```
1                       .REM    *
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16                                  IDENTIFICATION
17                                  --------------
18
19                      PRODUCT CODE:   AC-F9398-MC
20
21                      PRODUCT NAME:   CZRMSBO  RM05/3/2 DUAL PORT TEST, PT 2
22
23                      PRODUCT DATE:   APRIL 1981
24
25                      MAINTAINER:     CX DIAGNOSTIC GROUP
26
27                      AUTHOR:         MIKE LEAVITT
28
29
30
31      THE INFORMATION IN  THIS  DOCUMENT IS SUBJECT TO CHANGE  WITHOUT
32      NOTICE AND SHOULD  NOT BE CONSTRUED AS A COMMITMENT  BY  DIGITAL
33      EQUIPMENT  CORPORATION.  DIGITAL  EQUIPMENT  CORPORATION ASSUMES
34      NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.
35
36      THE SOFTWARE DESCRIBED  IN  THIS DOCUMENT  IS  FURNISHED UNDER A
37      LICENSE  AND  MAY  ONLY BE USED OR COPIED IN ACCORDANCE WITH THF
38      TERMS OF SUCH LICENSE.
39
40      DIGITAL EQUIPMENT  CORPORATION ASSUMES NO RESPONSIBILITY FOR THE
41      USE  OR  RELIABILITY OF ITS  SOFTWARE  ON  EQUIPMENT THAT IS NOT
42      SUPPLIED BY DIGITAL.
43
44           COPYRIGHT (C) 1980,1981 DIGITAL EQUIPMENT CORPORATION
```

CONTENTS

1. ABSTRACT

THE RM05/3/2 DUAL PORT LOGIC TEST PERFORMS A SERIES OF TESTS WHICH VERIFY THAT THE RM05/3/2 DUAL PORT LOGIC IS FUNCTIONING PROPERLY. ONLY THE CONTROL LOGIC IS TESTED BY THIS PROGRAM; DATA HANDLING IN THE DUAL PORT MODE IS NOT TESTED BY THIS PROGRAM.

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

THIS PROGRAM IS THE SECOND PART OF THE RM05/3/2 DUAL PORT OPTION LOGIC TEST, AND IS USED TO TEST THE "PORT SELECT" SWITCH.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
8K MEMORY
KW11-L OR KW11-P CLOCK
TERMINAL
RH11 OR RH70
1 - DISK DRIVE (RM05, RM03 OR RM02)
RM DUAL PORT TEST CABLE (P/N: 7010507-02)

2.2 PREREQUISITE PROGRAMS

A. RM05/3/2 DISKLESS TEST, PART 1 & 2

B. RM05/3/2 FUNCTIONAL TEST, PART 1, 2 & 3

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

C. RM05/3/2 DUAL PORT LOGIC TEST, PART 1

2.3 OTHER PROGRAMS

DYNAMIC OPERATION OF THE DUAL PORT OPTION IS TESTED BY THE RM05/3/2 PERFORMANCE EXERCISER PROGRAM.

3. LOADING PROCEDURES

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

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 RH11 OR RH70 ADDRESS TO BE CHANGED.

4.2 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 APPROPIATE 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.

5. 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
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 RM05/3/2 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 RM05/3/2 WILL RESPOND TO A DIFFERENT MASSBUS ADDRESS.

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

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

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

                        THE TEST CABLE CONNECTION TO THE DRIVE UNDER TEST WILL
                        DEPEND ON WHICH PROCESSOR, RH11/RH70 IS TO TEST THE DRIVE.
                        IF THE DRIVE IS TO BE TESTED BY THE PROCESSOR ON PORT A,
                        CONNECT THE MASSBUS CABLE FROM THE RH11/RH70 TO J3 OF THE
                        RM05/3/2 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
                        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 APPROPIATE 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 RM05/3/2 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 4 MINUTES PER PASS (DEPENDING ON OPERATOR INTERVENTION EFFICIENCY).

7.4 REQUIRED TESTS

IF THE PROGRAM IS BEING EXECUTED IN SINGLE TEST MOLE, 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

IF SW<09> IS SET, THE PROGRAM WILL LOOP ON A FAILING TEST UNTIL EITHER THE SWITCH IS RESET OR THE ERROR STOPS OCCURING. BECAUSE THE PROGRAM MUST RESET THE RM05/3/2 TO A KNOWN STATE BEFORE LOOPING ON THE ERROR, THE TEST FOR SW<09> IS PERFORMED AT THE END OF THE TEST - NOT AT THE POINT WHERE THE ERROR WAS DETECTED.

8. TEST DESCRIPTIONS

8.1 METHOD USED TO VERIFY THAT DRIVE IS IN NEUTRAL

THE PROGRAM DETERMINES THE THE DRIVE IS IN NEUTRAL BY CHECKING THE CONTENTS OF THE DRIVE STATUS REGISTER (RMDS) THROUGH BOTH PORTS. THE PROGRAM MASKS OUT THE PORT DEPENDENT BITS

('ATA' & 'VV') AND VERIFIES THAT CORRECT STATUS IS READ THROUGH BOTH PORTS. (THE CORRECT STATUS IS 'MOL', 'PGM', 'DPR', & 'DRY'.) IF NEITHER PORT SEES ALL ZEROS FROM RMDS, THE PROGRAM CONCLUDES THAT THE DRIVE IS IN NEUTRAL AND THAT ANY BIT DISCREPANCY BETWEEN PORTS INDICATES A FAILURE IN THE PATH FOR THAT BIT.

8.2 METHOD USED TO VERIFY THAT THE DRIVE HAS BEEN SEIZED

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

TEST 1 DRIVE ACCESS TEST

VERIFY THAT THE DRIVE CAN BE ACCESSED THROUGH BOTH PORTS

A. SELECT DRIVE, VERIFY THAT THE DRIVE IS PRESENT, THAT THE DRIVE IS A DUAL PORT RM05/3/2, 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.

TEST 2 SET 'VV' FOR PORT A

SET VOLUME VALID

A. ISSUE A DRIVE CLEAR COMMAND THROUGH PORT A.

B. ISSUE A READIN PRESET COMMAND THROUGH PORT A. VERIFY THAT THE 'VV' BIT IS SET FOR PORT A.

C. ISSUE A RELEASE COMMAND THROUGH PORT A. VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION BIT IS SET.

TEST 3 SET 'VV' FOR PORT B

SET VOLUME VALID

A. ISSUE A DRIVE CLEAR COMMAND THROUGH PORT B.

B. ISSUE A READIN PRESET COMMAND THROUGH PORT B. VERIFY THAT THE 'VV' BIT IS SET FOR PORT B.

C. ISSUE A RELEASE COMMAND THROUGH PORT B. VERIFY THAT THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION BIT IS SET.

```
343
344
345                     TEST 4  MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A
346
347                     MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT A
348
349                       A.  WRITE 0'S INTO RMDS THROUGH PORT A AND VERIFY THAT THE
350                             DRIVE HAS BEEN SEIZED.
351
352                       B.  WAIT FOR TIMEOUT TO OCCUR.  MEASURE THE DURATION OF THE TIMEOUT
353                             ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
354
355                       C.  VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS
356                             TO NEUTRAL
357
358
359                     TEST 5  MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT B
360
361                     MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT B
362
363                       A.  WRITE 0'S INTO RMDS THROUGH PORT B AND VERIFY THAT THE
364                             DRIVE HAS BEEN SEIZED.
365
366                       B.  WAIT FOR TIMEOUT TO OCCUR.  MEASURE THE DURATION OF THE TIMEOUT
367                             ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
368
369                       C.  VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS
370                             TO NEUTRAL
371
372
373                     TEST 6  TEST 'PORT SELECT' SWITCH, DRIVE CYCLED UP
374
375                     TEST THE OPERATION OF THE 'PORT SELECT' SWITCH (DRIVE CYCLED UP).
376
377                       A.  SWITCH TO PORT 'A' POSITION.  VERIFY THAT THE DRIVE IS IN
378                             NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
379                             PORTS, ARE CORRECT.
380
381                       B.  SWITCH TO PORT 'B' POSITION. VERIFY THAT THE DRIVE IS IN
382                             NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
383                             PORTS, ARE CORRECT.
384
385                       C.  RETURN THE 'PORT SELECT' SWITCH TO THE 'A/B' POSITION.  VERIFY
386                             THE DRIVE STATE.
387
388
389                     TEST 7  TEST 'PORT SELECT' SWITCH LOCKED ON PORT A
390
391                     TEST THE OPERATION OF THE 'PORT SELECT' SWITCH (DRIVE CYCLED DOWN).
392
393                       A.  CYCLE THE DRIVE DOWN.
394
395                       B.  SWITCH TO PORT 'A' POSITION.  VERIFY THAT THE DRIVE IS IN
396                             NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
397                             PORTS, ARE CORRECT.
398
399                       C.  SWITCH THE 'PORT SELECT' SWITCH TO A; CYCLE THE DRIVE UP.
```

```
400
401                          D.  WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-A IS RESET, AND
402                              THAT 'ATA-A IS SET.
403
404                          E.  ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
405                              PORT A.
406
407                          F.  VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT B AND
408                              'NED' SETS WHEN ATEMPTING TO ACCESS THE DRIVE THROUGH
409                              PORT B.  ATTEMPT TO SET PORT REQUEST BY WRITING 0'S
410                              INTO RMDS THROUGH PORT B.
411
412                          G.  ISSUE A RELEASE COMMAND THROUGH PORT A.  VERIFY THAT THE
413                              DRIVE REMAINS LOCKED ON PORT A.
414
415
416                      TEST 10 TEST 'PORT SELECT' SWITCH LOCKED ON PORT B
417
418                      TEST THE OPERATION OF THE 'PORT SELECT' SWITCH (DRIVE CYCLED DOWN).
419
420                          A.  CYCLE THE DRIVE DOWN.
421
422                          B.  SWITCH TO PORT 'B' POSITION.  VERIFY THAT THE DRIVE IS IN
423                              NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
424                              PORTS, ARE CORRECT.
425
426                          C.  SWITCH THE 'PORT SELECT' SWITCH TO B; CYCLE THE DRIVE UP.
427
428                          D.  WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-B IS RESET, AND
429                              THAT 'ATA-B IS SET.
430
431                          E.  ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
432                              PORT B.
433
434                          F.  VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT A AND
435                              'NED' SETS WHEN ATEMPTING TO ACCESS THE DRIVE THROUGH
436                              PORT A.  ATTEMPT TO SET PORT REQUEST BY WRITING 0'S
437                              INTO RMDS THROUGH PORT A.
438
439                          G.  ISSUE A RELEASE COMMAND THROUGH PORT B.  VERIFY THAT THE
440                              DRIVE REMAINS LOCKED ON PORT B.
441
442                          H.  CYCLE THE DRIVE DOWN.  CHANGE THE 'PORT SELECT' SWITCH TO
443                              A/B; CYCLE THE DRIVE UP.
444
445                          I.  VERIFY THAT BOTH PORTS CAN ACCESS THE DRIVE, THAT BOTH ATTENTION
446                              BITS ARE SET, AND THAT BOTH 'VV' BITS ARE RESET.
447
448
449                      ^
```

```
                        ;*LAST REVISION 04-APR-81

                        .TITLE  CZRMSB0 RM05/3/2 DU POR TST 2
                        ;*COPYRIGHT (C) 1981
                        ;*DIGITAL EQUIPMENT CORPORATION
                        ;*COLORADO SPGS., CO. 80919
                        ;*
                        ;*PROGRAM BY MIKE LEAVITT
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81
                        ;*
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR

                        .SBTTL  BASIC DEFINITIONS

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

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

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0      = %0            ;;GENERAL REGISTER
000001                  R1      = %1            ;;GENERAL REGISTER
000002                  R2      = %2            ;;GENERAL REGISTER
000003                  R3      = %3            ;;GENERAL REGISTER
000004                  R4      = %4            ;;GENERAL REGISTER
000005                  R5      = %5            ;;GENERAL REGISTER
000006                  R6      = %6            ;;GENERAL REGISTER
000007                  R7      = %7            ;;GENERAL REGISTER
000006                  SP      = %6            ;;STACK POINTER
000007                  PC      = %7            ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0     = 0             ;;PRIORITY LEVEL 0
000040                  PR1     = 40            ;;PRIORITY LEVEL 1
```

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

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

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

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

                                ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                000004          ERRVEC  = 4             ;;TIME OUT AND OTHER ERRORS
                000010          RESVEC  = 10            ;;RESERVED AND ILLEGAL INSTRUCTIONS
                000014          TBITVEC = 14            ;;"T" BIT
                000014          TRTVEC  = 14            ;;TRACE TRAP
                000014          BPTVEC  = 14            ;;BREAKPOINT TRAP (BPT)
                000020          IOTVEC  = 20            ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                000024          PWRVEC  = 24            ;;POWER FAIL
                000030          EMTVEC  = 30            ;;EMULATOR TRAP (EMT) **ERROR**
                000034          TRAPVEC = 34            ;;"TRAP" TRAP
                000060          TKVEC   = 60            ;;TTY KEYBOARD VECTOR
                000064          TPVEC   = 64            ;;TTY PRINTER VECTOR
                000240          PIRQVEC = 240           ;;PROGRAM INTERRUPT REQUEST VECTOR
    520
    521                         .SBTTL  RH/RM  REGISTERS
    522
    523                         ;CONTROL AND STATUS REGISTER 1 (RMCS1)
    524
    525         000100          IE      = 100                   ;INTERRUPT ENABLE (BIT #6)
    526         000200          RDY     = 200                   ;READY (BIT #7)
    527         000400          A16     = 400                   ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
    528         001000          A17     = 1000                  ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
    529         002000          PSEL    = 2000                  ;PORT SELECT (BIT #10)
    530         020000          MCPE    = 20000                 ;MASSBUSS PARITY ERROR (BIT #13)
    531         040000          TRE     = 40000                 ;TRANSFER ERROR (BIT #14)
    532         100000          SC      = 100000                ;SPECIAL CONDITION (BIT #15)
    533
    534                         ;WORD COUNT REGISTER (RMWC)
    535                         ;(EACH BIT IS CALLED BY BIT NUMBER)
    536
    537                         ;BUS ADDRESS REGISTER (RMBA)
    538                         ;(EACH BIT IS CALLED BY BIT NUMBER)
    539
    540                         ;CONTROL AND STATUS REGISTER 2 (RMCS2)
    541
    542         000001          U0      = 1                     ;UNIT SELECT (BIT #0)
    543         000002          U1      = 2                     ;UNIT SELECT (BIT #1)
    544         000004          U3      = 4                     ;UNIT SELECT (BIT #2)
    545         000010          BAI     = 10                    ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
    546         000020          PAT     = 20                    ;MASSBUS PARITY TEST (BIT #4)
    547         000040          CLR     = 40                    ;CLEAR (BIT #5)
    548         000100          IR      = 100                   ;INPUT READY (BIT #6)
    549         000200          OR      = 200                   ;OUTPUT READY (BIT #7)
    550         000400          MDPE    = 400                   ;MASS BUS PARITY ERROR (BIT #8)
    551         001000          MXF     = 1000                  ;MISSED TRANSFER ERROR (BIT #9)
    552         002000          PGE     = 2000                  ;PROGRAM ERROR (BIT #10)
    553         004000          NEM     = 4000                  ;NON EXISTENT MEMORY (BIT #11)
    554         010000          NED     = 10000                 ;NON EXISTENT DRIVE (BIT #12)
    555         020000          UPE     = 20000                 ;UNIBUS PARITY ERROR (BIT #13)
    556         040000          WCE     = 40000                 ;WRITE CHECK ERROR (BIT #14)
    557         100000          DLT     = 100000                ;DATA LATE (BIT #15)
```

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

                                .SBTTL  RM REGISTERS

                                ;CONTROL AND STATUS 1 REGISTER. (#00)

000001                          GO      = 1                     ;GO BIT (BIT #0)
000002                          F0      = 2                     ;FUNCTION CODE BIT #1
000004                          F1      = 4                     ;FUNCTION CODE BIT #2
000010                          F2      = 10                    ;FUNCTION CODE BIT #3
000020                          F3      = 20                    ;FUNCTION CODF BIT #4
000040                          F4      = 40                    ;FUNCTION CODE BIT #5
004000                          DVA     = 4000                  ;DEVICE AVAILABLE (BIT #11)

                                ;DRIVE STATUS REGISTER (RMDS) (#01)

                                ;DF5     = 1                    DRIVE FORWARD 5''/SEC.  (BIT #0)
000002                          DFF20   = 2                     ;DRIVE FORWARD 20''/SEC.  (BIT #1)
000004                          DIGB    = 4                     ;DRIVE TO INNER GUARD BAND (BIT #2)
000010                          GRV     = 10                    ;GO REVERSE (BIT #3)
000020                          DL64    = 20                    ;DIFFERENCE LESS THAN 64 (BIT #4)
000040                          DE1     = 40                    ;DIFFERENCE EQUALS 1 (BIT #5)
000100                          VV      = 100                   ;VOLUME VALID (BIT #6)
000200                          DRY     = 200                   ;DRIVE READY (BIT #7)
000400                          DPR     = 400                   ;DRIVE PRESENT (BIT #8)
001000                          PGM     = 1000                  ;PROGRAMABLE (BIT #9)
002000                          LBT     = 2000                  ;LAST SECTOR TRANSFERRED (BIT #10)
004000                          WRL     = 4000                  ;WRITE LOCK (BIT #11)
010000                          MOL     = 10000                 ;MEDIUM ON-LINE (BIT #12)
020000                          PIP     = 20000                 ;POSITIONING OPERATION IN PROGRESS (BIT #13)
040000                          ERR     = 40000                 ;COMPOSITE ERROR (BIT #14)
100000                          ATA     = 100000                ;ATTENTION ACTIVE (BIT #15)

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

000001                          ILF     = 1                     ;ILLEGAL FUNCTION (BIT #0)
000002                          ILR     = 2                     ;ILLEGAL REGISTER (BIT #1)
000004                          RMR     = 4                     ;REGISTER MODIFICATION REFUSED (BIT #2)
000010                          PAR     = 10                    ;PARITY ERROR (BIT #3)
000020                          FER     = 20                    ;FORMAT ERROR (BIT #4)
000040                          WCF     = 40                    ;WRITE CLOCK FAIL (BIT #5)
000100                          ECH     = 100                   ;ECC HARD ERROR (BIT #6)
000200                          HCE     = 200                   ;HEADER COMPARE ERROR (BIT #7)
000400                          HCRC    = 400                   ;HEADER CRC ERROR (BIT #8)
001000                          AOE     = 1000                  ;ADDRESS OVERFLOW ERROR (BIT #9)
002000                          IAE     = 2000                  ;INVALID ADDRESS ERROR (BIT #10)
004000                          WLE     = 4000                  ;WRITE LOCK ERROR (BIT #11)
010000                          DTE     = 10000                 ;DRIVE TIMING ERROR (BIT #12)
020000                          OPI     = 20000                 ;OPERATION INCOMPLETE (BIT #13)
040000                          UNS     = 40000                 ;DRIVE UNSAFE (BIT #14)
100000                          DCK     = 100000                ;DATA CHECK ERROR (BIT 15)

                                ;MAINTAINABILITY REGISTER (RMMR1)(#03)

000001                          DMD     = 1                     ;DIAGINOSTIC MODE (BIT #0)
```

```
                        ;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)

000001          AT0     = 1                     ;DEVICE 0 (BIT #0)
000002          AT1     = 2                     ;DEVICE 1 (BIT #1)
000004          AT2     = 4                     ;DEVICE 2 (BIT #2)
000010          AT3     = 10                    ;DEVICE 3 (BIT #3)
000020          AT4     = 20                    ;DEVICE 4 (BIT #4)
000040          AT5     = 40                    ;DEVICE 5 (BIT #5)
000100          AT6     = 100                   ;DEVICE 6 (BIT #6)
000200          AT7     = 200                   ;DEVICE 7 (BIT #7)

                        ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;DRIVE TYPE REGISTER (RMDT) (#06)

000001          DT00    = 1                     ;DRIVE TYPE NUMBER BIT 1
000002          DT01    = 2                     ;DRIVE TYPE NUMBER BIT 2
000004          DT02    = 4                     ;DRIVE TYPE NUMBER BIT 3
000010          DT03    = 10                    ;DRIVE TYPE NUMBER BIT 4
000020          DT04    = 20                    ;DRIVE TYPE NUMBER BIT 5
000040          DT05    = 40                    ;DRIVE TYPE NUMBER BIT 6
000100          DT06    = 100                   ;DRIVE TYPE NUMBER BIT 7
000200          DT07    = 200                   ;DRIVE TYPE NUMBER BIT 8
000400          DT08    = 400                   ;DRIVE TYPE NUMBER BIT 9
004000          DRQ     = 4000                  ;DRIVE REQUEST REQUIRED (BIT #11)
020000          MOH     = 20000                 ;MOVING HEAD (BIT #13)
040000          TAP     = 40000                 ;TAPE DRIVE (BIT #14)
100000          NBA     = 100000                ;NOT BLOCK ADDRESSED (BIT #15)

                        ;LOOK-AHEAD REGISTER (RMLA) (#07)

000100          SC0     = 100                   ;SECTOR COUNT FIELD 0 (BIT #6)
000200          SC1     = 200                   ;SECTOR COUNT FIELD 1 (BIT #7)
000400          SC2     = 400                   ;SECTOR COUNT FIELD 2 (BIT #8)
001000          SC3     = 1000                  ;SECTOR COUNT FIELD 3 (BIT #9)
002000          SC4     = 2000                  ;SECTOR COUNT FIELD 4 (BIT #10)

                        ;RM ERROR REGISTER #2 (RMER2) (#10)

000010          DPE     = 10                    ;DATA PARITY ERROR  (BIT #3)
000200          DVC     = 200                   ;DEVICE CHECK  (BIT #7)
002000          LBC     = 2000                  ;LOSS OF BIT CLOCK (BIT #10)
004000          LSC     = 4000                  ;LOSS OF SYSTEM CLOCK (BIT #11)
010000          IVC     = 10000                 ;INVALID COMMAND (BIT #12)
020000          OPE     = 20000                 ;OPERATOR ERROR  (BIT #13)
100000          SKI     = 100000                ;SEEK INCOMPLETE (BIT #14)

                        ;OFFSET REGISTER (RMOF) (#11)

000200          OFD     = 200                   ;OFFSET FORWARD (BIT #5)
002000          HCI     = 2000                  ;HEADER COMPARE INHIBIT (BIT #10)
004000          ECI     = 4000                  ;ERROR CORRECTION CODE INHIBIT (BIT #11)
010000          FMT16   = 10000                 ;FORMAT BIT (BIT #12

                        ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
```

```
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;SERIAL NUMBER REGISTER (RMSN) (#14)
                        ;(EACH IS CALLED BY BIT NUMBER)

                        ;ECC POSITION REGISTER (RMEC1) (#16)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;ECC PATTERN REGISTER (RMEC2) (#17)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        .SBTTL  DEFINITIONS OF THE RH/RM ADDRESS INDEXES

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

```
                        .SBTTL  TRAP CATCHER

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

                        .SBTTL  STARTING ADDRESS(ES)

000200  000137  001766          JMP     @#START         ;;JUMP TO STARTING ADDRESS OF PROGRAM
2
3 000204  000137  001774        JMP     @#START1        ;START AND CHANGE THE RH/RM ADDRESS
4
5                       .SBTTL  ACT11 HOOKS

                        ;;*****************************************************************
                        ;HOOKS REQUIRED BY ACT11
000210                          $SVPC=.                 ;SAVE PC
000046                          .=46
000046  013366                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052                          . 52
000052  020000                  .WORD   20000           ;;2)SET LOC.52 TO 20000
000210                          .=$SVPC                 ;; RESTORE PC
6
```

```
0                               .SBTTL  COMMON TAGS

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

        001100                          .-1100
001100                          $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                  $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102      000                 $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103      000                 $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                  $ERTTL:  WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114      000                 $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115      001                 $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                          .WORD   0
001134      000                 $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135      000                 $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                          .WORD   0
001140  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001146  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154      000                 $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155      002                 $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156      012                 $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
001157      000                 $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE' FLAG (BIT<07> 0 YES)
001160  000000                  $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                                ;;WHICH  ($REGO) WAS OBTAINED
001162  000000                  $REGO:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
001166  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001170  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
001172  000000                  $TMP3:  .WORD   0               ;;USER DEFINED
001174  000000                  $TMP4:  .WORD   0               ;;USER DEFINED
001176  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001200  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001202      207     377     377 $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001206      077                 $QUES.  .ASCII  /?/             ;;QUESTION MARK
001207      015                 $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001210      012     000         $LF:    .ASCIZ  <12>            ;;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  OPPRT:  .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 DOG 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 MESSACE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT


  001276                        $ERRTB:
   1
   2                            ;ERROR 1
   3
   4 001276  020501                     EM1     ;DRIVE IS NON-EXISTENT ('NED' BIT SET)
   5 001300  023244                     DH1
   6 001302  024614                     DT1
   7 001304  025034                     DF1
   8
   9                            ;ERROR 2
  10
  11 001306  020547                     EM2     ;WRONG DRIVE TYPE
  12 001310  023315                     DH2
  13 001312  024630                     DT2
  14 001314  025041                     DF2
  15
  16                            ;ERROR 3
  17
  18 001316  020570                     EM3     ;PORT SELECT SWITCH ON DRIVE NOT IN 'A/B'
  19 001320  023244                     DH1
  20 001322  024614                     DT1
  21 001324  025034                     DF1
  22
  23                            ;ERROR 4
  24
  25 001326  020642                     EM4     ;DRIVE NOT ON LINE
  26 001330  023315                     DH2
  27 001332  024630                     DT2
  28 001334  025041                     DF2
  29
  30                            ;ERROR 5
  31
  32 001336  020664                     EM5     ;SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME
  33 001340  023371                     DH5
  34 001342  024646                     DT5
  35 001344  025047                     DF5
  36
  37                            ;ERROR 6
  38
  39 001346  020746                     EM6     ;TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS
  40 001350  023440                     DH6
  41 001352  024662                     DT6
  42 001354  025054                     DF6
```

```
                                ;ERROR 7
                                
001356  021020                        EM7     ;TIMEOUT ONE-SHOT IS LESS THAN 500 MS
001360  023467                        DH7
001362  024672                        DT7
001364  025057                        DF7

                                ;ERROR 10

001366  021065                        EM10    ;READIN PRESET DOES NOT SET VOLUME VALID FOR THE PORT
001370  023244                        DH1
001372  024614                        DT1
001374  025034                        DF1

                                ;ERROR 11

001376  021153                        EM11    ;'GO' BIT RESET DURING UNLOAD COMMAND
001400  023244                        DH1
001402  024614                        DT1
001404  025034                        DF1

                                ;ERROR 12

001406  021220                        EM12    ;INCORRECT STATUS DURING UNLOAD COMMAND
001410  023244                        DH1
001412  024614                        DT1
001414  025034                        DF1

                                ;ERROR 13

001416  021267                        EM13    ;DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND
001420  023534                        DH13
001422  024704                        DT13
001424  025054                        DF6

                                ;ERROR 14

001426  021354                        EM14    ;ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD
001430  023611                        DH14
001432  024714                        DT14
001434  025063                        DF14

                                ;ERROR 15

001436  021436                        EM15    ;ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD'
001440  023244                        DH1
001442  024614                        DT1
001444  025034                        DF1

                                ;ERROR 16

001446  021522                        EM16    ;DRIVE NOT IN NEUTRAL AFTER UNLOAD WITH 'PORT
                                                ;SELECT' SWITCH MOVED FROM 'A/B'
001450  023534                        DH13
001452  024704                        DT13
001454  025054                        DF6
```

```
100
101                                     ;ERROR 17
102
103 001456  021641                              EM17    ;DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP
104 001460  023730                              DH17
105 001462  024732                              DT17
106 001464  025071                              DF17
107
108                                     ;ERROR 20
109
110 001466  021724                              EM20    ;DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP
111 001470  023730                              DH17
112 001472  024732                              DT17
113 001474  025071                              DF17
114
115                                     ;ERROR 21
116
117 001476  022007                              EM21    ;STATUS INCORRECT FOR PORT AFTER CYCLE UP
118 001500  023244                              DH1
119 001502  024614                              DT1
120 001504  025034                              DF1
121
122                                     ;ERROR 22
123
124 001506  022060                              EM22    ;REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT
125 001510  023244                              DH1
126 001512  024614                              DT1
127 001514  025034                              DF1
128
129                                     ;ERROR 23
130
131 001516  022156                              EM23    ;'NED' SET WHEN RMDS ACCESSED THROUGH PORT NOT SWITCHED
132 001520  023244                              DH1
133 001522  024614                              DT1
134 001524  025034                              DF1
135
136                                     ;ERROR 24
137
138 001526  022251                              EM24    ;DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/
139 001530  023747                              DH24
140 001532  024740                              DT24
141 001534  025057                              DF7
142
143                                     ;ERROR 25
144
145 001536  022331                              EM25    ;RH/RM DIDN'T RESPOND TO ADDRESSING
146 001540  024052                              DH25
147 001542  024752                              DT25
148 001544  025073                              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

                                ;ERROR 30

001566  022374                          EM30    ;DRIVE NOT SEIZED BY PORT 'N'
001570  024061                          DH30
001572  024756                          DT30
001574  025074                          DF30

                                ;ERROR 31

001576  022425                          EM31    ;WRONG STATUS SEEN BY THE SEIZING PORT
001600  023315                          DH2
001602  024630                          DT2
001604  025041                          DF2

                                ;ERROR 32

001606  022473                          EM32    ;REGISTER CONTENTS INCORRECT
001610  023315                          DH2
001612  024630                          DT2
001614  025041                          DF2

                                ;ERROR 33

001616  022523                          EM33    ;CONTROL BUS PARITY ERROR WHILE READING REGISTER
001620  023244                          DH1
001622  024614                          DT1
001624  025034                          DF1

                                ;ERROR 34

001626  022607                          EM34    ;CAN'T ACCESS DRIVE THROUGH EITHER PORT
001630  024203                          DH34
001632  024776                          DT34
001634  025103                          DF34

                                ;ERROR 35

001636  022656                          EM35    ;DRIVE NOT IN NEUTRAL AFTER RELEASE, REQUEST NOT SET
001640  024277                          DH35
001642  025010                          DT35
001644  025057                          DF7

                                ;ERROR 36

001646  022743                          EM36    ;DRIVE NOT IN NEUTRAL AFTER TIMEOUT, REQUEST NOT SET
001650  024277                          DH35
001652  025010                          DT35
001654  025057                          DF7

                                ;ERROR 37
```

ERROR POINTER TABLE

```
209
210 001656  023030                    EM37    ;REGISTER CONTENTS INCORRECT AFTER RELEASE/TIMEOUT
211 001660  024374                    DH37
212 001662  024756                    DT30
213 001664  025074                    DF30
214
215                             ;ERROR 40
216
217 001666  023111                    EM40    ;DRIVE NOT SEIZED BY PORT AFTER RELEASE WITH REQUEST SET
218 001670  024516                    DH40
219 001672  025022                    DT40
220 001674  025057                    DF7
221
222                             ;ERROR 41
223
224 001676  023166                    EM41    ;REGISTER WRONG AFTER RELEASE WITH REQUEST SET
225 001700  024061                    DH30
226 001702  024756                    DT30
227 001704  025074                    DF30
228
```

```
                                        ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS

001706  011600                BADTMO: MOV     (SP),R0         ;SAVE PC WHERE THE TIME OUT OCCURED
001710  005740                        TST     -(R0)           ;ADJUST PC -2
001712  022626                        CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
001714  104401  001722                TYPE    ,65$            ;;TYPE ASCIZ STRING
001720  000417                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
001760                          64$:
001760  010046                        MOV     R0,-(SP)        ;SETUP FOR TYPING OUT PC
001762  104402                        TYPOC
001764  000240                        NOP                     ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
                                                                ;TO STOP ON UNEXPECTED TIMEOUT.


                                .SBTTL  START OF PROGRAM

001766  005037  001270        START:  CLR     CHGADR          ;CLEAR THE 'CHANGE RH/RM ADDRESS' INDICATOR
001772  000403                        BR      START2          ;GO TO THE START

001774  012737  177777  001270  START1: MOV    #-1,CHGADR      ;SET THE 'CHANGE RH/RM ADDRESS' INDICATOR

002002  000240                  START2: NOP
002004  005227  000000                  INC     #0              ;TTY LOOP, WAIT FOR INCREMENT
002010  001375                          BNE     .-4             ;OF WORD
002012  000005                          RESET                   ;CLEAR THE WORLD


                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002014  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
002020  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
002022  022706  001140                  CMP     #SWR,R6         ;;DONE?
002026  001374                          BNE     .-6             ;;LOOP BACK IF NO
002030  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
002034  012737  013620  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
002042  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
002050  012737  014202  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
002056  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
002064  012737  017540  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
002072  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
002100  013737  013224  013216          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
002106  005037  001176                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
002112  005037  001200                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
002116  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
002124  012737  002124  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
002132  012737  002132  001110          MOV     #.,$LPERR       ;;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.
002140  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
002144  012737  002200  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
002152  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
002160  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
002166  022777  177777  176744          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
002174  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT - -1
002176  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
002200  012716  002206          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
```

```
                002204  000002                          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

           25                                   ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
           26 002226  012737  001706  000004          MOV     #BADTMO,ERRVEC  ;SETUP FOR UNEXPECTED TIMEOUT
           27 002234  012737  000300  000006          MOV     #PR6,ERRVEC+2   ;LEVEL 6
           28
           29                                   .SBTTL  TYPE PROGRAM NAME
                                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
                002242  005227  177777                  INC     #-1             ;;FIRST TIME?
                002246  001037                          BNE     67$             ;;BRANCH IF NO
                002250  022737  013366  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
                002256  001433                          BEQ     67$             ;;BRANCH IF YES
                002260  104401  002266                  TYPE    ,68$            ;;TYPE ASCIZ STRING
                002264  000430                          BR      67$             ;;GET OVER THE ASCIZ
                                                ;;68$:  .ASCIZ  <CRLF>@CZRMSB0 - RM05/3/2 DUAL PORT LOGIC TEST, PT 2@<CRLF>
                002346                          67$:
                                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
                002346  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
                002352  001006                          BNE     69$             ;;BRANCH IF YES
                002354  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
                002362  001005                          BNE     70$             ;;BRANCH IF NO
                002364  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
                002366  000403                          BR      70$
                002370  112737  000001  001134  69$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
                002376                          70$:
           30
           31 002376  004737  016044          1$:     JSR     PC,$TKINT       ;SETUP THE TTY KEYBOARD
           32 002402  004737  002774                  JSR     PC,CHANGE       ;CHECK/CHANGE THE RH/RM ADDRESS
           33 002406  104401  017626                  TYPE    ,ENTERA         ;ENTER DRIVE ADDRESS
           34 002412  104412                          RDOCT                   ;GET THE ADDRESS
           35 002414  012637  001224                  MOV     (SP)+,PORTA     ;STORE THE ADDRESS
           36 002420  023727  001224  000007          CMP     PORTA,#7        ;SEE IF ADDRESS TOO LARGE
           37 002426  101403                          BLOS    2$              ;BR IF NOT
           38 002430  104401  017655                  TYPE    ,ADRERR         ;TYPE ADDRESS ERROR MESSAGE
           39 002434  000760                          BR      1$              ;TRY AGAIN
           40 002436  013737  001224  001226  2$:     MOV     PORTA,PORTB     ;GENERATE THE PORT B ADDRESS
           41 002444  005237  001226                  INC     PORTB           ;INCREMENT THE ADDRESS
           42 002450  042737  000016  001226          BIC     #16,PORTB       ;LEAVE BIT 0
           43 002456  013746  001224                  MOV     PORTA,-(SP)     ;PUT PORT A ADDRESS ON THE STACK
           44 002462  042716  177771                  BIC     #^C6,(SP)       ;SAVE BITS 1 & 2
           45 002466  052637  001226                  BIS     (SP)+,PORTB     ;SET BITS 1 & 2 IN PORT B ADDRESS
           46 002472  104401  017700                  TYPE    ,PORTAIS        ;'PORT A ADDRESS IS '
           47 002476  013746  001224                  MOV     PORTA,-(SP)     ;;SAVE PORTA FOR TYPEOUT
                                                                              ;;TYPE PORT A ADDRESS
                002502  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
                002504     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
                002505     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
           48 002506  104401  017727                  TYPE    ,PORTBIS        ;'PORT B ADDRESS IS '
           49 002512  013746  001226                  MOV     PORTB,-(SP)     ;;SAVE PORTB FOR TYPEOUT
                                                                              ;;TYPE PORT B ADDRESS
                002516  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
                002520     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
                002521     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
           50 002522  104401  001207                  TYPE    ,$CRLF          ;ANOTHER CR-LF
```

```
 51 002526  013737  001224  001230          MOV     PORTA,PORTC     ;GENERATE ADDRESS OF DRIVE NOT TESTED
 52 002534  062737  000006  001230          ADD     #6,PORTC        ;COMPLEMENT SOME BITS
 53 002542  042737  177770  001230          BIC     #^C7,PORTC      ;SAVE ONLY LOWER BITS
 54 002550  013701  001224                  MOV     PORTA,R1        ;USE PORT A ADDRESS AS INDEX
 55 002554  116137  025130  001232          MOVB    ATABIT(R1),ASR1  ;GET ATTENTION BIT FOR DRIVE
 58 002562  005037  001256                  CLR     TIMEA           ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
    002566  005037  001260                  CLR     TIMEAP          ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
    002572  005037  001262                  CLR     TIMEB           ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
    002576  005037  001264                  CLR     TIMEBP          ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
 59 002602  004737  013406                  JSR     PC,CKCLK        ;SETUP CLOCK
 60 002606  000137  002622                  JMP     EXEC            ;CLOCK HAS BEEN STARTED
 61 002612  104401  017756                  TYPE    ,NOCLOCK        ;NO CLOCK ON SYSTEM
 62 002616  000000                  3$:     HALT                    ;FATAL ERROR
 63 002620  000776                          BR      3$              ;INTERLOCK THE HALT
 64
 65                                 ;ROUTINE TO GET THE TEST NUMBER FROM THE OPERATOR
 66
 67 002622  000005                  EXEC:   RESET                   ;CLEAR EVERYTHING
 68 002624  005037  177776                  CLR     PS              ;CLEAR THE PROCESSOR STATUS WORD
 69 002630  104401  001207                  TYPE    ,$CRLF          ;CR-LF
 70 002634  013700  001272                  MOV     $RMADR,R0       ;RH/RM ADDRESS FOR INDEXING
 71 002640  012706  001100                  MOV     #STACK,SP       ;LOAD STACK POINTER
 72 002644  004737  013406                  JSR     PC,CKCLK        ;START THE CLOCK
 73 002650  000240                          NOP                     ;RETURN IF NO CLOCK
 74 002652  004737  016044                  JSR     PC,$TKINT       ;INITIALIZE THE KEYBOARD
 75 002656  005037  001266                  CLR     KYBCTL          ;CLEAR SINGLE TEST INDICATOR
 76 002662  005037  001100                  CLR     $PASS           ;CLEAR THE PASS COUNT
 77 002666  112737  000001  001115          MOVB    #1,$ERMAX       ;SET ERROR MAX TO 1
 78 002674  012737  002674  001106          MOV     #.,$LPADR       ;INITIAL SETTING FOR LOOP ADDRESS
 79 002702  012737  002702  001110          MOV     #.,$LPERR       ;INITIAL SETTING FOR LOOP ON ERROR ADDRESS
 80 002710  104401  020023          1$:     TYPE    ,TESTNO         ;ASK FOR TEST NUMBER
 81 002714  104412                          RDOCT                   ;GET THE NUMBER
 82 002716  012601                          MOV     (SP)+,R1        ;PUT ENTRY INTO R1
 83 002720  001002                          BNE     2$              ;BR IF NOT ZERO
 84 002722  000137  003122                  JMP     TST1            ;ENTER ZERO - PERFORM ALL TESTS
 85 002726  020137  025140          2$:     CMP     R1,MAXTN        ;SEE IF NUMBER GREATER THAN MAXIMUM
 86 002732  003403                          BLE     3$              ;BR IF LESS OR EQUAL
 87 002734  104401  020043                  TYPE    ,BADNO          ;BAD ENTRY
 88 002740  000763                          BR      1$              ;TRY AGAIN
 89 002742  005301                  3$:     DEC     R1              ;DECREMENT ENTRY
 90 002744  006301                          ASL     R1              ;SHIFT IT LEFT
 91 002746  016137  025110  002772          MOV     TSTADR(R1),4$   ;GET THE TEST ADDRESS
 92 002754  005237  001266                  INC     KYBCTL          ;SET SINGLE TEST INDICATOR
 93 002760  012737  000001  001104          MOV     #1,$ICNT        ;PRESET ITERATION COUNT
 94 002766  000177  000000                  JMP     @4$             ;GO TO THE SELECTED TEST
 95 002772  000000                  4$:     .WORD   0               ;TEST ADDRESS GOES HERE
 96
 97                                 ;CHANGE THE RH/RM UNIBUS ADDRESS USED BY THE PROGRAM
 98
 99 002774  005737  001270          CHANGE: TST     CHGADR          ;CHANGE THE ADDRESS ?
100 003000  001421                          BEQ     3$              ;BR IF NOT
101 003002  005037  001270                  CLR     CHGADR          ;CLEAR THE INDICATOR
102 003006  104401  020072          1$:     TYPE    ,ADDRIS         ;TYPE OUT WHAT THE PRESENT ADDRESS IS
103 003012  013746  001272                  MOV     $RMADR,-(SP)    ;PUT THE ADDRESS ON THE STACK
104 003016  104402                          TYPOC                   ;TYPE THE ACTUAL ADDRESS
105 003020  104401  001207                  TYPE    ,$CRLF          ;CR-LF
106 003024  104401  020127                  TYPE    ,NTRH           ;ASK FOR NEW ADDRESS
```

```
107 003030  104412                        RDOCT
108 003032  005716                        TST     (SP)            ;0 OR 'CR' ENTERED ?
109 003034  001402                        BEQ     2$              ;BR IF EITHER ENTERED (NO ADDRESS CHANGE)
110 003036  011637  001272                MOV     (SP),$RMADR     ;NEW RH/RM ADDRESS
111 003042  005726                  2$:   TST     (SP)+           ;CORRECT THE STACK POINTER
112 003044  012737  003064  000004  3$:   MOV     #4$,@#4         ;LOAD TRAP ADDRESS
113 003052  013700  001272                MOV     $RMADR,R0       ;GET RH/RM ADDRESS
114 003056  005760  000002                TST     RMWC(R0)        ;RESPONDS AT THAT ADDRESS ?
115 003062  000404                        BR      5$              ;BR IF YES
116 003064                          4$:
    003064  10402                         EMT     25
117 003066  062706  000004                ADD     #4,SP           ;RESET THE STACK POINTER
118 003072  000745                        BR      1$              ;GET ADDRESS AGAIN
119 003074  012737  000006  000004  5$:   MOV     #6,@#4          ;RESTORE THE VECTOR
120 003102  000207                        RTS     PC              ;RETURN
```

```
                                .SBTTL  TESTS

                        TST1AA: MOV     $RMADR,R0       ;;RESTORE R0 AFTER END OF PASS
                                MOV     #PR5,-(SP)      ;;PUT NEW PS ON STACK
                                MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
                                RTI                     ;;POP NEW PC AND PS
                        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 RM05, RM03 OR RM02 AND 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.
                        ;*
                        ;;*****************************************************************
                        TST1:
003122  005737  001266          TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
003126  001406                  BEQ     2$              ;BR IF NOT
003130  100002                  BPL     1$              ;BR IF JUST ENTERED TEST
003132  000137  002622          JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
003136  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
003144  012737  003160  001106  2$:     MOV     #TEST1,$LPADR   ;SETUP SCOPE LOOP ADDRESS
003152  012737  003160  001110          MOV     #TEST1,$LPERR   ;SETUP ERROR LOOP ADDRESS
003160                          TEST1:
003160  112737  000001  001102          MOVB    #1,$TSTNM       ;MOVE #1 TEST NUMBER
003166  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
003172  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

003200  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR MASSBUS

                        ;VERIFY THAT DRIVE IS PRESENT THROUGH PORTS A & B

003206  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
003214  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003222  005760  000012                  TST     RMDS(R0)        ;SEE IF DRIVE (PORT A) PRESENT
003226  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
003232  016037  000010  001126          MOV     RMCS2(R0),$BDDAT  ;GET CONTENTS OF RMCS2
003240  012737  000010  001122          MOV     #RMCS2,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003246  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
003252  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
003256  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
003264  042737  167777  001164          BIC     #^CNED,$TMP0  ;SAVE SPECIFIED BITS
003272  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
003300  001414                  BEQ     64$             ;BR IF OK
003302  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
003310  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
003316  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
003324  104001                  EMT     1
003326  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
```

```
003332  000240                  64$:    NOP
003334  005737  001244                  TST     CKERR           ;WAS 'NED' SET ?
003340  001403                          BEQ     .+10            ;BR IF NOT
003342  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;ISSUE MASSBUS INIT TO CLEAR 'NED'
003350  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
003356  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003364  005760  000012                  TST     RMDS(R0)        ;SEE IF DRIVE (PORT B) PRESENT
003370  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
003374  016037  000010  001126          MOV     RMCS2(R0),$BDDAT  ;GET CONTENTS OF RMCS2
003402  012737  000010  001122          MOV     #RMCS2,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003410  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
003414  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
003420  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
003426  042737  167777  001164          BIC     #^CNED,$TMP0  ;SAVE SPECIFIED BITS
003434  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
003442  001414                          BEQ     66$             ;BR IF OK
003444  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
003452  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
003460  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
003466  104001                          EMT     1
003470  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003474  000240                  66$:    NOP
003476  005737  001244                  TST     CKERR           ;WAS 'NED' SET ?
003502  001403                          BEQ     .+10            ;BR IF NOT
003504  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;ISSUE MASSBUS INIT TO CLEAR 'NED'
34
35                                      ;CONFIRM THAT DRIVE IS AN RM05, RM03 OR RM02 AND IS DUAL PORTED
36
40 003512  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
003520  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003526  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
003532  016037  000026  001126          MOV     RMDT(R0),$BDDAT  ;GET CONTENTS OF RMDT
003540  012737  000026  001122          MOV     #RMDT,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003546  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
003552  012737  024027  001124          MOV     #024027,$GDDAT  ;WHAT REGISTER SHOULD BE
003560  022737  024024  001126          CMP     #024024,$BDDAT  ;DUAL PORT RM03 ?
003566  001413                          BEQ     68$             ;YES !!
003570  022737  024025  001126          CMP     #024025,$BDDAT  ;DUAL PORT RM02 ?
003576  001407                          BEQ     68$             ;YES !.
003600  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS THE REGISTER OK ?
003606  001403                          BEQ     68$             ;BR IF OK
003610  104002                          EMT     2
003612  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003616  000240                  68$:    NOP
003620  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
003626  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003634  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
003640  016037  000026  001126          MOV     RMDT(R0),$BDDAT  ;GET CONTENTS OF RMDT
003646  012737  000026  001122          MOV     #RMDT,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003654  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
003660  012737  024027  001124          MOV     #024027,$GDDAT  ;WHAT REGISTER SHOULD BE
003666  022737  024024  001126          CMP     #024024,$BDDAT  ;DUAL PORT RM03 ?
003674  001413                          BEQ     70$             ;YES !!
003676  022737  024025  001126          CMP     #024025,$BDDAT  ;DUAL PORT RM02 ?
003704  001407                          BEQ     70$             ;YES !!
003706  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS THE REGISTER OK ?
003714  001403                          BEQ     70$             ;BR IF OK
```

```
003716  104002                          EMT     2
003720  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003724  000240                  70$:    NOP
41
42                                      ;VERIFY THROUGH BOTH PORTS THAT THE DRIVE IS ON LINE AND IN NEUTRAL
43
48 003726  113760  001224  000010      MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
   003734  013737  001224  001234      MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
   003742  005037  001244              CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
   003746  016037  000012  001126      MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
   003754  012737  000012  001122      MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
   003762  060037  001122              ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
   003766  012737  001000  001124      MOV     #PGM,$GDDAT  ;WHAT REGISTER SHOULD BE
   003774  013737  001126  001164      MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
   004002  042737  176777  001164      BIC     #^CPGM,$TMP0  ;SAVE SPECIFIED BITS
   004010  023737  001124  001164      CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
   004016  001414                      BEQ     72$             ;BR IF OK
   004020  013737  001126  001174      MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
   004026  042737  001000  001174      BIC     #PGM,$TMP4      ;CLEAR THE MASKED BITS
   004034  053737  001174  001124      BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
   004042  104003                      EMT     3
   004044  005137  001244              COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
   004050  000240                  72$:    NOP
   004052  005037  001244              CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
   004056  016037  000012  001126      MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
   004064  012737  000012  001122      MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
   004072  060037  001122              ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
   004076  012737  010600  001124      MOV     #MOL!DPR!DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
   004104  013737  001126  001164      MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
   004112  042737  167177  001164      BIC     #^C10600,$TMP0  ;SAVE SPECIFIED BITS
   004120  023737  001124  001164      CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
   004126  001414                      BEQ     74$             ;BR IF OK
   004130  013737  001126  001174      MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
   004136  042737  010600  001174      BIC     #10600,$TMP4    ;CLEAR THE MASKED BITS
   004144  053737  001174  001124      BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
   004152  104004                      EMT     4
   004154  005137  001244              COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
   004160  000240                  74$:    NOP
   004162  113760  001226  000010      MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
   004170  013737  001226  001234      MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
   004176  005037  001244              CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
   004202  016037  000012  001126      MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
   004210  012737  000012  001122      MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
   004216  060037  001122              ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
   004222  012737  001000  001124      MOV     #PGM,$GDDAT  ;WHAT REGISTER SHOULD BE
   004230  013737  001126  001164      MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
   004236  042737  176777  001164      BIC     #^CPGM,$TMP0  ;SAVE SPECIFIED BITS
   004244  023737  001124  001164      CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
   004252  001414                      BEQ     76$             ;BR IF OK
   004254  013737  001126  001174      MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
   004262  042737  001000  001174      BIC     #PGM,$TMP4      ;CLEAR THE MASKED BITS
   004270  053737  001174  001124      BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
   004276  104003                      EMT     3
   004300  005137  001244              COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
   004304  000240                  76$:    NOP
   004306  005037  001244              CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
   004312  016037  000012  001126      MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
```

```
004320  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004326  060037  001122                  ADD     R0,$BDADR     ;ADD RH/RM BASE ADDRESS
004332  012737  010600  001124          MOV     #MOL!DPR!DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
004340  013737  001126  001164          MOV     $BDDAT,$TMP0  ;MOVE REGISTER CONTENTS TO '$TMP0'
004346  042737  167177  001164          BIC     #^C10600,$TMP0  ;SAVE SPECIFIED BITS
004354  023737  001124  001164          CMP     $GDDAT,$TMP0  ;COMPARE THE BITS
004362  001414                          BEQ     78$           ;BR IF OK
004364  013737  001126  001174          MOV     $BDDAT,$TMP4  ;COPY 'BAD DATA'
004372  042737  010600  001174          BIC     #10600,$TMP4  ;CLEAR THE MASKED BITS
004400  053737  001174  001124          BIS     $TMP4,$GDDAT  ;'OR' WITH GOOD DATA FOR TYPEOUT
004406  104004                          EMT     4
004410  005137  001244                  COM     CKERR         ;SET THE REGISTER COMPARE ERROR INDICATOR
004414  000240                  78$:    NOP
49
50                                      ;VERIFY THAT DRIVE SERIAL NUMBER SEEN THROUGH BOTH PORTS IS THE SAME
51
52 004416  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A
53 004424  016037  000030  001124          MOV     RMSN(R0),$GDDAT ;STORE THE PORT A SERIAL NUMBER
54 004432  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
55 004440  016037  000030  001126          MOV     RMSN(R0),$BDDAT ;STORE THE PORT B SERIAL NUMBER
56 004446  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;ARE THEY THE SAME ?
57 004454  001406                          BEQ     1$              ;BR IF THEY ARE
58 004456  104005                          EMT     5
59 004460  032777  100000  174452          BIT     #SW15,@SWR      ;HALT ON ERROR ?
60 004466  001001                          BNE     1$              ;BR IF SET - PROGRAM HAS ALREADY HALTED
61 004470  000000                          HALT                    ;HALT, POSSIBLE CABLE CONNECTION PROBLEM
62 004472  000004                  1$:     SCOPE                   ;LOOP ?
63
97
98
                                        ;;*****************************************************************
                                        ;*TEST 2        SET 'VV' FOR PORT A
                                        ;*
                                        ;*SET VOLUME VALID
                                        ;*
                                        ;*  A.  ISSUE A DRIVE CLEAR COMMAND THROUGH PORT A.
                                        ;*
                                        ;*  B.  ISSUE A READIN PRESET COMMAND THROUGH PORT A.  VERIFY
                                        ;*      THAT THE 'VV' BIT IS SET FOR PORT A.
                                        ;*
                                        ;*  C.  ISSUE A RELEASE COMMAND THROUGH PORT A.  VERIFY THAT
                                        ;*      THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION
                                        ;*      BIT IS SET.
                                        ;*
                                        ;;*****************************************************************
004474                                  TST2:
004474  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
004500  001406                          BEQ     2$              ;BR IF NOT
004502  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
004504  000137  002622                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
004510  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
004516  012737  004532  001106  2$:     MOV     #TEST2,$LPADR   ;SETUP SCOPE LOOP ADDRESS
004524  012737  004532  001110          MOV     #TEST2,$LPERR   ;SETUP ERROR LOOP ADDRESS
004532                                  TEST2:
004532  112737  000002  001102          MOVB    #2,$TSTNM       ;MOVE #2 TEST NUMBER
004540  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
004544  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION
```

```
004552  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
004560  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

                                ;SET VOLUME VALUE FOR PORT

004566  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
004574  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
004602  012760  010000  000032          MOV     #FMT16,RMOF(R0)  ;SET FMT16

                                ;VERIFY THAT THE DRIVE STATUS IS CORRECT

004610  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
004614  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
004622  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004630  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
004634  012737  011700  001124          MOV     #MOL!PGM!DPR.DRY!VV,$GDDAT  ;WHAT REGISTER SHOULD BE
004642  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
004650  042737  106077  001164          BIC     #^C71700,$TMP0  ;SAVE SPECIFIED BITS
004656  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
004664  001414                          BEQ     64$             ;BR IF OK
004666  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
004674  042737  071700  001174          BIC     #71700,$TMP4    ;CLEAR THE MASKED BITS
004702  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
004710  104010                          EMT     10
004712  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
004716  000240                  64$:    NOP

                                ;RELEASE THE DRIVE FROM PORT A

004720  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
004726  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
004734  012760  000013  000000          MOV     #13,RMCS1(R0)   ;ISSUE RELEASE THROUGH PORT A

                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

004742  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
004746  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
004754  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
004760  012737  011600  001124          MOV     #MOL!PGM!DPR!DRY,$GDDAT  ;COMPARISON CONSTANT
004766  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
004774  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
005002  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
005010  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005016  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
005024  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
005032  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
005040  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005046  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
005054  001006                          BNE     66$             ;BR IF NOT
005056  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
005062  001037                          BNE     68$             ;BR IF NOT
005064  104034                          EMT     34
005066  000137  005252                  JMP     70$             ;BYPASS THE REST OF THE CHECKS
005072  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
005100  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005106  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
```

```
;       SET 'VV' FOR PORT A

005114  005737  001164                TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
005120  001414                        BEQ     67$             ;BR IF ZERO
005122  013737  001224  001234        MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005130  013737  001172  001126        MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
005136  113760  001224  000010        MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
005144  005737  001166                TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
005150  001004                        BNE     68$             ;BR IF NOT
005152  012737  177777  001250  67$:  MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
005160  104036                        EMT     36
005162  013737  001170  001126  68$:  MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
005170  013737  001224  001234        MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
005176  042737  100100  001170        BIC     #ATA!VV,$TMP2   ;DON'T CHECK ATTN BIT OR VV BIT
005204  023737  001124  001170        CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
005212  001401                        BEQ     69$             ;BR IF OK FROM PORT A.
005214  104037                        EMT     37
005216  013737  001172  001126  69$:  MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
005224  013737  001226  001234        MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
005232  042737  100100  001172        BIC     #ATA!VV,$TMP3   ;DON'T CHECK ATTN BIT OR VV BIT
005240  023737  001124  001172        CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
005246  001401                        BEQ     70$             ;BR IF OK
005250  104037                        EMT     37
005252  000240                70$:    NOP
005254  000004                        SCOPE                   ;LOOP ?
99
                                ;;*****************************************************************
                                ;*TEST 3         SET 'VV' FOR PORT B
                                ;*
                                ;*SET VOLUME VALID
                                ;*
                                ;*  A.  ISSUE A DRIVE CLEAR COMMAND THROUGH PORT B.
                                ;*
                                ;*  B.  ISSUE A READIN PRESET COMMAND THROUGH PORT B.  VERIFY
                                ;*      THAT THE 'VV' BIT IS SET FOR PORT B.
                                ;*
                                ;*  C.  ISSUE A RELEASE COMMAND THROUGH PORT B.  VERIFY THAT
                                ;*      THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION
                                ;*      BIT IS SET.
                                ;*
                                ;;*****************************************************************
005256                          TST3:
005256  005737  001266                TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
005262  001406                        BEQ     2$              ;BR IF NOT
005264  100002                        BPL     1$              ;BR IF JUST ENTERED TEST
005266  000137  002622                JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
005272  012737  177777  001266  1$:   MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
005300  012737  005314  001106  2$:   MOV     #TEST3,$LPADR   ;SETUP SCOPE LOOP ADDRESS
005306  012737  005314  001110        MOV     #TEST3,$LPERR   ;SETUP ERROR LOOP ADDRESS
005314                          TEST3:
005314  112737  000003  001102        MOVB    #3,$TSTNM       ;MOVE #3 TEST NUMBER
005322  012706  001100                MOV     #STACK,SP       ;SETUP THE STACK POINTER
005326  012737  000001  001176        MOV     #1,$TIMES       ;;DO 1 ITERATION

005334  113760  001226  000010        MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
005342  013737  001226  001234        MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

                                ;SET VOLUME VALUE FOR PORT
```

```
005350  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
005356  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
005364  012760  010000  000032          MOV     #FMT16,RMOF(R0)  ;SET FMT16

                                ;VERIFY THAT THE DRIVE STATUS IS CORRECT

005372  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
005376  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
005404  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
005412  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
005416  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;WHAT REGISTER SHOULD BE
005424  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
005432  042737  106077  001164          BIC     #^C71700,$TMP0  ;SAVE SPECIFIED BITS
005440  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
005446  001414                          BEQ     64$             ;BR IF OK
005450  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
005456  042737  071700  001174          BIC     #71700,$TMP4    ;CLEAR THE MASKED BITS
005464  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
005472  104010                          EMT     10
005474  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
005500  000240                  64$:    NOP

                                ;RELEASE THE DRIVE FROM PORT B

005502  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
005510  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
005516  012760  000013  000000          MOV     #13,RMCS1(R0)   ;ISSUE RELEASE THROUGH PORT B

                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

005524  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
005530  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
005536  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
005542  012737  011600  001124          MOV     #MOL!PGM!DPR!DRY,$GDDAT  ;COMPARISON CONSTANT
005550  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
005556  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
005564  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
005572  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005600  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
005606  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
005614  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
005622  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005630  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
005636  001006                          BNE     66$             ;BR IF NOT
005640  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
005644  001037                          BNE     68$             ;BR IF NOT
005646  104034                          EMT     34
005650  000137  006034                  JMP     70$             ;BYPASS THE REST OF THE CHECKS
005654  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
005662  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005670  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
005676  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
005702  001414                          BEQ     67$             ;BR IF ZERO
005704  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005712  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
005720  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
005726  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
```

```
005732  001004                          BNE     68$             ;BR IF NOT
005734  012737  177777  001250  67$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
005742  104036                          EMT     36
005744  013737  001170  001126  68$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
005752  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
005760  042737  100100  001170          BIC     #ATA!VV,$TMP2   ;DON'T CHECK ATTN BIT OR VV BIT
005766  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
005774  001401                          BEQ     69$             ;BR IF OK FROM PORT A.
005776  104037                          EMT     37
006000  013737  001172  001126  69$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
006006  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
006014  042737  100100  001172          BIC     #ATA!VV,$TMP3   ;DON'T CHECK ATTN BIT OR VV BIT
006022  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
006030  001401                          BEQ     70$             ;BR IF OK
006032  104037                          EMT     37
006034  000240                  70$:    NOP
006036  000004                          SCOPE                   ;LOOP ?
100
105
157

                                ;;*****************************************************************
                                ;*TEST 4         MEASURE THE TIMFOUT 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
                                ;*
                                ;;*****************************************************************
006040                          TST4:
006040  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
006044  001406                          BEQ     2$              ;BR IF NOT
006046  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
006050  000137  002622                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
006054  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
006062  012737  006076  001106  2$:     MOV     #TEST4,$LPADR   ;SETUP SCOPE LOOP ADDRESS
006070  012737  006076  001110          MOV     #TEST4,$LPERR   ;SETUP ERROR LOOP ADDRESS
006076                          TEST4:
006076  112737  000004  001102          MOVB    #4,$TSTNM       ;MOVE #4 TEST NUMBER
006104  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
006110  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

006116  005037  001256                  CLR     TIMEA           ;CLEAR THE TIMEOUT VALUE STORAGE LOCATION
006122  005037  001260                  CLR     TIMEAP  ;CLEAR THE + 25% TOLERANCE LOCATION

                                ;START THE TIMER

006126  005037  001252                  CLR     TIME            ;CLEAR THE ELAPSED  IME COUNTER
006132  012737  003720  001254          MOV     #2000.,WATCH    ;SET WATCH TO 2000. MS
```

```
                                ;SEIZE THE DRIVE THROUGH PORT A

006140  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A
006146  013737  001224  001236          MOV     PORTA,SEIZPT  ;STORE SEIZING PORT'S ADDRESS
006 54  005060  000012                  CLR     RMDS(R0)        ;WRITE RMDS
006160  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
006166  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006174  013737  001226  001240          MOV     PORTB,OPPRT     ;'OPPOSITE' PORT ADDRESS
006202  016037  000012  001126          MOV     RMDS(R0),$BDDAT ;SEE IF DRIVE SEIZED BY PORT A
006210  010037  001122                  MOV     R0,$BDADR       ;RH/RM BASE ADDRESS
006214  062737  000012  001122          ADD     #RMDS,$BDADR    ;GENERATE BAD REGISTER ADDRESS
006222  005037  001124                  CLR     $GDDAT          ;REGISTER SHOULD BE ZERO
006226  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS THE REGISTER ZERO
006234  001403                          BEQ     64$             ;BR IF IT IS
006236  104030                          EMT     30
006240  000137  006754                  JMP     4$              ;BYPASS REST OF THE SUBTEST
006244                          64$:
006244  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
006252  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006260  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;SEE IF SEIZING PORT SEES CORRECT STATUS
006266  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;EXPECTED STATUS
006274  013737  001124  001166          MOV     $GDDAT,$TMP1  ;USE GOOD DATA AS A MASK
006302  005137  001166                  COM     $TMP1           ;COMPLEMENT THE EXPECTED STATUS
006306  013737  001126  001164          MOV     $BDDAT,$TMP0    ;SAVE THE ACTUAL STATUS
006314  043737  001166  001164          BIC     $TMP1,$TMP0     ;CLEAR UNWANTED BITS
006322  023737  001124  001164          CMP     $GDDAT,$TMP0    ;ARE THE EXPECTED STATUS BITS SET ?
006330  001401                          BEQ     65$             ;BR IF THEY ARE
006332  104031                          EMT     31
006334  000240                  65$:    NOP

                                ;WAIT FOR PORT A TO TIMEOUT

006336  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
006344  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006352  005760  000012          1$:     TST     RMDS(R0)        ;WAIT FOR THE DRIVE TO TIMEOUT
006356  001006                          BNE     2$              ;BR WHEN TIMEOUT OCCURS
006360  005737  001254                  TST     WATCH           ;CHECK WATCH
006364  001372                          BNE     1$              ;BR IF NOT ZERO
006366  104006                          EMT     6
006370  000137  006426                  JMP     3$              ;BYPASS THE REST OF THE TEST
006374  013737  001252  001256  2$:     MOV     TIME,TIMEA      ;SAVE THE ELAPSED TIME FOR PORT A
006402  004537  013572                  JSR     R5,TOLER        ;CALCULATE THE TOLERANCE
006406  001256                          .WORD   TIMEA           ;TIMEOUT VALUE FOR PORT A
006410  012637  001260                  MOV     (SP)+,TIMEAP    ;+25% TOLERANCE

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

006414  023727  001256  000764          CMP     TIMEA,#500.     ;IS TIMEOUT VALUE AT LEAST 500 MS ?
006422  103001                          BHIS    3$              ;BR IF IT IS
006424  104007                          EMT     7

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

006426                          3$:
                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

006426  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
```

T4 MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A

```
006432  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
006440  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
006444  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;COMPARISON CONSTANT
006452  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
006460  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
006466  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
006474  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
006502  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
006510  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS RFGISTER FROM PORT B.
006516  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
006524  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
006532  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
006540  001006                          BNE     66$             ;BR IF NOT
006542  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
006546  001045                          BNE     68$             ;BR IF NOT
006550  104034                          EMT     34
006552  000137  006752                  JMP     70$             ;BYPASS THE REST OF THE CHECKS
006556  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
006564  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
006572  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
006600  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
006604  001414                          BEQ     67$             ;BR IF ZERO
006606  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
006614  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
006622  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
006630  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
006634  001012                          BNE     68$             ;BR IF NOT
006636  012737  177777  001250  67$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
006644  012760  000011  000000          MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
006652  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
006660  104035                          EMT     35
006662  013737  001170  001126  68$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
006670  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
006676  042737  100000  001170          BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
006704  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
006712  001401                          BEQ     69$             ;BR IF OK FROM PORT A.
006714  104037                          EMT     37
006716  013737  001172  001126  69$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
006724  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
006732  042737  100000  001172          BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
006740  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
006746  001401                          BEQ     70$             ;BR IF OK
006750  104037                          EMT     37
006752  000240                  70$:    NOP
006754  000004                  4$:     SCOPE                   ;LOOP ?
158
```

```
;;*****************************************************************
;*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
                                ;*
                                ;;*****************************************************************
006756                          TST5:
006756  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
006762  001406                          BEQ     2$              ;BR IF NOT
006764  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
006766  000137  002622                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
006772  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
007000  012737  007014  001106  2$:     MOV     #TEST5,$LPADR   ;SETUP SCOPE LOOP ADDRESS
007006  012737  007014  001110          MOV     #TEST5,$LPERR   ;SETUP ERROR LOOP ADDRESS
007014                          TEST5:
007014  112737  000005  001102          MOVB    #5,$TSTNM       ;MOVE #5 TEST NUMBER
007022  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
007026  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

007034  005037  001262                  CLR     TIMEB           ;CLEAR THE TIMEOUT VALUE STORAGE LOCATION
007040  005037  001264                  CLR     TIMEBP  ;CLEAR THE + 25% TOLERANCE LOCATION

                                ;START THE TIMER

007044  005037  001252                  CLR     TIME            ;CLEAR THE ELAPSED TIME COUNTER
007050  012737  003720  001254          MOV     #2000.,WATCH    ;SET WATCH TO 2000. MS

                                ;SEIZE THE DRIVE THROUGH PORT B

007056  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
007064  013737  001226  001236          MOV     PORTB,SEIZPT  ;STORE SEIZING PORT'S ADDRESS
007072  005060  000012                  CLR     RMDS(R0)        ;WRITE RMDS
007076  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
007104  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007112  013737  001224  001240          MOV     PORTA,OPPRT     ;'OPPOSITE' PORT ADDRESS
007120  016037  000012  001126          MOV     RMDS(R0),$BDDAT ;SEE IF DRIVE SEIZED BY PORT B
007126  010037  001122                  MOV     R0,$BDADR       ;RH/RM BASE ADDRESS
007132  062737  000012  001122          ADD     #RMDS,$BDADR    ;GENERATE BAD REGISTER ADDRESS
007140  005037  001124                  CLR     $GDDAT          ;REGISTER SHOULD BE ZERO
007144  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS THE REGISTER ZERO
007152  001403                          BEQ     64$             ;BR IF IT IS
007154  104030                          EMT     30
007156  000137  007672                  JMP     4$              ;BYPASS REST OF THF SUBTEST
007162                          64$:
007162  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
007170  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007176  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;SEE IF SEIZING PORT SEES CORRECT STATUS
007204  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;EXPECTED STATUS
007212  013737  001124  001166          MOV     $GDDAT,$TMP1  ;USE GOOD DATA AS A MASK
007220  005137  001166                  COM     $TMP1           ;COMPLEMENT THE EXPECTED STATUS
007224  013737  001126  001164          MOV     $BDDAT,$TMP0    ;SAVE THE ACTUAL STATUS
007232  043737  001166  001164          BIC     $TMP1,$TMP0     ;CLEAR UNWANTED BITS
007240  023737  001124  001164          CMP     $GDDAT,$TMP0    ;ARE THE EXPECTED STATUS BITS SET ?
007246  001401                          BEQ     65$             ;BR IF THEY ARE
007250  104031                          EMT     31
007252  000240                  65$:    NOP

                                ;WAIT FOR PORT B TO TIMEOUT
```

```
007254  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
007262  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007270  005760  000012        1$:       TST     RMDS(R0)         ;WAIT FOR THE DRIVE TO TIMEOUT
007274  001006                          BNE     2$               ;BR WHEN TIMEOUT OCCURS
007276  005737  001254                  TST     WATCH            ;CHECK WATCH
007302  001372                          BNE     1$               ;BR IF NOT ZERO
007304  104006                          EMT     6
007306  000137  007344                  JMP     3$               ;BYPASS THE REST OF THE TEST
007312  013737  001252  001262  2$:     MOV     TIME,TIMEB       ;SAVE THE ELAPSED TIME FOR PORT B
007320  004537  013572                  JSR     R5,TOLER         ;CALCULATE THE TOLERANCE
007324  001262                          .WORD   TIMEB            ;TIMEOUT VALUE FOR PORT B
007326  012637  001264                  MOV     (SP)+,TIMEBP     ;+25% TOLERANCE

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

007332  023727  001262  000764          CMP     TIMEB,#500.      ;IS TIMEOUT VALUE AT LEAST 500 MS ?
007340  103001                          BHIS    3$               ;BR IF IT IS
007342  104007                          EMT     7

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

007344                          3$:
                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

007344  005037  001250                  CLR     RELERR           ;CLEAR THE 'RELEASE ERROR ' INDICATOR
007350  012737  000012  001122          MOV     #RMDS,$BDADR     ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
007356  060037  001122                  ADD     R0,$BDADR        ;ADD THE I/O BASE ADDRESS
007362  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;COMPARISON CONSTANT
007370  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
007376  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
007404  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
007412  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
007420  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
007426  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
007434  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
007442  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
007450  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
007456  001006                          BNE     66$             ;BR IF NOT
007460  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
007464  001045                          BNE     68$             ;BR IF NOT
007466  104034                          EMT     34
007470  000137  007670                  JMP     70$             ;BYPASS THE REST OF THE CHECKS
007474  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
007502  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
007510  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
007516  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
007522  001414                          BEQ     67$             ;BR IF ZERO
007524  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
007532  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
007540  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
007546  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
007552  001012                          BNE     68$             ;BR IF NOT
007554  012737  177777  001250  67$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
007562  012760  000011  000000          MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
007570  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
007576  104035                          EMT     35
```

```
;5      MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT B

 007600  013737  001170  001126  68$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
 007606  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
 007614  042737  100000  001170          BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
 007622  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
 007630  001401                          BEQ     69$             ;BR IF OK FROM PORT A.
 007632  104037                          EMT     37
 007634  013737  001172  001126  69$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
 007642  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
 007650  042737  100000  001172          BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
 007656  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
 007664  001401                          BEQ     70$             ;BR IF OK
 007666  104037                          EMT     37
 007670  000240                  70$:    NOP
 007672  000004                  4$:     SCOPE                   ;LOOP ?
159
160
177
178                                     ;;******************************************************** ********
                                        ;*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.
                                        ;*
                                        ;;*****************************************************************
 007674                                 TST6:
 007674  005737  001266                          TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
 007700  001406                          BEQ     2$              ;BR IF NOT
 007702  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
 007704  000137  002622                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
 007710  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
 007716  012737  007732  001106  2$:     MOV     #TEST6,$LPADR   ;SETUP SCOPE LOOP ADDRESS
 007724  012737  007732  001110          MOV     #TEST6,$LPERR   ;SETUP ERROR LOOP ADDRESS
 007732                                 TEST6:
 007732  112737  000006  001102          MOVB    #6,$TSTNM       ;MOVE #6 TEST NUMBER
 007740  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
 007744  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION
179
180                                     ;CLEAR ATTENTION BITS FOR BOTH PORTS

 007752  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT #A
 007760  005060  000012                  CLR     RMDS(R0)        ;SEIZE THE DRIVE
 007764  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE DRIVE CLEAR
 007772  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
 010000  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT #B
 010006  005060  000012                  CLR     RMDS(R0)        ;SEIZE THE DRIVE THROUGH PORT 'B'
 010012  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE DRIVE CLEAR
 010020  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
```

```
                                                TYPE    ,SWTCHA         ;SWITCH TO 'A'
                                                TYPE    ,CONTUE         ;PRESS 'CONTINUE'
                                                HALT
                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

010040  005037  001250                          CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
010044  012737  000012  001122                  MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
010052  060037  001122                          ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
010056  012737  011700  001124                  MOV     #MOL!PGM!DPR.DRY!VV,$GDDAT  ;COMPARISON CONSTANT
010064  113760  001224  000010                  MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010072  016037  000012  001170                  MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
010100  013737  001170  001164                  MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
010106  042737  100100  001164                  BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010114  113760  001226  000010                  MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010122  016037  000012  001172                  MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
010130  013737  001172  001166                  MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
010136  042737  100100  001166                  BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010144  023737  001164  001166                  CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
010152  001006                                  BNE     64$             ;BR IF NOT
010154  005737  001164                          TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
010160  001045                                  BNE     66$             ;BR IF NOT
010162  104034                                  EMT     34
010164  000137  010364                          JMP     68$             ;BYPASS THE REST OF THE CHECKS
010170  013737  001170  001126  64$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
010176  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010204  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010212  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
010216  001414                          BEQ     65$             ;BR IF ZERO
010220  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010226  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
010234  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010242  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
010246  001012                          BNE     66$             ;BR IF NOT
010250  012737  177777  001250  65$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
010256  012760  000011  000000          MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
010264  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
010272  104017                          EMT     17
010274  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
010302  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
010310  042737  100000  001170          BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
010316  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
010324  001401                          BEQ     67$             ;BR IF OK FROM PORT A.
010326  104037                          EMT     37
010330  013737  001172  001126  67$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
010336  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
010344  042737  100000  001172          BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
010352  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
010360  001401                          BEQ     68$             ;BR IF OK
010362  104037                          EMT     37
010364  000240                  68$:    NOP
                                                TYPE    ,SWTCHB         ;SWITCH TO 'B'
                                                TYPE    ,CONTUE         ;PRESS 'CONTINUE'
                                                HALT
                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL
```

TEST 'PORT SELECT' SWITCH, DRIVE CYCLED UP

```
010400  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
010404  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
010412  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
010416  012737  011700  001124          MOV     #MOL!PGM!DPR.DRY!VV,$GDDAT  ;COMPARISON CONSTANT
010424  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010432  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
010440  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
010446  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010454  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010462  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
010470  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
010476  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010504  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
010512  001006                          BNE     69$             ;BR IF NOT
010514  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
010520  001045                          BNE     71$             ;BR IF NOT
010522  104034                          EMT     34
010524  000137  010724                  JMP     73$             ;BYPASS THE REST OF THE CHECKS
010530  013737  001170  001126  69$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
010536  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010544  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010552  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
010556  001414                          BEQ     70$             ;BR IF ZERO
010560  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010566  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
010574  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010602  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
010606  001012                          BNE     71$             ;BR IF NOT
010610  012737  177777  001250  70$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
010616  012760  000011  000000          MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
010624  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
010632  104020                          EMT     20
010634  013737  001170  001126  71$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
010642  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
010650  042737  100000  001170          BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
010656  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
010664  001401                          BEQ     72$             ;BR IF OK FROM PORT A.
010666  104037                          EMT     37
010670  013737  001172  001126  72$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
010676  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
010704  042737  100000  001172          BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
010712  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
010720  001401                          BEQ     73$             ;BR IF OK
010722  104037                          EMT     37
010724  000240                  73$:    NOP
191 010726  005737  001266              TST     KYBCTL          ;SINGLE TEST MODE ?
192 010732  001402                      BEQ     1$              ;BR IF NOT
193 010734  104401  020156              TYPE    ,SWTCHN         ;RETURN SWITCH TO 'A/B'
194 010740  000004                  1$:     SCOPE               ;LOOP ?
195
321
322
                                ;;*****************************************************************
                                ;*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 ATEMPTING 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.
                        ;*
                        ;;*****************************************************************
010742                  TST7:
010742  005737  001266          TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
010746  001406                  BEQ     2$              ;BR IF NO
010750  100002                  BPL     1$              ;BR IF JUST ENTERED TEST
010752  000137  002622          JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
010756  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
010764  012737  011000  001106  2$:     MOV     #TEST7,$LPADR   ;SETUP SCOPE LOOP ADDRESS
010772  012737  011000  001110          MOV     #TEST7,$LPERR   ;SETUP ERROR LOOP ADDRESS
011000                  TEST7:
011000  112737  000007  001102          MOVB    #7,$TSTNM       ;MOVE #7 TEST NUMBER
011006  012706  001100          MOV     #STACK,SP       ;SETUP THE STACK POINTER
011012  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

011020  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
011026  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011034  104401  020367          TYPE    ,CYCLED         ;'CYCLE DOWN THE DRIVE'
011040  104401  020224          TYPE    ,SWTCHA         ;SWITCH TO 'A'
011044  104401  020407          TYPE    ,CYCLEU         ;'CYCLE UP THE DRIVE'

011050  032760  010000  000012  1$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' RESET ?
011056  001374                  BNE     1$              ;BR IF NO (DRIVE NOT CYCLED DOWN)
011060  032760  010000  000012  2$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' SET ?
011066  001774                  BEQ     2$              ;BR IF NO (DRIVE NOT CYCLED UP)

                        ;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT A

011070  005037  001244          CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011074  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
011102  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011110  060037  001122          ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011114  012737  010600  001124          MOV     #ATA!MOL.DPR!DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
011122  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011130  042737  066077  001164          BIC     #^C111700,$TMP0  ;SAVE SPECIFIED BITS
011136  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
```

```
011144  001414                          BEQ     64$             ;BR IF OK
011146  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011154  042737  111700  001174          BIC     #111700,$TMP4   ;CLEAR THE MASKED BITS
011162  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011170  104021                          EMT     21
011172  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011176  000240                  64$:    NOP

                                ;SET VOLUME VALID FOR PORT A

011200  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
011206  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
011214  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.

011222  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
011230  013737  001226  001234          MOV     PORTB,PTNBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011236  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011242  016037  000012  001126          MOV     RMDS(R0),$BDDAT ;GET CONTENTS OF RMDS
011250  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011256  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011262  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
011266  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011274  042737  000077  001164          BIC     #^C177700,$TMP0 ;SAVE SPECIFIED BITS
011302  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
011310  001414                          BEQ     66$             ;BR IF OK
011312  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011320  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
011326  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011334  104022                          EMT     22
011336  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011342  000240                  66$:    NOP
011344  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011350  016037  000010  001126          MOV     RMCS2(R0),$BDDAT ;GET CONTENTS OF RMCS2
011356  012737  000010  001122          MOV     #RMCS2,$BDADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011364  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011370  012737  010000  001124          MOV     #NED,$GDDAT     ;WHAT REGISTER SHOULD BE
011376  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011404  042737  167777  001164          BIC     #^CNED,$TMP0    ;SAVE SPECIFIED BITS
011412  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
011420  001414                          BEQ     68$             ;BR IF OK
011422  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011430  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
011436  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011444  104023                          EMT     23
011446  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011452  000240                  68$:    NOP
011454  005060  000012                  CLR     RMDS(R0)        ;TRY TO SET REQUEST BY WRITING THROUGH
                                                                ;THE LOCKED OUT PORT (PORT 'B')

                                ;VERIFY THAT DRIVE STAYS LOCKED ON PORT A

011460  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A
011466  013737  001224  001234          MOV     PORTA,PTNBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011474  012760  000013  000012          MOV     #13,RMDS(R0)    ;ISSUE A RELEASE THROUGH PORT A
```

TEST 'PORT SELECT' SWITCH ON PORT A

```
011502  013737  001224  001236          MOV     PORTA,SEIZPT    ;ADDRESS OF 'LOCKED ON' PORT
011510  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
011516  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011524  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011530  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
011536  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011544  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011550  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
011554  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011562  042737  000077  001164          BIC     #^C177700,$TMP0  ;SAVE SPECIFIED BITS
011570  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
011576  001414                          BEQ     70$             ;BR IF OK
011600  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011606  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
011614  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011622  104024                          EMT     24
011624  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011630  000240                  70$:    NOP

                                ;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST

011632  105737  001103                  TSTB    $ERFLG          ;DID AN ERROR OCCUR
011636  001412                          BEQ     3$              ;BR IF NOT
011640  032777  001000  167272          BIT     #SW09,@SWR      ;SEE IF LOOP ON ERROR (SWR9 - 1)
011646  001406                          BEQ     3$              ;BR IF NOT
011650  105037  001103                  CLRB    $ERFLG          ;CLEAR THE ERROR FLAG
011654  005037  001176                  CLR     $TIMES          ;CLEAR THE MAX ITERATION COUNT
011660  000177  167224                  JMP     @$LPERR         ;GO TO THE LOOP ADDRESS
011664  005737  001266          3$:     TST     KYBCTL          ;IN SINGLE TEST MODE ?
011670  001460                          BEQ     6$              ;BR IF NOT
011672  032777  040000  167240          BIT     #SW14,@SWR      ;LOOP ON TEST ?
011700  001054                          BNE     6$              ;BR IF LOOPING
011702  104401  020367                  TYPE    ,CYCLED         ;TYPE 'CYCLE DOWN'
011706  104401  020156                  TYPE    ,SWTCHN         ;'SWITCH TO A/B'
011712  104401  020407                  TYPE    ,CYCLEU         ;'CYCLE THE DRIVE UP'
011716  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
011724  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011732  032760  010000  000012  4$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' RESET ?
011740  001374                          BNE     4$              ;BR IF NO (DRIVE NOT CYCLED DOWN)
011742  032760  010000  000012  5$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' SET ?
011750  001774                          BEQ     5$              ;BR IF NO (DRIVE NOT CYCLED UP)

                                ;SET VOLUME VALID FOR BOTH PORTS

011752  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR THROUGH PORT A
011760  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET THROUGH PORT A
011766  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE PORT A
011774  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
012002  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012010  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET THROUGH PORT B
012016  012760  010000  000032          MOV     #FMT16,RMOF(R0) ;SET FMT16
012024  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE PORT B
012032  012737  011610  001254  6$:     MOV     #5000.,WATCH    ;SPINDLE MOTOR 'COOL DOWN' DELAY
012040  005737  001254          7$:     TST     WATCH           ;FINSIHED ?
012044  001375                          BNE     7$              ;BR IF NOT
012046  000004                          SCOPE                   ;LOOP ?
012050  000400                          BR      TST10           ;;GO TO NEXT TEST
```

```
323

                        ;*****************************************************************
                        ;*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 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 ATEMPTINC 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.
                        ;*
                        ;*  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.
                        ;*
                        ;*****************************************************************
012052                  TST10:
012052  005737  001266          TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
012056  001406                  BEQ     2$              ;BR IF NOT
012060  100002                  BPL     1$              ;BR IF JUST ENTERED TEST
012062  000137  002622          JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
012066  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
012074  012737  012110  001106  2$:     MOV     #TEST10,$LPADR  ;SETUP SCOPE LOOP ADDRESS
012102  012737  012110  001110          MOV     #TEST10,$LPERR  ;SETUP ERROR LOOP ADDRESS
012110                  TEST10:
012110  112737  000010  001102          MOVB    #10,$TSTNM      ;MOVE #10 TEST NUMBER
012116  012706  001100          MOV     #STACK,SP       ;SETUP THE STACK POINTER
012122  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

012130  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
012136  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012144  104401  020367          TYPE    ,CYCLED         ;'CYCLE DOWN THE DRIVE'
012150  104401  020261          TYPE    ,SWTCHB         ;SWITCH TO 'B'
012154  104401  020407          TYPE    ,CYCLEU         ;'CYCLE UP THE DRIVE'

012160  032760  010000  000012  1$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' RESET ?
012166  001374                  BNE     1$              ;BR IF NO (DRIVE NOT CYCLED DOWN)
012170  032760  010000  000012  2$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' SET ?
```

```
012176  001774                          BEQ     2$              ;BR IF NO (DRIVE NOT CYCLED UP)

                                ;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT B

012200  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012204  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
012212  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012220  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012224  012737  110600  001124          MOV     #ATA!MOL!DPR:DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
012232  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012240  042737  066077  001164          BIC     #^C111700,$TMP0  ;SAVE SPECIFIED BITS
012246  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012254  001414                          BEQ     64$             ;BR IF OK
012256  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012264  042737  111700  001174          BIC     #111700,$TMP4   ;CLEAR THE MASKED BITS
012272  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012300  104021                          EMT     21
012302  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012306  000240                  64$:    NOP

                                ;SET VOLUME VALID FOR PORT B

012310  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
012316  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
012324  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.

012332  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
012340  013737  001224  001234          MOV     PORTA,PTNBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012346  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012352  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
012360  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012366  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012372  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
012376  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012404  042737  000077  001164          BIC     #^C177700,$TMP0  ;SAVE SPECIFIED BITS
012412  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012420  001414                          BEQ     66$             ;BR IF OK
012422  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012430  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
012436  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012444  104022                          EMT     22
012446  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012452  000240                  66$:    NOP
012454  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012460  016037  000010  001126          MOV     RMCS2(R0),$BDDAT  ;GET CONTENTS OF RMCS2
012466  012737  000010  001122          MOV     #RMCS2,$BDADR   ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012474  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012500  012737  010000  001124          MOV     #NED,$GDDAT     ;WHAT REGISTER SHOULD BE
012506  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012514  042737  167777  001164          BIC     #^CNED,$TMP0    ;SAVE SPECIFIED BITS
012522  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012530  001414                          BEQ     68$             ;BR IF OK
012532  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012540  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
```

```
012546  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012554  104023                          EMT     23
012556  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012562  000240                  68$:    NOP
012564  005060  000012                  CLR     RMDS(R0)        ;TRY TO SET REQUEST BY WRITING THROUGH
                                                                ;THE LOCKED OUT PORT (PORT 'A')

                                ;VERIFY THAT DRIVE STAYS LOCKED ON PORT B

012570  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
012576  013737  001226  001234          MOV     PORTB,PTNBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012604  012760  000013  000012          MOV     #13,RMDS(R0)    ;ISSUE A RELEASE THROUGH PORT B
012612  013737  001226  001236          MOV     PORTB,SEIZPT    ;ADDRESS OF 'LOCKED ON' PORT
012620  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A
012626  013737  001224  001234          MOV     PORTA,PTNBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012634  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012640  016037  000012  001126          MOV     RMDS(R0),$BDDAT ;GET CONTENTS OF RMDS
012646  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012654  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012660  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
012664  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012672  042737  000077  001164          BIC     #^C177700,$TMP0 ;SAVE SPECIFIED BITS
012700  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012706  001414                          BEQ     70$             ;BR IF OK
012710  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012716  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
012724  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012732  104024                          EMT     24
012734  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012740  000240                  70$:    NOP

                                ;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST

012742  105737  001103                  TSTB    $ERFLG          ;DID AN ERROR OCCUR
012746  001412                          BEQ     3$              ;BR IF NOT
012750  032777  001000  166162          BIT     #SW09,@SWR      ;SEE IF LOOP ON ERROR (SWR9 - 1)
012756  001406                          BEQ     3$              ;BR IF NOT
012760  105037  001103                  CLRB    $ERFLG          ;CLEAR THE ERROR FLAG
012764  005037  001176                  CLR     $TIMES          ;CLEAR THE MAX ITERATION COUNT
012770  000177  166114                  JMP     @$LPERR         ;GO TO THE LOOP ADDRESS
012774  032777  040000  166136  3$:     BIT     #SW14,@SWR      ;LOOP ON TEST ?
013002  001054                          BNE     6$              ;BR IF LOOPING
013004  104401  020367                  TYPE    ,CYCLED         ;TYPE 'CYCLE DOWN'
013010  104401  020156                  TYPE    ,SWTCHN         ;'SWITCH TO A/B'
013014  104401  020407                  TYPE    ,CYCLEU         ;'CYCLE THE DRIVE UP'
013020  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
013026  013737  001226  001234          MOV     PORTB,PTNBR     ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
013034  032760  010000  000012  4$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' RESET ?
013042  001374                          BNE     4$              ;BR IF NO (DRIVE NOT CYCLED DOWN)
013044  032760  010000  000012  5$:     BIT     #MOL,RMDS(R0)   ;IS 'MOL' SET ?
013052  001774                          BEQ     5$              ;BR IF NO (DRIVE NOT CYCLED UP)

                                ;SET VOLUME VALID FOR BOTH PORTS

013054  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR THROUGH PORT B
013062  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET THROUGH PORT B
013070  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE PORT B
```

SEQ 0050

```
013076  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
013104  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
013112  012760  000021  000000          MOV     #21,RMCS1(R0)    ;ISSUE A READIN PRESET THROUGH PORT A
013120  012760  010000  000032          MOV     #FMT16,RMOF(R0)  ;SET FMT16
013126  012760  000013  000000          MOV     #13,RMCS1(R0)    ;RELEASE PORT A
013134  012737  011610  001254  6$:     MOV     #5000.,WATCH     ;SPINDLE MOTOR 'COOL DOWN' DELAY
013142  005737  001254          7$:     TST     WATCH            ;FINISHED ?
013146  001375                          BNE     7$               ;BR IF NOT
013150  000004                          SCOPE                    ;LOOP ?
013152  000137  013160                  JMP     $EOP             ;GO TO THE END OF PASS ROUTINE
    324
    325                         ;;*****************************************************************
    326                         ;PUT NEWTEST HERE
    327                         ;;*****************************************************************
013156  000004                TST11:  SCOPE
```

```
1                                       .SBTTL  END OF PASS ROUTINE

                                        ;;**********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY''
                                        ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO TST1AA

013160                                  $EOP:
013160  005737  001266                          TST     KYBCTL          ;ENTERED TEST VIA KEYBOARD COMMAND ?
013164  001402                                  BEQ     .+6             ;BR IF NOT
013166  000137  002622                          JMP     EXEC            ;RETURN TO KEYBOARD CONTROL
013172  005037  001102                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
013176  005037  001176                          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
013202  005237  001100                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
013206  042737  100000  001100                BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
013214  005327                                  DEC     (PC)+           ;;LOOP?
013216  000001                          $EOPCT: .WORD   1
013220  003066                                  BGT     $DOAGN          ;;YES
013222  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
013224  000001                          $ENDCT: .WORD   1
013226  013216                                  $EOPCT
013230  104401  013236                          TYPE    ,65$            ;;TYPE ASCIZ STRING
013234  000407                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ,;65$:  .ASCIZ  <12><15>/END PASS #/
013254                                  64$:
013254  013746  001100                          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                        ;;TYPE PASS NUMBER
013260  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
013262  005737  001112                          TST     $ERTTL          ;;SEE IF ANY ERRORS THIS PASS
013266  001431                                  BEQ     $GT42P          ;;BR IF NO ERRORS TO REPORT
013270  104401  013276                          TYPE    ,67$            ;;TYPE ASCIZ STRING
013274  000421                                  BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
013340                                  66$:
013340  013746  001112                          MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
013344  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
013346  005037  001112                          CLR     $ERTTL          ;;CLEAR ERROR TOTAL
013352  104401  001207                  $GT42P: TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
013356  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
013362  001405                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
013364  000005                                  RESET                   ;;CLEAR THE WORLD
013366  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
013370  000240                                  NOP                     ;;SAVE ROOM
013372  000240                                  NOP                     ;;FOR
013374  000240                                  NOP                     ;;ACT11
013376                                  $DOAGN:
013376  000137                                  JMP     @(PC)+          ;;RETURN
013400  003104                          $RTNAD: .WORD   TST1AA
013402     377     377     000          $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                                .EVEN
```

```
                                        .SBTTL  CLOCK SUBROUTINES
                                ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
013406  012737  013456  000004  CKCLK:  MOV     #CKCLK1,@#ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
013414  005037  000006          CLR     @#ERRVEC+2        ;NEW PSW
013420  005777  165566          TST     @$LKCSR         ;CHECK FOR KW11-P
013424  013701  001216          MOV     $LPVEC,R1       ;KW11-P VECTOR ADDRESS
013430  012721  013540          MOV     #CLOCK,(R1)+    ;SET UP KW11-P VECTOR
013434  012711  000300          MOV     #300,(R1)       ;PSW - PRI 6
013440  012777  177777  165546  MOV     #-1,@$LKCSB     ;LOAD COUNTER BUFFER WITH 1'S
013446  012777  000135  165536  MOV     #135,@$LKCSR    ;SET CLOCK - CNT UP, 16MS, CONT INT
013454  000425                  BR      CKCLK3
013456  062706  000004  CKCLK1: ADD     #4,SP           ;RESTORE THE STACK POINTER
013462  012737  013520  000004  MOV     #CKCLK2,@#ERRVEC  ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
013470  005777  165524          TST     @$LKS           ;LOOK FOR KW11-L
013474  013701  001222          MOV     $LLVEC,R1       ;KW11-L VECTOR ADDRESS
013500  012721  013540          MOV     #CLOCK,(R1)+    ;SET UP KW11-L VECTOR
013504  012711  000300          MOV     #300,(R1)       ;PSW - PRI 6
013510  012777  000100  165502  MOV     #100,@$LKS      ;SET KW11-L INTERRUPT
013516  000404                  BR      CKCLK3
013520  062706  000004  CKCLK2: ADD     #4,SP           ;RESTORE THE STACK POINTER
013524  062716  000002          ADD     #2,(SP)         ;INCREMENT RETURN, NO CLOCK
013530  012737  000006  000004  CKCLK3: MOV     #6,@#ERRVEC     ;RESTORE THE ERROR VECTOR
013536  000207                  RTS     PC
                                ;ROUTINE TO COUNT CLOCK TICKS
013540  062737  000021  001252  CLOCK:  ADD     #17.,TIME       ;ADD 17 MS TO ELAPSED TIME COUNTER
013546  005737  001254          TST     WATCH           ;IS WATCH ALREADY ZERO ?
013552  001406                  BEQ     1$              ;BR IF IT IS
013554  162737  000021  001254  SUB     #17.,WATCH      ;SUBTRACT 17 MS FROM WATCH DOG COUNTER
013562  100002                  BPL     1$              ;BR IF NOT MINUS
013564  005037  001254          CLR     WATCH           ;CLEAR WATCH DOG COUNTER
013570  000002          1$:     RTI                     ;RETURN
                                ;ROUTINE TO CALCULATE + 25% TIME TOLERANCE VALUES
013572  005746          TOLER:  TST     -(SP)           ;MAKE ROOM ON THE STACK
013574  016616  000002          MOV     2(SP),(SP)      ;SAVE STACK
013600  013546                  MOV     @(R5)+,-(SP)    ;GET TIME VALUE
013602  011666  000004          MOV     (SP),4(SP)      ;MOVE TIME VALUE
013606  006216                  ASR     (SP)            ;DIVIDE BY 2
013610  006216                  ASR     (SP)            ;DIVIDE BY 2 AGAIN (FOR A TOTAL OF 4)
013612  062666  000002          ADD     (SP)+,2(SP)     ;CALCULATE UPPER LIMIT FOR TIMEOUT
013616  000205                  RTS     R5              ;RETURN WITH TOLERANCES ON THE STACK
```

```
                                .SBTTL  SCOPE HANDLER ROUTINE

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

013620                          $SCOPE:
013620  104407                         CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
013622  004737  014154                  JSR     PC,STOP
013626  032777  040000  165304  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
013634  001402                         BEQ     9$              ;;NO IF SW14=0
013636  000137  014136                  JMP     $OVER           ;;JUMP OVER SCOPE ROUTINE
013642                          9$:
                                ;#####START OF CODE FOR THE XOR TESTER#####
013642  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
013644  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
013650  012737  013670  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
013656  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
013662  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
013666  000517                         BR      $SVLAD          ;;GO TO THE NEXT TEST
013670  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
013672  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
013676  000517                         BR      $OVER           ;;LOOP ON THE PRESENT TEST
013700                          6$:;#####END OF CODE FOR THE XOR TESTER#####
013700  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
013704  001465                         BEQ     3$              ;;BR IF NO
013706  022737  177777  014524          CMP     #-1,CPSAVE      ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
013714  001455                         BEQ     2003$           ;;KICK AROUND ROUTINE IF SO
013716  013746  000004                  MOV     ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
013722  012737  013740  000004          MOV     #2000$,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
013730  013737  177766  014524          MOV     177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
013736  000406                         BR      2001$
013740  012737  177777  014524  2000$:  MOV     #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
013746  012716  013754                  MOV     #2001$,(SP)     ;;SETUP RETURN ADDRESS
013752  000002                         RTI
013754  012637  000004          2001$.  MOV     (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

013760  022737  177777  014524  2002$:  CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
013766  001430                         BEQ     2003$           ;;BRANCH IF SO
013770  032737  000001  014524          BIT     #BIT00,CPSAVE   ;;SEE IF THE POWER MONITOR BIT IS ON
013776  001424                         BEQ     2003$           ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
014000  042737  000001  177766          BIC     #BIT00,177766   ;;CLEAR THE BIT FOUND TO BE SET
014006  013746  001140                  MOV     SWR,-(SP)       ;;SAVE SWR ADDRESS
014012  017646  000000                  MOV     @(SP),-(SP)     ;;SAVE SWR VALUE
014016  012737  000176  001140          MOV     #176,SWR        ;;GET SOFTWARE SWR ADDRESS
014024  011677  165110                  MOV     (SP),@SWR       ;;GET CURRENT SWR VALUE
014030  042777  001000  165102          BIC     #BIT09,@SWR     ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
014036  104177                         EMT     177             ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
014040  012676  000000                  MOV     (SP)+,@'SP)     ;;RESTORE SWR TO ORIGINAL VALUE
014044  012637  001140                  MOV     (SP)+,SWR       ;;RESTORE SWR ADDRESS
```

```
                                        2003$:
014050                                  4$:     CLRB    $ERFLG          ;;ZERO THE ERRUR FLAG
014050  105037  001103                        CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
014054  005037  001176                  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
014060  032777  004000  165052                BNE     1$              ;;BR IF YES
014066  001011                                TST     $PASS           ;;IF FIRST PASS OF PROGRAM
014070  005737  001100                        BEQ     1$              ;;      INHIBIT ITERATIONS
014074  001406                                INC     $ICNT           ;;INCREMENT ITERATION COUNT
014076  005237  001104                        CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
014102  023737  001176  001104                BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
014110  002012                          1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
014112  012737  000001  001104                MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
014120  013737  014152  001176          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
014126  105237  001102                        MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
014132  011637  001106                  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
014136  013777  001102  164776                MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
014144  013716  001106                        RTI                     ;;FIXES PS
014150  000002                          $MXCNT: 5.                      ;;MAX. NUMBER OF ITERATIONS
014152  000005
    2
    3                                   ;DROP PRIORITY TO ALLOW CONSOLE INTERRUPT
    4
    5 014154                            STOP:
014154  012746  000140                        MOV     #PR3,-(SP)      ;;PUT NEW PS ON STACK
014160  012746  014166                        MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
014164  000002                              RTI                     ;;POP NEW PC AND PS
014166                              64$:
    6
    7                                   ;RAISE PRIORITY TO INHIBIT CONSOLE INTERRUPT
    8
    9 014166  012746  000240                MOV     #PR5,-(SP)      ;;PUT NEW PS ON STACK
014172  012746  014200                        MOV     #65$,-(SP)      ;;PUT NEW PC ON  STACK
014176  000002                              RTI                     ;;POP NEW PC AND PS
014200                              65$:
   10 014200  000207                        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       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

014202  10503/  014526        $ERROR: CLRB    IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
014206  104407                        CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
014210  113737  001102  001242        MOVB    $TSTNM,TSTNUM
014216  105237  001103        7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
014222  001775                        BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
014224  013777  001102  164710        MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
014232  032777  002000  164700        BIT     #BIT10,@SWR     ;;BELL ON ERROR?
014240  001402                        BEQ     1$              ;;NO - SKIP
014242  104401  001202                TYPE    ,$BELL          ;;RING BELL
014246  005237  001112        1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
014252  011637  001116                MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
014256  162737  000002  001116        SUB     #2,$ERRPC
014264  117737  164626  001114        MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
014272  032777  001000  164640        BIT     #BIT09,@SWR     ;;SEE IF LOOP ON ERROR IS SET
014300  001060                        BNE     1004$           ;;BRANCH AROUND ROUTINE IF SO
014302  122737  000177  001114        CMPB    #177,$ITEMB     ;;SEE IF THIS IS THE POWER FAIL CALL
014310  001454                        BEQ     1004$           ;;BRANCH AROUND ROUTINE IF IT IS
014312  105737  014526                TSTB    IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
014316  001047                        BNE     1003$           ;;BRANCH IF SO
014320  022737  177777  014524        CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
014326  001445                        BEQ     1004$           ;;BRANCH IF SO
014330  013746  000004                MOV     ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
014334  012737  014352  000004        MOV     #1000$,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
014342  013737  177766  014524        MOV     177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
014350  000406                        BR      1001$
014352  012737  177777  014524  1000$:  MOV     #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
014360  012716  014366                MOV     #1001$,(SP)     ;;SETUP RETURN ADDRESS
014364  000002                        RTI
014366  012637  000004        1001$:  MOV     (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

014372  022737  177777  014524  1002$:  CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
014400  001420                        BEQ     1004$           ;;BRANCH IF SO
014402  032737  000001  014524        BIT     #BIT00,CPSAVE   ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
014410  001414                        BEQ     1004$           ;;BRANCH IF OK
014412  042737  000001  177766        BIC     #BIT00,177766   ;;CLEAR THE BIT FOUND SET
014420  113737  001114  014526        MOVB    $ITEMB,IBSAVE   ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
014426  112737  000177  001114        MOVB    #177,$ITEMB     ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
014434  000402                        BR      1004$           ;;BRANCH OVER IBSAVE CLEARING

014436  105037  014526        1003$:  CLRB    IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
014442                          1004$:
014442  032777  020000  164470        BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
014450  001004                        BNE     20$             ;;SKIP TYPEOUTS
014452  004737  014530                JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
```

```
014456  10440'  001207                  TYPE    ,$CRLF
014462                          20$:
014462  105737  014526          2$:     TSTB    IBSAVE          ;;SEE IF IBSAVE IS LOADED
014466  001005                          BNE     3$              ;;BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
014470  005777  164444                  TST     @SWR            ;;HALT ON ERROR
014474  100002                          BPL     3$              ;;SKIP IF CONTINUE
014476  000000                          HALT                    ;;HALT ON ERROR!
014500  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
014502                          3$:
014502  022737  013366  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
014510  001001                          BNE     6$              ;;BRANCH IF NO
014512  000000                          HALT                    ;;YES
014514                          6$:
014514  105737  014526                  TSTB    IBSAVE          ;;SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
014520  001236                          BNE     7$              ;;BRANCH BACK TO CALL ORIGINAL ERROR
014522  000002                          RTI                     ;;RETURN
014524  000000                  CPSAVE: .WORD   0               ;;LOCATION TO SAVE CPU ERROR REG CONTENTS
014526  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.

014530                          $ERRTYP:
014530  104401  001207                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
014534  010046                  MOV     R0,-(SP)        ;;SAVE R0
014536  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
014540  153700  001114          BISB    @#$ITEMB,R0
014544  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                        ;;TYPE THE PC OF THE ERROR
014546  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                        ;;ERROR ADDRESS
014552  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
014554  000456                  BR      10$             ;;GET OUT
014556  122700  000177          1$:     CMPB    #177,R0         ;;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
014562  001006                  BNE     1000$           ;;BRANCH IF NOT
014564  113737  001102  015066          MOVB    $TSTNM,PFTSTN   ;;GET TEST NUMBER
014572  012700  014726          MOV     #PFECH,R0       ;;MOVE POWER FAIL ERROR CALL TABLE TO R0
014576  000406                  BR      1001$           ;;BRANCH TO CALL ERROR
014600  005300                  1000$:  DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
014602  006300                  ASL     R0              ;;       WORK FOR THE ERROR TABLE
014604  006300                  ASL     R0
014606  006300                  ASL     R0
014610  062700  001276          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
014614  012037  014624          1001$:  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
014620  001404                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
014622  104401                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
014624  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
014626  104401  001207          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
014632  012037  014642          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
014636  001404                  BEQ     5$              ;;SKIP TYPEOUT IF 0
014640  104401                  TYPE                    ;;TYPE THE "DATA HEADER"
014642  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
014644  104401  001207          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
014650  010146                  5$:     MOV     R1,-(SP)        ;;SAVE R1
014652  012001                  MOV     (R0)+,R1        ;;PICKUP "DATA TABLE" POINTER
014654  001415                  BEQ     9$              ;;BR IF NO DATA TO BE TYPED
014656  012000                  MOV     (R0)+,R0        ;;PICKUP "DATA FORMAT" POINTER
014660  105720                  6$:     TSTB    (R0)+           ;;"OCTAL" OR "DECIMAL"
014662  001003                  BNE     7$              ;;BR IF DECIMAL
014664  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
014666  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
014670  000402                  BR      8$
014672                          7$:
014672  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
014674  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
014676  005711                  8$:     TST     (R1)            ;;IS THERE ANOTHER NUMBER?
014700  001403                  BEQ     9$              ;;BR IF NO
014702  104401  014722          TYPE    ,10$            ;;TYPE TWO(2) SPACES
014706  000764                  BR      6$              ;;LOOP

014710  012601                  9$:     MOV     (SP)+,R1        ;;RESTORE R1
014712  012600                  10$:    MOV     (SP)+,R0        ;;RESTORE R0
```

```
014714  10440*  001207                    TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
014720  000207                            RTS     PC              ;;RETURN
014722     040     040     000  11$:      .ASCIZ  /  /            ;;TWO(2) SPACES
                                          .EVEN
014726  014736  015020  015052  PFECH:    PFECH1,PFECH2,PFECH3,PFECH4 ;;WORDS DEFINING TABLES BELOW
014736     120     117     127  PFECH1:   .ASCIZ  ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
015020     124     105     123  PFECH2:   .ASCIZ  ?TESTNO  ERR PC  CPUERREG?
                                          .EVEN
015052  015066  001116  014524  PFECH3:   .WORD   PFTSTN,$ERRPC,CPSAVE,0
015062     000     000     000  PFECH4:   .BYTE   0,0,0,0
015066  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
                                ;*

015070  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
015074  100002                          BPL     1$              ;;BR IF YES
015076  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
015100  000407                          BR      3$              ;;LEAVE
015102  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
015104  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
015110  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
015112  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
015114  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
015116  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
015120  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
015124  000002                          RTI                     ;;RETURN
015126  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
015132  001430                          BEQ     8$
015134  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
015140  001006                          BNE     5$
015142  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
015144  104401                          TYPE                    ;;TYPE A CR AND LF
015146  001207                          $CRLF
015150  105037  015356                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
015154  000755                          BR      2$              ;;GET NEXT CHARACTER
015156  004737  015240          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
015162  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
015166  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
015170  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
015174  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
015200  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
015202  004737  015240                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
015206  105337  015356                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
015212  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

015214  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
015220  004737  015240          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
015224  132737  000007  015356          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
015232  001372                          BNE     9$              ;;TAB STOP
015234  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
015236  000724                          BR      2$              ;;GET NEXT CHARACTER
```

```
015240                          $TYPEC:
015240  105777  163700                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?
015244  100022                  BPL     10$             ;;BR IF NOT
015246  017746  163674          MOV     @$TKB,-(SP)     ;;GET CHAR
015252  042716  177600          BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS
015256  122716  000023          CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF
015262  001012                  BNE     102$            ;;BR IF NOT
015264                          101$:
015264  105777  163654          TSTB    @$TKS           ;;WAIT FOR CHAR
015270  100375                  BPL     101$
015272  117716  163650          MOVB    @$TKB,(SP)      ;;GET CHAR
015276  042716  177600          BIC     #177600,(SP)    ;;STRIP IT
015302  122716  000021          CMPB    #$XON,(SP)      ;;WAS IT XON?
015306  001366                  BNE     101$            ;;BR IF NOT
015310                          102$:
015310  005726                  TST     (SP)+           ;;FIX STACK
015312                          10$:
015312  105777  163632          TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
015316  100375                  BPL     10$
015320  116677  000002  163624  MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
015326  122766  000015  000002  CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
015334  001003                  BNE     1$              ;;BRANCH IF NO
015336  105037  015356          CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
015342  000406                  BR      $TYPEX          ;;EXIT
015344  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
015352  001402                  BEQ     $TYPEX          ;;BRANCH IF YES
015354  105227                  INCB    (PC)+           ;;COUNT THE CHARACTER
015356  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
015360  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.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF D'GITS TO TYPE
                                ;*      .BYTE   M               ;;M-1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$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

015362  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
015366  116637  000001  015605          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
015374  112637  015607                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
015400  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
015404  000406                          BR      $TYPON
015406  112737  000001  015605  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
015414  112737  000006  015607          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
015422  112737  000005  015604  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
015430  010346                          MOV     R3,-(SP)        ;;SAVE R3
015432  010446                          MOV     R4,-(SP)        ;;SAVE R4
015434  010546                          MOV     R5,-(SP)        ;;SAVE R5
015436  113704  015607                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
015442  005404                          NEG     R4
015444  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
015450  110437  015606                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
015454  113704  015605                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
015460  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
015464  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
015466  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
015470  000404                          BR      3$              ;;GO DO MSB
015472  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
015474  006105                          ROL     R5
015476  006105                          ROL     R5
015500  010503                          MOV     R5,R3
015502  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
015504  105337  015606                  DECB    $OMODE          ;;TYPE THIS DIGIT?
015510  100016                          BPL     7$              ;;BR IF NO
015512  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
015516  001002                          BNE     4$              ;;TEST FOR 0
015520  005704                          TST     R4              ;;SUPPRESS THIS 0?
015522  001403                          BEQ     5$              ;;BR IF YES
015524  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

```
015526  052703  000060            BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
015532  052703  000040     5$:    BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
015536  110337  015602            MOVB    R3,8$           ;;SAVE FOR TYPING
015542  104401  015602            TYPE    ,8$             ;;GO TYPE THIS DIGIT
015546  105337  015604     7$:    DECB    $OCNT           ;;COUNT BY 1
015552  003347                    BGT     2$              ;;BR IF MORE TO DO
015554  002402                    BLT     6$              ;;BR IF DONE
015556  005204                    INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
015560  000744                    BR      2$              ;;GO DO THE LAST DIGIT
015562  012605     6$:            MOV     (SP)+,R5        ;;RESTORE R5
015564  012604                    MOV     (SP)+,R4        ;;RESTORE R4
015566  012603                    MOV     (SP)+,R3        ;;RESTORE R3
015570  016666  000002  000004    MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
015576  012616                    MOV     (SP)+,(SP)
015600  000002                    RTI                     ;;RETURN
015602     000     8$:            .BYTE   0               ;;STORAGE FOR ASCII DIGIT
015603     000                    .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
015604     000     $OCNT:         .BYTE   0               ;;OCTAL DIGIT COUNTER
015605     000     $OFILL:        .BYTE   0               ;;ZERO FILL SWITCH
015606  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

015610                          $TYPDS:
015610  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
015612  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015614  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
015616  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
015620  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
015622  012746  020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
015626  016605  000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
015632  100004                         BPL     1$              ;;BR IF INPUT IS POS.
015634  005405                         NEG     R5              ;;MAKE THE BINARY NUMBER POS.
015636  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
015644  005000                 1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
015646  012703  016024                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
015652  112723  000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
015656  005002                 2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
015660  016001  016014                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
015664  160105                 3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
015666  002402                         BLT     4$              ;;BR IF DONE
015670  005202                         INC     R2              ;;INCREASE THE BCD DIGIT BY 1
015672  000774                         BR      3$
015674  060105                 4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
015676  005702                         TST     R2              ;;CHECK IF BCD DIGIT=0
015700  001002                         BNE     5$              ;;FALL THROUGH IF 0
015702  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
015704  100407                         BMI     7$              ;;BR IF YES
015706  106316                 5$:     ASLB    (SP)            ;;MSD?
015710  103003                         BCC     6$              ;;BR IF NO
015712  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
015720  052702  000060         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
015724  052702  000040         7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
015730  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
015732  005720                         TST     (R0)+           ;;JUST INCREMENTING
015734  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
015740  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
015742  003002                         BGT     8$              ;;GO TO EXIT
015744  010502                         MOV     R5,R2           ;;GET THE LSD
015746  000764                         BR      6$              ;;GO CHANGE TO ASCII
015750  105726                 8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
015752  100003                         BPL     9$              ;;BR IF NO
015754  116663  177777  177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
015762  105013                 9$:     CLRB    (R3)            ;;SET THE TERMINATOR
015764  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
015766  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
015770  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
015772  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
015774  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
015776  104401  016024                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
016002  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
016010  012616                          MOV     (SP)+,(SP)
016012  000002                          RTI                     ;;RETURN TO USER
016014  023420                  $DTBL:  10000.
016016  001750                          1000.
016020  000144                          100.
016022  000012                          10.
016024                          $DBLK:  .BLKW   4
```

```
                                .SBTTL  TTY INPUT ROUTINE

                                ;;*****************************************************************
                                ;
                                .ENABL  LSB
016034  000000                  $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
016036  000000                  $TKQIN: .WORD   0               ;;INPUT POINTER
016040  000000                  $TKQOUT: .WORD  0               ;;OUTPUT POINTER
016042                          $TKQSRT: .BLKB  1               ;;TTY KEYBOARD QUEUE
        016043                  $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
                                ;
016044  005037  016034          $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
016050  012737  016042  016036        MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
016056  013737  016036  016040        MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
016064  012737  016114  000060        MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
016072  012737  000200  000062        MOV     #200,@#TKVEC+2  ;;'BR'' LEVEL 4
016100  005777  163042                TST     @$TKB           ;;CLEAR DONE FLAG
016104  012777  000100  163032        MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
016112  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)
                                ;
016114  117746  163026          $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
016120  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
016124  021627  000021                  CMP     (SP),#$XON      ;;IS IT A RANDOM XON?
016130  001002                          BNE     30$             ;;BRANCH IF NO
016132  005726                          TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK
016134  000002                          RTI                     ;;RETURN
016136                          30$:
016136  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL C?
016142  001007                          BNE     1$              ;;BRANCH IF NO
016144  104401  017242                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
016150  004737  016044                  JSR     PC,$TKINT       ;;INIT THE KEYBOARD
016154  005726                          TST     (SP)+           ;;CLEAN UP STACK
016156  000137  001766                  JMP     START           ;;CONTROL C RESTART
016162  021627  000007          1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
016166  001004                          BNE     2$              ;;BRANCH IF NO
016170  022737  000176  001140          CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
016176  001500                          BEQ     6$              ;;GO TO SWR CHANGE

016200                          2$:
016200  022737  000001  016034          CMP     #1,$TKCNT       ;;IS THE QUEUE FULL?
016206  001004                          BNE     3$              ;;BRANCH IF NO
016210  104401  001202                  TYPE    ,$BELL          ;;RING THE TTY BELL
```

```
016214  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
016216  000451                          BR      5$              ;;EXIT
016220  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
016224  001021                          BNE     32$             ;;BRANCH IF NO
016226  005077  162712                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
016232  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
016234  105777  162704          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
016240  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
016242  117746  162700                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
016246  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
016252  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
016256  001366                          BNE     31$             ;;BRANCH IF NO
016260  012777  000100  062656          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
016266  000002                          RTI                     ;;RETURN
016270  005237  016034          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
016274  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
016300  002405                          BLT     4$              ;;BRANCH IF YES
016302  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
016306  003002                          BGT     4$              ;;BRANCH IF YES
016310  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
016314  112677  177516          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
016320  005237  016036                  INC     $TKQIN          ;;UPDATE THE POINTER
016324  023727  016036  016043          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
016332  001003                          BNE     5$              ;;BRANCH IF NO
016334  012737  016042  016036          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
016342  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.
016344  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
016352  001124                          BNE     15$             ;;EXIT IF NOT
016354  105777  162564                  TSTB    @$TKS           ;;IS A CHAR WAITING?
016360  100121                          BPL     15$             ;;IF NOT, EXIT
016362  117746  162560                  MOVB    @$TKB,-(SP)     ;;YES
016366  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
016372  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
016376  001300                          BNE     2$              ;;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.
016400  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
016406  001674                          BEQ     2$              ;;BRANCH IF YES
016410  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
016412  004737  016044                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
016416  005077  162522                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
016422  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

016430  104401  017254                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
016434  104401  017261          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
016440  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
016444  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
016446  104401  017272                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
016452  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
016454  005046                          CLR     -(SP)           ;;THE NEW SWR
016456  105777  162462          7$:     TSTB    @$TKS           ;;CHAR THERE?
016462  100375                          BPL     7$              ;;IF NOT TRY AGAIN

016464  117746  162456                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
016470  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

016474  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
016500  001015                          BNE     9$              ;;BRANCH IF NOT
016502  104401  017242                  TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
016506  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
016512  123727  001135  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
016520  001003                          BNE     8$              ;;BRANCH IF NO
016522  012777  000100  162414          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
016530  000137  001766          8$:     JMP     START           ;;CONTROL-C RESTART


016534  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
016540  001005                          BNE     10$             ;;BRANCH IF NOT
016542  104401  017247                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
016546  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
016552  000737                          BR      19$             ;;LET'S TRY IT AGAIN


016554  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
016560  001022                          BNE     16$             ;;BRANCH IF NO
016562  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
016566  001403                          BEQ     11$             ;;BRANCH IF YES
016570  016677  000002  162342          MOV     2(SP),@SWR      ;;SAVE NEW SWR
016576  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
016602  104401  001207          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
016606  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
016614  001003                          BNE     15$             ;;BRANCH IF NOT
016616  012777  000100  162320          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
016624  000002          15$:    RTI                             ;;RETURN
016626  004737  015240          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
016632  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
016636  002420                          BLT     18$             ;;BRANCH IF YES
016640  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
016644  003015                          BGT     18$             ;;BRANCH IF YES
016646  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
016652  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
016656  001403                          BEQ     17$             ;;BRANCH IF YES
016660  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
016662  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
016664  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
016666  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
016672  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
016676  000667                          BR      7$              ;;GET THE NEXT ONE
016700  104401  001206          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
016704  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
                                ;

016706  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
016710  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
016716  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
016722  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
016724  012746  016732                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
016730  000002                          RTI                     ;;POP NEW PC AND PS
016732                          64$:
016732  005737  016034          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
016736  001775                          BEQ     1$
016740  005337  016034                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
016744  117766  177070  000004          MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
016752  005237  016040                  INC     $TKQOUT         ;;UPDATE THE POINTER
016756  023727  016040  016043          CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
016764  001003                          BNE     2$              ;;BRANCH IF NO
016766  012737  016042  016040          MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
016774  000002                  2$:     RTI                     ;;RETURN
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

016776  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
017000  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
017002  012703  017232          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
017006  022703  017242          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
017012  101456                          BLOS    4$              ;;BR IF YES
017014  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
017016  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
017020  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
017024  001022                          BNE     5$              ;;BR IF NO
017026  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
017030  001007                          BNE     6$              ;;BR IF NO
017032  112737  000134  017230          MOVB    #'\,9$          ;;TYPE A BACK SLASH
017040  104401  017230                  TYPE    ,9$
017044  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
017050  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
017052  020327  017232                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
017056  103434                          BLO     4$              ;;BR IF YES
017060  111337  017230                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
017064  104401  017230                  TYPE    ,9$             ;;GO TYPE
017070  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
017072  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
017074  001406                          BEQ     7$              ;;BR IF NO
017076  112737  000134  017230          MOVB    #'\,9$          ;;TYPE A BACK SLASH
017104  104401  017230                  TYPE    ,9$
017110  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
017112  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
017116  001003                          BNE     8$              ;;BR IF NO
```

```
017120  104401  017247                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
017124  000726                          BR      1$              ;;GO START OVER
017126  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
017132  001011                          BNE     3$              ;;BRANCH IF NO
017134  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
017136  104401  001207                  TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
017142  104401  017232                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
017146  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
017150  104401  001206          4$:     TYPE    ,$QUES          ;;TYPE A '?'
017154  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
017156  111337  017230          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
017162  104401  017230                  TYPE    ,9$
017166  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
017172  001305                          BNE     2$              ;;LOOP IF NOT RETURN
017174  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
017200  104401  001210                  TYPE    ,$LF            ;;TYPE A LINE FEED
017204  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
017206  012603                          MOV     (SP)+,R3        ;;RESTORE R3
017210  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
017212  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
017220  012766  017232  000004          MOV     #$TTYIN,4(SP)
017226  000002                          RTI                     ;;RETURN
017230     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
017231     000                  .BYTE   0                       ;;TERMINATOR
017232                  $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
017242     136     103     015  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL 'C'
017247     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
017254     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
017261     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR - /
017272     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
                                .EVEN
```

SEQ 0070

```
                                .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 INSURED 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

017304  011646                 $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
017306  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
017314  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
017316  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
017320  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
017322  104411                 1$:     RDLIN                   ;;READ AN ASCIZ LINE
017324  012600                         MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
017326  010037  017432                 MOV     R0,5$           ;;AND SAVE IT
017332  005001                         CLR     R1              ;;CLEAR DATA WORD
017334  005002                         CLR     R2
017336  112046                 2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
017340  001420                         BEQ     3$              ;;IF ZERO GET OUT
017342  122716  000060                 CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
017346  003026                         BGT     4$              ;;IS AN OCTAL DIGIT
017350  122716  000067                 CMPB    #'7,(SP)
017354  002423                         BLT     4$
017356  006301                         ASL     R1              ;;*2
017360  006102                         ROL     R2
017362  006301                         ASL     R1              ;;*4
017364  006102                         ROL     R2
017366  006301                         ASL     R1              ;;*8
017370  006102                         ROL     R2
017372  042716  177770                 BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
017376  062601                         ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
017400  000756                         BR      2$              ;;LOOP
017402  005726                 3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
017404  010166  000012                 MOV     R1,12(SP)       ;;SAVE THE RESULT
017410  010237  017442                 MOV     R2,$HIOCT
017414  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
017416  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
017420  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
017422  000002                         RTI                     ;;RETURN
017424  005726                 4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
017426  105010                         CLRB    (R0)            ;;SET A TERMINATOR
017430  104401                         TYPE                    ;;TYPE UP THRU THE BAD CHAR.
017432  000000                 5$:     .WORD   0
017434  104401  001206                 TYPE    ,$QUES          ;;"?" "CR" & "LF"
017440  000730                         BR      1$              ;;TRY AGAIN
017442  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

017444                  $SAVREG:
017444  010046                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
017446  010146                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
017450  010246                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
017452  010346                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
017454  010446                 MOV     R4,-(SP)        ;;PUSH R4 ON STACK
017456  010546                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
017460  016646  000022         MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
017464  016646  000022         MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
017470  016646  000022         MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
017474  016646  000022         MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
017500  000002                 RTI

                        ;*RESTORE R0-R5
                        ;*CALL:
                        ;*      RESREG
017502                  $RESREG:
017502  012666  000022         MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
017506  012666  000022         MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
017512  012666  000022         MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
017516  012666  000022         MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
017522  012605                 MOV     (SP)+,R5        ;;POP STACK INTO R5
017524  012604                 MOV     (SP)+,R4        ;;POP STACK INTO R4
017526  012603                 MOV     (SP)+,R3        ;;POP STACK INTO R3
017530  012602                 MOV     (SP)+,R2        ;;POP STACK INTO R2
017532  012601                 MOV     (SP)+,R1        ;;POP STACK INTO R1
017534  012600                 MOV     (SP)+,R0        ;;POP STACK INTO R0
017536  000002                 RTI
```

```
                                .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.

017540  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
017542  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
017546  005740                          TST     -(R0)           ;;BACKUP BY 2
017550  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
017552  006300                          ASL     R0              ;;POSITION FOR INDEXING
017554  016000  017574                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
017560  000200                          RTS     R0              ;;GO TO ROUTINE


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

017562  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
017564  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
017572  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
017574  017562                  $TRPAD: .WORD   $TRAP2
017576  015070                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
017600  015406                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
017602  015362                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
017604  015422                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
017606  015610                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

017610  016434                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

017612  016344                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
017614  016706                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
017616  016776                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
017620  017304                          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
017622  017444                          $SAVREG ;;CALL=SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
017624  017502                          $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
```

```
                                .SBTTL  TELETYPE MESSAGES

                ;
017626  200     105     116     ENTERA: .ASCIZ  <CRLF>/ENTER DRIVE ADDRESS: /
017655  040     077     111     ADRERR: .ASCIZ  / ?INVALID ADDRESS/<CRLF>
017700  200     120     117     PORTAIS: .ASCIZ <CRLF>/PORT 'A' ADDRESS IS: /
017727  200     120     117     PORTBIS: .ASCIZ <CRLF>/PORT 'B' ADDRESS IS: /
017756  200     123     131     NOCLOCK: .ASCIZ <CRLF>/SYSTEM MUST HAVE 'L' OR 'P' CLOCK/<CRLF><LF>
020023  012     105     116     TESTNO: .ASCIZ  <LF>/ENTER TEST #: /
020043  040     077     111     BADNO:  .ASCIZ  / ?INVALID TEST NUMBER/<CRLF>
020072  200     012     122     ADDRIS: .ASCIZ  <CRLF><LF>@RH/RM ADDRESS (RMCS1) IS: @
020127  012     105     116     NTRH:   .ASCIZ  <LF>@ENTER RH/RM ADDRESS: @
020156  200     122     105     SWTCHN: .ASCIZ  <CRLF>@RETURN 'PORT SELECT' SWITCH TO 'A/B'@
020224  200     123     127     SWTCHA: .ASCIZ  <CRLF>/SWITCH 'PORT SELECT' TO 'A'/
020261  200     123     127     SWTCHB: .ASCIZ  <CRLF>/SWITCH 'PORT SELECT' TO 'B'/
020316  200     124     110     CONTUE: .ASCIZ  <CRLF>/THEN PRESS 'CONTINUE' ON THE PROCESSOR/<CRLF>
020367  200     123     124     CYCLED: .ASCIZ  <CRLF>/STOP THE DRIVE/
020407  200     123     124     CYCLEU: .ASCIZ  <CRLF>/START THE DRIVE, THE PROGRAM WILL WAIT FOR 'MOL' TO SET/<CRLF>
```

```
                                        .SBTTL  TEST ERROR MESSAGES

020501  104  122  111  EM1:    .ASCIZ  /DRIVE IS NON-EXISTENT ('NED' BIT SET)/
020547  127  122  117  EM2:    .ASCIZ  /WRONG DRIVE TYPE/
020570  120  117  122  EM3:    .ASCIZ  @PORT SELECT SWITCH ON DRIVE, NOT IN 'A/B'@
020642  104  122  111  EM4:    .ASCIZ  /DRIVE NOT ON LINE/
020664  123  105  122  EM5:    .ASCIZ  /SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME/
020746  124  111  115  EM6:    .ASCIZ  /TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS/
021020  124  111  115  EM7:    .ASCIZ  /TIMEOUT ONE-SHOT IS LESS THAN 500 MS/
021065  122  105  101  EM10:   .ASCIZ  /READ IN PRESET DOES NOT SET VOLUME VALID FOR THE PORT/
021153  047  107  117  EM11:   .ASCIZ  /'GO' BIT RESET DURING UNLOAD COMMAND/
021220  111  116  103  EM12:   .ASCIZ  /INCORRECT STATUS DURING UNLOAD COMMAND/
021267  104  122  111  EM13:   .ASCIZ  /DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND/
021354  101  124  124  EM14:   .ASCIZ  /ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD/
021436  101  124  124  EM15:   .ASCIZ  /ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD'/
021522  104  122  111  EM16:   .ASCII  /DRIVE NOT IN NEUTRAL AFTER UNLOAD WITH 'PORT/<CR><LF>
021600  123  105  114          .ASCIZ  @SELECT' SWITCH MOVED FROM 'A/B' @
021641  104  122  111  EM17:   .ASCIZ  /DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP/
021724  104  122  111  EM20:   .ASCIZ  /DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP/
022007  123  124  101  EM21:   .ASCIZ  /STATUS INCORRECT FOR PORT AFTER CYCLE UP/
022060  122  105  107  EM22:   .ASCIZ  /REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT/
022156  047  116  105  EM23:   .ASCIZ  /'NED' NOT SET WHEN RMDS ACCESSED THROUGH PORT NOT SWITCHED/
022251  104  122  111  EM24:   .ASCIZ  /DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/
022331  122  110  057  EM25:   .ASCIZ  @RH/RM DIDN'T RESPOND TO ADDRESSING@
022374  104  122  111  EM30:   .ASCIZ  /DRIVE NOT SEIZED BY PORT/
022425  127  122  117  EM31:   .ASCIZ  /WRONG STATUS SEEN BY THE SEIZING PORT/
022473  122  105  107  EM32:   .ASCIZ  /REGISTER CONTENTS WRONG/
022523  103  117  116  EM33:   .ASCIZ  /CONTROL BUS PARITY ERROR READING INDICATED REGISTER/
022607  103  101  116  EM34:   .ASCIZ  /CAN'T ACCESS DRIVE THROUGH EITHER PORT/
022656  104  122  111  EM35:   .ASCIZ  /DRIVE NOT IN NEUTRAL AFTER RELEASE - REQUEST NOT SET/
022743  104  122  111  EM36:   .ASCIZ  /DRIVE NOT IN NEUTRAL AFTER TIMEOUT - REQUEST NOT SET/
023030  122  105  107  EM37:   .ASCIZ  /REGISTER CONTENTS WRONG AFTER RELEASE OR TIMEOUT/
023111  104  122  111  EM40:   .ASCIZ  /DRIVE IN NEUTRAL AFTER RELEASE - REQUEST SET/
023166  122  105  107  EM41:   .ASCIZ  /REGISTER WRONG AFTER RELEASE WITH REQUEST SET/
```

```
023244     124     105     123  DH1:    .ASCIZ  /TEST #  ERR PC  PORT #  REG ADR CONTENTS/
023315     124     105     123  DH2:    .ASCIZ  /TEST #  ERR PC  PORT #  REG ADR GOOD    BAD/
023371     124     105     123  DH5:    .ASCIZ  /TEST #  ERR PC  REG ADR PORT A  PORT B/
023440     124     105     123  DH6:    .ASCIZ  /TEST #  ERR PC  PORT #/
023467     124     105     123  DH7:    .ASCIZ  /TEST #  ERR PC  PORT #  TIME (IN MS)/
023534     040     040     040  DH13:   .ASCII  /                SEIZE/<CRLF>
023562     124     105     123          .ASCIZ  /TEST #  ERR PC  PORT #/
023611     040     040     040  DH14:   .ASCII  /                SEIZE   ERROR/<CRLF>
023647     124     105     123          .ASCIZ  /TEST #  ERR PC  PORT #  PORT #  REG ADR CONTENTS/
023730     124     105     123  DH17:   .ASCIZ  /TEST #  ERR PC/
023747     040     040     040  DH24:   .ASCII  /                LOCKED  SWITCHED TO/<CRLF>
024013     124     105     123          .ASCIZ  /TEST #  ERR PC  PORT #  PORT #/
024052     044     122     115  DH25:   .ASCIZ  /$RMADR/
024061     040     040     040  DH30:   .ASCII  /                SEIZE   ERROR/<CRLF>
024117     124     105     123          .ASCIZ  /TEST #  ERR PC  PORT #  PORT #  REG ADR GOOD    BAD/
024203     040     040     040  DH34:   .ASCII  /                PORT A  PORT B/<CRLF>
024242     124     105     123          .ASCIZ  /TEST #  ERR PC  RMDS    RMDS/
024277     040     040     040  DH35:   .ASCII  /                RELSNG  ERROR/<CRLF>
024335     124     105     123          .ASCIZ  /TEST #  ERR PC  PORT #  PORT #/
024374     040     040     040  DH37:   .ASCII  /                RELSNG  ERROR/<CRLF>
024432     124     105     123          .ASCIZ  /TEST #  ERR PC  PORT #  PORT #  REG ADR GOOD    BAD/
024516     040     040     040  DH40:   .ASCII  /                RELSNG  RQSTNG/<CRLF>
024555     124     105     123          .ASCIZ  /TEST #  ERR PC  PORT #  PORT #/

                                .EVEN

024614  001242  001116  001234  DT1:    .WORD   TSTNUM,$ERRPC,PTNBR,$BDADR,$BDDAT,0
024630  001242  001116  001234  DT2:    .WORD   TSTNUM,$ERRPC,PTNBR,$BDADR,$GDDAT,$BDDAT,0
024646  001242  001116  001122  DT5:    .WORD   TSTNUM,$ERRPC,$BDADR,$GDDAT,$BDDAT,0
024662  001242  001116  001234  DT6:    .WORD   TSTNUM,$ERRPC,PTNBR,0
024672  001242  001116  001234  DT7:    .WORD   TSTNUM,$ERRPC,PTNBR,TIME,0
024704  001242  001116  001236  DT13:   .WORD   TSTNUM,$ERRPC,SEIZPT,0
024714  001242  001116  001236  DT14:   .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,$BDADR,$BDDAT,0
024732  001242  001116  000000  DT17:   .WORD   TSTNUM,$ERRPC,0
024740  001242  001116  001236  DT24:   .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,0
024752  001272  000000          DT25:   .WORD   $RMADR,0
024756  001242  001116  001236  DT30:   .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,$BDADR,$GDDAT,$BDDAT,0
024776  001242  001116  001170  DT34:   .WORD   TSTNUM,$ERRPC,$TMP2,$TMP3,0
025010  001242  001116  001236  DT35:   .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,0
025022  001242  001116  001236  DT40:   .WORD   TSTNUM,$ERRPC,SEIZPT,OPPRT,0

025034     000     000     001  DF1:    .BYTE   0,0,1,0,0
025041     000     000     001  DF2:    .BYTE   0,0,1,0,0,0
025047     000     000     000  DF5:    .BYTE   0,0,0,0,0
025054     000     000     001  DF6:    .BYTE   0,0,1
025057     000     000     001  DF7:    .BYTE   0,0,1,1
025063     000     000     001  DF14:   .BYTE   0,0,1,1,0,0
025071     000     000          DF17:   .BYTE   0,0
025073     000                  DF25:   .BYTE   0
025074     000     000     001  DF30:   .BYTE   0,0,1,1,1,0,0
025103     000     000     000  DF34:   .BYTE   0,0,0,0
                                .EVEN
```

```
                        .SBTTL  CONSTANTS, TABLES, ETC
                        ;TABLE OF TEST STARTING ADDRESSES

025110  003122  TSTADR: .WORD   TST1            ;STARTING ADDRESS OF TEST 1
025112  004474          .WORD   TST2            ;STARTING ADDRESS OF TEST 2
025114  005256          .WORD   TST3            ;STARTING ADDRESS OF TEST 3
025116  006040          .WORD   TST4            ;STARTING ADDRESS OF TEST 4
025120  006756          .WORD   TST5            ;STARTING ADDRESS OF TEST 5
025122  007674          .WORD   TST6            ;STARTING ADDRESS OF TEST 6
025124  010742          .WORD   TST7            ;STARTING ADDRESS OF TEST 7
025126  012052          .WORD   TST10           ;STARTING ADDRESS OF TEST 10

                        ;ATTENTION BIT TABLE

025130  001    ATABIT: .BYTE   1               ;ATTENTION BIT FOR DRIVE 0
025131  002            .BYTE   2               ;ATTENTION BIT FOR DRIVE 1
025132  004            .BYTE   4               ;ATTENTION BIT FOR DRIVE 2
025133  010            .BYTE   10              ;ATTENTION BIT FOR DRIVE 3
025134  020            .BYTE   20              ;ATTENTION BIT FOR DRIVE 4
025135  040            .BYTE   40              ;ATTENTION BIT FOR DRIVE 5
025136  100            .BYTE   100             ;ATTENTION BIT FOR DRIVE 6
025137  200            .BYTE   200             ;ATTENTION BIT FOR DRIVE 7

025140  000011 MAXTN:  .WORD   11              ;MAXIMUM TEST NUMBER

        000200  .END    200
```

```
ADDRIS  020072
ADRERR  017655
ADE     = 001000
ASR1    001232
ATA     = 100000
ATABIT  025130
AT0     = 000001
AT1     = 000002
AT2     = 000004
AT3     = 000010
AT4     = 000020
AT5     = 000040
AT6     = 000100
AT7     = 000200
A16     = 000400
A17     = 001000
BADNO   020043
BADTMO  001706
BAI     = 000010
BIT0    = 000001
BIT00   = 000001
BIT01   = 000002
BIT02   = 000004
BIT03   = 000010
BIT04   = 000020
BIT05   = 000040
BIT06   = 000100
BIT07   = 000200
BIT08   = 000400
BIT09   = 001000
BIT1    = 000002
BIT10   = 002000
BIT11   = 004000
BIT12   = 010000
BIT13   = 020000
BIT14   = 040000
BIT15   = 100000
BIT2    = 000004
BIT3    = 000010
BIT4    = 000020
BIT5    = 000040
BIT6    = 000100
BIT7    = 000200
BIT8    = 000400
BIT9    = 001000
BPTVEC  = 000014
CHANGE  002774
CHGADR  001270
CKCLK   013406
CKCLK1  013456
CKCLK2  013520
CKCLK3  013530
CKERR   001244
CKSWR   = 104407
CLOCK   013540
CLR     = 000040
CONTUE  020316
CPSAVE  014524
CR      = 000015
CRLF    = 000200
CYCLED  020367
CYCLEU  020407
DCK     = 100000
DDISP   = 177570
DEI     = 000040
DFF20   = 000002
DF1     025034
DF14    025063
DF17    025071
DF2     025041
DF25    025073
DF30    025074
DF34    025103
DF5     025047
DF6     025054
DF7     025057
DH1     023244
DH13    023534
DH14    023611
DH17    023730
DH2     023315
DH24    023747
DH25    024052
DH30    024061
DH34    024203
DH35    024277
DH37    024374
DH40    024516
DH5     023371
DH6     023440
DH7     023467
DIGB    - 000004
DISPLA  001142
DISPRE  000174
DLT     = 100000
DL64    = 000020
DMD     = 000001
DPE     = 000010
DPR     = 000400
DRQ     = 004000
DRY     = 000200
DSWR    = 177570
DTE     = 010000
DT00    = 000001
DT01    = 000002
DT02    = 000004
DT03    = 000010
DT04    = 000020
DT05    = 000040
DT06    = 000100
DT07    = 000200
DT08    - 000400
DT1     024614
DT13    024704
DT14    024714
DT17    024732
DT2     024630
DT24    024740
DT25    024752
DT30    024756
DT34    024776
DT35    025010
DT40    025022
DT5     024646
DT6     024662
DT7     024672
DVA     = 004000
DVC     - 000200
ECH     - 000100
ECI     = 004000
EMTVEC  = 000030
EM1     020501
EM10    021065
EM11    021153
EM12    021220
EM13    021267
EM14    021354
EM15    021436
EM16    021522
EM17    021641
EM2     020547
EM20    021724
EM21    022007
EM22    022060
EM23    022156
EM24    022251
EM25    022331
EM3     020570
EM30    022374
EM31    022425
EM32    022473
EM33    022523
EM34    022607
EM35    022656
EM36    022743
EM37    023030
EM4     020642
EM40    023111
EM41    023166
EM5     020664
EM6     020746
EM7     021020
ENTERA  017626
ERR     - 040000
ERROR   = 104000
ERRVEC  = 000004
EXEC    002622
FER     - 000020
FMT16   = 010000
F0      000002
F1      - 000004
F2      = 000010
F3      = 000020
F4      = 000040
GO      = 000001
GRV     = 000010
GTSWR   = 104406
HCE     = 000200
HCI     = 002000
HCRC    = 000400
HT      = 000011
IAE     = 002000
IBSAVE  014526
IE      = 000100
ILF     = 000001
ILR     = 000002
IOTVEC  = 000020
IR      = 000100
IVC     = 010000
KYBCTL  001266
LBC     = 002000
LBT     = 002000
LF      = 000012
LSC     = 004000
MAXTN   025140
MCPE    = 020000
MDPE    - 000400
MOH     = 020000
MOL     = 010000
MXF     = 001000
NBA     = 100000
NED     = 010000
NEM     = 004000
NOATA   = 000000
NOCLOC  017756
NOSEIZ  001246
NTRH    020127
OFD     = 000200
OPE     = 020000
OPI     = 020000
OPPRT   001240
OR      = 000200
PAR     = 000010
PAT     = 000020
PFECH   014726
PFECH1  014736
PFECH2  015020
PFECH3  015052
PFECH4  015062
PFTSTN  015066
PGE     002000
PGM     - 001000
PIP     - 020000
PIRQ    = 177772
PIRQVE  - 000240
PORTA   001224
PORTAI  017700
PORTB   001226
PORTBI  017727
PORTC   001230
PR0     = 000000
PR1     = 000040
PR2     = 000100
PR3     = 000140
PR4     = 000200
PR5     = 000240
PR6     = 000300
PR7     = 000340
PS      = 177776
PSEL    = 002000
PSW     = 177776
PTNBR   001234
PWRVEC  = 000024
RDCHR   = 104410
RDLIN   = 104411
RDOCT   = 104412
RDY     = 000200
RELERR  001250
RELOK   - 000001
RESREG  = 104414
RESVEC  = 000010
RMAS    = 000016
RMBA    = 000004
RMCS1   = 000000
RMCS2   = 000010
RMDA    = 000006
RMDB    = 000022
RMDC    = 000034
RMDS    = 000012
RMDT    = 000026
RMEC1   = 000044
RMEC2   = 000046
RMER1   = 000014
RMER2   = 000042
RMLA    = 000020
RMMR1   = 000024
RMMR2   - 000040
RMOF    = 000032
RMR     = 000004
RMSN    = 000030
RMWC    = 000002
R6      = %000006
R7      = %000007
SAVREG  = 104413
SC      = 100000
SCOPE   = 000004
SC0     = 000100
SC1     = 000200
SC2     = 000400
SC3     = 001000
SC4     = 002000
SEIZPT  001236
SKI     = 100000
STACK   - 001100
START   001766
```

SYMBOL TABLE

```
START1  001774          TEST3   005314          WCE   - 040000          $GTSWR  016434          $SVPC = 000210
START2  002002          TEST4   006076          WCF   - 000040          $GT42P  013352          $SWR  = 166000
STKLMT= 177774          TEST5   007014          WLE   = 004000          $HD   = 000000          $SWRMK= 000000
STOP    014154          TEST6   007732          WRL   = 004000          $HIUCT  017442          $TIMES  001176
SWR     001140          TEST7   011000          $AUTOB  001134          $ICNT   001104          $TKB    001146
SWREG   000176          TIME    001252          $BDADR  001122          $INTAG  001135          $TKCNT  016034
SWTCHA  020224          TIMEA   001256          $BDDAT  001126          $ITEMB  001114          $TKINT  016044
SWTCHB  020261          TIMEAP  001260          $BELL   001202          $LF     001210          $TKQEN= 016043
SWTCHN  020156          TIMEB   001262          $CHARC  015356          $LKCSB  001214          $TKQIN  016036
SW0   = 000001          TIMEBP  001264          $CKSWR  016344          $LKCSR  001212          $TKQOU  016040
SW00  - 000001          TKVEC = 000060          $CMTAG  001100          $LKS    001220          $TKQSR  016042
SW01  = 000002          TOLER   013572          $CM1  - 000001          $LLVEC  001222          $TKS    001144
SW02  - 000004          TPVEC = 000064          $CM2  = 000002          $LPADR  001106          $TKSRV  016114
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  017242          $MNEW   017272          $TMP2   001170
SW06    000100          TSTADR  025110          $CNTLG  017254          $MSWR   017261          $TMP3   001172
SW07    000200          TSTNUM  001242          $CNTLU  017247          $MXCNT  014152          $TMP4   001174
SW08    000400          TST1    003122          $CRLF   001207          $NULL   001154          $TN   = 000012
SW09    001000          TST1AA  003104          $DBLK   016024          $NWTST- 000000          $TPB    001152
SW1   - 000002          TST10   012052          $DOAGN  013376          $OCNT   015604          $TPFLG  001157
SW10    002000          TST11   013156          $DTBL   016014          $OMODE  015606          $TPS    001150
SW11    004000          TST2    004474          $ENDAD  013366          $OVER   014136          $TRAP   017540
SW12  = 010000          TST3    005256          $ENDCT  013224          $PASS   001100          $TRAP2  017562
SW13  - 020000          TST4    006040          $ENULL  013402          $QUES   001206          $TRP  - 000015
SW14  - 040000          TST5    006756          $EOP    013160          $RDCHR  016706          $TRPAD  017574
SW15  - 100000          TST6    007674          $EOPCT  013216          $RDLIN  016776          $TSTNM  001102
SW2   - 000004          TST7    010742          $ERFLG  001103          $RDOCT  017304          $TTYIN  017232
SW3   - 000010          TYPDS = 104405          $ERMAX  001115          $RDSZ - 000010          $TYPDS  015610
SW4   - 000020          TYPE  = 104401          $ERROR  014202          $REGAD  001160          $TYPE   015070
SW5   - 000040          TYPOC = 104402          $ERRPC  001116          $REGO   001162          $TYPEC  015240
SW6   - 000100          TYPON - 104404          $ERRTB  001276          $RESRE  017502          $TYPEX  015360
SW7   - 000200          TYPOS = 104403          $ERRTY  014530          $RMADR  001272          $TYPOC  015406
SW8   - 000400          UNS   = 040000          $ERTTL  001112          $RMVEC  001274          $TYPON  015422
SW9   - 001000          UPE   = 020000          $ESCAP  001200          $RTNAD  013400          $TYPOS  015362
TAP   = 040000          U0    = 000001          $FILLC  001156          $SAVRE  017444          $XOFF - 000023
TBITVE- 000014          U1    = 000002          $FILLS  001155          $SCOPE  013620          $XON  = 000021
TESTNO  020023          U3    = 000004          $GDADR  001120          $SETUP= 000127          $XTSTR  013642
TEST1   003160          VV    = 000100          $GDDAT  001124          $STUP   177777          $$GET4- 000000
TEST10  012110          VVSET - 000001          $GET42  013356          $SVLAD  014126          $OFILL  015605
TEST2   004532          WATCH   001254

. ABS.  025142      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  54784 WORDS  ( 214 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  70 PAGES
(ZRMSB,BIN,CZRMSB/C=(ZRMSB.DOC,CZRMSB,SYSMAC/M
```

$$GET4	11-1	11-1#												
$OFILL	17-1	17-1#	17-1*	17-1*										
$40CAT	13-1	14-1												
$AUTOB	6-0#	9-29*	19-1	19-1	19-1									
$BDADR	6-0#	10-33*	10-33*	10-33*	10-33*	10-40*	10-40*	10-40*	10-40*	10-48*	10-48*	10-48*	10-48*	10-48*
	10-48*	10-48*	10-48*	10-98*	10-98*	10-98*	10-98*	10-99*	10-99*	10-99*	10-99*	10-157*	10-157*	10-157*
	10-157*	10-158*	10-158*	10-158*	10-158*	10-185*	10-185*	10-190*	10-190*	10-322*	10-322*	10-322*	10-322*	10-322*
	10-322*	10-322*	10-322*	10-323*	10-323*	10-323*	10-323*	10-323*	10-323*	10-323*	10-323*	25-27	25-28	25-29
	25-33	25-37												
$BDDAT	6-0#	10-33	10-33	10-33	10-33	10-33*	10-33*	10-40	10-40	10-40	10-40	10-40	10-40	10-40*
	10-40*	10-48	10-48	10-48	10-48	10-48	10-48	10-48	10-48	10-48*	10-48*	10-48*	10-48*	10-55*
	10-56	10-98	10-98	10-98*	10-98*	10-98*	10-98*	10-98*	10-99	10-99	10-99*	10-99*	10-99*	10-99*
	10-99*	10-157	10-157	10-157*	10-157*	10-157*	10-157*	10-157*	10-157*	10-158	10-158	10-158*	10-158*	10-158*
	10-158*	10-158*	10-158*	10-185*	10-185*	10-185*	10-185*	10-190*	10-190*	10-190*	10-190*	10-322	10-322	10-322
	10-322	10-322	10-322	10-322	10-322	10-322*	10-322*	10-322*	10-322*	10-323	10-323	10-323	10-323	10-323
	10-323	10-323	10-323	10-323*	10-323*	10-323*	10-323*	25-27	25-28	25-29	25-33	25-37		
$BELL	6-0#	14-1	14-1	14-1	19-1	19-1	19-1							
$CHARC	16-1	16-1#	16-1*	16-1*	16-1*									
$CKSWR	19-1#	22-1	22-1											
$CM1	6-0	6-0	6-0#	6-0#										
$CM2	6-0	6-0	6-0#	6-0#										
$CM3	6-0	6-0	6-0#											
$CM4	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0#	6-0#	6-0#	6-0#
	6-0#	6-0#												
$CMTAG	6-0#	9-24	9-24	9-24	9-24	9-24	9-24	9-24						
$CNTLC	19-1	19-1	19-1	19-1	19-1#									
$CNTLG	19-1	19-1#												
$CNTLU	19-1	19-1	19-1#											
$CRLF	6-0#	9-50	9-69	9-105	11-1	14-1	14-1	14-1	15-1	15-1	15-1	15-1	16-1	16-1
	16-1	19-1	19-1	19-1	19-1	20-1	20-1							
$DBLK	18-1	18-1	18-1#											
$DOAGN	11-1	11-1	11-1#											
$DTBL	18-1	18-1#												
$ENDAD	5-5	9-29	11-1#	14-1										
$ENDCT	9-24	11-1#												
$ENULL	11-1#													
$EOP	10-323	11-1#												
$EOPCT	9-24*	11-1	11-1#											
$ERFLG	6-0#	10-322	10-322*	10-323	10-323*	13-1	13-1	13-1	13-1	13-1*	14-1	14-1	14-1*	
$ERMAX	6-0#	9-24*	9-77*	13-1	13-1									
$ERROR	9-24	14-1#												
$ERRPC	6-0#	14-1	14-1	14-1	14-1*	14-1*	15-1	15-1	25-27	25-28	25-29	25-30	25-31	25-32
	25-33	25-34	25-35	25-37	25-38	25-39	25-40							
$ERRTB	8-0#	15-1												
$ERRTY	14-1	15-1#												
$ERTTL	6-0#	11-1	11-1	11-1*	14-1	14-1	14-1*							
$ESCAP	6-0#	9-24*												
$FILLC	6-0#	16-1	16-1	16-1										
$FILLS	6-0#	16-1	16-1											
$GDADR	6-0#													
$GDDAT	6-0#	10-33	10-33	10-33*	10-33*	10-33*	10-33*	10-40	10-40	10-40*	10-40*	10-48	10-48	10-48
	10-48	10-48*	10-48*	10-48*	10-48*	10-48*	10-48*	10-48*	10-48*	10-53*	10-56	10-98	10-98	10-98
	10-98*	10-98*	10-98*	10-99	10-99	10-99	10-99*	10-99*	10-99*	10-157	10-157	10-157	10-157	10-157
	10-157*	10-157*	10-157*	10-158	10-158	10-158	10-158	10-158	10-158*	10-158*	10-158*	10-185	10-185	10-185*
	10-190	10-190	10-190*	10-322	10-322	10-322	10-322	10-322*	10-322*	10-322*	10-322*	10-322*	10-322*	10-322*
	10-322*	10-323	10-323	10-323	10-323	10-323*	10-323*	10-323*	10-323*	10-323*	10-323*	10-323*	10-323*	25-28

	25-29	25-37												
$GET42	11-1#													
$GT42P	11-1	11-1#												
$GTSWR	19-1#	22-1	22-1											
$HD	4-516	4-516	4-516											
$HIOCT	20-1#	20-1*												
$ICNT	6-0#	9-93*	13-1	13-1	13-1	13-1*	13-1*							
$INTAG	6-0#	19-1	19-1	19-1	19-1	19-1*								
$ITEMB	6-0#	14-1	14-1	14-1	14-1	14-1*	14-1*	15-1						
$LF	6-0#	14-1	14-1	16-1	16-1	19-1	19-1	19-1	20-1	20-1				
$LKCSB	7-0#	12-11*												
$LKCSR	7-0#	12-7	12-12*											
$LKS	7-0#	12-16	12-20*											
$LLVEC	7-0#	12-17												
$LPADR	6-0#	9-24*	9-78*	10-20*	10-98*	10-99*	10-157*	10-158*	10-178*	10-322*	10-323*	13-1	13-1	13-1
	13-1*													
$LPERR	6-0#	9-24*	9-79*	10-20*	10-98*	10-99*	10-157*	10-158*	10-178*	10-322	10-322*	10-323	10-323*	
$LPVEC	7-0#	12-8												
$MAIL	9-24	9-29	13-1	14-1	16-1									
$MNEW	19-1	19-1#												
$MSWR	19-1	19-1#												
$MXCNT	13-1	13-1	13-1	13-1#										
$NULL	6-0#	16-1	16-1	16-1										
$NWTST	10-20	10-20	10-20#	10-20#	10-98	10-98	10-98#	10-98#	10-99	10-99	10-99#	10-99#	10-157	10-157
	10-157#	10-157#	10-158	10-158	10-158#	10-158#	10-178	10-178	10-178#	10-178#	10-322	10-322	10-322#	10-322#
	10-323	10-323	10-323#	10-323#	10-327#									
$OCNT	17-1#	17-1*	17-1*											
$OMODF	17-1	17-1#	17-1*	17-1*	17-1*	17-1*								
$OVER	13-1	13-1	13-1	13-1#										
$PASS	6-0#	9-76*	11-1	11-1	11-1	11-1*	11-1*	13-1	13-1	13-1				
$QUES	6-0#	14-1	14-1	16-1	16-1	19-1	19-1	19-1	19-1	20-1	20-1	20-1		
$R2A	22-1													
$RDCHR	19-1#	22-1	22-1											
$RDDEC	22-1													
$RDLIN	19-1#	22-1	22-1											
$RDOCT	20-1#	22-1	22-1											
$RDSZ	19-1	19-1#												
$REGO	6-0#													
$REGAD	6-0#													
$RESRE	21-1#	22-1												
$RMADR	7-0#	9-70	9-103	9-110*	9-113	10-17	25-36							
$RMVEC	7-0#													
$RTNAD	11-1#													
$SAVRE	21-1#	22-1	22-1											
$SCOPE	9-24	13-1#												
$SETUP	4-705	4-705	4-705	4-705	4-705	4-705#	4-705#	4-705#	4-705#	4-705#	4-705#	9-24	9-24	9-24
	9-24	9-24	9-24	9-24	9-24	9-24	9-24	9-24	9-24	9-29	9-29	9-29	11-1	11-1
	13-1	14-1	14-1	14-1	14-1	19-1	19-1	19-1	19-1	19-1				
$STUP	4-705	4-705	4-705	4-705	4-705	4-705#	4-705#	4-705#	4-705#	4-705#	4-705#	4-705#	4-705#	4-705#
	4-705#													
$SVLAD	13-1	13-1#												
$SVPC	5-5	5-5#												
$SWR	4-512#	4-516	4-517	4-517	4-517	4-517	4-517	4-517	4-517	4-517	6-0	6-0	6-0	9-24
	9-24	9-24	9-24	9-24	10-20	10-98	10-99	10-157	10-158	10-178	10-322	10-323	10-327	11-1
	11-1	11-1	11-1	11-1	12-48#	13-1	13-1	13-1	13-1	13-1	13-1	13-1	13-1	13-1
	13-1	13-1	13-1	13-1	13-1	13-1	13-1	13-1	14-1	14-1	14-1	14-1	14-1	14-1

```
$SWRMK  13-1
$TESTN  15-1
$TIMES   6-0#    9-24*   10-20*  10-98*  10-99*  10-157*  10-158*  10-178*  10-322*  10-322*  10-323*  10-323*  11-1*   13-1
        13-1     13-1    13-1*   13-1*
$TKB     6-0#    16-1    16-1    19-1    19-1    19-1     19-1     19-1     19-1     19-1
$TKCNT  19-1     19-1    19-1#   19-1*   19-1*   19-1*
$TKINT   9-31    9-74    19-1    19-1    19-1#
$TKQEN  19-1     19-1    19-1#
$TKQIN  19-1     19-1    19-1#   19-1*   19-1*   19-1*    19-1*
$TKQOU  19-1     19-1    19-1#   19-1*   19-1*   19-1*
$TKQSR  19-1     19-1    19-1    19-1#
$TKS     6-0#    16-1    16-1    19-1    19-1    19-1     19-1     19-1     19-1*    19-1*    19-1*    19-1*    19-1*   19-1*
$TKSRV  19-1     19-1#
$TMP0    6-0#    10-33   10-33   10-33*  10-33*  10-33*   10-33*   10-48    10-48    10-48    10-48    10-48*   10-48*  10-48*
        10-48*   10-48*  10-48*  10-48*  10-48*  10-98    10-98    10-98    10-98    10-98*   10-98*   10-98*   10-98*  10-99
        10-99    10-99    10-99   10-99*  10-99*  10-99*   10-99*   10-157   10-157   10-157   10-157   10-157*  10-157* 10-157*
        10-157*  10-158   10-158  10-158  10-158  10-158*  10-158*  10-158*  10-158*  10-185   10-185   10-185   10-185* 10-185*
        10-190   10-190   10-190  10-190* 10-190* 10-322   10-322   10-322   10-322   10-322*  10-322*  10-322*  10-322* 10-322*
        10-322*  10-322*  10-322* 10-323  10-323  10-323   10-323   10-323*  10-323*  10-323*  10-323*  10-323*  10-323* 10-323*
        10-323*
$TMP1    6-0#    10-98    10-98   10-98*  10-98*  10-99    10-99    10-99*   10-99*   10-157   10-157   10-157   10-157* 10-157*
        10-157*  10-157*  10-158  10-158  10-158  10-158*  10-158*  10-158*  10-158*  10-185   10-185   10-185*  10-185* 10-190
        10-190   10-190*  10-190*
$TMP2    6-0#    10-98    10-98   10-98   10-98   10-98*   10-98*   10-99    10-99    10-99    10-99    10-99*   10-99*  10-157
        10-157   10-157   10-157  10-157* 10-157* 10-158   10-158   10-158   10-158   10-158*  10-158*  10-185   10-185  10-185
        10-185   10-185*  10-185* 10-190  10-190  10-190   10-190   10-190*  10-190*  25-38
$TMP3    6-0#    10-98    10-98   10-98   10-98   10-98*   10-98*   10-99    10-99    10-99    10-99*   10-99*   10-157
        10-157   10-157   10-157  10-157* 10-157* 10-158   10-158   10-158   10-158   10-158*  10-158*  10-185   10-185  10-185
        10-185   10-185*  10-185* 10-190  10-190  10-190   10-190   10-190*  10-190*  25-38
$TMP4    6-0#    10-33    10-33   10-33*  10-33*  10-33*   10-33*   10-48    10-48    10-48    10-48    10-48*   10-48*  10-48*
        10-48*   10-48*   10-48*  10-48*  10-48*  10-98    10-98*   10-98*   10-99    10-99*   10-99*   10-322   10-322  10-322
        10-322   10-322*  10-322* 10-322* 10-322* 10-322*  10-322*  10-322*  10-322*  10-323   10-323   10-323   10-323  10-323*
        10-323*  10-323*  10-323* 10-323* 10-323* 10-323*  10-323*
$TN      4-513#   4-516   10-20   10-20   10-20   10-20#   10-98    10-98    10-98    10-98#   10-99    10-99    10-99   10-99#
        10-157   10-157   10-157  10-157# 10-158  10-158   10-158   10-158#  10-178   10-178   10-178#  10-322   10-322
        10-322   10-322   10-322# 10-323  10-323  10-323   10-323#  10-327   10-327   10-327#  26-20
$TPB     6-0#    16-1     16-1    16-1*
$TPFLG   6-0#    16-1     16-1    16-1
$TPS     6-0#    16-1     16-1    16-1
$TRAP    9-24    22-1#
$TRAP2  22-1     22-1#
$TRP    22-1     22-1     22-1    22-1    22-1    22-1     22-1     22-1     22-1     22-1     22-1     22-1     22-1    22-1
        22-1     22-1     22-1    22-1    22-1    22-1     22-1     22-1     22-1     22-1     22-1     22-1     22-1    22-1
        22-1     22-1     22-1    22-1    22-1    22-1     22-1     22-1     22-1     22-1     22-1     22-1     22-1    22-1
        22-1     22-1     22-1    22-1    22-1    22-1     22-1#    22-1#    22-1#    22-1#    22-1#    22-1#    22-1#   22-1#
        22-1#    22-1#    22-1#   22-1#   22-1#
$TRPAD  22-1     22-1#
$TSTNM   6-0#    10-20*   10-98*  10-99*  10-157*  10-158*  10-178*  10-322*  10-323*  11-1*    13-1     13-1     13-1    13-1
        13-1*    14-1     14-1    14-1    14-1     15-1
$TTYIN  19-1     19-1     19-1    19-1    19-1     19-1#
$TYPBN  22-1
$TYPDS  18-1#    22-1     22-1
$TYPE   16-1#    22-1     22-1
$TYPEC  16-1     16-1     16-1    16-1#   19-1
$TYPEX  16-1     16-1     16-1#
$TYPOC  17-1#    22-1     22-1
```

STYPON 17-1 17-1# 22-1

SEG 0083

```
DT01    4-633#
DT02    4-634#
DT03    4-635#
DT04    4-636#
DT05    4-637#
DT06    4-638#
DT07    4-639#
DT08    4-640#
DT'     8-6      8-20     8-55     8-62     8-69     8-90     8-119    8-126    8-133    8-184    25-27#
DT13    8-76     8-98     25-32#
DT14    8-83     25-33#
DT17    8-105    8-112    25-34#
DT2     8-13     8-27     8-170    8-177    25-28#
DT24    8-140    25-35#
DT25    8-147    25-36#
DT30    8-163    8-212    8-226    25-37#
DT34    8-191    25-38#
DT35    8-198    8-205    25-39#
DT40    8-219    25-40#
DT5     8-34     25-29#
DT6     8-41     25-30#
DT7     8-48     25-31#
DTE     4-607#
DVA     4-572#
DVC     4-657#
ECH     4-601#
ECI     4-668#
EM1     8-4      24-3#
EM10    8-53     24-10#
EM11    8-60     24-11#
EM12    8-67     24-12#
EM13    8-74     24-13#
EM14    8-81     24-14#
EM15    8-88     24-15#
EM16    8-95     24-16#
EM17    8-103    24-18#
EM2     8-11     24-4#
EM20    8-110    24-19#
EM21    8-117    24-20#
EM22    8-124    24-21#
EM23    3-131    24-22#
EM24    8-138    24-23#
EM25    8-145    24-24#
EM3     8-18     24-5#
EM30    8-161    24-25#
EM31    8-168    24-26#
EM32    8-175    24-27#
EM33    8-182    24-28#
EM34    8-189    24-29#
EM35    8-196    24-30#
EM36    8-203    24-31#
EM37    8-210    24-32#
EM4     8-25     24-6#
EM40    8-217    24-33#
EM41    8-224    24-34#
EM5     8-32     24-7#
```

```
OPPRT   7-0#    10-157*  10-158*  25-40
OR      4-549#
PAR     4-598#
PAT     4-546#
PFECH   15-1    15-1#
PFECH1  15-1    15-1#
PFECH2  15-1    15-1#
PFECH3  15-1    15-1#
PFECH4  15-1    15-1#
PFTSTN  15-1    15-1#    15-1*
PGE     4-552#
PGM     4-585#  10-48    10-48    10-48    10-48    10-48    10-48    10-98    10-98    10-99    10-99    10-157   10-157   10-158
        10-158  10-185   10-190
PIP     4-589#
PIRO    4-519#
PIROVE  4-519#
PORTA   7-0#     9-35*    9-36     9-40     9-43     9-47     9-51     9-54     10-33    10-33    10-40    10-40    10-48    10-48
        10-52   10-98    10-98    10-98    10-98    10-98    10-98    10-98    10-98    10-99    10-99    10-99    10-99    10-157
        10-157  10-157   10-157   10-157   10-157   10-157   10-157   10-158   10-158   10-158   10-158   10-158   10-158   10-158
        10-158  10-158   10-180   10-185   10-185   10-185   10-185   10-190   10-190   10-190   10-190   10-322   10-322   10-322
        10-322  10-322   10-322   10-322   10-323   10-323   10-323   10-323   10-323   10-323
PORTAI  9-46    23-5#
PORTB   7-0#     9-40*    9-41*    9-42*    9-45*    9-49     10-33    10-33    10-40    10-40    10-48    10-48    10-54    10-98
        10-98   10-98    10-98    10-99    10-99    10-99    10-99    10-99    10-99    10-99    10-99    10-157   10-157   10-157
        10-157  10-157   10-157   10-157   10-157   10-157   10-158   10-158   10-158   10-158   10-158   10-158   10-158   10-158
        10-180  10-185   10-185   10-185   10-185   10-190   10-190   10-190   10-190   10-322   10-322   10-322   10-322   10-322
        10-322  10-323   10-323   10-323   10-323   10-323   10-323   10-323
PORTBI  9-48    23-6#
PORTC   7-0#     9-51*    9-52*    9-53*
PRO     4-519#
PR1     4-519#
PR2     4-519#
PR3     4-519#  13-5
PR4     4-519#
PR5     4-519#  10-18    13-9
PR6     4-519#  9-27
PR7     4-519#
PS      4-519   4-519#   9-68*
PSEL    4-529#
PSW     4-519#
PTNBR   7-0#     10-33*   10-33*   10-40*   10-40*   10-48*   10-48*   10-98*   10-98*   10-98*   10-98*   10-98*   10-98*   10-99*
        10-99*  10-99*   10-99*   10-99*   10-99*   10-157*  10-157*  10-157*  10-157*  10-157*  10-157*  10-157*  10-158*  10-158*
        10-158* 10-158*  10-158*  10-158*  10-158*  10-185*  10-185*  10-185*  10-185*  10-190*  10-190*  10-190*  10-190*  10-322*
        10-322* 10-322*  10-322*  10-322*  10-322*  10-323*  10-323*  10-323*  10-323*  10-323*  10-323*  25-27    25-28    25-30
        25-31   25-33    25-35    25-37    25-39
PWRVEC  4-519#
R6      4-519#  9-24     9-24*    9-24*
R7      4-519#
RDCHR   19-1    22-1#
RDLIN   20-1    22-1#
RDOCT   9-34    9-81     9-107    22-1#
RDY     4-526#
RELERR  7-0#    10-98*   10-98*   10-99*   10-99*   10-157*  10-157*  10-158*  10-158*  10-185*  10-185*  10-190*  10-190*
RELOK   8-231#  10-98    10-99    10-102#  10-157   10-158   10-185   10-190
RFSREG  22-1#
RFSVEC  4-519#
```

RMBA	4-687#													
RMCS1	4-685#	10-98*	10-98*	10-98*	10-99*	10-99*	10-99*	10-157*	10-157*	10-158*	10-158*	10-180*	10-180*	10-180*
	10-180*	10-185*	10-185*	10-190*	10-190*	10-322*	10-322*	10-322*	10-322*	10-322*	10-322*	10-322*	10-323*	10-323*
	10-323*	10-323*	10-323*	10-323*	10-323*									
RMCS2	4-689#	10-22*	10-33	10-33	10-33	10-33	10-33*	10-33*	10-33*	10-33*	10-40*	10-40*	10-48*	10-48*
	10-52*	10-54*	10-98*	10-98*	10-98*	10-98*	10-98*	10-98*	10-99*	10-99*	10-99*	10-99*	10-99*	10-99*
	10-157*	10-157*	10-157*	10-157*	10-157*	10-157*	10-157*	10-157*	10-158*	10-158*	10-158*	10-158*	10-158*	10-158*
	10-158*	10-158*	10-180*	10-180*	10-185*	10-185*	10-185*	10-185*	10-190*	10-190*	10-190*	10-190*	10-322	10-322
	10-322*	10-322*	10-322*	10-322*	10-322*	10-322*	10-323	10-323	10-323*	10-323*	10-323*	10-323*	10-323*	10-323*
RMDA	4-688#													
RMDB	4-694#													
RMDC	4-699#													
RMDS	4-690#	10-33	10-33	10-48	10-48	10-48	10-48	10-48	10-48	10-48	10-48	10-98	10-98	10-98
	10-98	10-98	10-99	10-99	10-99	10-99	10-99	10-157	10-157	10-157	10-157	10-157	10-157	10-157
	10-157*	10-158	10-158	10-158	10-158	10-158	10-158	10-158	10-158*	10-180*	10-180*	10-185	10-185	10-185
	10-190	10-190	10-190	10-322	10-322	10-322	10-322	10-322	10-322	10-322	10-322	10-322	10-322	10-322*
	10-322*	10-323	10-323	10-323	10-323	10-323	10-323	10-323	10-323	10-323	10-323	10-323*	10-323*	
RMDT	4-696#	10-40	10-40	10-40	10-40									
RMEC1	4-702#													
RMEC2	4-703#													
RMER1	4-691#													
RMER2	4-701#													
RMLA	4-693#													
RMMR1	4-695#													
RMMR2	4-700#													
RMOF	4-698#	10-98*	10-99*	10-322*	10-322*	10-323*	10-323*							
RMR	4-597#													
RMSN	4-697#	10-53	10-55											
RMWC	4-686#	9-114												
SAVREG	22-1#													
SC	4-532#													
SC0	4-648#													
SC1	4-649#													
SC2	4-650#													
SC3	4-651#													
SC4	4-652#													
SCOPE	4-519#	10-62	10-98	10-99	10-157	10-158	10-194	10-322	10-323	10-327				
SEIZPT	7-0#	10-157*	10-158*	10-322*	10-323*	25-32	25-33	25-35	25-37	25-39	25-40			
SKI	4-662#													
STACK	4-519#	9-24	9-71	10-20	10-98	10-99	10-157	10-158	10-178	10-322	10-323			
START	5-1	9-14#	19-1	19-1										
START1	5-3	9-17#												
START2	9-15	9-19#												
STKLMT	4-519#													
STOP	13-1	13-5#												
SW0	4-519#													
SW00	4-519	4-519#												
SW01	4-519	4-519#												
SW02	4-519	4-519#												
SW03	4-519	4-519#												
SW04	4-519	4-519#												
SW05	4-519	4-519#												
SW06	4-519	4-519#												
SW07	4-519	4-519#												
SW08	4-519	4-519#												
SW09	4-519	4-519#	10-322	10-323										

SW1 4-519W

SEQ 0095

CROSS REFERENCE TABLE (CREF V01-05)

SW10	4-519#													
SW11	4-519#													
SW12	4-519#													
SW13	4-519#													
SW14	4-519#	10-322	10-323											
SW15	4-519#	10-59												
SW2	4-519#													
SW3	4-519#													
SW4	4-519#													
SW5	4-519#													
SW6	4-519#													
SW7	4-519#													
SW8	4-519#													
SW9	4-519#													
SWR	6-0#	9-24	9-24	9-24*	9-24*	9-29	10-59	10-322	10-322	10-323	10-323	13-1	13-1	13-1
	13-1*	13-1*	13-1*	13-1*	14-1	14-1	14-1	14-1	19-1	19-1	19-1*			
SWREG	5-1#	9-24	9-29	19-1	19-1	19-1								
SWTCHA	10-181	10-322	23-13#											
SWTCHB	10-186	10-323	23-14#											
SWTCHN	10-193	10-322	10-323	23-12#										
TAP	4-643#													
TBITVE	4-519#													
TEST1	10-20	10-20	10-20#											
TEST10	10-323	10-323	10-323#											
TEST2	10-98	10-98	10-98#											
TEST3	10-99	10-99	10-99#											
TEST4	10-157	10-157	10-157#											
TEST5	10-158	10-158	10-158#											
TEST6	10-178	10-178	10-178#											
TEST7	10-322	10-322	10-322#											
TESTNO	9-80	23-8#												
TIME	7-0#	10-157	10-157*	10-158	10-158*	12-29*	25-31							
TIMEA	7-0#	9-58*	10-157	10-157	10-157*	10-157*								
TIMEAP	7-0#	9-58*	10-157*	10-157*										
TIMEB	7-0#	9-58*	10-158	10-158	10-158*	10-158*								
TIMEBP	7-0#	9-58*	10-158*	10-158*										
TKVEC	4-519#	19-1*	19-1*											
TOLER	10-157	10-158	12-39#											
TPVEC	4-519#													
TRAPVE	4-519#	9-24*	9-24*											
TRE	4-531#													
TRTVEC	4-519#													
TST1	9-84	10-20#	26-4											
TST10	10-322	10-323#	26-7											
TST11	10-327#													
TST1AA	10-17#	11-1												
TST2	10-98#	26-7												
TST3	10-99#	26-7												
TST4	10-157#	26-7												
TST5	10-158#	26-7												
TST6	10-178#	26-7												
TST7	10-322#	26-7												
TSTADR	9-91	26-4#												
TSTNUM	7-0#	14-1*	25-27	25-28	25-29	25-30	25-31	25-32	25-33	25-34	25-35	25-37	25-38	25-39
	25-40													
TYPDS	11-1	11-1	15-1	22-1#										

9-6 9-29 9-33 9-38 9-46 9-48 9-50 9-61 9-69 9-80 9-87 9-102 9-105 9-106

	10-181	10-182	10-186	10-187	10-193	10-322	10-322	10-322	10-322	10-322	10-322	10-323	10-323	10-323
	10-323	10-323	10-323	11-1	11-1	11-1	14-1	14-1	15-1	15-1	15-1	15-1	15-1	15-1
	15-1	16-1	17-1	18-1	19-1	19-1	19-1	19-1	19-1	19-1	19-1	19-1	19-1	19-1
	19-1	19-1	19-1	19-1	19-1	19-1	19-1	19-1	20-1	20-1	22-1#			
TYPOC	9-8	9-104	15-1	15-1	19-1	22-1#								
TYPON	22-1#													
TYPOS	9-47	9-49	22-1#											
U0	4-542#													
U1	4-543#													
U3	4-544#													
UNS	4-609#													
UPE	4-555#													
VV	4-582#	10-98	10-98	10-98	10-98	10-98	10-99	10-99	10-99	10-99	10-99	10-157	10-157	10-157
	10-157	10-158	10-158	10-158	10-158	10-185	10-185	10-185	10-190	10-190	10-190			
VVSET	8-230#	10-98	10-98	10-98	10-98	10-98	10-99	10-99	10-99	10-99	10-99	10-103#	10-157	10-157
	10-157	10-157	10-157	10-157	10-157	10-157	10-158	10-158	10-158	10-158	10-158	10-158	10-158	10-158
	10-185	10-185	10-185	10-185	10-185	10-185	10-190	10-190	10-190	10-190	10-190	10-190		
WATCH	7-0#	10-157	10-157*	10-158	10-158*	10-322	10-322*	10-323	10-323*	12-30	2-32*	12-34*		
WCE	4-556#													
WCF	4-600#													
WLE	4-606#													
WRL	4-587#													

$$CMRE	5-41#	6-0												
$$CMTM	5-41#	6-0	6-0	6-0	6-0	6-0								
$$ESCA	4-519#													
$$NEWT	4-519#	10-20	10-98	10-99	10-157	10-158	10-178	10-322	10-323	10-327				
$$SET	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1#	
$$SKIP	4-519#	10-322												
.$ACT1	4-508#	5-5												
.$CATC	4-509#	5-1												
.$CMTA	4-509#	5-41												
.$EOP	4-509#	11-1												
.$ERRO	4-509#	14-1												
.$ERRT	4-508#	15-1												
.$RDOC	4-510#	20-1												
.$READ	4-510#	19-1												
.$SAVE	4-510#	21-1												
.$SCOP	4-509#	13-1												
.$TRAP	4-510#	22-1												
.$TYPD	4-510#	18-1												
.$TYPE	4-509#	16-1												
.$TYPO	4-510#	17-1												
.EQUAT	4-508#	4-519												
.HEADE	4-508#	4-516												
.SETUP	4-508#	4-705												
.SWRHI	4-508#	4-517												
.SWRLO	4-508#	4-517#												
AA	10-106#	10-157	10-158											
BB	10-64#	10-98	10-99											
CHECK	4-122#	10-33	10-33	10-40	10-40	10-48	10-48	10-48	10-48	10-98	10-99	10-322	10-322	10-322
	10-322	10-323	10-323	10-323	10-323									
CLRATA	4-417#	10-180												
COMMEN	4-519#													
ENDCOM	4-519#													
ERROR	4-519#	9-116	10-33	10-33	10-40	10-40	10-48	10-48	10-48	10-48	10-58	10-98	10-98	10-98
	10-98	10-98	10-99	10-99	10-99	10-99	10-99	10-157	10-157	10-157	10-157	10-157	10-157	10-157
	10-157	10-158	10-158	10-158	10-158	10-158	10-158	10-158	10-158	10-185	10-185	10-185	10-185	10-190
	10-190	10-190	10-190	10-322	10-322	10-322	10-322	10-323	10-323	10-323	10-323	13-1		
ESCAPE	4-519#													
FF	10-196#	10-322	10-323											
GETPRI	4-519#													
GETSWR	4-519#	9-29	9-29#											
MORETA	5-7#	6-0												
MSG	10-3#	10-20	10-98	10-98#	10-99	10-99#	10-157	10-157#	10-158	10-158#	10-161#	10-178	10-322	10-322#
	10-323	10-323#												
MULT	4-519#													
NEUTRA	4-273#	10-98	10-99	10-157	10-158	10-185	10-190							
NEWTST	4-519#	10-20	10-98	10-99	10-157	10-158	10-178	10-322	10-323	10-327				
NWTST	4-471#	10-20	10-98	10-99	10-157	10-158	10-178	10-322	10-323					
POP	4-519#	18-1	20-1	21-1										
PUSH	4-519#	18-1	20-1	21-1										
RELEAS	4-202#	10-98	10-99											
REPORT	4-519#													
SFIZE	4-32#	10-157	10-158											
SELECT	4-247#	10-33	10-33	10-40	10-40	10-48	10-48	10-98	10-98	10-99	10-99	10-157	10-157	10-157
	10-158	10-158	10-158	10-322	10-322	10-322	10-322	10-322	10-322	10-323	10-323	10-323	10-323	10-323
	10-323													
SETATA	4-450#													

SETPRI	4-519#	10-18	13-5	13-9	19-1									
SETTRA	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1	22-1#	
SETUP	4-519#	9-24												
SKIP	4-519#	10-322												
SLASH	4-519#													
STARS	4-519#	5-5	6-0	6-0	10-20	10-20	10-98	10-98	10-99	10-99	10-157	10-157	10-158	10-158
	10-178	10-178	10-322	10-322	10-323	10-323	10-325	10-327	11-1	13-1	14-1	15-1	16-1	17-1
	18-1	19-1	19-1	19-1	19-1	19-1	20-1	21-1	22-1					
SWITCH	4-373#													
SWRSU	4-519#	9-24	9-24#											
TIMER	4-497#	10-157	10-158											
TRMTRP	22-1#													
TYPBIN	4-519#													
TYPDEC	4-519#	11-1	11-1	15-1										
TYPNAM	4-519#	9-29												
TYPNUM	4-519#													
TYPOCS	4-519#	9-47	9-49											
TYPOCT	4-519#	15-1	15-1	19-1										
TYPTXT	4-519#	9-6	11-1	11-1										
USEREO	10-328#	11-1												