOVB     @ $TKB, -(SP)
016050 042716 177600 BIC     #^C177, (SP)
016054 021627 000021 CMP     (SP), # $XON
016060 001002 BNE     30$
016062 005726 TST     (SP)+
016064 000002 RTI
016066
016066 021627 000003 30$:
016072 001007 CMP     (SP), #3
016074 104401 BNE     1$
016100 004737 017172 TYPE    ,SCNTLC
016104 005726 JSR     PC,$TKINT
016106 000137 001766 TST     (SP)+
016112 021627 000007 JMP     START
016116 001004 1$:
016120 022737 000176 001140 CMP     #SWREG,SWR
016126 001500 BEQ     6$

016130
016130 022737 000001 015764 2$:
016136 001004 CMP     #1,$TKCNT
016140 104401 001202 BNE     3$
        TYPE    ,SBELL

```

```

016144 005726          TST      (SP)+          ;;CLEAN CHARACTER OFF OF STACK
016146 000451          BR       5$              ;;EXIT
016150 021627 000023  3$:    CMP      (SP),#23          ;;IS IT A CONTROL-S?
016154 001021          BNE     32$              ;;BRANCH IF NO
016156 005077 162762  CLR     @STKS            ;;DISABLE TTY KEYBOARD INTERRUPTS
016162 005726          TST      (SP)+          ;;CLEAN CHAR OFF STACK
016164 105777 162754  31$:  TSTB   @STKS            ;;WAIT FOR A CHAR
016170 100375          BPL     31$              ;;LOOP UNTIL ITS THERE
016172 117746 162750  MOVB   @STKB,-(SP)       ;;GET THE CHARACTER
016176 042716 177600  BIC    #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
016202 022627 000021  CMP    (SP)+,#21        ;;IS IT A CONTROL-Q?
016206 001366          BNE     31$              ;;BRANCH IF NO
016210 012777 000100 162726 MOV    #100,@STKS       ;;REENABLE TTY KEYBOARD INTERRUPTS
016216 000002          RTI                    ;;RETURN
016220 005237 015764  32$:  INC    $TKCNT          ;;COUNT THIS CHARACTER
016224 021627 000140  CMP    (SP),#140        ;;IS IT UPPER CASE?
016230 002405          BLT    4$                ;;BRANCH IF YES
016232 021627 000175  CMP    (SP),#175        ;;IS IT A SPECIAL CHAR?
016236 003002          BGT    4$                ;;BRANCH IF YES
016240 042716 000040  BIC    #40,(SP)         ;;MAKE IT UPPER CASE
016244 112677 177516  4$:  MOVB   (SP)+,@$TKQIN    ;;AND PUT IT IN QUEUE
016250 005237 015766          INC    $TKQIN           ;;UPDATE THE POINTER
016254 023727 015766 015773 CMP    $TKQIN,$$TKQEND  ;;GO OFF THE END?
016262 001003          BNE     5$              ;;BRANCH IF NO
016264 012737 015772 015766 MOV    $$TKQRT,$$TKQIN  ;;RESET THE POINTER
016272 000002          5$:  RTI                    ;;RETURN
  
```

```

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

```

016274 022737 000176 001140 $CKSWR: CMP    #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
016302 001124          BNE     15$              ;;EXIT IF NOT
016304 105777 162634  TSTB   @STKS            ;;IS A CHAR WAITING?
016310 100121          BPL     15$              ;;IF NOT, EXIT
016312 117746 162630  MOVB   @STKB,-(SP)       ;;YES
016316 042716 177600  BIC    #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
016322 021627 000007  CMP    (SP),#7          ;;IS IT A CONTROL-G?
016326 001300          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                                ;;AND EXIT
  
```

```

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

```

016330 123727 001134 000001 6$:  CMPB   $AUTOB,#1        ;;ARE WE RUNNING IN AUTO-MODE?
016336 001674          BEQ    2$                ;;BRANCH IF YES
016340 005726          TST    (SP)+            ;;CLEAR CONTROL-G OFF STACK
016342 004737 015774  JSR    PC,$TKINT        ;;FLUSH THE TTY INPUT QUEUE
016346 005077 162572  CLR    @STKS            ;;DISABLE TTY KEYBOARD INTERRUPTS
016352 112737 000001 001135 MOVB   #1,$INTAG        ;;SET INTERRUPT MODE INDICATOR

016360 104401 017204          TYPE   ,$CNTLG          ;;ECHO THE CONTROL-G (^G)
016364 104401 017211  $GTSWR: TYPE   ,$MSWR     ;;TYPE CURRENT CONTENTS
016370 013746 000176          MOV    $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
016374 104402          TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  
```

```

016376 104401 017222          TYPE      ,SMNEW      ;;PROMPT FOR NEW SWR
016402 005046          19$: CLR      -(SP)      ;;CLEAR COUNTER
016404 005046          CLR      -(SP)      ;;THE NEW SWR
016406 105777 162532      7$: TSTB     @STKS     ;;CHAR THERE?
016412 100375          BPL      7$        ;;IF NOT TRY AGAIN

016414 117746 162526      MOVB     @STKB,-(SP)  ;;PICK UP CHAR
016420 042716 177600      BIC     #^C177,(SP) ;;MAKE IT 7-BIT ASCII

016424 021627 000003      CMP      (SP),#3     ;;IS IT A CONTROL-C?
016430 001015          BNE     9$          ;;BRANCH IF NOT
016432 104401 017172      TYPE     ,SCNTLC    ;;YES, ECHO CONTROL-C (^C)
016436 062706 000006      ADD     #6,SP       ;;CLEAN UP STACK
016442 123727 001135 000001 CMPB     $INTAG,#1   ;;REENABLE TTY KEYBOARD INTERRUPTS?
016450 001003          BNE     8$          ;;BRANCH IF NO
016452 012777 000100 162464 8$: MOV     #100,@STKS  ;;ALLOW TTY KEYBOARD INTERRUPTS
016460 000137 001766      JMP     START      ;;CONTROL-C RESTART

016464 021627 000025      9$: CMP     (SP),#25  ;;IS IT A CONTROL-U?
016470 001005          BNE     10$         ;;BRANCH IF NOT
016472 104401 017177      TYPE     ,SCNTLU    ;;YES, ECHO CONTROL-U (^U)
016476 062706 000006      20$: ADD     #6,SP   ;;IGNORE PREVIOUS INPUT
016502 000737          BR      19$         ;;LET'S TRY IT AGAIN

016504 021627 000015      10$: CMP     (SP),#15  ;;IS IT A <CR>?
016510 001022          BNE     16$         ;;BRANCH IF NO
016512 005766 000004      TST     4(SP)      ;;YES, IS IT THE FIRST CHAR?
016516 001403          BEQ     11$         ;;BRANCH IF YES
016520 016677 000002 162412 MOV     2(SP),@SWR  ;;SAVE NEW SWR
016526 062706 000006      11$: ADD     #6,SP   ;;CLEAR UP STACK
016532 104401 001207      14$: TYPE     ,SCRLF  ;;ECHO <CR> AND <LF>
016536 123727 001135 000001 CMPB     $INTAG,#1   ;;RE-ENABLE TTY KBD INTERRUPTS?
016544 001003          BNE     15$         ;;BRANCH IF NOT
016546 012777 000100 162370 MOV     #100,@STKS  ;;RE-ENABLE TTY KBD INTERRUPTS
016554 000002          RTI     ;;RETURN
016556 004737 015170      15$: JSR     PC,$TYPEC  ;;ECHO CHAR
016562 021627 000060      16$: CMP     (SP),#60  ;;CHAR < 0?
016566 002420          BLT     18$         ;;BRANCH IF YES
016570 021627 000067      CMP     (SP),#67   ;;CHAR > 7?
016574 003015          BGT     18$         ;;BRANCH IF YES
016576 042726 000060      BIC     #60,(SP)+  ;;STRIP-OFF ASCII
016602 005766 000002      TST     2(SP)      ;;IS THIS THE FIRST CHAR
016606 001403          BEQ     17$         ;;BRANCH IF YES
016610 006316          ASL     (SP)       ;;NO, SHIFT PRESENT
016612 006316          ASL     (SP)       ;;CHAR OVER TO MAKE
016614 006316          ASL     (SP)       ;;ROOM FOR NEW ONE.
016616 005266 000002      17$: INC     2(SP)    ;;KEEP COUNT OF CHAR
016622 056616 177776      BIS     -2(SP),(SP) ;;SET IN NEW CHAR
016626 000667          BR      7$        ;;GET THE NEXT ONE
016630 104401 001206      18$: TYPE     ,SQUES  ;;TYPE ?<CR><LF>
016634 000720          BR      20$      ;;SIMULATE CONTROL-U

.DSABL  LSB

```

;;*****

```

: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *      RDCHR          :: GET A CHARACTER FROM THE QUEUE
: *      RETURN HERE   :: CHARACTER IS ON THE STACK
: *                   :: WITH PARITY BIT STRIPPED OFF
:
016636 011646          SRDCHR: MOV      (SP),-(SP)      :: PUSH DOWN THE PC AND
016640 016666 000004 000002  MOV      4(SP),2(SP)    :: THE PS
016646 005066 000004          CLR      4(SP)        :: GET READY FOR A CHARACTER
016652 005046          CLR      -(SP)         :: PUT NEW PS ON STACK
016654 012746 016662          MOV      #64$,-(SP)    :: PUT NEW PC ON STACK
016660 000002          RTI                          :: POP NEW PC AND PS
016662
016662 005737 015764 64$:   TST      $TKCNT          :: WAIT ON A CHARACTER
016666 001775 1$:         BEQ      1$
016670 005337 015764          DEC      $TKCNT          :: DECREMENT THE COUNTER
016674 117766 177070 000004  MOVB   @$TKQOUT,4(SP)  :: GET ONE CHARACTER
016702 005237 015770          INC      $TKQOUT        :: UPDATE THE POINTER
016706 023727 015770 015773  CMP     $TKQOUT,$$TKQEND :: DID IT GO OFF OF THE END?
016714 001003          BNE     2$           :: BRANCH IF NO
016716 012737 015772 015770  MOV     $$TKQRT,$$TKQOUT :: RESET THE POINTER
016724 000002          RTI                          :: RETURN
2$:
: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:
: *      RDLIN          :: INPUT A STRING FROM THE TTY
: *      RETURN HERE   :: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: *                   :: TERMINATOR WILL BE A BYTE OF ALL 0'S
:
016726 010346          SRDLIN: MOV      R3, -(SP)      :: SAVE R3
016730 005046          CLR      -(SP)        :: CLEAR THE RUBOUT KEY
016732 012703 017162 1$:   MOV     $$TTYIN,R3      :: GET ADDRESS
016736 022703 017172 2$:   CMP     $$TTYIN+8.,R3    :: BUFFER FULL?
016742 101456          BLOS   4$           :: BR IF YES
016744 104410          RDCHR          :: GO READ ONE CHARACTER FROM THE TTY
016746 112613          MOVB   (SP)+,(R3)      :: GET CHARACTER
016750 122713 000177 10$:  CMPB   #177,(R3)        :: IS IT A RUBOUT
016754 001022          BNE   5$           :: BR IF NO
016756 005716          TST   (SP)           :: IS THIS THE FIRST RUBOUT?
016760 001007          BNE   6$           :: BR IF NO
016762 112737 000134 017160  MOVB   #' \ ,9$        :: TYPE A BACK SLASH
016770 104401 017160          TYPE   ,9$
016774 012716 177777          MOV   #-1,(SP)      :: SET THE RUBOUT KEY
017000 005303 6$:       DEC     R3              :: BACKUP BY ONE
017002 020327 017162          CMP   R3,$$TTYIN    :: STACK EMPTY?
017006 103434          BLO   4$           :: BR IF YES
017010 111337 017160          MOVB   (R3),9$      :: SETUP TO TYPEOUT THE DELETED CHAR.
017014 104401 017160          TYPE   ,9$        :: GO TYPE
017020 000746          BR    2$           :: GO READ ANOTHER CHAR.
017022 005716 5$:       TST   (SP)           :: RUBOUT KEY SET?
017024 001406          BEQ   7$           :: BR IF NO
017026 112737 000134 017160  MOVB   #' \ ,9$        :: TYPE A BACK SLASH
017034 104401 017160          TYPE   ,9$
017040 005016          CLR   (SP)         :: CLEAR THE RUBOUT KEY
017042 122713 000025 7$:  CMPB   #25,(R3)      :: IS CHARACTER A CTRL U?
017046 001003          BNE   8$           :: BR IF NO
8$:

```

```

017050 104401 017177          TYPE      $CNTLU          ;;TYPE A CONTROL 'U'
017054 000726                BR          1$              ;;GO START OVER
017056 122713 000022      8$:  CMPB      #22,(R3)          ;;IS CHARACTER A '^R'?
017062 001011                BNE          3$              ;;BRANCH IF NO
017064 105013                CLRB      (R3)              ;;CLEAR THE CHARACTER
017066 104401 001207          TYPE      ,SCLF          ;;TYPE A 'CR' & 'LF'
017072 104401 017162          TYPE      ,STTYIN         ;;TYPE THE INPUT STRING
017076 000717                BR          2$              ;;GO PICKUP ANOTHER CHACTER
017100 104401 001206      4$:  TYPE      ,SQUES          ;;TYPE A '?'
017104 000712                BR          1$              ;;CLEAR THE BUFFER AND LOOP
017106 111337 017160      3$:  MOVB      (R3),9$          ;;ECHO THE CHARACTER
017112 104401 017160          TYPE      ,9$              ;;
017116 122723 000015          CMPB      #15,(R3)+        ;;CHECK FOR RETURN
017122 001305                BNE          2$              ;;LOOP IF NOT RETURN
017124 105063 177777          CLRB      -1(R3)          ;;CLEAR RETURN (THE 15)
017130 104401 001210          TYPE      ,SLF          ;;TYPE A LINE FEED
017134 005726                TST          (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
017136 012603                MOV          (SP)+,R3        ;;RESTORE R3
017140 011646                MOV          (SP),-(SP)     ;;ADJUST THE STACK AND PUT ADDRESS OF THE
017142 016666 000004 000002  MOV          4(SP),2(SP)    ;;FIRST ASCII CHARACTER ON IT
017150 012766 017162 000004  MOV          #STTYIN,4(SP)  ;;
017156 000002                RTI          ;;RETURN
017160          000          9$:  .BYTE      0          ;;STORAGE FOR ASCII CHAR. TO TYPE
017161          000                .BYTE      0          ;;TERMINATOR
017162                $TTYIN: .BLKB      8.          ;;RESERVE 8 BYTES FOR TTY INPUT
017172          136          103      015 $CNTLC: .ASCIZ  /^C/<15><12>  ;;CONTROL 'C'
017177          136          125      015 $CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
017204          136          107      015 $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
017211          015          012      123 $MSWR: .ASCIZ  <15><12>/SWR = /
017222          040          040      116 $MNEW: .ASCIZ  / NEW = /
          .EVEN
  
```

.SBTTL READ AN OCTAL NUMBER FROM THE TTY

```

*****
*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
*CHANGE IT TO BINARY.
*THE INPUT CHARACTERS WILL BE CHECKED TO INSURE THEY ARE LEGAL
*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
*CALL:
*      RDOCT          ::READ AN OCTAL NUMBER
*      RETURN HERE   ::LOW ORDER BITS ARE ON TOP OF THE STACK
*                   ::HIGH ORDER BITS ARE IN $HIOCT
  
```

```

017234 011646          SRDOCT: MOV      (SP),-(SP)      ::PROVIDE SPACE FOR THE
017236 016666 000004 000002 MOV      4(SP),2(SP)    ::INPUT NUMBER
017244 010046          MOV      R0,-(SP)      ::PUSH R0 ON STACK
017246 010146          MOV      R1,-(SP)      ::PUSH R1 ON STACK
017250 010246          MOV      R2,-(SP)      ::PUSH R2 ON STACK
017252 104411          1$:  RDLIN          ::READ AN ASCII LINE
017254 012600          MOV      (SP)+,R0      ::GET ADDRESS OF 1ST CHARACTER
017256 010037 017362  MOV      R0,5$          ::AND SAVE IT
017262 005001          CLR      R1          ::CLEAR DATA WORD
017264 005002          CLR      R2
017266 112046          2$:  MOVB     (R0)+,-(SP)    ::PICKUP THIS CHARACTER
017270 001420          BEQ      3$          ::IF ZERO GET OUT
017272 122716 000060  CMPB     #'0,(SP)      ::MAKE SURE THIS CHARACTER
017276 003026          BGT      4$          ::IS AN OCTAL DIGIT
017300 122716 000067  CMPB     #'7,(SP)
017304 002423          BLT      4$
017306 006301          ASL      R1          ::*2
017310 006102          ROL      R2
017312 006301          ASL      R1          ::*4
017314 006102          ROL      R2
017316 006301          ASL      R1          ::*8
017320 006102          ROL      R2
017322 042716 177770  BIC     #'^C7,(SP)    ::STRIP THE ASCII JUNK
017326 062601          ADD     (SP)+,R1    ::ADD IN THIS DIGIT
017330 000756          BR      2$          ::LOOP
017332 005726          3$:  TST     (SP)+      ::CLEAN TERMINATOR FROM STACK
017334 010166 000012  MOV     R1,12(SP)    ::SAVE THE RESULT
017340 010237 017372  MOV     R2,$HIOCT
017344 012602          MOV     (SP)+,R2    ::POP STACK INTO R2
017346 012601          MOV     (SP)+,R1    ::POP STACK INTO R1
017350 012600          MOV     (SP)+,R0    ::POP STACK INTO R0
017352 000002          RTI          ::RETURN
017354 005726          4$:  TST     (SP)+      ::CLEAN PARTIAL FROM STACK
017356 105010          CLRB   (R0)        ::SET A TERMINATOR
017360 104401          TYPE          ::TYPE UP THRU THE BAD CHAR.
017362 000000          5$:  .WORD   0
017364 104401 001206  TYPE     $QUES      ::'"' 'CR' & 'LF'
017370 000730          BR      1$        ::TRY AGAIN
017372 000000          $HIOCT: .WORD   0 ::HIGH ORDER BITS GO HERE
  
```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

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

```

017374
017374 010046
017376 010146
017400 010246
017402 010346
017404 010446
017406 010546
017410 016646 000022
017414 016646 000022
017420 016646 000022
017424 016646 000022
017430 000002
    
```

```

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

*RESTORE R0-R5

```

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

```

017432
017432 012666 000022
017436 012666 000022
017442 012666 000022
017446 012666 000022
017452 012605
017454 012604
017456 012603
017460 012602
017462 012601
017464 012600
017466 000002
    
```

2

.SBTTL TRAP DECODER

```

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

```

017470 010046
017472 016600 000002
017476 005740
017500 111000
    
```

```

$TRAP:
MOV R0,-(SP)      ;;SAVE R0
MOV 2(SP),R0      ;;GET TRAP ADDRESS
TST -(R0)         ;;BACKUP BY 2
MOVB (R0),R0      ;;GET RIGHT BYTE OF TRAP
    
```



```

017502 006300          ASL      R0          ::POSITION FOR INDEXING
017504 016000 017524  MOV      $TRPAD(R0),R0  ::INDEX TO TABLE
017510 000200          RTS      R0          ::GO TO ROUTINE
  
```

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

```

017512 011646          $TRAP2: MOV      (SP),-(SP)  ::MOVE THE PC DOWN
017514 016666 000004 000002  MOV      4(SP),2(SP)  ::MOVE THE PSW DOWN
017522 000002          RTI          ::RESTORE THE PSW
  
```

.SBTTL TRAP TABLE

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

```

:          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)

017524 017512          $GTSWR  ::CALL=GTSWR   TRAP+6(104406)  GET SOFT-SWR SETTING

017540 016364          $CKSWR  ::CALL=CKSWR   TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
017542 016274          $RDCHR  ::CALL=RDCHR   TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
017544 016636          $RDLIN  ::CALL=RDLIN   TRAP+11(104411) TTY TYPEIN STRING ROUTINE
017546 016726          $RDOCT  ::CALL=RDOCT   TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
017550 017234          $SAVREG ::CALL=SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
017552 017374          $RESREG ::CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
017554 017432
  
```

.SBTTL TELETYPE MESSAGES

| | | | | |
|----|--------|-----|-----|--|
| 1 | | | | |
| 2 | | | | |
| 3 | 017556 | 200 | 105 | 116 ENTERA: .ASCIZ <CRLF>/ENTER DRIVE ADDRESS: / |
| 4 | 017605 | 040 | 077 | 111 ADRRER: .ASCIZ / ?INVALID ADDRESS/<CRLF> |
| 5 | 017630 | 200 | 120 | 117 PORTAIS: .ASCIZ <CRLF>/PORT 'A' ADDRESS IS: / |
| 6 | 017657 | 200 | 120 | 117 PORTBIS: .ASCIZ <CRLF>/PORT 'B' ADDRESS IS: / |
| 7 | 017706 | 200 | 123 | 131 NOCLOCK: .ASCIZ <CRLF>/SYSTEM MUST HAVE 'L' OR 'P' CLOCK/<CRLF><LF> |
| 8 | 017753 | 012 | 105 | 116 TESTNO: .ASCIZ <LF>/ENTER TEST #: / |
| 9 | 017773 | 040 | 077 | 111 BADNO: .ASCIZ / ?INVALID TEST NUMBER/<CRLF> |
| 10 | 020022 | 200 | 012 | 122 ADDRIS: .ASCIZ <CRLF><LF>@RH/RM ADDRESS (RMCS1) IS: @ |
| 11 | 020057 | 012 | 105 | 116 NTRH: .ASCIZ <LF>@ENTER RH/RM ADDRESS: @ |
| 12 | 020106 | 200 | 122 | 105 SWTCHN: .ASCIZ <CRLF>@RETURN 'PORT SELECT' SWITCH TO 'A/B'@ |
| 13 | 020154 | 200 | 123 | 127 SWTCHA: .ASCIZ <CRLF>/SWITCH 'PORT SELECT' TO 'A'/' |
| 14 | 020211 | 200 | 123 | 127 SWTCHB: .ASCIZ <CRLF>/SWITCH 'PORT SELECT' TO 'B'/' |
| 15 | 020246 | 200 | 124 | 110 CONTUE: .ASCIZ <CRLF>/THEN PRESS 'CONTINUE' ON THE PROCESSOR/<CRLF> |
| 16 | 020317 | 200 | 123 | 124 CYCLED: .ASCIZ <CRLF>/STOP THE DRIVE/ |
| 17 | 020337 | 200 | 123 | 124 CYCLEU: .ASCIZ <CRLF>/START THE DRIVE, THE PROGRAM WILL WAIT FOR 'MOL' TO SET/<CRLF> |

```

1          .SBTTL  TEST ERROR MESSAGES
2
3 020431    104    122    111  EM1:  .ASCIZ  /DRIVE IS NON-EXISTENT ('NED' BIT SET)/
4 020477    127    122    117  EM2:  .ASCIZ  /WRONG DRIVE TYPE/
5 020520    120    117    122  EM3:  .ASCIZ  @PORT SELECT SWITCH ON DRIVE NOT IN 'A/B'@
6 020571    104    122    111  EM4:  .ASCIZ  /DRIVE NOT ON LINE/
7 020613    123    105    122  EM5:  .ASCIZ  /SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME/
8 020675    124    111    115  EM6:  .ASCIZ  /TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS/
9 020747    124    111    115  EM7:  .ASCIZ  /TIMEOUT ONE-SHOT IS LESS THAN 500 MS/
10 021014   122    105    101  EM10: .ASCIZ  /READ IN PRESET DOES NOT SET VOLUME VALID FOR THE PORT/
11 021102   047    107    117  EM11: .ASCIZ  /'GO' BIT RESET DURING UNLOAD COMMAND/
12 021147   111    116    103  EM12: .ASCIZ  /INCORRECT STATUS DURING UNLOAD COMMAND/
13 021216   104    122    111  EM13: .ASCIZ  /DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND/
14 021303   101    124    124  EM14: .ASCIZ  /ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD/
15 021365   101    124    124  EM15: .ASCIZ  /ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD'/
16 021451   104    122    111  EM16: .ASCII  /DRIVE NOT IN NEUTRAL AFTER UNLOAD, WITH 'PORT/<CR><LF>
17 021530   123    105    114  .ASCIZ  @SELECT' SWITCH MOVED FROM 'A/B' @
18 021571   104    122    111  EM17: .ASCIZ  /DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP/
19 021654   104    122    111  EM20: .ASCIZ  /DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP/
20 021737   123    124    101  EM21: .ASCIZ  /STATUS INCORRECT FOR PORT AFTER CYCLE UP/
21 022010   122    105    107  EM22: .ASCIZ  /REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT/
22 022106   047    116    105  EM23: .ASCIZ  /'NED' NOT SET WHEN RMDs ACCESSED THROUGH PORT NOT SWITCHED/
23 022201   104    122    111  EM24: .ASCIZ  /DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/
24 022261   122    110    057  EM25: .ASCIZ  @RH/RM DIDN'T RESPOND TO ADDRESSING@
25 022324   104    122    111  EM30: .ASCIZ  /DRIVE NOT SEIZED BY PORT/
26 022355   127    122    117  EM31: .ASCIZ  /WRONG STATUS SEEN BY THE SEIZING PORT/
27 022423   122    105    107  EM32: .ASCIZ  /REGISTER CONTENTS WRONG/
28 022453   103    117    116  EM33: .ASCIZ  /CONTROL BUS PARITY ERROR READING INDICATED REGISTER/
29 022537   103    101    116  EM34: .ASCIZ  /CAN'T ACCESS DRIVE THROUGH EITHER PORT/
30 022606   104    122    111  EM35: .ASCIZ  /DRIVE NOT IN NEUTRAL AFTER RELEASE - REQUEST NOT SET/
31 022673   104    122    111  EM36: .ASCIZ  /DRIVE NOT IN NEUTRAL AFTER TIMEOUT - REQUEST NOT SET/
32 022760   122    105    107  EM37: .ASCIZ  /REGISTER CONTENTS WRONG AFTER RELEASE OR TIMEOUT/
33 023041   104    122    111  EM40: .ASCIZ  /DRIVE IN NEUTRAL AFTER RELEASE - REQUEST SET/
34 023116   122    105    107  EM41: .ASCIZ  /REGISTER WRONG AFTER RELEASE WITH REQUEST SET/
    
```


1
2
3
4 025040 003112
7 025042 004424
025044 005206
025046 005770
025050 006706
025052 007624
025054 010672
025056 012002
8
9
10
11 025060 001
12 025061 002
13 025062 004
14 025063 010
15 025064 020
16 025065 040
17 025066 100
18 025067 200
19
22 025070 000011
23
24 000200

.SBTTL CONSTANTS, TABLES, ETC
:TABLE OF TEST STARTING ADDRESSES

TSTADR: .WORD TST1 ;STARTING ADDRESS OF TEST 1
.WORD TST2 ;STARTING ADDRESS OF TEST 2
.WORD TST3 ;STARTING ADDRESS OF TEST 3
.WORD TST4 ;STARTING ADDRESS OF TEST 4
.WORD TST5 ;STARTING ADDRESS OF TEST 5
.WORD TST6 ;STARTING ADDRESS OF TEST 6
.WORD TST7 ;STARTING ADDRESS OF TEST 7
.WORD TST10 ;STARTING ADDRESS OF TEST 10

:ATTENTION BIT TABLE

ATABIT: .BYTE 1 ;ATTENTION BIT FOR DRIVE 0
.BYTE 2 ;ATTENTION BIT FOR DRIVE 1
.BYTE 4 ;ATTENTION BIT FOR DRIVE 2
.BYTE 10 ;ATTENTION BIT FOR DRIVE 3
.BYTE 20 ;ATTENTION BIT FOR DRIVE 4
.BYTE 40 ;ATTENTION BIT FOR DRIVE 5
.BYTE 100 ;ATTENTION BIT FOR DRIVE 6
.BYTE 200 ;ATTENTION BIT FOR DRIVE 7

MAXTN: .WORD 11 ;MAXIMUM TEST NUMBER

.END 200

| | | | | | | | | | |
|--------|----------|--------|----------|--------|----------|--------|----------|--------|----------|
| ADDRIS | 020022 | CPSAVE | 014454 | DT14 | 024644 | F2 | = 000010 | PORTBI | 017657 |
| ADRERR | 017605 | CR | = 000015 | DT17 | 024662 | F3 | = 000020 | PORTC | 001230 |
| AOE | = 001000 | CRLF | = 000200 | DT2 | 024560 | F4 | = 000040 | PRO | = 000000 |
| ASR1 | 001232 | CYCLED | 020317 | DT24 | 024670 | GO | = 000001 | PR1 | = 000040 |
| ATA | = 100000 | CYCLEU | 020337 | DT25 | 024702 | GRV | = 000010 | PR2 | = 000100 |
| ATABIT | 025060 | DCK | = 100000 | DT30 | 024706 | GTSWR | = 104406 | PR3 | = 000140 |
| ATO | = 000001 | DDISP | = 177570 | DT34 | 024726 | HCE | = 000200 | PR4 | = 000200 |
| AT1 | = 000002 | DE1 | = 000040 | DT35 | 024740 | HCI | = 002000 | PR5 | = 000240 |
| AT2 | = 000004 | DF20 | = 000002 | DT40 | 024752 | HCRC | = 000400 | PR6 | = 000300 |
| AT3 | = 000010 | DF1 | 024764 | DT5 | 024576 | HT | = 000011 | PR7 | = 000340 |
| AT4 | = 000020 | DF14 | 025013 | DT6 | 024612 | IAE | = 002000 | PS | = 177776 |
| AT5 | = 000040 | DF17 | 025021 | DT7 | 024622 | IBSAVE | 014456 | PSEL | = 002000 |
| AT6 | = 000100 | DF2 | 024771 | DVA | = 004000 | IE | = 000100 | PSW | = 177776 |
| AT7 | = 000200 | DF25 | 025023 | DVC | = 000200 | ILF | = 000001 | PTNBR | 001234 |
| A16 | = 000400 | DF30 | 025024 | ECH | = 000100 | ILR | = 000002 | PWRVEC | = 000024 |
| A17 | = 001000 | DF34 | 025033 | ECI | = 004000 | IOTVEC | = 000020 | RDCHR | = 104410 |
| BADNO | 017773 | DF5 | 024777 | EMTVEC | = 000030 | IR | = 000100 | RDLIN | = 104411 |
| BADTMO | 001706 | DF6 | 025004 | EM1 | 020431 | IVC | = 010000 | RDOCT | = 104412 |
| BAI | = 000010 | DF7 | 025007 | EM10 | 021014 | KYBCTL | 001266 | RDY | = 000200 |
| BIT0 | = 000001 | DH1 | 023174 | EM11 | 021102 | LBC | = 002000 | RELERR | 001250 |
| BIT00 | = 000001 | DH13 | 023464 | EM12 | 021147 | LBT | = 002000 | RELOK | = 000001 |
| BIT01 | = 000002 | DH14 | 023541 | EM13 | 021216 | LF | = 000012 | RESREG | = 104414 |
| BIT02 | = 000004 | DH17 | 023660 | EM14 | 021303 | LSC | = 004000 | RESVEC | = 000010 |
| BIT03 | = 000010 | DH2 | 023245 | EM15 | 021365 | MAXTN | 025070 | RMAS | = 000016 |
| BIT04 | = 000020 | DH24 | 023677 | EM16 | 021451 | MCPE | = 020000 | RMBA | = 000004 |
| BIT05 | = 000040 | DH25 | 024002 | EM17 | 021571 | MDPE | = 000400 | RMCS1 | = 000000 |
| BIT06 | = 000100 | DH30 | 024011 | EM2 | 020477 | MOH | = 020000 | RMCS2 | = 000010 |
| BIT07 | = 000200 | DH34 | 024133 | EM20 | 021654 | MOL | = 010000 | RMDA | = 000006 |
| BIT08 | = 000400 | DH35 | 024227 | EM21 | 021737 | MXF | = 001000 | RMDB | = 000022 |
| BIT09 | = 001000 | DH37 | 024324 | EM22 | 022010 | NBA | = 100000 | RMDC | = 000034 |
| BIT1 | = 000002 | DH40 | 024446 | EM23 | 022106 | NED | = 010000 | RMDS | = 000012 |
| BIT10 | = 002000 | DH5 | 023321 | EM24 | 022201 | NEM | = 004000 | RMDT | = 000026 |
| BIT11 | = 004000 | DH6 | 023370 | EM25 | 022261 | NOATA | = 000000 | RMEC1 | = 000044 |
| BIT12 | = 010000 | DH7 | 023417 | EM3 | 020520 | NOCLOC | 017706 | RMEC2 | = 000046 |
| BIT13 | = 020000 | DIGB | = 000004 | EM30 | 022324 | NOSEIZ | 001246 | RMER1 | = 000014 |
| BIT14 | = 040000 | DISPLA | 001142 | EM31 | 022355 | NTRH | 020057 | RMER2 | = 000042 |
| BIT15 | = 100000 | DISPRE | 000174 | EM32 | 022423 | OFD | = 000200 | RMLA | = 000020 |
| BIT2 | = 000004 | DLT | = 100000 | EM33 | 022453 | OPE | = 020000 | RMMR1 | = 000024 |
| BIT3 | = 000010 | DL64 | = 000020 | EM34 | 022537 | OPI | = 020000 | RMMR2 | = 000040 |
| BIT4 | = 000020 | DMD | = 000001 | EM35 | 022606 | OPPRT | 001240 | RMOF | = 000032 |
| BIT5 | = 000040 | DPE | = 000010 | EM36 | 022673 | OR | = 000200 | RMR | = 000004 |
| BIT6 | = 000100 | DPR | = 000400 | EM37 | 022760 | PAR | = 000010 | RMSN | = 000030 |
| BIT7 | = 000200 | DRQ | = 004000 | EM4 | 020571 | PAT | = 000020 | RMWC | = 000002 |
| BIT8 | = 000400 | DRY | = 000200 | EM40 | 023041 | PFECH | 014656 | R6 | = 000006 |
| BIT9 | = 001000 | DSWR | = 177570 | EM41 | 023116 | PFECH1 | 014666 | R7 | = 000007 |
| BPTVEC | = 000014 | DTE | = 010000 | EM5 | 020613 | PFECH2 | 014750 | SAVREG | = 104413 |
| CHANGE | 002764 | DT00 | = 000001 | EM6 | 020675 | PFECH3 | 015002 | SC | = 100000 |
| CHGADR | 001270 | DT01 | = 000002 | EM7 | 020747 | PFECH4 | 015012 | SCOPE | = 000004 |
| CKCLK | 013336 | DT02 | = 000004 | ENTERA | 017556 | PFTSTN | 015016 | SC0 | = 000100 |
| CKCLK1 | 013406 | DT03 | = 000010 | ERR | = 040000 | PGE | = 002000 | SC1 | = 000200 |
| CKCLK2 | 013450 | DT04 | = 000020 | ERROR | = 104000 | PGM | = 001000 | SC2 | = 000400 |
| CKCLK3 | 013460 | CT05 | = 000040 | ERRVEC | = 000004 | PIP | = 020000 | SC3 | = 001000 |
| CKERR | 001244 | DT06 | = 000100 | EXEC | 002610 | PIRQ | = 177772 | SC4 | = 002000 |
| CKSWR | = 104407 | DT07 | = 000200 | FER | = 000020 | PIRQVE | = 000240 | SEIZPT | 001236 |
| CLOCK | 013470 | DT08 | = 000400 | FMT16 | = 010000 | PORTA | 001224 | SKI | = 100000 |
| CLR | = 000040 | DT1 | 024544 | F0 | = 000002 | PORTAI | 017630 | STACK | = 001100 |
| CONTUE | 020246 | DT13 | 024634 | F1 | = 000004 | PORTB | 001226 | START | 001766 |

| | | | | | | | | | |
|---------|----------|---------|----------|---------|----------|----------|----------|----------|----------|
| START1 | 001776 | TEST3 | 005244 | WCE | = 040000 | \$GTSWR | 016364 | \$SVPC | = 000210 |
| START2 | 002004 | TEST4 | 006026 | WCF | = 000040 | \$GT42P | 013302 | \$SWR | = 166000 |
| STKLMT= | 177774 | TEST5 | 006744 | WLE | = 004000 | \$HD | = 000000 | \$SWRMK= | 000000 |
| STOP | 014104 | TEST6 | 007662 | WRL | = 004000 | \$HIOCT | 017372 | \$TIMES | 001176 |
| SWR | 001140 | TEST7 | 010730 | \$AUTOB | 001134 | \$ICNT | 001104 | \$TKB | 001146 |
| SWREG | 000176 | TIME | 001252 | \$BDADR | 001122 | \$INTAG | 001135 | \$TKCNT | 015764 |
| SWTCHA | 020154 | TIMEA | 001256 | \$BDDAT | 001126 | \$ITEMB | 001114 | \$TKINT | 015774 |
| SWTCHB | 020211 | TIMEAP | 001260 | \$BELL | 001202 | \$LF | 001210 | \$TKQEN= | 015773 |
| SWTCHN | 020106 | TIMEB | 001262 | \$CHARC | 015306 | \$LKCSB | 001214 | \$TKQIN | 015766 |
| SWO | = 000001 | TIMEBP | 001264 | \$CKSWR | 016274 | \$LKCSR | 001212 | \$TKQOU | 015770 |
| SW00 | = 000001 | TKVEC | = 000060 | \$CMTAG | 001100 | \$LKS | 001220 | \$TKQSR | 015772 |
| SW01 | = 000002 | TOLER | 013522 | \$CM1 | = 000001 | \$LLVEC | 001222 | \$TKS | 001144 |
| SW02 | = 000004 | TPVEC | = 000064 | \$CM2 | = 000002 | \$LPADR | 001106 | \$TKSRV | 016044 |
| SW03 | = 000010 | TRAPVE= | 000034 | \$CM3 | = 000001 | \$LPERR | 001110 | \$TMP0 | 001164 |
| SW04 | = 000020 | TRE | = 040000 | \$CM4 | = 000005 | \$LPVEC | 001216 | \$TMP1 | 001166 |
| SW05 | = 000040 | TRTVEC= | 000014 | \$CNTLC | 017172 | \$MNEW | 017222 | \$TMP2 | 001170 |
| SW06 | = 000100 | TSTADR | 025040 | \$CNTLG | 017204 | \$MSWR | 017211 | \$TMP3 | 001172 |
| SW07 | = 000200 | TSTNUM | 001242 | \$CNTLU | 017177 | \$MXCNT | 014102 | \$TMP4 | 001174 |
| SW08 | = 000400 | TST1 | 003112 | \$CRLF | 001207 | \$NULL | 001154 | \$TN | = 000012 |
| SW09 | = 001000 | TST1AA | 003074 | \$DBLK | 015754 | \$NWTST= | 000000 | \$TPB | 001152 |
| SW1 | = 000002 | TST10 | 012002 | \$DOAGN | 013326 | \$OCNT | 015534 | \$TPFLG | 001157 |
| SW10 | = 002000 | TST11 | 013106 | \$DTBL | 015744 | \$OMODE | 015536 | \$TPS | 001150 |
| SW11 | = 004000 | TST2 | 004424 | \$ENDAD | 013316 | \$OVER | 014066 | \$TRAP | 017470 |
| SW12 | = 010000 | TST3 | 005206 | \$ENDCT | 013154 | \$PASS | 001100 | \$TRAP2 | 017512 |
| SW13 | = 020000 | TST4 | 005770 | \$ENULL | 013332 | \$QUES | 001206 | \$TRP | = 000015 |
| SW14 | = 040000 | TST5 | 006706 | \$EOP | 013110 | \$RDCHR | 016636 | \$TRPAD | 017524 |
| SW15 | = 100000 | TST6 | 007624 | \$EOPCT | 013146 | \$RDLIN | 016726 | \$TSTNM | 001102 |
| SW2 | = 000004 | TST7 | 010672 | \$ERFLG | 001103 | \$RDOCT | 017234 | \$TTYIN | 017162 |
| SW3 | = 000010 | TYPDS | = 104405 | \$ERMAX | 001115 | \$RDSZ | = 000010 | \$TYPDS | 015540 |
| SW4 | = 000020 | TYPE | = 104401 | \$ERROR | 014132 | \$REGAD | 001160 | \$TYPE | 015020 |
| SW5 | = 000040 | TYPOC | = 104402 | \$ERRPC | 001116 | \$REGO | 001162 | \$TYPEC | 015170 |
| SW6 | = 000100 | TYPON | = 104404 | \$ERRTB | 001276 | \$RESRE | 017432 | \$TYPEX | 015310 |
| SW7 | = 000200 | TYPOS | = 104403 | \$ERRTY | 014460 | \$RMADR | 001272 | \$TYPOC | 015336 |
| SW8 | = 000400 | UNS | = 040000 | \$ERTTL | 001112 | \$RMVEC | 001274 | \$TYPON | 015352 |
| SW9 | = 001000 | UPE | = 020000 | \$ESCAP | 001200 | \$RTNAD | 013330 | \$TYPOS | 015312 |
| TAP | = 040000 | U0 | = 000001 | \$FILLC | 001156 | \$SAVRE | 017374 | \$XOFF | = 000023 |
| TBITVE= | 000014 | U1 | = 000002 | \$FILLS | 001155 | \$SCOPE | 013550 | \$XON | = 000021 |
| TESTNO | 017753 | U3 | = 000004 | \$GDADR | 001120 | \$SETUP= | 000127 | \$XTSTR | 013572 |
| TEST1 | 003150 | VV | = 000100 | \$GDDAT | 001124 | \$STUP | = 177777 | \$GET4= | 000000 |
| TEST10 | 012040 | VVSET | = 000001 | \$GET42 | 013306 | \$SVLAD | 014056 | \$OFILL | 015535 |
| TEST2 | 004462 | WATCH | 001254 | | | | | | |

. ABS. 025072 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 54784 WORDS (214 PAGES)
 DYNAMIC MEMORY AVAILABLE FOR 71 PAGES
 ,A:CZRNIA/C=A:CZRNIA.DOC,CZRNIA,SYSMAC/M

| | | | | | | | | | | | | | | |
|---------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| | 23-29 | 23-37 | | | | | | | | | | | | |
| \$GET42 | 10-1# | | | | | | | | | | | | | |
| \$GT42P | 10-1 | 10-1# | | | | | | | | | | | | |
| \$GTSWR | 18-1# | 20-2 | 20-2 | | | | | | | | | | | |
| \$HD | 4-510 | 4-510 | 4-510 | | | | | | | | | | | |
| \$HIOCT | 19-1# | 19-1* | | | | | | | | | | | | |
| \$ICNT | 6-0# | 9-92* | 12-1 | 12-1 | 12-1 | 12-1* | 12-1* | | | | | | | |
| \$INTAG | 6-0# | 18-1 | 18-1 | 18-1 | 18-1 | 18-1* | 18-1* | | | | | | | |
| \$ITEMB | 6-0# | 13-1 | 13-1 | 13-1 | 13-1 | 13-1* | 13-1* | 14-1 | | | | | | |
| \$LF | 6-0# | 13-1 | 13-1 | 15-1 | 15-1 | 18-1 | 18-1 | 18-1 | 19-1 | 19-1 | | | | |
| \$LKCSB | 7-0# | 11-11* | | | | | | | | | | | | |
| \$LKCSR | 7-0# | 11-7 | 11-12* | | | | | | | | | | | |
| \$LKS | 7-0# | 11-16 | 11-20* | | | | | | | | | | | |
| \$LLVEC | 7-0# | 11-17 | | | | | | | | | | | | |
| \$LPADR | 6-0# | 9-23* | 9-76* | 9-137* | 9-215* | 9-216* | 9-274* | 9-275* | 9-295* | 9-439* | 9-440* | 12-1 | 12-1 | 12-1 |
| \$LPERR | 12-1* | | | | | | | | | | | | | |
| \$LPVEC | 6-0# | 9-23* | 9-77* | 9-137* | 9-215* | 9-216* | 9-274* | 9-275* | 9-295* | 9-439 | 9-439* | 9-440 | 9-440* | |
| \$MAIL | 7-0# | 11-8 | | | | | | | | | | | | |
| \$MNEW | 9-23 | 9-28 | 12-1 | 13-1 | 15-1 | | | | | | | | | |
| \$MSWR | 18-1 | 18-1# | | | | | | | | | | | | |
| \$MXCNT | 18-1 | 18-1# | | | | | | | | | | | | |
| \$NULL | 12-1 | 12-1 | 12-1 | 12-1# | | | | | | | | | | |
| \$NWTST | 6-0# | 15-1 | 15-1 | 15-1 | | | | | | | | | | |
| | 9-137 | 9-137 | 9-137# | 9-137# | 9-215 | 9-215 | 9-215# | 9-215# | 9-216 | 9-216 | 9-216# | 9-216# | 9-274 | 9-274 |
| | 9-274# | 9-274# | 9-275 | 9-275 | 9-275# | 9-275# | 9-295 | 9-295 | 9-295# | 9-295# | 9-439 | 9-439 | 9-439# | 9-439# |
| | 9-440 | 9-440 | 9-440# | 9-440# | 9-444# | | | | | | | | | |
| \$OCNT | 16-1# | 16-1* | 16-1* | | | | | | | | | | | |
| \$OMODE | 16-1 | 16-1# | 16-1* | 16-1* | 16-1* | 16-1* | | | | | | | | |
| \$OVER | 12-1 | 12-1 | 12-1 | 12-1# | | | | | | | | | | |
| \$PASS | 6-0# | 9-74* | 10-1 | 10-1 | 10-1 | 10-1* | 10-1* | 12-1 | 12-1 | 12-1 | | | | |
| \$QUES | 6-0# | 13-1 | 13-1 | 15-1 | 15-1 | 18-1 | 18-1 | 18-1 | 18-1 | 19-1 | 19-1 | 19-1 | | |
| \$R2A | 20-2 | | | | | | | | | | | | | |
| \$RDCHR | 18-1# | 20-2 | 20-2 | | | | | | | | | | | |
| \$RDDEC | 20-2 | | | | | | | | | | | | | |
| \$RDLIN | 18-1# | 20-2 | 20-2 | | | | | | | | | | | |
| \$RDOCT | 19-1# | 20-2 | 20-2 | | | | | | | | | | | |
| \$RDSZ | 18-1 | 18-1# | | | | | | | | | | | | |
| \$REGO | 6-0# | | | | | | | | | | | | | |
| \$REGAD | 6-0# | | | | | | | | | | | | | |
| \$RESRE | 20-1# | 20-2 | | | | | | | | | | | | |
| \$RMADR | 7-0# | 9-68 | 9-102 | 9-109* | 9-112 | 9-134 | 23-36 | | | | | | | |
| \$RMVEC | 7-0# | | | | | | | | | | | | | |
| \$RTNAD | 10-1# | | | | | | | | | | | | | |
| \$CAVRE | 20-1# | 20-2 | 20-2 | | | | | | | | | | | |
| \$SCOPE | 9-23 | 12-1# | | | | | | | | | | | | |
| \$SETUP | 4-699 | 4-699 | 4-699 | 4-699 | 4-699 | 4-699# | 4-699# | 4-699# | 4-699# | 4-699# | 4-699# | 9-23 | 9-23 | 9-23 |
| | 9-23 | 9-23 | 9-23 | 9-23 | 9-23 | 9-23 | 9-23 | 9-23 | 9-23 | 9-28 | 9-28 | 9-28 | 10-1 | 10-1 |
| | 12-1 | 13-1 | 13-1 | 13-1 | 13-1 | 18-1 | 18-1 | 18-1 | 18-1 | 18-1 | 18-1 | 18-1 | | |
| \$STUP | 4-699 | 4-699 | 4-699 | 4-699 | 4-699 | 4-699# | 4-699# | 4-699# | 4-699# | 4-699# | 4-699# | 4-699# | 4-699# | 4-699# |
| | 4-699# | | | | | | | | | | | | | |
| \$SVLAD | 12-1 | 12-1# | | | | | | | | | | | | |
| \$SVPC | 5-5 | 5-5# | | | | | | | | | | | | |
| \$SWR | 4-506# | 4-510 | 4-511 | 4-511 | 4-511 | 4-511 | 4-511 | 4-511 | 4-511 | 4-511 | 6-0 | 6-0 | 6-0 | 9-23 |
| | 9-23 | 9-23 | 9-23 | 9-23 | 9-137 | 9-215 | 9-216 | 9-274 | 9-275 | 9-295 | 9-439 | 9-440 | 9-444 | 10-1 |
| | 10-1 | 10-1 | 10-1 | 10-1 | 11-48# | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 |
| | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 12-1 | 13-1 | 13-1 | 13-1 | 13-1 | 13-1 |

| | | | | | | | | | | | | | | |
|--------|--------|--------|-------|-------|--------|-------|--------|-------|--------|--------|--------|-------|--------|-------|
| EM4 | 8-25 | 22-6# | | | | | | | | | | | | |
| EM40 | 8-217 | 22-33# | | | | | | | | | | | | |
| EM41 | 8-224 | 22-34# | | | | | | | | | | | | |
| EM5 | 8-32 | 22-7# | | | | | | | | | | | | |
| EM6 | 8-39 | 22-8# | | | | | | | | | | | | |
| EM7 | 8-46 | 22-9# | | | | | | | | | | | | |
| EMTVEC | 4-513# | 9-23* | 9-23* | | | | | | | | | | | |
| ENTERA | 9-31 | 21-3# | | | | | | | | | | | | |
| ERR | 4-584# | | | | | | | | | | | | | |
| ERROR | 4-513# | | | | | | | | | | | | | |
| ERRVEC | 4-513# | 9-23 | 9-23* | 9-23* | 9-25* | 9-26* | 11-5* | 11-6* | 11-15* | 11-24* | 12-1 | 12-1 | 12-1* | 12-1* |
| | 12-1* | 12-1* | 12-1* | 13-1 | 13-1* | 13-1* | | | | | | | | |
| EXEC | 9-58 | 9-65# | 9-137 | 9-215 | 9-216 | 9-274 | 9-275 | 9-295 | 9-439 | 9-440 | 10-1 | | | |
| F0 | 4-561# | | | | | | | | | | | | | |
| F1 | 4-562# | | | | | | | | | | | | | |
| F2 | 4-563# | | | | | | | | | | | | | |
| F3 | 4-564# | | | | | | | | | | | | | |
| F4 | 4-565# | | | | | | | | | | | | | |
| FER | 4-593# | | | | | | | | | | | | | |
| FMT16 | 4-663# | 9-215 | 9-216 | 9-439 | 9-439 | 9-440 | 9-440 | | | | | | | |
| GNS | 5-1 | 5-1 | 9-6 | 9-28 | 10-1 | 10-1 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 |
| | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 | 20-2 |
| | 20-2 | 20-2 | | | | | | | | | | | | |
| GO | 4-560# | | | | | | | | | | | | | |
| GRV | 4-573# | | | | | | | | | | | | | |
| GTSWR | 9-28 | 20-2# | | | | | | | | | | | | |
| HCE | 4-596# | | | | | | | | | | | | | |
| HCI | 4-661# | | | | | | | | | | | | | |
| HCRC | 4-597# | | | | | | | | | | | | | |
| HT | 4-513# | 15-1 | 15-1 | | | | | | | | | | | |
| IAE | 4-599# | | | | | | | | | | | | | |
| IBSAVE | 13-1 | 13-1 | 13-1 | 13-1 | 13-1 | 13-1# | 13-1* | 13-1* | 13-1* | | | | | |
| IE | 4-519# | | | | | | | | | | | | | |
| ILF | 4-589# | | | | | | | | | | | | | |
| ILR | 4-590# | | | | | | | | | | | | | |
| IOTVEC | 4-513# | 9-23* | 9-23* | | | | | | | | | | | |
| IR | 4-542# | | | | | | | | | | | | | |
| IVC | 4-654# | | | | | | | | | | | | | |
| KYBCTL | 7-0# | 9-73* | 9-91* | 9-137 | 9-137* | 9-215 | 9-215* | 9-216 | 9-216* | 9-274 | 9-274* | 9-275 | 9-275* | 9-295 |
| | 9-295* | 9-308 | 9-439 | 9-439 | 9-439* | 9-440 | 9-440* | 10-1 | | | | | | |
| LBC | 4-652# | | | | | | | | | | | | | |
| LBT | 4-580# | | | | | | | | | | | | | |
| LF | 4-513# | 15-1 | 15-1 | 21-7 | 21-8 | 21-10 | 21-11 | 22-16 | | | | | | |
| LSC | 4-653# | | | | | | | | | | | | | |
| MAXTN | 9-85 | 24-22# | | | | | | | | | | | | |
| MCPE | 4-524# | | | | | | | | | | | | | |
| MDPE | 4-544# | | | | | | | | | | | | | |
| MOH | 4-636# | | | | | | | | | | | | | |
| MOL | 4-582# | 9-165 | 9-165 | 9-215 | 9-215 | 9-216 | 9-216 | 9-274 | 9-274 | 9-275 | 9-275 | 9-302 | 9-307 | 9-439 |
| | 9-439 | 9-439 | 9-439 | 9-439 | 9-440 | 9-440 | 9-440 | 9-440 | 9-440 | | | | | |
| MXF | 4-545# | | | | | | | | | | | | | |
| NBA | 4-638# | | | | | | | | | | | | | |
| NED | 4-548# | 9-150 | 9-150 | 9-150 | 9-150 | 9-439 | 9-439 | 9-439 | 9-440 | 9-440 | 9-440 | | | |
| NEM | 4-547# | | | | | | | | | | | | | |
| NOATA | 8-232# | 9-215 | 9-215 | 9-216 | 9-216 | 9-274 | 9-274 | 9-274 | 9-274 | 9-275 | 9-275 | 9-275 | 9-275 | 9-302 |
| | 9-302 | 9-302 | 9-302 | 9-307 | 9-307 | 9-307 | 9-307 | | | | | | | |

