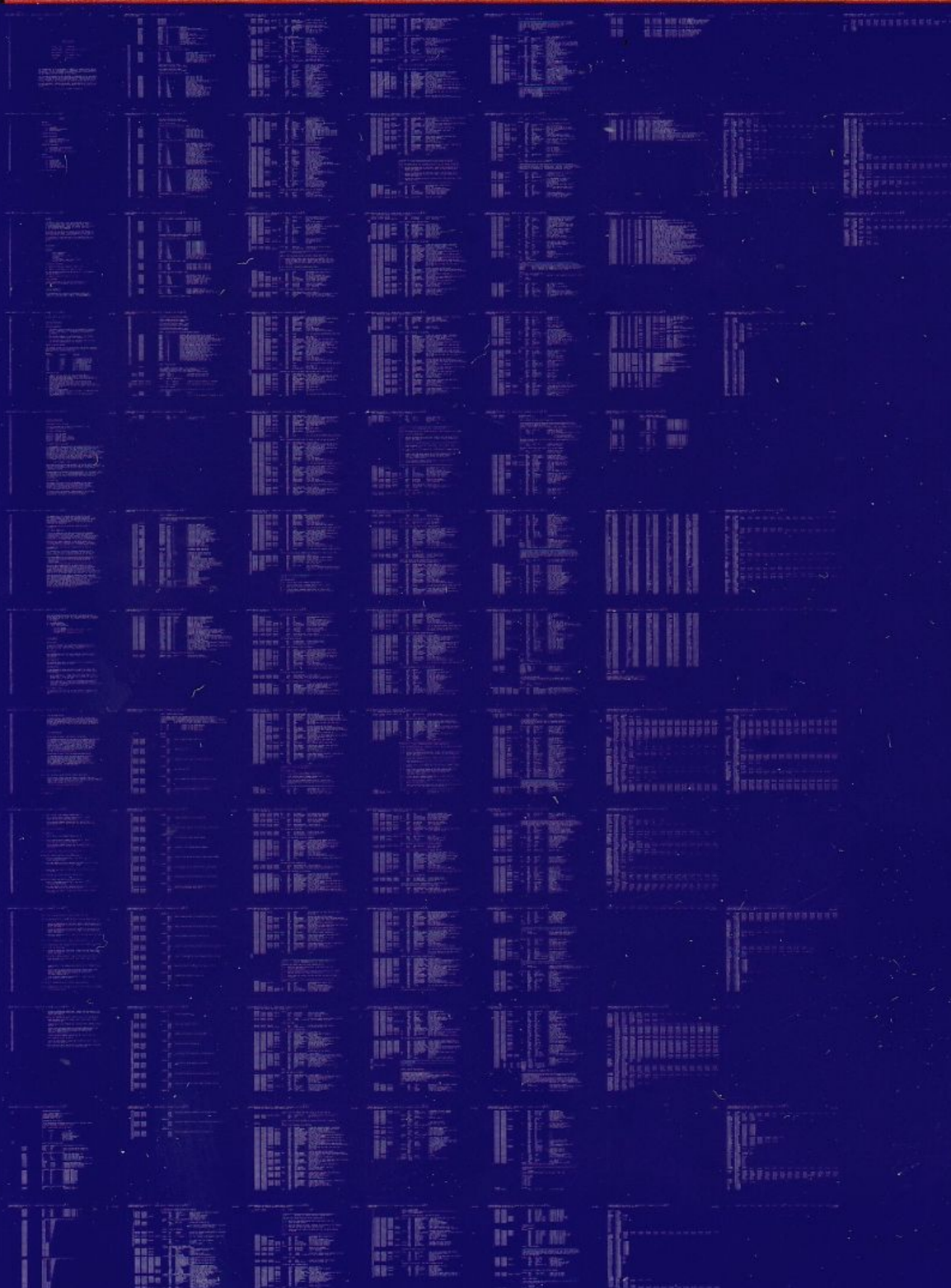
RM05/3/2 RM05/3/2 DU POR TST2 AH-F940A-MC
CZRMSA0 FICHE 1 OF 1



```
.REM    ^
```

IDENTIFICATION

PRODUCT CODE: AC-F939A-MC
PRODUCT NAME: CZRMSA0 RM05/3/2 DU POR TST 2
DATE CREATED: APRIL 1980
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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
16K OF MEMORY
KW11-L OR KW11-P CLOCK
TERMINAL
RH11 OR RH70
1 - DISK DRIVE (RM05, RM03 OR RM02)
RM DUAL PORT TEST CABLE

2.2 PREREQUISITE PROGRAMS

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

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

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

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 (DCL) ADDRESS.

C. THE PROGRAM CAN BE STARTED AT LOCATION 210(8) TO ALLOW THE RH11 OR RH70 ADDRESS TO BE CHANGED.

4.2 UNIBUS & VECTOR ADDRESS

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

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

4.3 OPERATOR ACTION

A. CONNECT THE DUAL PORT TEST CABLE BETWEEN BUS A & BUS B ON THE DRIVE BEING TESTED. (SEE SECTION 5.4)
B. LOAD THE PROGRAM INTO MEMORY IN THE PROCESSOR CONTROLLING THE MASSBUS USED FOR TESTING.
C. SWITCH THE 'PORT SELECT' SWITCH ON THE DRIVE TO BE TESTED TO THE 'A/B' POSITION. CYCLE THE DRIVE UP.
D. LOAD THE APPROPIATE STARTING ADDRESS (200(8) OR 24C(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 OPERATIONAL SWITCH SETTINGS

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

THE SWITCH SETTINGS ARE:

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

5.2 'SOFTWARE' SWITCH REGISTER

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34)
THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS
NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE
'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE
SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD
ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL
RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM
IS AT A HIGHER PRIORITY PROCESSING AN RM05/3/2 INTERRUPT. THE
'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE
TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

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

5.3 TEST SELECTION

INDIVIDUAL TESTS ARE SELECTED IN RESPONSE TO THE 'ENTER
TEST NUMBER:' MESSAGE. ANY VALID TEST NUMBER CAN BE
ENTERED. EACH ENTRY MUST BE TERMINATED BY A CARRIAGE
RETURN (CR). THE LOOP ON TEST SWITCH, SW<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 J8.

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 60 SECONDS PER PASS (DEPENDING ON OPERATOR INTERVENTION EFFICIENCY).

7.4 REQUIRED TESTS

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

A. TEST 2 AND TEST 3. THESE TESTS SET 'VV-A' AND 'VV-B' RESPECTIVLY. 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

```
343
344                                   A.  ISSUE A DRIVE CLEAR COMMAND THROUGH PORT A.
345
346                                   B.  ISSUE A READIN PRESET COMMAND THROUGH PORT A.  VERIFY
347                                         THAT THE 'VV' BIT IS SET FOR PORT A.
348
349                                   C.  ISSUE A RELEASE COMMAND THROUGH PORT A.  VERIFY THAT
350                                         THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION
351                                         BIT IS SET.
352
353
354                                 TEST 3  SET 'VV' FOR PORT B
355
356
357                                 SET VOLUME VALID
358
359                                   A.  ISSUE A DRIVE CLEAR COMMAND THROUGH PORT B.
360
361
362                                   B.  ISSUE A READIN PRESET COMMAND THROUGH PORT B.  VERIFY
363                                         THAT THE 'VV' BIT IS SET FOR PORT B.
364
365                                   C.  ISSUE A RELEASE COMMAND THROUGH PORT B.  VERIFY THAT
366                                         THE DRIVE RETURNED TO NEUTRAL AND THAT NEITHER ATTENTION
367                                         BIT IS SET.
368
369
370                                 TEST 4  MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A
371
372
373                                 MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT A
374
375                                   A.  WRITE 0'S INTO RMDS THROUGH PORT A AND VERIFY THAT THE
376                                         DRIVE HAS BEEN SEIZED.
377
378                                   B.  WAIT FOR TIMEOUT TO OCCUR.  MEASURE THE DURATION OF THE TIMEOUT
379                                         ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
380
381
382
383                                   C.  VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS
384                                         TO NEUTRAL
385
386
387                                 TEST 5  MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT B
388
389
390                                 MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT B
391
392                                   A.  WRITE 0'S INTO RMDS THROUGH PORT B AND VERIFY THAT THE
393                                         DRIVE HAS BEEN SEIZED.
394
395                                   B.  WAIT FOR TIMEOUT TO OCCUR.  MEASURE THE DURATION OF THE TIMEOUT
396                                         ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
397
398                                   C.  VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS
399                                         TO NEUTRAL
```

```
400
401
402                          TEST 6  TEST 'CONTROLLER SELECT' SWITCH, DRIVE CYCLED UP
403
404
405                          TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED UP).
406
407
408                            A.    SWITCH TO CONTROLLER 'A' POSITION.  VERIFY THAT THE DRIVE IS IN
409                                  NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
410                                  PORTS, ARE CORRECT.
411
412                            B.    SWITCH TO CONTROLLER 'B' POSITION. VERIFY THAT THE DRIVE IS IN
413                                  NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
414                                  PORTS, ARE CORRECT.
415
416                            C.  RETURN THE 'CONTROLLER SELECT' SWITCH TO THE 'A/B' POSITION.  VERIFY
417                                  THE DRIVE STATE.
418
419
420                          TEST 7  TEST 'CONTROLLER SELECT' SWITCH LOCKED ON PORT A
421
422
423                          TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).
424
425                            A.    CYCLE THE DRIVE DOWN.
426
427                            B.    SWITCH TO CONTROLLER A POSITION.  VERIFY THAT THE DRIVE IS IN
428                                  NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
429                                  PORTS, ARE CORRECT.
430
431                            C.  SWITCH THE 'CONTROLLER SELECT' SWITCH TO A; CYCLE THE DRIVE UP.
432
433                            D.    WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-A IS RESET, AND
434                                  THAT 'ATA-A IS SET.
435
436
437
438                            E.    ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
439                                  PORT A.
440
441                            F.    VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT B AND
442                                  'NED' SETS WHEN ATEMPTING TO ACCESS THE DRIVE THROUGH
443                                  PORT B.  ATTEMPT TO SET PORT REQUEST BY WRITING 0'S
444                                  INTO RMDS THROUGH PORT B.
445
446                            G.    ISSUE A RELEASE COMMAND THROUGH PORT A.  VERIFY THAT THE
447                                  DRIVE REMAINS LOCKED ON PORT A.
448
449
450                          TEST 10 TEST 'CONTROLLER SELECT' SWITCH LOCKED ON PORT B
451
452
453                          TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).
454
455                            A.    CYCLE THE DRIVE DOWN.
456
```

```
                        B.    SWITCH TO CONTROLLER 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 'CONTROLLER SELECT' SWITCH TO B; CYCLE THE DRIVE UP.

                        D.    WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-B IS RESET, AND
                              THAT 'ATA-B IS SET.

                        E.    ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
                              PORT B.

                        F.    VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT A AND
                              'NED' SETS WHEN ATEMPTING 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 'CONTROLLER SELECT' SWITCH TO
                              A/B; CYCLE THE DRIVE UP.

                        I.  VERIFY THAT BOTH PORTS CAN ACCESS THE DRIVE, THAT BOTH ATTENTION
                              BITS ARE SET, AND THAT BOTH 'VV' BITS ARE RESET.
```

```
                        ;PROGRAM REVISION #001

                        .TITLE  CZRMSAO RM05/3/2 DU POR TST 2
                        ;*COPYRIGHT (C) 1980
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY MIKE LEAVITT
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C4), 1980.
                        ;*
                        .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
529
530                     .SBTTL  RH11/RH70 REGISTERS
531
532                     ;CONTROL AND STATUS REGISTER 1 (RMCS1)
533
534     000100          IE      = 100                   ;INTERRUPT ENABLE (BIT #6)
535     000200          RDY     = 200                   ;READY (BIT #7)
536     000400          A16     = 400                   ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
537     001000          A17     = 1000                  ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
538     002000          PSEL    = 2000                  ;PORT SELECT (BIT #10)
539     020000          MCPE    = 20000                 ;MASSBUSS PARITY ERROR (BIT #13)
540     040000          TRE     = 40000                 ;TRANSFER ERROR (BIT #14)
541     100000          SC      = 100000                ;SPECIAL CONDITION (BIT #15)
542
543                     ;WORD COUNT REGISTER (RMWC)
544                     ;(EACH BIT IS CALLED BY BIT NUMBER)
545
546                     ;BUS ADDRESS REGISTER (RMBA)
547                     ;(EACH BIT IS CALLED BY BIT NUMBER)
548
549                     ;CONTROL AND STATUS REGISTER 2 (RMCS2)
550
551     000001          U0      = 1                     ;UNIT SELECT (BIT #0)
552     000002          U1      = 2                     ;UNIT SELECT (BIT #1)
553     000004          U2      = 4                     ;UNIT SELECT (BIT #2)
554     000010          BAI     = 10                    ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
555     000020          PAT     = 20                    ;MASSBUS PARITY TEST (BIT #4)
556     000040          CLR     = 40                    ;CLEAR (BIT #5)
557     000100          IR      = 100                   ;INPUT READY (BIT #6)
558     000200          OR      = 200                   ;OUTPUT READY (BIT #7)
559     000400          MDPE    = 400                   ;MASS BUS PARITY ERROR (BIT #8)
560     001000          MXF     = 1000                  ;MISSED TRANSFER ERROR (BIT #9)
561     002000          PGE     = 2000                  ;PROGRAM ERROR (BIT #10)
562     004000          NEM     = 4000                  ;NON EXISTENT MEMORY (BIT #11)
563     010000          NED     = 10000                 ;NON EXISTENT DRIVE (BIT #12)
564     020000          UPE     = 20000                 ;UNIBUS PARITY ERROR (BIT #13)
565     040000          WCE     = 40000                 ;WRITE CHECK ERROR (BIT #14)
566     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 CODE 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

  000204    000137  001774              JMP     @#START1        ;START AND CHANGE THE RH/RM ADDRESS

                                .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
    721
    757
```

```
      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            $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                                ;;WHICH  ($REG0) WAS OBTAINED
      001162  000000            $REG0:  .WORD   0               ;;CONTAINS (($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
                                ;;*****************************************************************
```

```
      0                         .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
```

```
      0                                 .SBTTL  ERROR POINTER TABLE

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

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


    001276                              $ERRTB:
  1
  2                                     ;ERROR 1
  3
  4 001276  017703                              EM1     ;DRIVE IS NON-EXISTENT ('NED' BIT SET)
  5 001300  022460                              DH1
  6 001302  024030                              DT1
  7 001304  024250                              DF1
  8
  9                                     ;ERROR 2
 10
 11 001306  017751                              EM2     ;WRONG DRIVE TYPE
 12 001310  022531                              DH2
 13 001312  024044                              DT2
 14 001314  024255                              DF2
 15
 16                                     ;ERROR 3
 17
 18 001316  017772                              EM3     ;CONTROLLER SELECT SWITCH ON DRIVE NOT IN 'A/B'
 19 001320  022460                              DH1
 20 001322  024030                              DT1
 21 001324  024250                              DF1
 22
 23                                     ;ERROR 4
 24
 25 001326  020051                              EM4     ;DRIVE NOT ON LINE
 26 001330  022531                              DH2
 27 001332  024044                              DT2
 28 001334  024255                              DF2
 29
 30                                     ;ERROR 5
 31
 32 001336  020073                              EM5     ;SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME
 33 001340  022605                              DH5
 34 001342  024062                              DT5
 35 001344  024263                              DF5
 36
 37                                     ;ERROR 6
 38
 39 001346  020155                              EM6     ;TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS
 40 001350  022654                              DH6
 41 001352  024076                              DT6
 42 001354  024270                              DF6
```

```
                                ;ERROR 7
                                
001356  020227                          EM7     ;TIMEOUT ONE-SHOT IS LESS THAN 500 MS
001360  022703                          DH7
001362  024106                          DT7
001364  024273                          DF7

                                ;ERROR 10

001366  020274                          EM10    ;READIN PRESET DOES NOT SET VOLUME VALID FOR THE PORT
001370  022460                          DH1
001372  024030                          DT1
001374  024250                          DF1

                                ;ERROR 11

001376  020362                          EM11    ;'GO' BIT RESET DURING UNLOAD COMMAND
001400  022460                          DH1
001402  024030                          DT1
001404  024250                          DF1

                                ;ERROR 12

001406  020427                          EM12    ;INCORRECT STATUS DURING UNLOAD COMMAND
001410  022460                          DH1
001412  024030                          DT1
001414  024250                          DF1

                                ;ERROR 13

001416  020476                          EM13    ;DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND
001420  022750                          DH13
001422  024120                          DT13
001424  024270                          DF6

                                ;ERROR 14

001426  020563                          EM14    ;ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD
001430  023025                          DH14
001432  024130                          DT14
001434  024277                          DF14

                                ;ERROR 15

001436  020645                          EM15    ;ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD'
001440  022460                          DH1
001442  024030                          DT1
001444  024250                          DF1

                                ;ERROR 16

001446  020731                          EM16    ;DRIVE NOT IN NEUTRAL AFTER UNLOAD WITH 'CONTROLLER
                                                ;SELECT' SWITCH MOVED FROM 'A/B'
001450  022750                          DH13
001452  024120                          DT13
001454  024270                          DF6
```

```
                                ;ERROR 17

001456  021055                          EM17    ;DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP
001460  023144                          DH17
001462  024146                          DT17
001464  024305                          DF17

                                ;ERROR 20

001466  021140                          EM20    ;DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP
001470  023144                          DH17
001472  024146                          DT17
001474  024305                          DF17

                                ;ERROR 21

001476  021223                          EM21    ;STATUS INCORRECT FOR PORT AFTER CYCLE UP
001500  022460                          DH1
001502  024030                          DT1
001504  024250                          DF1

                                ;ERROR 22

001506  021274                          EM22    ;REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT
001510  022460                          DH1
001512  024030                          DT1
001514  024250                          DF1

                                ;ERROR 23

001516  021372                          EM23    ;'NED' SET WHEN RMDS ACCESSED THROUGH PORT NOT SWITCHED
001520  022460                          DH1
001522  024030                          DT1
001524  024250                          DF1

                                ;ERROR 24

001526  021465                          EM24    ;DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/  ➡
001530  023163                          DH24
001532  024154                          DT24
001534  024273                          DF7

                                ;ERROR 25

001536  021545                          EM25    ;RH/RM DIDN'T RESPOND TO ADDRESSING
001540  023266                          DH25
001542  024166                          DT25
001544  024307                          DF25

                                ;ERROR 26

001546  000000                          0       ;UNUSED ERROR MESSAGES
001550  000000                          0
001552  000000                          0
001554  000000                          0
```

```
                                        ;ERROR 27

        001556  000000                            0       ;UNUSED ERROR MESSAGES
        001560  000000                            0
        001562  000000                            0
        001564  000000                            0

    159                                 ;ERROR 30
    160
    161 001566  021610                            EM30    ;DRIVE NOT SEIZED BY PORT 'N'
    162 001570  023275                            DH30
    163 001572  024172                            DT30
    164 001574  024310                            DF30
    165
    166                                 ;ERROR 31
    167
    168 001576  021641                            EM31    ;WRONG STATUS SEEN BY THE SEIZING PORT
    169 001600  022531                            DH2
    170 001602  024044                            DT2
    171 001604  024255                            DF2
    172
    173                                 ;ERROR 32
    174
    175 001606  021707                            EM32    ;REGISTER CONTENTS INCORRECT
    176 001610  022531                            DH2
    177 001612  024044                            DT2
    178 001614  024255                            DF2
    179
    180                                 ;ERROR 33
    181
    182 001616  021737                            EM33    ;CONTROL BUS PARITY ERROR WHILE READING REGISTER
    183 001620  022460                            DH1
    184 001622  024030                            DT1
    185 001624  024250                            DF1
    186
    187                                 ;ERROR 34
    188
    189 001626  022023                            EM34    ;CAN'T ACCESS DRIVE THROUGH EITHER PORT
    190 001630  023417                            DH34
    191 001632  024212                            DT34
    192 001634  024317                            DF34
    193
    194                                 ;ERROR 35
    195
    196 001636  022072                            EM35    ;DRIVE NOT IN NEUTRAL AFTER RELEASE, REQUEST NOT SET
    197 001640  023513                            DH35
    198 001642  024224                            DT35
    199 001644  024273                            DF7
    200
    201                                 ;ERROR 36
    202
    203 001646  022157                            EM36    ;DRIVE NOT IN NEUTRAL AFTER TIMEOUT, REQUEST NOT SET
    204 001650  023513                            DH35
    205 001652  024224                            DT35
    206 001654  024273                            DF7
    207
    208                                 ;ERROR 37
```

```
209
210 001656  022244                    EM37    ;REGISTER CONTENTS INCORRECT AFTER RELEASE/TIMEOUT
211 001660  023610                    DH37
212 001662  024172                    DT30
213 001664  024310                    DF30
214
215                             ;ERROR 40
216
217 001666  022325                    EM40    ;DRIVE NOT SEIZED BY PORT AFTER RELEASE WITH REQUEST SET
218 001670  023732                    DH40
219 001672  024236                    DT40
220 001674  024273                    DF7
221
222                             ;ERROR 41
223
224 001676  022402                    EM41    ;REGISTER WRONG AFTER RELEASE WITH REQUEST SET
225 001700  023275                    DH30
226 001702  024172                    DT30
227 001704  024310                    DF30
228
234
```

```
                                        ;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     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
002042  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
002050  012737  014002  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
002056  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
002064  012737  016654  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
002072  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
002100  013737  013232  013224          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
```

```
                                                RTI
002204  000002
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 002242  012746  000140               MOV     #PR3,-(SP)      ;;PUT NEW PS ON STACK
   002246  012746  002254               MOV     #67$,-(SP)      ;;PUT NEW PC ON  STACK
   002252  000002                       RTI                     ;;POP NEW PC AND PS
   002254                       67$:
29
30                                      .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
   002254  005227  177777               INC     #-1             ;;FIRST TIME?
   002260  001054                       BNE     68$             ;;BRANCH IF NO
   002262  022737  013366  000042       CMP     #$ENDAD,@#42    ;;ACT-11?
   002270  001450                       BEQ     68$             ;;BRANCH IF YES
   002272  104401  002330               TYPE    ,69$            ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
   002276  005737  000042               TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
   002302  001006                       BNE     70$             ;;BRANCH IF YES
   002304  023727  001140  000176       CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
   002312  001005                       BNE     71$             ;;BRANCH IF NO
   002314  104406                       GTSWR                   ;;GET SOFT-SWR SETTINGS
   002316  000403                       BR      71$
   002320  112737  000001  001134 70$:  MOVB    #1,$AUTOP       ;;SET AUTO-MODE INDICATOR
   002326                       71$:
   002326  000431                       BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  <CRLF>@CZRMSA0 - RM05/3/2 DUAL PORT LOGIC TEST, PART 2@<CRLF>
   002412                       68$:
31
32 002412  004737  015264       1$:     JSR     PC,$TKINT       ;SETUP THE TTY KEYBOARD
33 002416  004737  003010               JSR     PC,CHANGE       ;CHECK/CHANGE THE RH/RM ADDRESS
34 002422  104401  016742               TYPE    ,ENTERA         ;ENTER DRIVE ADDRESS
35 002426  104412                       RDOCT                   ;GET THE ADDRESS
36 002430  012637  001224               MOV     (SP)+,PORTA     ;STORE THE ADDRESS
37 002434  023727  001224  000007       CMP     PORTA,#7        ;SEE IF ADDRESS TOO LARGE
38 002442  101403                       BLOS    2$              ;BR IF NOT
39 002444  104401  016771               TYPE    ,ADRERR         ;TYPE ADDRESS ERROR MESSAGE
40 002450  000760                       BR      1$              ;TRY AGAIN
41 002452  013737  001224  001226 2$:  MOV     PORTA,PORTB     ;GENERATE THE PORT B ADDRESS
42 002460  005237  001226               INC     PORTB           ;INCREMENT THE ADDRESS
43 002464  042737  000016  001226       BIC     #16,PORTB       ;LEAVE BIT 0
44 002472  013746  001224               MOV     PORTA,-(SP)     ;PUT PORT A ADDRESS ON THE STACK
45 002476  042716  177771               BIC     #^C6,(SP)       ;SAVE BITS 1 & 2
46 002502  052637  001226               BIS     (SP)+,PORTB     ;SET BITS 1 & 2 IN PORT B ADDRESS
47 002506  104401  017013               TYPE    ,PORTAIS        ;'PORT A ADDRESS IS '
48 002512  013746  001224               MOV     PORTA,-(SP)     ;;SAVE PORTA FOR TYPEOUT
                                                                ;;TYPE PORT A ADDRESS
   002516  104403                       TYPOS                   ;;GO TYPE--OCTAL ASCII
   002520     001                       .BYTE   1               ;;TYPE 1 DIGIT(S)
   002521     000                       .BYTE   0               ;;SUPPRESS LEADING ZEROS
49 002522  104401  017040               TYPE    ,PORTBIS        ;'PORT B ADDRESS IS '
50 002526  013746  001226               MOV     PORTB,-(SP)     ;;SAVE PORTB FOR TYPEOUT
                                                                ;;TYPE PORT B ADDRESS
```

```
                002532  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
                002534     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
                002535     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
     51         002536  104401  001207                  TYPE    ,$CRLF          ;ANOTHER CR-LF
     52         002542  013737  001224  001230          MOV     PORTA,PORTC     ;GENERATE ADDRESS OF DRIVE NOT TESTED
     53         002550  062737  000006  001230          ADD     #6,PORTC        ;COMPLEMENT SOME BITS
     54         002556  042737  177770  001230          BIC     #^C7,PORTC      ;SAVE ONLY LOWER BITS
     55         002564  013701  001224                  MOV     PORTA,R1        ;USE PORT A ADDRESS AS INDEX
     56         002570  116137  024344  001232          MOVB    ATABIT(R1),ASR1  ;GET ATTENTION BIT FOR DRIVE
     59         002576  005037  001256                  CLR     TIMEA           ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
                002602  005037  001260                  CLR     TIMEAP          ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
                002606  005037  001262                  CLR     TIMEB           ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
                002612  005037  001264                  CLR     TIMEBP          ;CLEAR TIMEOUT ONE-SHOT VALUE LOCATION
     60         002616  004737  013406                  JSR     PC,CKCLK        ;SETUP CLOCK
     61         002622  000137  002636                  JMP     EXEC            ;CLOCK HAS BEEN STARTED
     62         002626  104401  017065                  TYPE    ,NOCLOCK        ;NO CLOCK ON SYSTEM
     63         002632  000000                  3$:     HALT                    ;FATAL ERROR
     64         002634  000776                          BR      3$              ;INTERLOCK THE HALT
     65
     66
     67                                         ;ROUTINE TO GET THE TEST NUMBER FROM THE OPERATOR
     68         002636  000005                  EXEC:   RESET                   ;CLEAR EVERYTHING
     69         002640  005037  177776                  CLR     PS              ;CLEAR THE PROCESSOR STATUS WORD
     70         002644  104401  001207                  TYPE    ,$CRLF          ;CR-LF
     71         002650  013700  001272                  MOV     $RMADR,R0       ;RH/RM ADDRESS FOR INDEXING
     72         002654  012706  001100                  MOV     #STACK,SP       ;LOAD STACK POINTER
     73         002660  004737  013406                  JSR     PC,CKCLK        ;START THE CLOCK
     74         002664  000240                          NOP                     ;RETURN IF NO CLOCK
     75         002666  004737  015264                  JSR     PC,$TKINT       ;INITIALIZE THE KEYBOARD
     76         002672  005037  001266                  CLR     KYBCTL          ;CLEAR SINGLE TEST INDICATOR
     77         002676  005037  001100                  CLR     $PASS           ;CLEAR THE PASS COUNT
     78         002702  112737  000001  001115          MOVB    #1,$ERMAX       ;SET ERROR MAX TO 1
     79         002710  012737  002710  001106          MOV     #.,$LPADR       ;INITIAL SETTING FOR LOOP ADDRESS
     80         002716  012737  002716  001110          MOV     #.,$LPERR       ;INITIAL SETTING FOR LOOP ON ERROR ADDRESS
     81         002724  104401  017132          1$:     TYPE    ,TESTNO         ;ASK FOR TEST NUMBER
     82         002730  104412                          RDOCT                   ;GET THE NUMBER
     83         002732  012601                          MOV     (SP)+,R1        ;PUT ENTRY INTO R1
     84         002734  001002                          BNE     2$              ;BR IF NOT ZERO
     85         002736  000137  003132                  JMP     TST1            ;ENTER ZERO - PERFORM ALL TESTS
     86         002742  020137  024354          2$:     CMP     R1,MAXTN        ;SEE IF NUMBER GREATER THAN MAXIMUM
     87         002746  003403                          BLE     3$              ;BR IF LESS OR EQUAL
     88         002750  104401  017152                  TYPE    ,BADNO          ;BAD ENTRY
     89         002754  000763                          BR      1$              ;TRY AGAIN
     90         002756  005301                  3$:     DEC     R1              ;DECREMENT ENTRY
     91         002760  006301                          ASL     R1              ;SHIFT IT LEFT
     92         002762  016137  024324  003006          MOV     TSTADR(R1),4$   ;GET THE TEST ADDRESS
     93         002770  005237  001266                  INC     KYBCTL          ;SET SINGLE TEST INDICATOR
     94         002774  012737  000001  001104          MOV     #1,$ICNT        ;PRESET ITERATION COUNT
     95         003002  000177  000000                  JMP     @4$             ;GO TO THE SELECTED TEST
     96         003006  000000                  4$:     .WORD   0               ;TEST ADDRESS GOES HERE
     97
     98                                         ;CHANGE THE RH/RM UNIBUS ADDRESS USED BY THE PROGRAM
     99
    100         003010  005737  001270          CHANGE: TST     CHGADR          ;CHANGE THE ADDRESS ?
    101         003014  001421                          BEQ     3$              ;BR IF NOT
    102         003016  005037  001270                  CLR     CHGADR          ;CLEAR THE INDICATOR
    103         003022  104401  017200          1$:     TYPE    ,ADDRIS         ;TYPE OUT WHAT THE PRESENT ADDRESS IS
```

```
                                                ;PUT THE ADDRESS ON THE STACK
                                                ;TYPE THE ACTUAL ADDRESS
                                                ;CR-LF
                                                ;ASK FOR NEW ADDRESS
                                                ;0 OR 'CR' ENTERED ?
                                                ;BR IF EITHER ENTERED (NO ADDRESS CHANGE)
                                                ;NEW RH/RM ADDRESS
                                                ;CORRECT THE STACK POINTER
                                                ;LOAD TRAP ADDRESS
                                                ;GET RH/RM ADDRESS
                                                ;RESPONDS AT THAT ADDRESS ?
                                                ;BR IF YES
                                                ;RESET THE STACK POINTER
                                                ;GET ADDRESS AGAIN
                                                ;RESTORE THE VECTOR
                                                ;RETURN
                                                ;;RESTORE R0 AFTER END OF PASS
                                                ;CLEAR MASSBUS
                                                ;;*****************************************************************
                                                ;*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.
                                                ;*
                                                ;;*****************************************************************
                                                ;PERFORMING ONLY SINGLE TESTS ?
                                                ;BR IF NOT
                                                ;BR IF JUST ENTERED TEST
                                                ;RETURN & GET NEXT TEST NUMBER
                                                ;SET SINGLE TEST INDICATOR
                                                ;SETUP SCOPE LOOP ADDRESS
                                                ;SETUP ERROR LOOP ADDRESS
                                                ;MOVE #1 TEST NUMBER
                                                ;SETUP THE STACK POINTER
                                                ;;DO 1 ITERATION

                                                ;VERIFY THAT DRIVE IS PRESENT THROUGH PORTS A & B

                                                ;SELECT PORT A
                                                ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
                                                ;SEE IF DRIVE (PORT A) PRESENT
                                                ;CLEAR THE 'CHECK ERROR' INDICATOR
```

```
003234  016037  000010  001126          MOV     RMCS2(R0),$BDDAT  ;GET CONTENTS OF RMCS2
003242  012737  000010  001122          MOV     #RMCS2,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003250  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
003254  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
003260  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
003266  042737  167777  001164          BIC     #^CNED,$TMP0  ;SAVE SPECIFIED BITS
003274  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
003302  001414                          BEQ     64$             ;BR IF OK
003304  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
003312  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
003320  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
003326  104001                          EMT     1
003330  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003334  000240                  64$:    NOP
003336  005737  001244                  TST     CKERR           ;WAS 'NED' SET ?
003342  001403                          BEQ     .+10            ;BR IF NOT
003344  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;ISSUE MASSBUS INIT TO CLEAR 'NED'
003352  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
003360  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003366  005760  000012                  TST     RMDS(R0)        ;SEE IF DRIVE (PORT B) PRESENT
003372  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
003376  016037  000010  001126          MOV     RMCS2(R0),$BDDAT  ;GET CONTENTS OF RMCS2
003404  012737  000010  001122          MOV     #RMCS2,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003412  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
003416  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
003422  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
003430  042737  167777  001164          BIC     #^CNED,$TMP0  ;SAVE SPECIFIED BITS
003436  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
003444  001414                          BEQ     66$             ;BR IF OK
003446  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
003454  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
003462  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
003470  104001                          EMT     1
003472  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
003476  000240                  66$:    NOP
003500  005737  001244                  TST     CKERR           ;WAS 'NED' SET ?
003504  001403                          BEQ     .+10            ;BR IF NOT
003506  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;ISSUE MASSBUS INIT TO CLEAR 'NED'
152
153                             ;CONFIRM THAT DRIVE IS AN RM05, RM03 OR RM02 AND IS DUAL PORTED
154
158 003514  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
    003522  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
    003530  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
    003534  016037  000026  001126          MOV     RMDT(R0),$BDDAT  ;GET CONTENTS OF RMDT
    003542  012737  000026  001122          MOV     #RMDT,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
    003550  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
    003554  012737  024027  001124          MOV     #024027,$GDDAT  ;WHAT REGISTER SHOULD BE
    003562  022737  024024  001126          CMP     #024024,$BDDAT  ;DUAL PORT RM03 ?
    003570  001413                          BEQ     68$             ;YES !!
    003572  022737  024025  001126          CMP     #024025,$BDDAT  ;DUAL PORT RM02 ?
    003600  001407                          BEQ     68$             ;YES !!
    003602  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS THE REGISTER OK ?
    003610  001403                          BEQ     68$             ;BR IF OK
    003612  104002                          EMT     2
    003614  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
    003620  000240                  68$:    NOP
```

```
003622  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
003630  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003636  005037  001244                  CLR     CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
003642  016037  000026  001126          MOV     RMDT(R0),$BDDAT  ;GET CONTENTS OF RMDT
003650  012737  000026  001122          MOV     #RMDT,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003656  060037  001122                  ADD     R0,$BDADR        ;ADD RH/RM BASE ADDRESS
003662  012737  024027  001124          MOV     #024027,$GDDAT  ;WHAT REGISTER SHOULD BE
003670  022737  024024  001126          CMP     #024024,$BDDAT  ;DUAL PORT RM03 ?
003676  001413                          BEQ     70$              ;YES !!
003700  022737  024025  001126          CMP     #024025,$BDDAT  ;DUAL PORT RM02 ?
003706  001407                          BEQ     70$              ;YES !!
003710  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS THE REGISTER OK ?
003716  001403                          BEQ     70$              ;BR IF OK
003720  104002                          EMT     2
003722  005137  001244                  COM     CKERR            ;SET THE REGISTER COMPARE ERROR INDICATOR
003726  000240                  70$:    NOP
159
160                                     ;VERIFY THROUGH BOTH PORTS THAT THE DRIVE IS ON LINE AND IN NEUTRAL
161
166 003730  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
003736  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
003744  005037  001244                  CLR     CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
003750  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
003756  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
003764  060037  001122                  ADD     R0,$BDADR        ;ADD RH/RM BASE ADDRESS
003770  012737  001000  001124          MOV     #PGM,$GDDAT  ;WHAT REGISTER SHOULD BE
003776  013737  001126  001164          MOV     $BDDAT,$TMP0     ;MOVE REGISTER CONTENTS TO '$TMP0'
004004  042737  176777  001164          BIC     #^CPGM,$TMP0  ;SAVE SPECIFIED BITS
004012  023737  001124  001164          CMP     $GDDAT,$TMP0     ;COMPARE THE BITS
004020  001414                          BEQ     72$              ;BR IF OK
004022  013737  001126  001174          MOV     $BDDAT,$TMP4     ;COPY 'BAD DATA'
004030  042737  001000  001174          BIC     #PGM,$TMP4       ;CLEAR THE MASKED BITS
004036  053737  001174  001124          BIS     $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
004044  104003                          EMT     3
004046  005137  001244                  COM     CKERR            ;SET THE REGISTER COMPARE ERROR INDICATOR
004052  000240                  72$:    NOP
004054  005037  001244                  CLR     CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
004060  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
004066  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004074  060037  001122                  ADD     R0,$BDADR        ;ADD RH/RM BASE ADDRESS
004100  012737  010600  001124          MOV     #MOL!DPR!DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
004106  013737  001126  001164          MOV     $BDDAT,$TMP0     ;MOVE REGISTER CONTENTS TO '$TMP0'
004114  042737  167177  001164          BIC     #^C10600,$TMP0  ;SAVE SPECIFIED BITS
004122  023737  001124  001164          CMP     $GDDAT,$TMP0     ;COMPARE THE BITS
004130  001414                          BEQ     74$              ;BR IF OK
004132  013737  001126  001174          MOV     $BDDAT,$TMP4     ;COPY 'BAD DATA'
004140  042737  010600  001174          BIC     #10600,$TMP4     ;CLEAR THE MASKED BITS
004146  053737  001174  001124          BIS     $TMP4,$GDDAT     ;'OR' WITH GOOD DATA FOR TYPEOUT
004154  104004                          EMT     4
004156  005137  001244                  COM     CKERR            ;SET THE REGISTER COMPARE ERROR INDICATOR
004162  000240                  74$:    NOP
004164  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
004172  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
004200  005037  001244                  CLR     CKERR            ;CLEAR THE 'CHECK ERROR' INDICATOR
004204  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
004212  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004220  060037  001122                  ADD     R0,$BDADR        ;ADD RH/RM BASE ADDRESS
```

```
004224  012737  001000  001124          MOV     #PGM,$GDDAT  ;WHAT REGISTER SHOULD BE
004232  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
004240  042737  176777  001164          BIC     #^CPGM,$TMP0  ;SAVE SPECIFIED BITS
004246  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
004254  001414                          BEQ     76$             ;BR IF OK
004256  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
004264  042737  001000  001174          BIC     #PGM,$TMP4      ;CLEAR THE MASKED BITS
004272  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
004300  104003                          EMT     3
004302  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
004306  000240                  76$:    NOP
004310  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
004314  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
004322  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004330  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
004334  012737  010600  001124          MOV     #MOL!DPR!DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
004342  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
004350  042737  167177  001164          BIC     #^C10600,$TMP0  ;SAVE SPECIFIED BITS
004356  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
004364  001414                          BEQ     78$             ;BR IF OK
004366  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
004374  042737  010600  001174          BIC     #10600,$TMP4    ;CLEAR THE MASKED BITS
004402  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
004410  104004                          EMT     4
004412  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
004416  000240                  78$:    NOP
167
168                             ;VERIFY THAT DRIVE SERIAL NUMBER SEEN THROUGH BOTH PORTS IS THE SAME
169
170 004420  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A
171 004426  016037  000030  001124          MOV     RMSN(R0),$GDDAT ;STORE THE PORT A SERIAL NUMBER
172 004434  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
173 004442  016037  000030  001126          MOV     RMSN(R0),$BDDAT ;STORE THE PORT B SERIAL NUMBER
174 004450  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;ARE THEY THE SAME ?
175 004456  001406                          BEQ     1$              ;BR IF THEY ARE
176 004460  104005                          EMT     5
177 004462  032777  100000  174450          BIT     #SW15,@SWR      ;HALT ON ERROR ?
178 004470  001001                          BNE     1$              ;BR IF SET - PROGRAM HAS ALREADY HALTED
179 004472  000000                          HALT                    ;HALT, POSSIBLE CABLE CONNECTION PROBLEM
180 004474  000004                  1$:     SCOPE                   ;LOOP ?
181
215
216
```

```
;;*****************************************************************
;*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.
;*
```

```
                                ;;*****************************************************************
004476                          TST2:
004476  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
004502  001406                          BEQ     2$              ;BR IF NOT
004504  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
004506  000137  002636                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
004512  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
004520  012737  004534  001106  2$:     MOV     #TEST2,$LPADR   ;SETUP SCOPE LOOP ADDRESS
004526  012737  004534  001110          MOV     #TEST2,$LPERR   ;SETUP ERROR LOOP ADDRESS
004534                          TEST2:
004534  112737  000002  001102          MOVB    #2,$TSTNM       ;MOVE #2 TEST NUMBER
004542  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
004546  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

004554  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
004562  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

                                ;SET VOLUME VALUE FOR PORT

004570  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
004576  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
004604  012760  010000  000032          MOV     #FMT16,RMOF(R0)  ;SET FMT16

                                ;VERIFY THAT THE DRIVE STATUS IS CORRECT

004612  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
004616  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
004624  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
004632  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
004636  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;WHAT REGISTER SHOULD BE
004644  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
004652  042737  106077  001164          BIC     #^C71700,$TMP0  ;SAVE SPECIFIED BITS
004660  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
004666  001414                          BEQ     64$             ;BR IF OK
004670  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
004676  042737  071700  001174          BIC     #71700,$TMP4    ;CLEAR THE MASKED BITS
004704  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
004712  104010                          EMT     10
004714  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
004720  000240                  64$:    NOP

                                ;RELEASE THE DRIVE FROM PORT A

004722  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
004730  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
004736  012760  000013  000000          MOV     #13,RMCS1(R0)   ;ISSUE RELEASE THROUGH PORT A

                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

004744  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
004750  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
004756  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
004762  012737  011600  001124          MOV     #MOL!PGM!DPR!DRY,$GDDAT  ;COMPARISON CONSTANT
004770  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
004776  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
005004  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
005012  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
```

```
005020  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
005026  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
005034  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
005042  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005050  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
005056  001006                          BNE     66$             ;BR IF NOT
005060  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
005064  001037                          BNE     68$             ;BR IF NOT
005066  104034                          EMT     34
005070  000137  005254                  JMP     70$             ;BYPASS THE REST OF THE CHECKS
005074  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
005102  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005110  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
005116  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
005122  001414                          BEQ     67$             ;BR IF ZERO
005124  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005132  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
005140  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
005146  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
005152  001004                          BNE     68$             ;BR IF NOT
005154  012737  177777  001250  67$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
005162  104036                          EMT     36
005164  013737  001170  001126  68$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
005172  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
005200  042737  100100  001170          BIC     #ATA!VV,$TMP2   ;DON'T CHECK ATTN BIT OR VV BIT
005206  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
005214  001401                          BEQ     69$             ;BR IF OK FROM PORT A.
005216  104037                          EMT     37
005220  013737  001172  001126  69$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
005226  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
005234  042737  100100  001172          BIC     #ATA!VV,$TMP3   ;DON'T CHECK ATTN BIT OR VV BIT
005242  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
005250  001401                          BEQ     70$             ;BR IF OK
005252  104037                          EMT     37
005254  000240                  70$:    NOP
005256  000004                          SCOPE                   ;LOOP ?
217

                                ;;*****************************************************************
                                ;*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.
                                ;*
                                ;;*****************************************************************
005260                          TST3:
005260  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
005264  001406                          BEQ     2$              ;BR IF NOT
005266  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
005270  000137  002636                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
```

```
005274  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
005302  012737  005316  001106  2$:     MOV     #TEST3,$LPADR   ;SETUP SCOPE LOOP ADDRESS
005310  012737  005316  001110          MOV     #TEST3,$LPERR   ;SETUP ERROR LOOP ADDRESS
005316                          TEST3:
005316  112737  000003  001102          MOVB    #3,$TSTNM       ;MOVE #3 TEST NUMBER
005324  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
005330  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

005336  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
005344  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT

                                ;SET VOLUME VALUE FOR PORT

005352  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
005360  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
005366  012760  010000  000032          MOV     #FMT16,RMOF(R0)  ;SET FMT16

                                ;VERIFY THAT THE DRIVE STATUS IS CORRECT

005374  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
005400  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
005406  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
005414  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
005420  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;WHAT REGISTER SHOULD BE
005426  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
005434  042737  106077  001164          BIC     #^C71700,$TMP0  ;SAVE SPECIFIED BITS
005442  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
005450  001414                          BEQ     64$             ;BR IF OK
005452  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
005460  042737  071700  001174          BIC     #71700,$TMP4    ;CLEAR THE MASKED BITS
005466  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
005474  104010                          EMT     10
005476  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
005502  000240                  64$:    NOP

                                ;RELEASE THE DRIVE FROM PORT B

005504  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
005512  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
005520  012760  000013  000000          MOV     #13,RMCS1(R0)   ;ISSUE RELEASE THROUGH PORT B

                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

005526  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
005532  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
005540  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
005544  012737  011600  001124          MOV     #MOL!PGM!DPR!DRY,$GDDAT  ;COMPARISON CONSTANT
005552  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
005560  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
005566  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
005574  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005602  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
005610  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
005616  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
005624  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
005632  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
005640  001006                          BNE     66$             ;BR IF NOT
```

```
005642  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
005646  001037                          BNE     68$             ;BR IF NOT
005650  104034                          EMT     34
005652  000137  006036                  JMP     70$             ;BYPASS THE REST OF THE CHECKS
005656  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
005664  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005672  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
005700  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
005704  001414                          BEQ     67$             ;BR IF ZERO
005706  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
005714  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
005722  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
005730  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
005734  001004                          BNE     68$             ;BR IF NOT
005736  012737  177777  001250  67$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
005744  104036                          EMT     36
005746  013737  001170  001126  68$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
005754  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
005762  042737  100100  001170          BIC     #ATA!VV,$TMP2   ;DON'T CHECK ATTN BIT OR VV BIT
005770  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
005776  001401                          BEQ     69$             ;BR IF OK FROM PORT A.
006000  104037                          EMT     37
006002  013737  001172  001126  69$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
006010  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
006016  042737  100100  001172          BIC     #ATA!VV,$TMP3   ;DON'T CHECK ATTN BIT OR VV BIT
006024  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
006032  001401                          BEQ     70$             ;BR IF OK
006034  104037                          EMT     37
006036  000240                  70$:    NOP
006040  000004                          SCOPE                   ;LOOP ?
```

218
223
275

```
;;*********************************************************************
;*TEST 4        MEASURE THE TIMEOUT ONE-SHOT THROUGH PORT A
;*
;*MEASURE THE TIMEOUT ONE-SHOT VALUE THROUGH PORT A
;*
;*  A.  WRITE 0'S INTO RMDS THROUGH PORT A AND VERIFY THAT THE
;*      DRIVE HAS BEEN SEIZED.
;*
;*  B.  WAIT FOR TIMEOUT TO OCCUR.  MEASURE THE DURATION OF THE TIMEOUT
;*      ONE-SHOT AND SAVE THE VALUE FOR LATER USE.
;*
;*  C.  VERIFY THAT THE TIMEOUT OCCURRED AND THAT THE DRIVE RETURNS
;*      TO NEUTRAL
;*
;;*********************************************************************
006042  TST4:
006042  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
006046  001406                          BEQ     2$              ;BR IF NOT
006050  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
006052  000137  002636                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
006056  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
006064  012737  006100  001106  2$:     MOV     #TEST4,$LPADR   ;SETUP SCOPE LOOP ADDRESS
006072  012737  006100  001110          MOV     #TEST4,$LPERR   ;SETUP ERROR LOOP ADDRESS
```

```
006100                                  TEST4:
006100  112737  000004  001102                MOVB    #4,$TSTNM       ;MOVE #4 TEST NUMBER
006106  012706  001100                        MOV     #STACK,SP       ;SETUP THE STACK POINTER
006112  012737  000001  001176                MOV     #1,$TIMES       ;;DO 1 ITERATION

006120  005037  001256                        CLR     TIMEA           ;CLEAR THE TIMEOUT VALUE STORAGE LOCATION
006124  005037  001260                        CLR     TIMEAP  ;CLEAR THE + 25% TOLERANCE LOCATION

                                        ;START THE TIMER

006130  005037  001252                        CLR     TIME            ;CLEAR THE ELAPSED TIME COUNTER
006134  012737  003720  001254                MOV     #2000.,WATCH    ;SET WATCH TO 2000. MS

                                        ;SEIZE THE DRIVE THROUGH PORT A

006142  113760  001224  000010                MOVB    PORTA,RMCS2(R0) ;SELECT PORT A
006150  013737  001224  001236                MOV     PORTA,SEIZPT  ;STORE SEIZING PORT'S ADDRESS
006156  005060  000012                        CLR     RMDS(R0)        ;WRITE RMDS
006162  113760  001226  000010                MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
006170  013737  001226  001234                MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006176  013737  001226  001240                MOV     PORTB,OPPRT     ;'OPPOSITE' PORT ADDRESS
006204  016037  000012  001126                MOV     RMDS(R0),$BDDAT ;SEE IF DRIVE SEIZED BY PORT A
006212  010037  001122                        MOV     R0,$BDADR       ;RH/RM BASE ADDRESS
006216  062737  000012  001122                ADD     #RMDS,$BDADR    ;GENERATE BAD REGISTER ADDRESS
006224  005037  001124                        CLR     $GDDAT          ;REGISTER SHOULD BE ZERO
006230  023737  001124  001126                CMP     $GDDAT,$BDDAT   ;IS THE REGISTER ZERO
006236  001403                                BEQ     64$             ;BR IF IT IS
006240  104030                                EMT     30
006242  000137  006756                        JMP     4$              ;BYPASS REST OF THE SUBTEST
006246                                  64$:
006246  113760  001224  000010                MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
006254  013737  001224  001234                MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006262  016037  000012  001126                MOV     RMDS(R0),$BDDAT  ;SEE IF SEIZING PORT SEES CORRECT STATUS
006270  012737  011700  001124                MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;EXPECTED STATUS
006276  013737  001124  001166                MOV     $GDDAT,$TMP1  ;USE GOOD DATA AS A MASK
006304  005137  001166                        COM     $TMP1           ;COMPLEMENT THE EXPECTED STATUS
006310  013737  001126  001164                MOV     $BDDAT,$TMP0    ;SAVE THE ACTUAL STATUS
006316  043737  001166  001164                BIC     $TMP1,$TMP0     ;CLEAR UNWANTED BITS
006324  023737  001124  001164                CMP     $GDDAT,$TMP0    ;ARE THE EXPECTED STATUS BITS SET ?
006332  001401                                BEQ     65$             ;BR IF THEY ARE
006334  104031                                EMT     31
006336  000240                          65$:  NOP

                                        ;WAIT FOR PORT A TO TIMEOUT

006340  113760  001226  000010                MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
006346  013737  001226  001234                MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
006354  005760  000012                  1$:   TST     RMDS(R0)        ;WAIT FOR THE DRIVE TO TIMEOUT
006360  001006                                BNE     2$              ;BR WHEN TIMEOUT OCCURS
006362  005737  001254                        TST     WATCH           ;CHECK WATCH
006366  001372                                BNE     1$              ;BR IF NOT ZERO
006370  104006                                EMT     6
006372  000137  006430                        JMP     3$              ;BYPASS THE REST OF THE TEST
006376  013737  001252  001256          2$:   MOV     TIME,TIMEA      ;SAVE THE ELAPSED TIME FOR PORT A
006404  004537  013572                        JSR     R5,TOLER        ;CALCULATE THE TOLERANCE
006410  001256                                .WORD   TIMEA           ;TIMEOUT VALUE FOR PORT A
006412  012637  001260                        MOV     (SP)+,TIMEAP    ;+25% TOLERANCE
```

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

006416  023727  001256  000764            CMP     TIMEA,#500.     ;IS TIMEOUT VALUE AT LEAST 500 MS ?
006424  103001                            BHIS    3$              ;BR IF IT IS
006426  104007                            EMT     7

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

006430                                  3$:
                                        ;VERIFY THAT THE DRIVE IS IN NEUTRAL

006430  005037  001250                    CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
006434  012737  000012  001122            MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
006442  060037  001122                    ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
006446  012737  011700  001124            MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;COMPARISON CONSTANT
006454  113760  001224  000010            MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
006462  016037  000012  001170            MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
006470  013737  001170  001164            MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
006476  042737  100100  001164            BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
006504  113760  001226  000010            MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
006512  016037  000012  001172            MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
006520  013737  001172  001166            MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
006526  042737  100100  001166            BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
006534  023737  001164  001166            CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
006542  001006                            BNE     66$             ;BR IF NOT
006544  005737  001164                    TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
006550  001045                            BNE     68$             ;BR IF NOT
006552  104034                            EMT     34
006554  000137  006754                    JMP     70$             ;BYPASS THE REST OF THE CHECKS
006560  013737  001170  001126  66$:      MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
006566  013737  001226  001234            MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
006574  113760  001226  000010            MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
006602  005737  001164                    TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
006606  001414                            BEQ     67$             ;BR IF ZERO
006610  013737  001224  001234            MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
006616  013737  001172  001126            MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
006624  113760  001224  000010            MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
006632  005737  001166                    TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
006636  001012                            BNE     68$             ;BR IF NOT
006640  012737  177777  001250  67$:      MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
006646  012760  000011  000000            MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
006654  012760  000013  000000            MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
006662  104035                            EMT     35
006664  013737  001170  001126  68$:      MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
006672  013737  001224  001234            MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
006700  042737  100000  001170            BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
006706  023737  001124  001170            CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
006714  001401                            BEQ     69$             ;BR IF OK FROM PORT A.
006716  104037                            EMT     37
006720  013737  001172  001126  69$:      MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
006726  013737  001226  001234            MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
006734  042737  100000  001172            BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
006742  023737  001124  001172            CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
006750  001401                            BEQ     70$             ;BR IF OK
006752  104037                            EMT     37
006754  000240                  70$:      NOP
```

```
006756  000004                  4$:     SCOPE                   ;LOOP ?
276

                                ;;*****************************************************************
                                ;*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
                                ;*
                                ;;*****************************************************************
006760                          TST5:
006760  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
006764  001406                          BEQ     2$              ;BR IF NOT
006766  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
006770  000137  002636                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
006774  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
007002  012737  007016  001106  2$:     MOV     #TEST5,$LPADR   ;SETUP SCOPE LOOP ADDRESS
007010  012737  007016  001110          MOV     #TEST5,$LPERR   ;SETUP ERROR LOOP ADDRESS
007016                          TEST5:
007016  112737  000005  001102          MOVB    #5,$TSTNM       ;MOVE #5 TEST NUMBER
007024  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
007030  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

007036  005037  001262                  CLR     TIMEB           ;CLEAR THE TIMEOUT VALUE STORAGE LOCATION
007042  005037  001264                  CLR     TIMEBP  ;CLEAR THE + 25% TOLERANCE LOCATION

                                ;START THE TIMER

007046  005037  001252                  CLR     TIME            ;CLEAR THE ELAPSED TIME COUNTER
007052  012737  003720  001254          MOV     #2000.,WATCH    ;SET WATCH TO 2000. MS

                                ;SEIZE THE DRIVE THROUGH PORT B

007060  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B
007066  013737  001226  001236          MOV     PORTB,SEIZPT  ;STORE SEIZING PORT'S ADDRESS
007074  005060  000012                  CLR     RMDS(R0)        ;WRITE RMDS
007100  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
007106  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007114  013737  001224  001240          MOV     PORTA,OPPRT     ;'OPPOSITE' PORT ADDRESS
007122  016037  000012  001126          MOV     RMDS(R0),$BDDAT ;SEE IF DRIVE SEIZED BY PORT B
007130  010037  001122                  MOV     R0,$BDADR       ;RH/RM BASE ADDRESS
007134  062737  000012  001122          ADD     #RMDS,$BDADR    ;GENERATE BAD REGISTER ADDRESS
007142  005037  001124                  CLR     $GDDAT          ;REGISTER SHOULD BE ZERO
007146  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS THE REGISTER ZERO
007154  001403                          BEQ     64$             ;BR IF IT IS
007156  104030                          EMT     30
007160  000137  007674                  JMP     4$              ;BYPASS REST OF THE SUBTEST
007164                          64$:
007164  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
```

```
007172  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007200  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;SEE IF SEIZING PORT SEES CORRECT STATUS
007206  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;EXPECTED STATUS
007214  013737  001124  001166          MOV     $GDDAT,$TMP1  ;USE GOOD DATA AS A MASK
007222  005137  001166                  COM     $TMP1           ;COMPLEMENT THE EXPECTED STATUS
007226  013737  001126  001164          MOV     $BDDAT,$TMPO    ;SAVE THE ACTUAL STATUS
007234  043737  001166  001164          BIC     $TMP1,$TMPO     ;CLEAR UNWANTED BITS
007242  023737  001124  001164          CMP     $GDDAT,$TMPO    ;ARE THE EXPECTED STATUS BITS SET ?
007250  001401                          BEQ     65$             ;BR IF THEY ARE
007252  104031                          EMT     31
007254  000240                  65$:    NOP

                                ;WAIT FOR PORT B TO TIMEOUT

007256  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
007264  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
007272  005760  000012          1$:     TST     RMDS(R0)        ;WAIT FOR THE DRIVE TO TIMEOUT
007276  001006                          BNE     2$              ;BR WHEN TIMEOUT OCCURS
007300  005737  001254                  TST     WATCH           ;CHECK WATCH
007304  001372                          BNE     1$              ;BR IF NOT ZERO
007306  104006                          EMT     6
007310  000137  007346                  JMP     3$              ;BYPASS THE REST OF THE TEST
007314  013737  001252  001262  2$:     MOV     TIME,TIMEB      ;SAVE THE ELAPSED TIME FOR PORT B
007322  004537  013572                  JSR     R5,TOLER        ;CALCULATE THE TOLERANCE
007326  001262                          .WORD   TIMEB           ;TIMEOUT VALUE FOR PORT B
007330  012637  001264                  MOV     (SP)+,TIMEBP    ;+25% TOLERANCE

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

007334  023727  001262  000764          CMP     TIMEB,#500.     ;IS TIMEOUT VALUE AT LEAST 500 MS ?
007342  103001                          BHIS    3$              ;BR IF IT IS
007344  104007                          EMT     7

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

007346                          3$:
                                ;VERIFY THAT THE DRIVE IS IN NEUTRAL

007346  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
007352  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
007360  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
007364  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;COMPARISON CONSTANT
007372  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
007400  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
007406  013737  001170  001164          MOV     $TMP2,$TMPO     ;COPY IT INTO '$TMPO'
007414  042737  100100  001164          BIC     #ATA!VV,$TMPO   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
007422  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
007430  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
007436  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
007444  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
007452  023737  001164  001166          CMP     $TMPO,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
007460  001006                          BNE     66$             ;BR IF NOT
007462  005737  001164                  TST     $TMPO           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
007466  001045                          BNE     68$             ;BR IF NOT
007470  104034                          EMT     34
007472  000137  007672                  JMP     70$             ;BYPASS THE REST OF THE CHECKS
007476  013737  001170  001126  66$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
```

```
007504  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
007512  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
007520  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
007524  001414                          BEQ     67$             ;BR IF ZERO
007526  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
007534  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
007542  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
007550  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
007554  001012                          BNE     68$             ;BR IF NOT
007556  012737  177777  001250  67$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
007564  012760  000011  000000          MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
007572  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
007600  104035                          EMT     35
007602  013737  001170  001126  68$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
007610  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
007616  042737  100000  001170          BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
007624  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
007632  001401                          BEQ     69$             ;BR IF OK FROM PORT A.
007634  104037                          EMT     37
007636  013737  001172  001126  69$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
007644  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
007652  042737  100000  001172          BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
007660  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
007666  001401                          BEQ     70$             ;BR IF OK
007670  104037                          EMT     37
007672  000240                  70$:    NOP
007674  000004                  4$:     SCOPE                   ;LOOP ?
277
278
295
296
                                ;;*******************************************************************
                                ;*TEST 6        TEST 'CONTROLLER SELECT' SWITCH, DRIVE CYCLED UP
                                ;*
                                ;*TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED UP).
                                ;*
                                ;*  A.  SWITCH TO CONTROLLER 'A' POSITION.  VERIFY THAT THE DRIVE IS IN
                                ;*      NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
                                ;*      PORTS, ARE CORRECT.
                                ;*
                                ;*  B.  SWITCH TO CONTROLLER 'B' POSITION. VERIFY THAT THE DRIVE IS IN
                                ;*      NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
                                ;*      PORTS, ARE CORRECT.
                                ;*
                                ;*  C.  RETURN THE 'CONTROLLER SELECT' SWITCH TO THE 'A/B' POSITION.  VERIFY
                                ;*      THE DRIVE STATE.
                                ;*
                                ;;*******************************************************************
007676                  TST6:
007676  005737  001266                  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
007702  001406                          BEQ     2$              ;BR IF NOT
007704  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
007706  000137  002636                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
007712  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
007720  012737  007734  001106  2$:     MOV     #TEST6,$LPADR   ;SETUP SCOPE LOOP ADDRESS
007726  012737  007734  001110          MOV     #TEST6,$LPERR   ;SETUP ERROR LOOP ADDRESS
007734                  TEST6:
007734  112737  000006  001102          MOVB    #6,$TSTNM       ;MOVE #6 TEST NUMBER
```

```
                                                                D 4
007742  012706  001100                    MOV     #STACK,SP       ;SETUP THE STACK POINTER
007746  012737  000001  001176            MOV     #1,$TIMES       ;;DO 1 ITERATION
297
298                             ;CLEAR ATTENTION BITS FOR BOTH PORTS

007754  113760  001224  000010            MOVB    PORTA,RMCS2(R0) ;SELECT PORT #A
007762  005060  000012                    CLR     RMDS(R0)        ;SEIZE THE DRIVE
007766  012760  000011  000000            MOV     #11,RMCS1(R0)   ;ISSUE DRIVE CLEAR
007774  012760  000013  000000            MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
010002  113760  001226  000010            MOVB    PORTB,RMCS2(R0) ;SELECT PORT #B
010010  005060  000012                    CLR     RMDS(R0)        ;SEIZE THE DRIVE THROUGH PORT 'B'
010014  012760  000011  000000            MOV     #11,RMCS1(R0)   ;ISSUE DRIVE CLEAR
010022  012760  000013  000000            MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
299 010030  104401  017352                TYPE    ,SWTCHA         ;SWITCH TO 'A'
300 010034  104401  017516                TYPE    ,CONTUE         ;PRESS 'CONTINUE'
301 010040  000000                        HALT
302
303                             ;VERIFY THAT THE DRIVE IS IN NEUTRAL

010042  005037  001250                    CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
010046  012737  000012  001122            MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
010054  060037  001122                    ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
010060  012737  011700  001124            MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;COMPARISON CONSTANT
010066  113760  001224  000010            MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010074  016037  000012  001170            MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
010102  013737  001170  001164            MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
010110  042737  100100  001164            BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010116  113760  001226  000010            MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010124  016037  000012  001172            MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
010132  013737  001172  001166            MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
010140  042737  100100  001166            BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010146  023737  001164  001166            CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
010154  001006                            BNE     64$             ;BR IF NOT
010156  005737  001164                    TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
010162  001045                            BNE     66$             ;BR IF NOT
010164  104034                            EMT     34
010166  000137  010366                    JMP     68$             ;BYPASS THE REST OF THE CHECKS
010172  013737  001170  001126  64$:      MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
010200  013737  001226  001234            MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010206  113760  001226  000010            MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010214  005737  001164                    TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
010220  001414                            BEQ     65$             ;BR IF ZERO
010222  013737  001224  001234            MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010230  013737  001172  001126            MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
010236  113760  001224  000010            MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010244  005737  001166                    TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
010250  001012                            BNE     66$             ;BR IF NOT
010252  012737  177777  001250  65$:      MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
010260  012760  000011  000000            MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
010266  012760  000013  000000            MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
010274  104017                            EMT     17
010276  013737  001170  001126  66$:      MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
010304  013737  001224  001234            MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
010312  042737  100000  001170            BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
010320  023737  001124  001170            CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
010326  001401                            BEQ     67$             ;BR IF OK FROM PORT A.
010330  104037                            EMT     37
```

```
                                                                                    E 4
010332  013737  001172  001126  67$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
010340  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
010346  042737  100000  001172          BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
010354  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
010362  001401                          BEQ     68$             ;BR IF OK
010364  104037                          EMT     37
010366  000240                  68$:    NOP
304 010370  104401  017434              TYPE    ,SWTCHB         ;SWITCH TO 'B'
305 010374  104401  017516              TYPE    ,CONTUE         ;PRESS 'CONTINUE'
306 010400  000000                      HALT
307
308                                     ;VERIFY THAT THE DRIVE IS IN NEUTRAL

010402  005037  001250                  CLR     RELERR          ;CLEAR THE 'RELEASE ERROR ' INDICATOR
010406  012737  000012  001122          MOV     #RMDS,$BDADR    ;FORM THE ADDRESS OF RMDS FOR TYPEOUT
010414  060037  001122                  ADD     R0,$BDADR       ;ADD THE I/O BASE ADDRESS
010420  012737  011700  001124          MOV     #MOL!PGM!DPR!DRY!VV,$GDDAT  ;COMPARISON CONSTANT
010426  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010434  016037  000012  001170          MOV     RMDS(R0),$TMP2  ;GET THE DRIVE STATUS REGISTER FROM PORT A.
010442  013737  001170  001164          MOV     $TMP2,$TMP0     ;COPY IT INTO '$TMP0'
010450  042737  100100  001164          BIC     #ATA!VV,$TMP0   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010456  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010464  016037  000012  001172          MOV     RMDS(R0),$TMP3  ;GET THE DRIVE STATUS REGISTER FROM PORT B.
010472  013737  001172  001166          MOV     $TMP3,$TMP1     ;COPY IT INTO '$TMP1'
010500  042737  100100  001166          BIC     #ATA!VV,$TMP1   ;CLEAR PORT DEPENDENT BITS FROM THE COPY
010506  023737  001164  001166          CMP     $TMP0,$TMP1     ;IS THE STATUS REGISTER THE SAME FROM BOTH PORTS ?
010514  001006                          BNE     69$             ;BR IF NOT
010516  005737  001164                  TST     $TMP0           ;REGISTERS ARE THE SAME: ARE THEY ZERO ?
010522  001045                          BNE     71$             ;BR IF NOT
010524  104034                          EMT     34
010526  000137  010726                  JMP     73$             ;BYPASS THE REST OF THE CHECKS
010532  013737  001170  001126  69$:    MOV     $TMP2,$BDDAT    ;SET UP POSSIBLE BAD DATA FOR ERROR MESSAGE
010540  013737  001226  001234          MOV     PORTB,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010546  113760  001226  000010          MOVB    PORTB,RMCS2(R0) ;SELECT PORT B.
010554  005737  001164                  TST     $TMP0           ;SEE IF STATUS EQ 0 FROM PORT A.
010560  001414                          BEQ     70$             ;BR IF ZERO
010562  013737  001224  001234          MOV     PORTA,PTNBR     ;SEIZING PORT IF TEST SHOWS DRIVE NOT IN NEUTRAL
010570  013737  001172  001126          MOV     $TMP3,$BDDAT    ;'BAD DATA' FOR ERROR TYPE OUT
010576  113760  001224  000010          MOVB    PORTA,RMCS2(R0) ;SELECT PORT A.
010604  005737  001166                  TST     $TMP1           ;SEE IF STATUS EQ ZERO FROM PORT B.
010610  001012                          BNE     71$             ;BR IF NOT
010612  012737  177777  001250  70$:    MOV     #-1,RELERR      ;SET 'RELEASE ERROR' INDICATOR
010620  012760  000011  000000          MOV     #11,RMCS1(R0)   ;CLEAR THE DRIVE
010626  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE THE DRIVE
010634  104020                          EMT     20
010636  013737  001170  001126  71$:    MOV     $TMP2,$BDDAT    ;LOOK FOR BIT FAILURES WHEN RMDS READ
010644  013737  001224  001234          MOV     PORTA,PTNBR     ;CHANGE PORT NUMBER
010652  042737  100000  001170          BIC     #ATA,$TMP2      ;DON'T CHECK THE ATTN BIT
010660  023737  001124  001170          CMP     $GDDAT,$TMP2    ;ALL BITS OK ?
010666  001401                          BEQ     72$             ;BR IF OK FROM PORT A.
010670  104037                          EMT     37
010672  013737  001172  001126  72$:    MOV     $TMP3,$BDDAT    ;CHECK RMDS FOR BIT FAILURES - FROM PORT B.
010700  013737  001226  001234          MOV     PORTB,PTNBR     ;CHANGE PORT NUMBER
010706  042737  100000  001172          BIC     #ATA,$TMP3      ;DON'T CHECK THE ATTN BIT
010714  023737  001124  001172          CMP     $GDDAT,$TMP3    ;SEE IF READ OK FROM PORT B.
010722  001401                          BEQ     73$             ;BR IF OK
010724  104037                          EMT     37
```

```
                                                                F 4
        010726  000240                    73$:    NOP
    309 010730  005737  001266                    TST     KYBCTL          ;SINGLE TEST MODE ?
    310 010734  001402                            BEQ     1$              ;BR IF NOT
    311 010736  104401  017264                    TYPE    ,SWTCHN         ;RETURN SWITCH TO 'A/B'
    312 010742  000004                    1$:     SCOPE                   ;LOOP ?
    313
    445
    446

                                        ;;*********************************************************************
                                        ;*TEST 7        TEST 'CONTROLLER SELECT' SWITCH ON PORT A
                                        ;*
                                        ;*TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).
                                        ;*
                                        ;*  A.  CYCLE THE DRIVE DOWN.
                                        ;*
                                        ;*  B.  SWITCH TO CONTROLLER A POSITION.  VERIFY THAT THE DRIVE IS IN
                                        ;*      NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
                                        ;*      PORTS, ARE CORRECT.
                                        ;*
                                        ;*  C.  SWITCH THE 'CONTROLLER SELECT' SWITCH TO A; CYCLE THE DRIVE UP.
                                        ;*
                                        ;*  D.  WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-A IS RESET, AND
                                        ;*      THAT 'ATA-A IS SET.
                                        ;*
                                        ;*  E.  ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
                                        ;*      PORT A.
                                        ;*
                                        ;*  F.  VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT B AND
                                        ;*      'NED' SETS WHEN 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.
                                        ;*
                                        ;;*********************************************************************
    010744                              TST7:
    010744  005737  001266                      TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
    010750  001406                              BEQ     2$              ;BR IF NOT
    010752  100002                              BPL     1$              ;BR IF JUST ENTERED TEST
    010754  000137  002636                      JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
    010760  012737  177777  001266    1$:       MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
    010766  012737  011002  001106    2$:       MOV     #TEST7,$LPADR   ;SETUP SCOPE LOOP ADDRESS
    010774  012737  011002  001110              MOV     #TEST7,$LPERR   ;SETUP ERROR LOOP ADDRESS
    011002                              TEST7:
    011002  112737  000007  001102              MOVB    #7,$TSTNM       ;MOVE #7 TEST NUMBER
    011010  012706  001100                      MOV     #STACK,SP       ;SETUP THE STACK POINTER
    011014  012737  000001  001176              MOV     #1,$TIMES       ;;DO 1 ITERATION

    011022  113760  001224  000010              MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
    011030  013737  001224  001234              MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
    011036  104401  017567                      TYPE    ,CYCLED         ;'CYCLE DOWN THE DRIVE'

                                        ;WAIT FOR 'MOL' TO RESET

    011042  032760  010000  000012    1$:       BIT     #MOL,RMDS(R0)   ;TEST 'MOL'
    011050  001374                              BNE     1$              ;BR IF IT IS STILL SET
```

```
011052  104401  017352                  TYPE    ,SWTCHA         ;SWITCH TO 'A'
011056  104401  017610                  TYPE    ,CYCLEU         ;'CYCLE UP THE DRIVE'

                                ;WAIT FOR DRIVE TO CYCLE UP AFTER SWITCH CHANGED

011062  032760  010000  000012  2$:     BIT     #MOL,RMDS(R0)   ;TEST 'MOL' AGAIN
011070  001774                          BEQ     2$              ;BR IF NOT SET

                                ;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT A

011072  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011076  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
011104  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011112  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011116  012737  110600  001124          MOV     #ATA!MOL!DPR!DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
011124  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011132  042737  066077  001164          BIC     #^C111700,$TMP0  ;SAVE SPECIFIED BITS
011140  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
011146  001414                          BEQ     64$             ;BR IF OK
011150  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011156  042737  111700  001174          BIC     #111700,$TMP4   ;CLEAR THE MASKED BITS
011164  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011172  104021                          EMT     21
011174  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011200  000240                  64$:    NOP

                                ;SET VOLUME VALID FOR PORT A

011202  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
011210  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
011216  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.

011224  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
011232  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011240  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011244  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
011252  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011260  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011264  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
011270  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011276  042737  000077  001164          BIC     #^C177700,$TMP0  ;SAVE SPECIFIED BITS
011304  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
011312  001414                          BEQ     66$             ;BR IF OK
011314  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011322  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
011330  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011336  104022                          EMT     22
011340  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011344  000240                  66$:    NOP
011346  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011352  016037  000010  001126          MOV     RMCS2(R0),$BDDAT  ;GET CONTENTS OF RMCS2
011360  012737  000010  001122          MOV     #RMCS2,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011366  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011372  012737  010000  001124          MOV     #NED,$GDDAT  ;WHAT REGISTER SHOULD BE
```

```
011400  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011406  042737  167777  001164          BIC     #^CNED,$TMP0  ;SAVE SPECIFIED BITS
011414  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
011422  001414                          BEQ     68$             ;BR IF OK
011424  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011432  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
011440  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011446  104023                          EMT     23
011450  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011454  000240                  68$:    NOP
011456  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

011462  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
011470  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011476  012760  000013  000012          MOV     #13,RMDS(R0)    ;ISSUE A RELEASE THROUGH PORT A
011504  013737  001224  001236          MOV     PORTA,SEIZPT    ;ADDRESS OF 'LOCKED ON' PORT
011512  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
011520  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011526  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
011532  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
011540  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
011546  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
011552  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
011556  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
011564  042737  000077  001164          BIC     #^C177700,$TMP0  ;SAVE SPECIFIED BITS
011572  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
011600  001414                          BEQ     70$             ;BR IF OK
011602  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
011610  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
011616  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
011624  104024                          EMT     24
011626  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
011632  000240                  70$:    NOP

                                ;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST

011634  105737  001103                  TSTB    $ERFLG          ;DID AN ERROR OCCUR
011640  001412                          BEQ     3$              ;BR IF NOT
011642  032777  001000  167270          BIT     #SW09,@SWR      ;SEE IF LOOP ON ERROR (SWR9 = 1)
011650  001406                          BEQ     3$              ;BR IF NOT
011652  105037  001103                  CLRB    $ERFLG          ;CLEAR THE ERROR FLAG
011656  005037  001176                  CLR     $TIMES          ;CLEAR THE MAX ITERATION COUNT
011662  000177  167222                  JMP     @$LPERR         ;GO TO THE LOOP ADDRESS
011666  005737  001266          3$:     TST     KYBCTL          ;IN SINGLE TEST MODE ?
011672  001460                          BEQ     6$              ;BR IF NOT
011674  032777  040000  167236          BIT     #SW14,@SWR      ;LOOP ON TEST ?
011702  001054                          BNE     6$              ;BR IF LOOPING
011704  104401  017567                  TYPE    ,CYCLED         ;TYPE 'CYCLE DOWN'
011710  104401  017264                  TYPE    ,SWTCHN         ;'SWITCH TO A/B'
011714  104401  017610                  TYPE    ,CYCLEU         ;'CYCLE THE DRIVE UP'
011720  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
011726  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
011734  032760  010000  000012  4$:     BIT     #MOL,RMDS(R0)   ;CHECK 'MOL'
011742  001374                          BNE     4$              ;BR IF SET (DRIVE NOT CYCLED DOWN)
```

```
011744  032760  010000  000012  5$:     BIT     #MOL,RMDS(R0)   ;CHECK 'MOL' AGAIN
011752  001774                          BEQ     5$              ;BR IF NOT SET (DRIVE NOT CYCLED UP)

                                ;SET VOLUME VALID FOR BOTH PORTS

011754  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR THROUGH PORT A
011762  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET THROUGH PORT A
011770  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE PORT A
011776  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
012004  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012012  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET THROUGH PORT B
012020  012760  010000  000032          MOV     #FMT16,RMOF(R0) ;SET FMT16
012026  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE PORT B
012034  104401  001207          6$:     TYPE    ,$CRLF          ;CR-LF
012040  012737  011610  001254          MOV     #5000.,WATCH    ;SPINDLE MOTOR 'COOL DOWN' DELAY
012046  005737  001254          7$:     TST     WATCH           ;FINSIHED ?
012052  001375                          BNE     7$              ;BR IF NOT
012054  000004                          SCOPE                   ;LOOP ?
012056  000400                          BR      TST10           ;;GO TO NEXT TEST
447

                                ;;*****************************************************************
                                ;*TEST 10       TEST 'CONTROLLER SELECT' SWITCH ON PORT B
                                ;*
                                ;*TEST THE OPERATION OF THE 'CONTROLLER SELECT' SWITCH (DRIVE CYCLED DOWN).
                                ;*
                                ;*  A.  CYCLE THE DRIVE DOWN.
                                ;*
                                ;*  B.  SWITCH TO CONTROLLER B POSITION.  VERIFY THAT THE DRIVE IS IN
                                ;*      NEUTRAL AND THAT THE STATUS BITS IN RMDS, AS READ THROUGH BOTH
                                ;*      PORTS, ARE CORRECT.
                                ;*
                                ;*  C.  SWITCH THE 'CONTROLLER SELECT' SWITCH TO B; CYCLE THE DRIVE UP.
                                ;*
                                ;*  D.  WHEN THE DRIVE CYCLES UP, VERIFY THAT 'VV-B IS RESET, AND
                                ;*      THAT 'ATA-B IS SET.
                                ;*
                                ;*  E.  ISSUE A DRIVE CLEAR COMMAND AND A READIN PRESET COMMAND THROUGH
                                ;*      PORT B.
                                ;*
                                ;*  F.  VERIFY THAT THE DRIVE CANNOT BE ACCESSED THROUGH PORT A AND
                                ;*      'NED' SETS WHEN ATEMPTING 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 'CONTROLLER SELECT' SWITCH TO
                                ;*      A/B; CYCLE THE DRIVE UP.
                                ;*
                                ;*  I.  VERIFY THAT BOTH PORTS CAN ACCESS THE DRIVE, THAT BOTH ATTENTION
                                ;*      BITS ARE SET, AND THAT BOTH 'VV' BITS ARE RESET.
                                ;*
                                ;*****************************************************************
012060
012060  005737  001266  TST10:  TST     KYBCTL          ;PERFORMING ONLY SINGLE TESTS ?
012064  001406                  BEQ     2$              ;BR IF NOT
```

```
012066  100002                          BPL     1$              ;BR IF JUST ENTERED TEST
012070  000137  002636                  JMP     EXEC            ;RETURN & GET NEXT TEST NUMBER
012074  012737  177777  001266  1$:     MOV     #-1,KYBCTL      ;SET SINGLE TEST INDICATOR
012102  012737  012116  001106  2$:     MOV     #TEST10,$LPADR  ;SETUP SCOPE LOOP ADDRESS
012110  012737  012116  001110          MOV     #TEST10,$LPERR  ;SETUP ERROR LOOP ADDRESS
012116                          TEST10:
012116  112737  000010  001102          MOVB    #10,$TSTNM      ;MOVE #10 TEST NUMBER
012124  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
012130  012737  000001  001176          MOV     #1,$TIMES       ;;DO 1 ITERATION

012136  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
012144  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012152  104401  017567                  TYPE    ,CYCLED         ;'CYCLE DOWN THE DRIVE'

                                ;WAIT FOR 'MOL' TO RESET

012156  032760  010000  000012  1$:     BIT     #MOL,RMDS(R0)   ;TEST 'MOL'
012164  001374                          BNE     1$              ;BR IF IT IS STILL SET
012166  104401  017434                  TYPE    ,SWTCHB         ;SWITCH TO 'B'
012172  104401  017610                  TYPE    ,CYCLEU         ;'CYCLE UP THE DRIVE'

                                ;WAIT FOR DRIVE TO CYCLE UP AFTER SWITCH CHANGED

012176  032760  010000  000012  2$:     BIT     #MOL,RMDS(R0)   ;TEST 'MOL' AGAIN
012204  001774                          BEQ     2$              ;BR IF NOT SET

                                ;DRIVE IS CYCLED UP, CHECK STATUS THROUGH PORT B

012206  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012212  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
012220  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012226  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012232  012737  110600  001124          MOV     #ATA!MOL!DPR!DRY,$GDDAT  ;WHAT REGISTER SHOULD BE
012240  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012246  042737  066077  001164          BIC     #^C111700,$TMP0  ;SAVE SPECIFIED BITS
012254  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012262  001414                          BEQ     64$             ;BR IF OK
012264  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012272  042737  111700  001174          BIC     #111700,$TMP4   ;CLEAR THE MASKED BITS
012300  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012306  104021                          EMT     21
012310  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012314  000240                  64$:    NOP

                                ;SET VOLUME VALID FOR PORT B

012316  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR
012324  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET
012332  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.

012340  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
012346  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012354  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012360  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
```

```
012366  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012374  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012400  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
012404  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012412  042737  000077  001164          BIC     #^C177700,$TMP0  ;SAVE SPECIFIED BITS
012420  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012426  001414                          BEQ     66$             ;BR IF OK
012430  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012436  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
012444  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012452  104022                          EMT     22
012454  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012460  000240                  66$:    NOP
012462  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012466  016037  000010  001126          MOV     RMCS2(R0),$BDDAT  ;GET CONTENTS OF RMCS2
012474  012737  000010  001122          MOV     #RMCS2,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012502  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012506  012737  010000  001124          MOV     #NED,$GDDAT  ;WHAT REGISTER SHOULD BE
012514  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012522  042737  167777  001164          BIC     #^CNED,$TMP0  ;SAVE SPECIFIED BITS
012530  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012536  001414                          BEQ     68$             ;BR IF OK
012540  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012546  042737  010000  001174          BIC     #NED,$TMP4      ;CLEAR THE MASKED BITS
012554  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012562  104023                          EMT     23
012564  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012570  000240                  68$:    NOP
012572  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

012576  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
012604  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012612  012760  000013  000012          MOV     #13,RMDS(R0)    ;ISSUE A RELEASE THROUGH PORT B
012620  013737  001226  001236          MOV     PORTB,SEIZPT    ;ADDRESS OF 'LOCKED ON' PORT
012626  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
012634  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
012642  005037  001244                  CLR     CKERR           ;CLEAR THE 'CHECK ERROR' INDICATOR
012646  016037  000012  001126          MOV     RMDS(R0),$BDDAT  ;GET CONTENTS OF RMDS
012654  012737  000012  001122          MOV     #RMDS,$BDADR  ;FORM REGISTER ADDRESS OF ERROR MESSAGE
012662  060037  001122                  ADD     R0,$BDADR       ;ADD RH/RM BASE ADDRESS
012666  005037  001124                  CLR     $GDDAT          ;WHAT REGISTER SHOULD BE
012672  013737  001126  001164          MOV     $BDDAT,$TMP0    ;MOVE REGISTER CONTENTS TO '$TMP0'
012700  042737  000077  001164          BIC     #^C177700,$TMP0  ;SAVE SPECIFIED BITS
012706  023737  001124  001164          CMP     $GDDAT,$TMP0    ;COMPARE THE BITS
012714  001414                          BEQ     70$             ;BR IF OK
012716  013737  001126  001174          MOV     $BDDAT,$TMP4    ;COPY 'BAD DATA'
012724  042737  177700  001174          BIC     #177700,$TMP4   ;CLEAR THE MASKED BITS
012732  053737  001174  001124          BIS     $TMP4,$GDDAT    ;'OR' WITH GOOD DATA FOR TYPEOUT
012740  104024                          EMT     24
012742  005137  001244                  COM     CKERR           ;SET THE REGISTER COMPARE ERROR INDICATOR
012746  000240                  70$:    NOP

                        ;IF ERROR OCCURRED, CHECK FOR LOOP ON TEST
```

```
012750  105737  001103                  TSTB    $ERFLG          ;DID AN ERROR OCCUR
012754  001412                          BEQ     3$              ;BR IF NOT
012756  032777  001000  166154          BIT     #SW09,@SWR      ;SEE IF LOOP ON ERROR (SWR9 = 1)
012764  001406                          BEQ     3$              ;BR IF NOT
012766  105037  001103                  CLRB    $ERFLG          ;CLEAR THE ERROR FLAG
012772  005037  001176                  CLR     $TIMES          ;CLEAR THE MAX ITERATION COUNT
012776  000177  166106                  JMP     @$LPERR         ;GO TO THE LOOP ADDRESS
013002  032777  040000  166130  3$:     BIT     #SW14,@SWR      ;LOOP ON TEST ?
013010  001054                          BNE     6$              ;BR IF LOOPING
013012  104401  017567                  TYPE    ,CYCLED         ;TYPE 'CYCLE DOWN'
013016  104401  017264                  TYPE    ,SWTCHN         ;'SWITCH TO A/B'
013022  104401  017610                  TYPE    ,CYCLEU         ;'CYCLE THE DRIVE UP'
013026  113760  001226  000010          MOVB    PORTB,RMCS2(R0)  ;SELECT PORT B
013034  013737  001226  001234          MOV     PORTB,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
013042  032760  010000  000012  4$:     BIT     #MOL,RMDS(R0)   ;CHECK 'MOL'
013050  001374                          BNE     4$              ;BR IF SET (DRIVE NOT CYCLED DOWN)
013052  032760  010000  000012  5$:     BIT     #MOL,RMDS(R0)   ;CHECK 'MOL' AGAIN
013060  001774                          BEQ     5$              ;BR IF NOT SET (DRIVE NOT CYCLED UP)

                                ;SET VOLUME VALID FOR BOTH PORTS

013062  012760  000011  000000          MOV     #11,RMCS1(R0)   ;ISSUE A DRIVE CLEAR THROUGH PORT B
013070  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET THROUGH PORT B
013076  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE PORT B
013104  113760  001224  000010          MOVB    PORTA,RMCS2(R0)  ;SELECT PORT A
013112  013737  001224  001234          MOV     PORTA,PTNBR  ;MOVE PORT ADDRESS TO LOCATION FOR TYPEOUT
013120  012760  000021  000000          MOV     #21,RMCS1(R0)   ;ISSUE A READIN PRESET THROUGH PORT A
013126  012760  010000  000032          MOV     #FMT16,RMOF(R0) ;SET FMT16
013134  012760  000013  000000          MOV     #13,RMCS1(R0)   ;RELEASE PORT A
013142  012737  011610  001254  6$:     MOV     #5000.,WATCH    ;SPINDLE MOTOR 'COOL DOWN' DELAY
013150  005737  001254          7$:     TST     WATCH           ;FINISHED ?
013154  001375                          BNE     7$              ;BR IF NOT
013156  000004                          SCOPE                   ;LOOP ?
013160  000137  013166                  JMP     $EOP            ;GO TO THE END OF PASS ROUTINE
448
449                             ;;*****************************************************************
450                             ;PUT NEWTEST HERE
451                             ;;*****************************************************************
013164  000004                  TST11:  SCOPE
452
458
459                             .SBTTL  END OF PASS ROUTINE

                                ;;*****************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
                                ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO TST1AA

013166                          $EOP:
013166  005737  001266                  TST     KYBCTL          ;ENTERED TEST VIA KEYBOARD COMMAND ?
013172  001402                          BEQ     .+6             ;BR IF NOT
013174  000137  002636                  JMP     EXEC            ;RETURN TO KEYBOARD CONTROL
013200  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
013204  005037  001176                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
```

```
013210  005237  001100                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
013214  042737  100000  001100          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
013222  005327                          DEC     (PC)+           ;;LOOP?
013224  000001          $EOPCT: .WORD   1
013226  003063                  BGT     $DOAGN          ;;YES
013230  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
013232  000001          $ENDCT: .WORD   1
013234  013224                  $EOPCT
013236  104401  013244          TYPE    ,65$            ;;TYPE ASCIZ STRING
013242  000407                  BR      64$             ;;GET OVER THE ASCIZ
                        ;;65$:  .ASCIZ  <12><15>/END PASS #/
013262                  64$:
013262  013746  001100          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                        ;;TYPE PASS NUMBER
013266  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
013270  104401  013276          TYPE    ,67$            ;;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 $ERTTL FOR TYPEOUT
                                                        ;;TOTAL NUMBER OF ERRORS
013344  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
013346  104401  001207          TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
013352  005037  001112          CLR     $ERTTL          ;;CLEAR ERROR TOTAL
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  003120          $RTNAD: .WORD   TST1AA
013402     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                .EVEN
460
```

```
                                        .SBTTL  SUBROUTINES
                                        ;;*****************************************************************
                                        .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                  $SCOPE:
013620  104407                  CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
013622  032777  040000  165310  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
013630  001055                  BNE     $OVER           ;;YES IF SW14=1
                        ;#####START OF CODE FOR THE XOR TESTER#####
013632  000416          $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
013634  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
013640  012737  013660  000004  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
013646  005737  177060          TST     @#177060        ;;TIME OUT ON XOR?
013652  012637  000004          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
013656  000436                  BR      $SVLAD          ;;GO TO THE NEXT TEST
013660  022626          5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
013662  012637  000004          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
013666  000436                  BR      $OVER           ;;LOOP ON THE PRESENT TEST
013670                  6$:;#####END OF CODE FOR THE XOR TESTER#####
013670  105737  001103  2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
013674  001404                  BEQ     3$              ;;BR IF NO
013676  105037  001103  4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
013702  005037  001176          CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
013706  032777  004000  165224  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
013714  001011                  BNE     1$              ;;BR IF YES
013716  005737  001100          TST     $PASS           ;;IF FIRST PASS OF PROGRAM
013722  001406                  BEQ     1$              ;;      INHIBIT ITERATIONS
013724  005237  001104          INC     $ICNT           ;;INCREMENT ITERATION COUNT
013730  023737  001176  001104  CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
013736  002012                  BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
013740  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
013746  013737  014000  001176  MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
013754  105237  001102  $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
013760  011637  001106          MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
013764  013777  001102  165150  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
013772  013716  001106          MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
013776  000002                  RTI                     ;;FIXES PS
014000  000004          $MXCNT: 4                       ;;MAX. NUMBER OF ITERATIONS
                        .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

014002                          $ERROR:
014002  104407                         CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
014004  113737  001102  001242        MOVB    $TSTNM,TSTNUM
014012  105237  001103        7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
014016  001775                        BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
014020  013777  001102  165114        MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
014026  032777  002000  165104        BIT     #BIT10,@SWR     ;;BELL ON ERROR?
014034  001402                        BEQ     1$              ;;NO - SKIP
014036  104401  001202                TYPE    ,$BELL          ;;RING BELL
014042  005237  001112        1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
014046  011637  001116                MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
014052  162737  000002  001116        SUB     #2,$ERRPC
014060  117737  165032  001114        MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
014066  032777  020000  165044        BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
014074  001004                        BNE     20$             ;;SKIP TYPEOUTS
014076  004737  014134                JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
014102  104401  001207                TYPE    ,$CRLF
014106                          20$:
014106  005777  165026        2$:     TST     @SWR            ;;HALT ON ERROR
014112  100002                        BPL     3$              ;;SKIP IF CONTINUE
014114  000000                        HALT                    ;;HALT ON ERROR!
014116  104407                        CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
014120                          3$:
014120  022737  013366  000042        CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
014126  001001                        BNE     6$              ;;BRANCH IF NO
014130  000000                        HALT                    ;;YES
014132                          6$:
014132  000002                        RTI                     ;;RETURN
6                               .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.

014134                          $ERRTYP:
014134  104401  001207                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
014140  010046                        MOV     R0,-(SP)        ;;SAVE R0
014142  005000                        CLR     R0              ;;PICKUP THE ITEM INDEX
014144  153700  001114                BISB    @#$ITEMB,R0
014150  001004                        BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
014152  013746  001116                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
014156  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
014160  000445                        BR      10$             ;;GET OUT
014162  005300                1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
014164  006300                        ASL     R0              ;;      WORK FOR THE ERROR TABLE
014166  006300                        ASL     R0
014170  006300                        ASL     R0
014172  062700  001276                ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
014176  012037  014206                MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
014202  001404                        BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
014204  104401                        TYPE                    ;;TYPE THE "ERROR MESSAGE"
```

```
014206  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
014210  104401  001207                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
014214  012037  014224            3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
014220  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
014222  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
014224  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
014226  104401  001207                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
014232  010146                  5$:     MOV     R1,-(SP)        ;;SAVE R1
014234  012001                          MOV     (R0)+,R1        ;;PICKUP "DATA TABLE" POINTER
014236  001415                          BEQ     9$              ;;BR IF NO DATA TO BE TYPED
014240  012000                          MOV     (R0)+,R0        ;;PICKUP "DATA FORMAT" POINTER
014242  105720                  6$:     TSTB    (R0)+           ;;"OCTAL" OR "DECIMAL"
014244  001003                          BNE     7$              ;;BR IF DECIMAL
014246  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
014250  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
014252  000402                          BR      8$
014254                          7$:
014254  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
014256  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
014260  005711                  8$:     TST     (R1)            ;;IS THERE ANOTHER NUMBER?
014262  001403                          BEQ     9$              ;;BR IF NO
014264  104401  014304                  TYPE    ,11$            ;;TYPE TWO(2) SPACES
014270  000764                          BR      6$              ;;LOOP

014272  012601                  9$:     MOV     (SP)+,R1        ;;RESTORE R1
014274  012600                  10$:    MOV     (SP)+,R0        ;;RESTORE R0
014276  104401  001207                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
014302  000207                          RTS     PC              ;;RETURN
014304    040     040     000   11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
                                        .EVEN
 7                              .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
                                ;*

014310  105737  001157  $TYPE:   TSTB    $TPFLG          ;;IS THERE A TERMINAL?
014314  100002                  BPL     1$              ;;BR IF YES
014316  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
014320  000407                  BR      3$              ;;LEAVE
014322  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
014324  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
014330  112046          2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
014332  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
014334  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
014336  012600          60$:    MOV     (SP)+,R0        ;;RESTORE R0
```

```
014340  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
014344  000002                          RTI                     ;;RETURN
014346  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
014352  001430                          BEQ     8$
014354  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
014360  001006                          BNE     5$
014362  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
014364  104401                          TYPE                    ;;TYPE A CR AND LF
014366  001207                          $CRLF
014370  105037  014576                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
014374  000755                          BR      2$              ;;GET NEXT CHARACTER
014376  004737  014460          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
014402  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
014406  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
014410  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
014414  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
014420  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
014422  004737  014460                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
014426  105337  014576                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
014432  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

014434  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
014440  004737  014460          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
014444  132737  000007  014576          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
014452  001372                          BNE     9$              ;;TAB STOP
014454  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
014456  000724                          BR      2$              ;;GET NEXT CHARACTER
014460                          $TYPEC:
014460  105777  164460                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?
014464  100022                          BPL     10$             ;;BR IF NOT
014466  017746  164454                  MOV     @$TKB,-(SP)     ;;GET CHAR
014472  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS
014476  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF
014502  001012                          BNE     102$            ;;BR IF NOT
014504                          101$:
014504  105777  164434                  TSTB    @$TKS           ;;WAIT FOR CHAR
014510  100375                          BPL     101$
014512  117716  164430                  MOVB    @$TKB,(SP)      ;;GET CHAR
014516  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT
014522  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?
014526  001366                          BNE     101$            ;;BR IF NOT
014530                          102$:
014530  005726                          TST     (SP)+           ;;FIX STACK
014532                          10$:
014532  105777  164412                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
014536  100375                          BPL     10$
014540  116677  000002  164404          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
014546  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
014554  001003                          BNE     1$              ;;BRANCH IF NO
014556  105037  014576                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
014562  000406                          BR      $TYPEX          ;;EXIT
014564  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
014572  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
014574  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
```

```
014576  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
014600  000207                  $TYPEX: RTS     PC

      8                         .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

014602  017646  000000  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
014606  116637  000001  015025  MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
014614  112637  015027          MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
014620  062716  000002          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
014624  000406                  BR      $TYPON
014626  112737  000001  015025  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
014634  112737  000006  015027          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
014642  112737  000005  015024  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
014650  010346                  MOV     R3,-(SP)        ;;SAVE R3
014652  010446                  MOV     R4,-(SP)        ;;SAVE R4
014654  010546                  MOV     R5,-(SP)        ;;SAVE R5
014656  113704  015027          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
014662  005404                  NEG     R4
014664  062704  000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
014670  110437  015026          MOVB    R4,$OMODE       ;;SAVE IT FOR USE
014674  113704  015025          MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
014700  016605  000012          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
014704  005003                  CLR     R3              ;;CLEAR THE OUTPUT WORD
014706  006105          1$:     ROL     R5              ;;ROTATE MSB INTO "C"
014710  000404                  BR      3$              ;;GO DO MSB
014712  006105          2$:     ROL     R5              ;;FORM THIS DIGIT
014714  006105                  ROL     R5
014716  006105                  ROL     R5
014720  010503                  MOV     R5,R3
014722  006103          3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
014724  105337  015026          DECB    $OMODE          ;;TYPE THIS DIGIT?
014730  100016                  BPL     7$              ;;BR IF NO
014732  042703  177770          BIC     #177770,R3      ;;GET RID OF JUNK
014736  001002                  BNE     4$              ;;TEST FOR 0
```

```
014740  005704                          TST     R4              ;;SUPPRESS THIS 0?
014742  001403                          BEQ     5$              ;;BR IF YES
014744  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
014746  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
014752  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
014756  110337  015022                  MOVB    R3,8$           ;;SAVE FOR TYPING
014762  104401  015022                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
014766  105337  015024          7$:     DECB    $OCNT           ;;COUNT BY 1
014772  003347                          BGT     2$              ;;BR IF MORE TO DO
014774  002402                          BLT     6$              ;;BR IF DONE
014776  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
015000  000744                          BR      2$              ;;GO DO THE LAST DIGIT
015002  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
015004  012604                          MOV     (SP)+,R4        ;;RESTORE R4
015006  012603                          MOV     (SP)+,R3        ;;RESTORE R3
015010  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
015016  012616                          MOV     (SP)+,(SP)
015020  000002                          RTI                     ;;RETURN
015022     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
015023     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
015024     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
015025     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
015026  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
  9                                     .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

015030                                  $TYPDS:
015030  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
015032  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015034  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
015036  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
015040  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
015042  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
015046  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
015052  100004                          BPL     1$              ;;BR IF INPUT IS POS.
015054  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
015056  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
015064  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
015066  012703  015244                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
015072  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
015076  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
015100  016001  015234                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
015104  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
015106  002402                          BLT     4$              ;;BR IF DONE
015110  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
015112  000774                          BR      3$
015114  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
015116  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
```

```
015120  001002                          BNE     5$              ;;FALL THROUGH IF 0
015122  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
015124  100407                          BMI     7$              ;;BR IF YES
015126  106316                  5$:     ASLB    (SP)            ;;MSD?
015130  103003                          BCC     6$              ;;BR IF NO
015132  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
015140  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
015144  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
015150  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
015152  005720                          TST     (R0)+           ;;JUST INCREMENTING
015154  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
015160  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
015162  003002                          BGT     8$              ;;GO TO EXIT
015164  010502                          MOV     R5,R2           ;;GET THE LSD
015166  000764                          BR      6$              ;;GO CHANGE TO ASCII
015170  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
015172  100003                          BPL     9$              ;;BR IF NO
015174  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
015202  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
015204  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
015206  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
015210  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
015212  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
015214  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
015216  104401  015244                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
015222  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
015230  012616                          MOV     (SP)+,(SP)
015232  000002                          RTI                     ;;RETURN TO USER
015234  023420                  $DTBL:  10000.
015236  001750                          1000.
015240  000144                          100.
015242  000012                          10.
015244                          $DBLK:  .BLKW   4
10                              .SBTTL  TTY INPUT ROUTINE

                                ;;**********************************************************************
                                .ENABL  LSB
015254  000000                  $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
015256  000000                  $TKQIN: .WORD   0               ;;INPUT POINTER
015260  000000                  $TKQOUT: .WORD  0               ;;OUTPUT POINTER
015262                          $TKQSRT: .BLKB  1               ;;TTY KEYBOARD QUEUE
        015263                  $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
                                ;*
015264  005037  015254          $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
015270  012737  015262  015256          MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
015276  013737  015256  015260          MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
015304  012737  015334  000060          MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
015312  012737  000200  000062          MOV     #200,@#TKVEC+2  ;;"BR" LEVEL 4
```

```
015320  005777  163622                  TST     @$TKB           ;;CLEAR DONE FLAG
015324  012777  000100  163612          MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
015332  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.
                                ;
015334  117746  163606          $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
015340  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
015344  021627  000007          1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
015350  001004                          BNE     2$              ;;BRANCH IF NO
015352  022737  000176  001140          CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
015360  001500                          BEQ     6$              ;;GO TO SWR CHANGE

015362                          2$:
015362  022737  000001  015254          CMP     #1,$TKCNT       ;;IS THE QUEUE FULL?
015370  001004                          BNE     3$              ;;BRANCH IF NO
015372  104401  001202                  TYPE    ,$BELL          ;;RING THE TTY BELL
015376  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
015400  000451                          BR      5$              ;;EXIT
015402  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
015406  001021                          BNE     32$             ;;BRANCH IF NO
015410  005077  163530                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
015414  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
015416  105777  163522          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
015422  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
015424  117746  163516                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
015430  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
015434  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
015440  001366                          BNE     31$             ;;BRANCH IF NO
015442  012777  000100  163474          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
015450  000002                          RTI                     ;;RETURN
015452  005237  015254          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
015456  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
015462  002405                          BLT     4$              ;;BRANCH IF YES
015464  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
015470  003002                          BGT     4$              ;;BRANCH IF YES
015472  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
015476  112677  177554          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
015502  005237  015256                  INC     $TKQIN          ;;UPDATE THE POINTER
015506  023727  015256  015263          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
015514  001003                          BNE     5$              ;;BRANCH IF NO
015516  012737  015262  015256          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
015524  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.
015526  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
015534  001104                          BNE     15$             ;;EXIT IF NOT
015536  105777  163402                  TSTB    @$TKS           ;;IS A CHAR WAITING?
015542  100101                          BPL     15$             ;;IF NOT, EXIT
015544  117746  163376                  MOVB    @$TKB,-(SP)     ;;YES
```

```
015550  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
015554  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
015560  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.
015562  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
015570  001674                          BEQ     2$              ;;BRANCH IF YES
015572  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
015574  004737  015264                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
015600  005077  163340                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
015604  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

015612  104401  016370                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
015616  104401  016375  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
015622  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
015626  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
015630  104401  016406                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
015634  005046          19$:    CLR     -(SP)           ;;CLEAR COUNTER
015636  005046                          CLR     -(SP)           ;;THE NEW SWR
015640  105777  163300          7$:     TSTB    @$TKS           ;;CHAR THERE?
015644  100375                          BPL     7$              ;;IF NOT TRY AGAIN

015646  117746  163274                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
015652  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

015656  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
015662  001005                          BNE     10$             ;;BRANCH IF NOT
015664  104401  016363                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
015670  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
015674  000757                          BR      19$             ;;LET'S TRY IT AGAIN


015676  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
015702  001022                          BNE     16$             ;;BRANCH IF NO
015704  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
015710  001403                          BEQ     11$             ;;BRANCH IF YES
015712  016677  000002  163220          MOV     2(SP),@SWR      ;;SAVE NEW SWR
015720  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
015724  104401  001207          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
015730  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
015736  001003                          BNE     15$             ;;BRANCH IF NOT
015740  012777  000100  163176          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
015746  000002          15$:    RTI             ;;RETURN
015750  004737  014460          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
015754  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
015760  002420                          BLT     18$             ;;BRANCH IF YES
015762  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
015766  003015                          BGT     18$             ;;BRANCH IF YES
015770  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
015774  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
016000  001403                          BEQ     17$             ;;BRANCH IF YES
```

```
016002  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
016004  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
016006  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
016010  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
016014  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
016020  000707                          BR      7$              ;;GET THE NEXT ONE
016022  104401  001206          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
016026  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
                                ;

016030  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
016032  016666  000004  000002  MOV     4(SP),2(SP)     ;;THE PS
016040  005066  000004          CLR     4(SP)           ;;GET READY FOR A CHARACTER
016044  005046                  CLR     -(SP)           ;;PUT NEW PS ON STACK
016046  012746  016054          MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
016052  000002                  RTI                     ;;POP NEW PC AND PS
016054                  64$:
016054  005737  015254  1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
016060  001775                  BEQ     1$
016062  005337  015254          DEC     $TKCNT          ;;DECREMENT THE COUNTER
016066  117766  177166  000004  MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
016074  005237  015260          INC     $TKQOUT         ;;UPDATE THE POINTER
016100  023727  015260  015263  CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
016106  001003                  BNE     2$              ;;BRANCH IF NO
016110  012737  015262  015260  MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
016116  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

016120  010346          $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
016122  005046                  CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
016124  012703  016354  1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
016130  022703  016363  2$:     CMP     #$TTYIN+7,R3    ;;BUFFER FULL?
016134  101456                  BLOS    4$              ;;BR IF YES
016136  104410                  RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
016140  112613                  MOVB    (SP)+,(R3)      ;;GET CHARACTER
016142  122713  000177  10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
016146  001022                  BNE     5$              ;;BR IF NO
016150  005716                  TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
016152  001007                  BNE     6$              ;;BR IF NO
016154  112737  000134  016352  MOVB    #'\,9$          ;;TYPE A BACK SLASH
016162  104401  016352          TYPE    ,9$
016166  012716  177777          MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
016172  005303          6$:     DEC     R3              ;;BACKUP BY ONE
```

```
016174  020327  016354                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
016200  103434                          BLO     4$              ;;BR IF YES
016202  111337  016352                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
016206  104401  016352                  TYPE    ,9$             ;;GO TYPE
016212  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
016214  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
016216  001406                          BEQ     7$              ;;BR IF NO
016220  112737  000134  016352          MOVB    #'\,9$          ;;TYPE A BACK SLASH
016226  104401  016352                  TYPE    ,9$
016232  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
016234  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
016240  001003                          BNE     8$              ;;BR IF NO
016242  104401  016363                  TYPE    ,$CNTLU         ;;TYPE A CONTROL ''U''
016246  000726                          BR      1$              ;;GO START OVER
016250  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A ''^R''?
016254  001011                          BNE     3$              ;;BRANCH IF NO
016256  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
016260  104401  001207                  TYPE    ,$CRLF          ;;TYPE A ''CR'' & ''LF''
016264  104401  016354                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
016270  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
016272  104401  001206          4$:     TYPE    ,$QUES          ;;TYPE A '?'
016276  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
016300  111337  016352          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
016304  104401  016352                  TYPE    ,9$
016310  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
016314  001305                          BNE     2$              ;;LOOP IF NOT RETURN
016316  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
016322  104401  001210                  TYPE    ,$LF            ;;TYPE A LINE FEED
016326  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
016330  012603                          MOV     (SP)+,R3        ;;RESTORE R3
016332  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
016334  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
016342  012766  016354  000004          MOV     #$TTYIN,4(SP)
016350  000002                          RTI                     ;;RETURN
016352     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
016353     000                          .BYTE   0               ;;TERMINATOR
016354                          $TTYIN: .BLKB   7               ;;RESERVE 7 BYTES FOR TTY INPUT
016363     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL ''U''
016370     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL ''G''
016375     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
016406     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
                                .EVEN
11                              .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

016420  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
```

```
016422  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
016430  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
016432  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016434  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016436  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
016440  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
016442  010037  016546                  MOV     R0,5$           ;;AND SAVE IT
016446  005001                          CLR     R1              ;;CLEAR DATA WORD
016450  005002                          CLR     R2
016452  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
016454  001420                          BEQ     3$              ;;IF ZERO GET OUT
016456  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
016462  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
016464  122716  000067                  CMPB    #'7,(SP)
016470  002423                          BLT     4$
016472  006301                          ASL     R1              ;;*2
016474  006102                          ROL     R2
016476  006301                          ASL     R1              ;;*4
016500  006102                          ROL     R2
016502  006301                          ASL     R1              ;;*8
016504  006102                          ROL     R2
016506  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
016512  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
016514  000756                          BR      2$              ;;LOOP
016516  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
016520  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
016524  010237  016556                  MOV     R2,$HIOCT
016530  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
016532  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
016534  012600                          MOV     (SP)+,RC        ;;POP STACK INTO R0
016536  000002                          RTI                     ;;RETURN
016540  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
016542  105010                          CLRB    (R0)            ;;SET A TERMINATOR
016544  104401                          TYPE                    ;;TYPE UP THRU THE BAD CHAR.
016546  000000                  5$:     .WORD   0
016550  104401  001206                  TYPE    ,$QUES          ;;''?'' ''CR'' & ''LF''
016554  000730                          BR      1$              ;;TRY AGAIN
016556  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
12                                      .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

016560                                  $SAVREG:
016560  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
```

```
016562  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016564  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016566  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
016570  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
016572  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
016574  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
016600  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
016604  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
016610  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
016614  000002                          RTI

                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
016616                          $RESREG:
016616  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
016622  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
016626  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
016632  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
016636  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
016640  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
016642  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
016644  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
016646  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
016650  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
016652  000002                          RTI
  13                            .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.

016654  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
016656  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
016662  005740                          TST     -(R0)           ;;BACKUP BY 2
016664  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
016666  006300                          ASL     R0              ;;POSITION FOR INDEXING
016670  016000  016710                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
016674  000200                          RTS     R0              ;;GO TO ROUTINE

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

016676  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
016700  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
016706  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
016710  016676                  $TRPAD: .WORD   $TRAP2
```

```
        016712  014310                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
        016714  014626                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        016716  014602                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        016720  014642                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        016722  015030                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

        016724  015616                  $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

        016726  015526                  $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
        016730  016030                  $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
        016732  016120                  $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
        016734  016420                  $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
        016736  016560                  $SAVREG ;;CALL=SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
        016740  016616                  $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
    14
```

```
                                .SBTTL  TELETYPE MESSAGES

016742  200  105  116  ENTERA: .ASCIZ  <CRLF>/ENTER DRIVE ADDRESS: /
016771  040  111  116  ADRERR: .ASCIZ  / INVALID ADDRESS/<CRLF>
017013  200  120  117  PORTAIS: .ASCIZ <CRLF>/PORT A ADDRESS IS: /
017040  200  120  117  PORTBIS: .ASCIZ <CRLF>/PORT B ADDRESS IS: /
017065  200  123  131  NOCLOCK: .ASCIZ <CRLF>/SYSTEM MUST HAVE 'L' OR 'P' CLOCK/<CRLF><LF>
017132  012  105  116  TESTNO: .ASCIZ  <LF>/ENTER TEST #: /
017152  040  111  116  BADNO:  .ASCIZ  / INVALID TEST NUMBER/<CRLF>
017200  200  012  122  ADDRIS: .ASCIZ  <CRLF><LF>@RH/RM ADDRESS (RMCS1) IS: @
017235  012  105  116  NTRH'1: .ASCIZ  <LF>@ENTER RH/RM ADDRESS: @
017264  200  012  122  SWTCHN: .ASCIZ  <CRLF><LF>@RETURN 'CONTROLLER SELECT' SWITCH ON DRIVE TO 'A/B'@
017352  200  012  124  SWTCHA: .ASCIZ  <CRLF><LF>/TURN 'CONTROLLER SELECT' SWITCH ON DRIVE TO 'A'/
017434  200  012  124  SWTCHB: .ASCIZ  <CRLF><LF>/TURN 'CONTROLLER SELECT' SWITCH ON DRIVE TO 'B'/
017516  200  124  110  CONTUE: .ASCIZ  <CRLF>/THEN PRESS 'CONTINUE' ON THE PROCESSOR/<CRLF>
017567  200  012  123  CYCLED: .ASCIZ  <CRLF><LF>/STOP THE DRIVE/
017610  200  012  123  CYCLEU: .ASCIZ  <CRLF><LF>/START THE DRIVE - THE PROGRAM WILL WAIT FOR 'MOL' TO SET/

```

```
                                .SBTTL  TEST ERROR MESSAGES

017703  104 122 111  EM1:    .ASCIZ  /DRIVE IS NON-EXISTENT ('NED' BIT SET)/
017751  127 122 117  EM2:    .ASCIZ  /WRONG DRIVE TYPE/
017772  103 117 116  EM3:    .ASCIZ  @CONTROLLER SELECT SWITCH ON DRIVE NOT IN 'A/B'@
020051  104 122 111  EM4:    .ASCIZ  /DRIVE NOT ON LINE/
020073  123 105 122  EM5:    .ASCIZ  /SERIAL NUMBER READ THROUGH EACH PORT NOT THE SAME/
020155  124 111 115  EM6:    .ASCIZ  /TIMEOUT HAS NOT OCCURRED WITHIN 2 SECONDS/
020227  124 111 115  EM7:    .ASCIZ  /TIMEOUT ONE-SHOT IS LESS THAN 500 MS/
020274  122 105 101  EM10:   .ASCIZ  /READ IN PRESET DOES NOT SET VOLUME VALID FOR THE PORT/
020362  047 107 117  EM11:   .ASCIZ  /'GO' BIT RESET DURING UNLOAD COMMAND/
020427  111 116 103  EM12:   .ASCIZ  /INCORRECT STATUS DURING UNLOAD COMMAND/
020476  104 122 111  EM13:   .ASCIZ  /DRIVE DID NOT RETURN TO NEUTRAL AFTER UNLOAD COMMAND/
020563  101 124 124  EM14:   .ASCIZ  /ATTENTION BIT SET ON 'OPPOSITE PORT' AFTER UNLOAD/
020645  101 124 124  EM15:   .ASCIZ  /ATTENTION BIT NOT SET ON PORT WHICH ISSUED 'UNLOAD'/
020731  104 122 111  EM16:   .ASCII  /DRIVE NOT IN NEUTRAL AFTER UNLOAD WITH 'CONTROLLER/<CR><LF>
021015  123 105 114           .ASCIZ  @SELECT' SWITCH MOVED FROM 'A/B @
021055  104 122 111  EM17:   .ASCIZ  /DRIVE LOCKED ON PORT 'A' BY SWITCH WHILE CYCLED UP/
021140  104 122 111  EM20:   .ASCIZ  /DRIVE LOCKED ON PORT 'B' BY SWITCH WHILE CYCLED UP/
021223  123 124 101  EM21:   .ASCIZ  /STATUS INCORRECT FOR PORT AFTER CYCLE UP/
021274  122 105 107  EM22:   .ASCIZ  /REGISTER CONTENTS SEEN WHEN DRIVE SWITCHED ON 'OPPOSITE' PORT/
021372  047 116 105  EM23:   .ASCIZ  /'NED' NOT SET WHEN RMDS ACCESSED THROUGH PORT NOT SWITCHED/
021465  104 122 111  EM24:   .ASCIZ  /DRIVE SWITCHED TO LOCKED OUT PORT WHEN RELEASED/
021545  122 110 057  EM25:   .ASCIZ  @RH/RM DIDN'T RESPOND TO ADDRESSING@
021610  104 122 111  EM30:   .ASCIZ  /DRIVE NOT SEIZED BY PORT/
021641  127 122 117  EM31:   .ASCIZ  /WRONG STATUS SEEN BY THE SEIZING PORT/
021707  122 105 107  EM32:   .ASCIZ  /REGISTER CONTENTS WRONG/
021737  103 117 116  EM33:   .ASCIZ  /CONTROL BUS PARITY ERROR READING INDICATED REGISTER/
022023  103 101 116  EM34:   .ASCIZ  /CAN'T ACCESS DRIVE THROUGH EITHER PORT/
022072  104 122 111  EM35:   .ASCIZ  /DRIVE NOT IN NEUTRAL AFTER RELEASE - REQUEST NOT SET/
022157  104 122 111  EM36:   .ASCIZ  /DRIVE NOT IN NEUTRAL AFTER TIMEOUT - REQUEST NOT SET/
022244  122 105 107  EM37:   .ASCIZ  /REGISTER CONTENTS WRONG AFTER RELEASE OR TIMEOUT/
022325  104 122 111  EM40:   .ASCIZ  /DRIVE IN NEUTRAL AFTER RELEASE - REQUEST SET/
022402  122 105 107  EM41:   .ASCIZ  /REGISTER WRONG AFTER RELEASE WITH REQUEST SET/
```

```
022460     124     105     123  DH1:      .ASCIZ  /TEST #  ERR PC  PORT #  REG ADR CONTENTS/
022531     124     105     123  DH2:      .ASCIZ  /TEST #  ERR PC  PORT #  REG ADR GOOD     BAD/
022605     124     105     123  DH5:      .ASCIZ  /TEST #  ERR PC  REG ADR PORT A  PORT B/
022654     124     105     123  DH6:      .ASCIZ  /TEST #  ERR PC  PORT #/
022703     124     105     123  DH7:      .ASCIZ  /TEST #  ERR PC  PORT #  TIME (IN MS)/
022750     040     040     040  DH13:     .ASCII  /                SEIZE/<CRLF>
022776     124     105     123            .ASCIZ  /TEST #  ERR PC  PORT #/
023025     040     040     040  DH14:     .ASCII  /                SEIZE   ERROR/<CRLF>
023063     124     105     123            .ASCIZ  /TEST #  ERR PC  PORT #  PORT #  REG ADR CONTENTS/
023144     124     105     123  DH17:     .ASCIZ  /TEST #  ERR PC/
023163     040     040     040  DH24:     .ASCII  /                LOCKED  SWITCHED TO/<CRLF>
023227     124     105     123            .ASCIZ  /TEST #  ERR PC  PORT #  PORT #/
023266     044     122     115  DH25:     .ASCIZ  /$RMADR/
023275     040     040     040  DH30:     .ASCII  /                SEIZE   ERROR/<CRLF>
023333     124     105     123            .ASCIZ  /TEST #  ERR PC  PORT #  PORT #  REG ADR GOOD    BAD/
023417     040     040     040  DH34:     .ASCII  /                PORT A  PORT B/<CRLF>
023456     124     105     123            .ASCIZ  /TEST #  ERR PC  RMDS    RMDS/
023513     040     040     040  DH35:     .ASCII  /                RELSNG  ERROR/<CRLF>
023551     124     105     123            .ASCIZ  /TEST #  ERR PC  PORT #  PORT #/
023610     040     040     040  DH37:     .ASCII  /                RELSNG  ERROR/<CRLF>
023646     124     105     123            .ASCIZ  /TEST #  ERR PC  PORT #  PORT #  REG ADR GOOD    BAD/
023732     040     040     040  DH40:     .ASCII  /                RELSNG  RQSTNG/<CRLF>
023771     124     105     123            .ASCIZ  /TEST #  ERR PC  PORT #  PORT #/

                                .EVEN

024030  001242  001116  001234  DT1:      .WORD   TSTNUM,$ERRPC,PTNBR,$BDADR,$BDDAT,0
024044  001242  001116  001234  DT2:      .WORD   TSTNUM,$ERRPC,PTNBR,$BDADR,$GDDAT,$BDDAT,0
024062  001242  001116  001122  DT5:      .WORD   TSTNUM,$ERRPC,$BDADR,$GDDAT,$BDDAT,0
024076  001242  001116  001234  DT6:      .WORD   TSTNUM,$ERRPC,PTNBR,0
024106  001242  001116  001234  DT7:      .WORD   TSTNUM,$ERRPC,PTNBR,TIME,0
024120  001242  001116  001236  DT13:     .WORD   TSTNUM,$ERRPC,SEIZPT,0
024130  001242  001116  001236  DT14:     .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,$BDADR,$BDDAT,0
024146  001242  001116  000000  DT17:     .WORD   TSTNUM,$ERRPC,0
024154  001242  001116  001236  DT24:     .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,0
024166  001272  000000          DT25:     .WORD   $RMADR,0
024172  001242  001116  001236  DT30:     .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,$BDADR,$GDDAT,$BDDAT,0
024212  001242  001116  001170  DT34:     .WORD   TSTNUM,$ERRPC,$TMP2,$TMP3,0
024224  001242  001116  001236  DT35:     .WORD   TSTNUM,$ERRPC,SEIZPT,PTNBR,0
024236  001242  001116  001236  DT40:     .WORD   TSTNUM,$ERRPC,SEIZPT,OPPRT,0

024250     000     000     001  DF1:      .BYTE   0,0,1,0,0
024255     000     000     001  DF2:      .BYTE   0,0,1,0,0,0
024263     000     000     000  DF5:      .BYTE   0,0,0,0,0
024270     000     000     001  DF6:      .BYTE   0,0,1
024273     000     000     001  DF7:      .BYTE   0,0,1,1
024277     00C     000     001  DF14:     .BYTE   0,0,1,1,0,0
024305     000     000          DF17:     .BYTE   0,0
024307     000                  DF25:     .BYTE   0
024310     000     000     001  DF30:     .BYTE   0,0,1,1,0,0,0
024317     000     000     000  DF34:     .BYTE   0,0,0,0

                                .EVEN
```

SEQ 0070

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

024324  003132                  TSTADR: .WORD   TST1             ;STARTING ADDRESS OF TEST 1
024326  004476                          .WORD   TST2             ;STARTING ADDRESS OF TEST 2
024330  005260                          .WORD   TST3             ;STARTING ADDRESS OF TEST 3
024332  006042                          .WORD   TST4             ;STARTING ADDRESS OF TEST 4
024334  006760                          .WORD   TST5             ;STARTING ADDRESS OF TEST 5
024336  007676                          .WORD   TST6             ;STARTING ADDRESS OF TEST 6
024340  010744                          .WORD   TST7             ;STARTING ADDRESS OF TEST 7
024342  012060                          .WORD   TST10            ;STARTING ADDRESS OF TEST 10

                                ;ATTENTION BIT TABLE

024344      001                 ATABIT: .BYTE   1                ;ATTENTION BIT FOR DRIVE 0
024345      002                         .BYTE   2                ;ATTENTION BIT FOR DRIVE 1
024346      004                         .BYTE   4                ;ATTENTION BIT FOR DRIVE 2
024347      010                         .BYTE   10               ;ATTENTION BIT FOR DRIVE 3
024350      020                         .BYTE   20               ;ATTENTION BIT FOR DRIVE 4
024351      040                         .BYTE   40               ;ATTENTION BIT FOR DRIVE 5
024352      100                         .BYTE   100              ;ATTENTION BIT FOR DRIVE 6
024353      200                         .BYTE   200              ;ATTENTION BIT FOR DRIVE 7

024354  000011                  MAXTN:  .WORD   11               ;MAXIMUM TEST NUMBER

        000200                  .END    200
```

```
ADDRIS  017200        CR    = 000015        DT17    024146        F3    = 000020        PR6   = 000300
ADRERR  016771        CRLF  = 000200        DT2     024044        F4    = 000040        PR7   = 000340
AOE   = 001000        CYCLED  017567        DT24    024154        GO    = 000001        PS    = 177776
ASR1    001232        CYCLEU  017610        DT25    024166        GRV   = 000010        PSEL  = 002000
ATA   = 100000        DCK   = 100000        DT30    024172        GTSWR = 104406        PSW   = 177776
ATABIT  024344        DDISP = 177570        DT34    024212        HCE   = 000200        PTNBR   001234
AT0   = 000001        DE1   = 000040        DT35    024224        HCI   = 002000        PWRVEC= 000024
AT1   = 000002        DFF20 = 000002        DT40    024236        HCRC  = 000400        RDCHR = 104410
AT2   = 000004        DF1     024250        DT5     024062        HT    = 000011        RDLIN = 104411
AT3   = 000010        DF14    024277        DT6     024076        IAE   = 002000        RDOCT = 104412
AT4   = 000020        DF17    024305        DT7     024106        IE    = 000100        RDY   = 000200
AT5   = 000040        DF2     024255        DVA   = 004000        ILF   = 000001        RELERR  001250
AT6   = 000100        DF25    024307        DVC   = 000200        ILR   = 000002        RELOK = 000001
AT7   = 000200        DF30    024310        ECH   = 000100        IOTVEC= 000020        RESREG= 104414
A16   = 000400        DF34    024317        ECI   = 004000        IR    = 000100        RESVEC= 000010
A17   = 001000        DF5     024263        EMTVEC= 000030        IVC   = 010000        RMAS  = 000016
BADNO   017152        DF6     024270        EM1     017703        KYBCTL  001266        RMBA  = 000004
BADTMO  001706        DF7     024273        EM10    020274        LBC   = 002000        RMCS1 = 000000
BAI   = 000010        DH1     022460        EM11    020362        LBT   = 002000        RMCS2 = 000010
BIT0  = 000001        DH13    022750        EM12    020427        LF    = 000012        RMDA  = 000006
BIT00 = 000001        DH14    023025        EM13    020476        LSC   = 004000        RMDB  = 000022
BIT01 = 000002        DH17    023144        EM14    020563        MAXTN   024354        RMDC  = 000034
BIT02 = 000004        DH2     022531        EM15    020645        MCPE  = 020000        RMDS  = 000012
BIT03 = 000010        DH24    023163        EM16    020731        MDPE  = 000400        RMDT  = 000026
BIT04 = 000020        DH25    023266        EM17    021055        MOH   = 020000        RMEC1 = 000044
BIT05 = 000040        DH30    023275        EM2     017751        MOL   = 010000        RMEC2 = 000046
BIT06 = 000100        DH34    023417        EM20    021140        MXF   = 001000        RMER1 = 000014
BIT07 = 000200        DH35    023513        EM21    021223        NBA   = 100000        RMER2 = 000042
BIT08 = 000400        DH37    023610        EM22    021274        NED   = 010000        RMLA  = 000020
BIT09 = 001000        DH40    023732        EM23    021372        NEM   = 004000        RMMR1 = 000024
BIT1  = 000002        DH5     022605        EM24    021465        NOATA = 000000        RMMR2 = 000040
BIT10 = 002000        DH6     022654        EM25    021545        NOCLOC  017065        RMOF  = 000032
BIT11 = 004000        DH7     022703        EM3     017772        NOSEIZ  001246        RMR   = 000004
BIT12 = 010000        DIGB  = 000004        EM30    021610        NTRH11  017235        RMSN  = 000030
BIT13 = 020000        DISPLA  001142        EM31    021641        OFD   = 000200        RMWC  = 000002
BIT14 = 040000        DISPRE  000174        EM32    021707        OPE   = 020000        R6    =%000006
BIT15 = 100000        DLT   = 100000        EM33    021737        OPI   = 020000        R7    =%000007
BIT2  = 000004        DL64  = 000020        EM34    022023        OPPRT   001240        SAVREG= 104413
BIT3  = 000010        DMD   = 000001        EM35    022072        OR    = 000200        SC    = 100000
BIT4  = 000020        DPE   = 000010        EM36    022157        PAR   = 000010        SCOPE = 000004
BIT5  = 000040        DPR   = 000400        EM37    022244        PAT   = 000020        SC0   = 000100
BIT6  = 000100        DRQ   = 004000        EM4     020051        PGE   = 002000        SC1   = 000200
BIT7  = 000200        DRY   = 000200        EM40    022325        PGM   = 001000        SC2   = 000400
BIT8  = 000400        DSWR  = 177570        EM41    022402        PIP   = 020000        SC3   = 001000
BIT9  = 001000        DTE   = 010000        EM5     020073        PIRQ  = 177772        SC4   = 002000
BPTVEC= 000014        DT00  = 000001        EM6     020155        PIRQVE= 000240        SEIZPT  001236
CHANGE  003010        DT01  = 000002        EM7     020227        PORTA   001224        SKI   = 100000
CHGADR  001270        DT02  = 000004        ENTERA  016742        PORTAI  017013        STACK = 001100
CKCLK   013406        DT03  = 000010        ERR   = 040000        PORTB   001226        START   001766
CKCLK1  013456        DT04  = 000020        ERROR = 104000        PORTBI  017040        START1  001774
CKCLK2  013520        DT05  = 000040        ERRVEC= 000004        PORTC   001230        START2  002002
CKCLK3  013530        DT06  = 000100        EXEC    002636        PR0   = 000000        STKLMT= 177774
CKERR   001244        DT07  = 000200        FER   = 000020        PR1   = 000040        SWR     001140
CKSWR = 104407        DT08  = 000400        FMT16 = 010000        PR2   = 000100        SWREG   000176
CLOCK   013540        DT1     024030        F0    = 000002        PR3   = 000140        SWTCHA  017352
CLR   = 000040        DT13    024120        F1    = 000004        PR4   = 000200        SWTCHB  017434
CONTUE  017516        DT14    024130        F2    = 000010        PR5   = 000240        SWTCHN  017264
```

```
SW0    = 000001        TIMEAP  001260        $AUTOB  001134        $ICNT   001104        $SWRMK= 000000
SW00   = 000001        TIMEB   001262        $BDADR  001122        $INTAG  001135        $TIMES  001176
SW01   = 000002        TIMEBP  001264        $BDDAT  001126        $ITEMB  001114        $TKB    001146
SW02   = 000004        TKVEC = 000060        $BELL   001202        $LF     001210        $TKCNT  015254
SW03   = 000010        TOLER   013572        $CHARC  014576        $LKCSB  001214        $TKINT  015264
SW04   = 000020        TPVEC = 000064        $CKSWR  015526        $LKCSR  001212        $TKQEN= 015263
SW05   = 0C0040        TRAPVE= 000034        $CMTAG  001100        $LKS    001220        $TKQIN  015256
SW06   = 000100        TRE   = 040000        $CM1  = 000001        $LLVEC  001222        $TKQOU  015260
SW07   = 000200        TRTVEC= 000014        $CM2  = 000002        $LPADR  001106        $TKQSR  015262
SW08   = 000400        TSTADR  024324        $CM3  = 000001        $LPERR  001110        $TKS    001144
SW09   = 001000        TSTNUM  001242        $CM4  = 000005        $LPVEC  001216        $TKSRV  015334
SW1    = 000002        TST1    003132        $CNTLG  016370        $MNEW   016406        $TMP0   001164
SW10   = 002000        TST1AA  003120        $CNTLU  016363        $MSWR   016375        $TMP1   001166
SW11   = 004000        TST10   012060        $CRLF   001207        $MXCNT  014000        $TMP2   001170
SW12   = 010000        TST11   013164        $DBLK   015244        $NULL   001154        $TMP3   001172
SW13   = 020000        TST2    004476        $DOAGN  013376        $NWTST= 000000        $TMP4   001174
SW14   = 040000        TST3    005260        $DTBL   015234        $OCNT   015024        $TN   = 000012
SW15   = 100000        TST4    006042        $ENDAD  013366        $OMODE  015026        $TPB    001152
SW2    = 000004        TST5    006760        $ENDCT  013232        $OVER   013764        $TPFLG  001157
SW3    = 000010        TST6    007676        $ENULL  013402        $PASS   001100        $TPS    001150
SW4    = 000020        TST7    010744        $EOP    013166        $QUES   001206        $TRAP   016654
SW5    = 000040        TYPDS = 104405        $EOPCT  013224        $RDCHR  016030        $TRAP2  016676
SW6    = 000100        TYPE  = 104401        $ERFLG  001103        $RDLIN  016120        $TRP  = 000015
SW7    = 000200        TYPOC = 104402        $ERMAX  001115        $RDOCT  016420        $TRPAD  016710
SW8    = 000400        TYPON = 104404        $ERROR  014002        $RDSZ = 000007        $TSTNM  001102
SW9    = 001000        TYPOS = 104403        $ERRPC  001116        $REGAD  001160        $TTYIN  016354
TAP    = 040000        UNS   = 040000        $ERRTB  001276        $REG0   001162        $TYPDS  015030
TBITVE= 000014         UPE   = 020000        $ERRTY  014134        $RESRE  016616        $TYPE   014310
TESTNO  017132         U0    = 000001        $ERTTL  001112        $RMADR  001272        $TYPEC  014460
TEST1   003170         U1    = 000002        $ESCAP  001200        $RMVEC  001274        $TYPEX  014600
TEST10  012116         U3    = 000004        $FILLC  001156        $RTNAD  013400        $TYPOC  014626
TEST2   004534         VV    = 000100        $FILLS  001155        $SAVRE  016560        $TYPON  014642
TEST3   005316         VVSET = 000001        $GDADR  001120        $SCOPE  013620        $TYPOS  014602
TEST4   006100         WATCH   001254        $GDDAT  001124        $SETUP= 000127        $XOFF = 000023
TEST5   007016         WCE   = 040000        $GET42  013356        $STUP = 177777        $XON  = 000021
TEST6   007734         WCF   = 000040        $GTSWR  015616        $SVLAD  013754        $XTSTR  013632
TEST7   011002         WLE   = 004000        $HD   = 000000        $SVPC = 000210        $$GET4= 000000
TIME    001252         WRL   = 004000        $HIOCT  016556        $SWR  = 166000        $0FILL  015025
TIMEA   001256

. ABS.  024356     000
        000000     001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  44816 WORDS  ( 176 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  69 PAGES
CZRMSA.BIN,CZRMSA/C=CZRMSA.DOC,CZRMSA,SYSMAC/M
```

$TIMES 5-0# 8-24* 8-140* 8-216* 8-217* 8-275* 8-276* K 6 8-296* 8-446* 8-446* 8-447* 8-447* 8-459* 10-4

8-446* 8-446* 8-446* 8-447* 8-447* 8-447* 8-447* 8-447* 8-447* 8-447* 8-447*

B 7

SEQ 0079

ERR 4-599#

SEQ 0083

PGM	4-594#	8-166	8-166	8-166	8-166	8-166	8-166 H 7	8-216	8-216	8-217	8-217	8-275	8-275	8-276 SEQ 0085

SW8 4-528#

L 7

SEQ 0089

VV	4-591#	8-216	8-216	8-216	8-216	8-216	8-217 N 7	8-217	8-217	8-217	8-217	8-275	8-275	8-275 SEQ 0091

	8-275	8-276	8-276	8-276	8-276	8-303	8-303	8-303	8-308	8-308	8-308		
VVSET	7-230#	8-216	8-216	8-216	8-216	8-216	8-217	8-217	8-217	8-217	8-217	8-221#	8-275	8-275
	8-275	8-275	8-275	8-275	8-275	8-275	8-276	8-276	8-276	8-276	8-276	8-276	8-276	8-276
	8-303	8-303	8-303	8-303	8-303	8-303	8-308	8-308	8-308	8-308	8-308	8-308		
WATCH	6-0#	8-275	8-275*	8-276	8-276*	8-446	8-446*	8-447	8-447*	9-32	9-34*	9-36*		
WCE	4-565#													
WCF	4-609#													
WLE	4-615#													
WRL	4-596#													

```
$$CMRE    4-758#   5-0
$$CMTM    4-758#   5-0      5-0      5-0      5-0      5-0
$$ESCA    4-528#
$$NEWT    4-528#   8-140    8-216    8-217    8-275    8-276    8-296    8-446    8-447    8-451
$$SET    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13#
$$SKIP    4-528#   8-446
.$ACT1    4-516#   4-720
.$CATC    4-517#   4-716
.$CMTA    4-517#   4-758
.$EOP     4-517#   8-459
.$ERRO    4-517#  10-5
.$ERRT    4-516#  10-6
.$RDOC    4-518#  10-11
.$READ    4-518#  10-10
.$SAVE    4-518#  10-12
.$SCOP    4-517#  10-4
.$TRAP    4-518#  10-13
.$TYPD    4-518#  10-9
.$TYPE    4-517#  10-7
.$TYPO    4-518#  10-8
.EQUAT    4-516#   4-528
.HEADE    4-516#   4-525
.SETUP    4-516#   4-714
.SWRHI    4-516#   4-526
.SWRLO    4-516#   4-526#
AA        8-224#   8-275    8-276
BB        8-182#   8-216    8-217
CHECK     4-130#   8-151    8-151    8-158    8-158    8-166    8-166    8-166    8-166    8-216    8-217    8-446    8-446    8-446
          8-446    8-447    8-447    8-447    8-447
CLRATA    4-425#   8-298
COMMEN    4-528#
ENDCOM    4-528#
ERROR     4-8#     8-117    8-151    8-151    8-158    8-158    8-166    8-166    8-166    8-166    8-176    8-216    8-216    8-216
          8-216    8-216    8-217    8-217    8-217    8-217    8-217    8-275    8-275    8-275    8-275    8-275    8-275    8-275
          8-275    8-276    8-276    8-276    8-276    8-276    8-276    8-276    8-276    8-303    8-303    8-303    8-303    8-308
          8-308    8-308    8-308    8-446    8-446    8-446    8-446    8-447    8-447    8-447    8-447
ESCAPE    4-528#
FF        8-314#   8-446    8-447
GETPRI    4-528#
GETSWR    4-528#   8-30     8-30#
MORETA    4-722#   5-0
MSG       8-123#   8-140    8-216    8-216#   8-217    8-217#   8-275    8-275#   8-276    8-276#   8-279#   8-296    8-446    8-446#
          8-447    8-447#
MULT      4-528#
NEUTRA    4-281#   8-216    8-217    8-275    8-276    8-303    8-308
NEWTST    4-528#   8-140    8-216    8-217    8-275    8-276    8-296    8-446    8-447    8-451
NWTST     4-479#   8-140    8-216    8-217    8-275    8-276    8-296    8-446    8-447
POP       4-528#  10-9     10-11    10-12
PUSH      4-528#  10-9     10-11    10-12
RELEAS    4-210#   8-216    8-217
REPORT    4-528#
SEIZE     4-40#    8-275    8-276
SELECT    4-255#   8-151    8-151    8-158    8-158    8-166    8-166    8-216    8-216    8-217    8-217    8-275    8-275    8-275
          8-276    8-276    8-276    8-446    8-446    8-446    8-446    8-446    8-446    8-447    8-447    8-447    8-447    8-447
          8-447
SETATA    4-458#
```

```
SETPRI   4-528#   8-28    10-10
SETTRA  10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13    10-13#
SETUP    4-528#   8-24
SKIP     4-528#   8-446
SLASH    4-528#
STARS    4-528#   4-720    5-0      5-0      8-140    8-140    8-216    8-216    8-217    8-217    8-275    8-275    8-276    8-276
         8-296    8-296    8-446    8-446    8-447    8-447    8-449    8-451    8-459    9-2      10-4     10-5     10-6     10-7
        10-8     10-9     10-10    10-10    10-10    10-10    10-10    10-11    10-12    10-13
SWITCH   4-381#
SWRSU    4-528#   8-24     8-24#
TIMER    4-505#   8-275    8-276
TRMTRP  10-13#
TYPBIN   4-528#
TYPDEC   4-528#   8-459    8-459    10-6
TYPNAM   4-528#   8-30
TYPNUM   4-528#
TYPOCS   4-528#   8-48     8-50
TYPOCT   4-528#   10-6     10-6     10-10
TYPTXT   4-528#   8-6      8-459    8-459
USEREO   8-453#   8-459
```