# DT07

UNIBUS SWITCH DIAG

## CRDTAA0

AH-F170A-MC

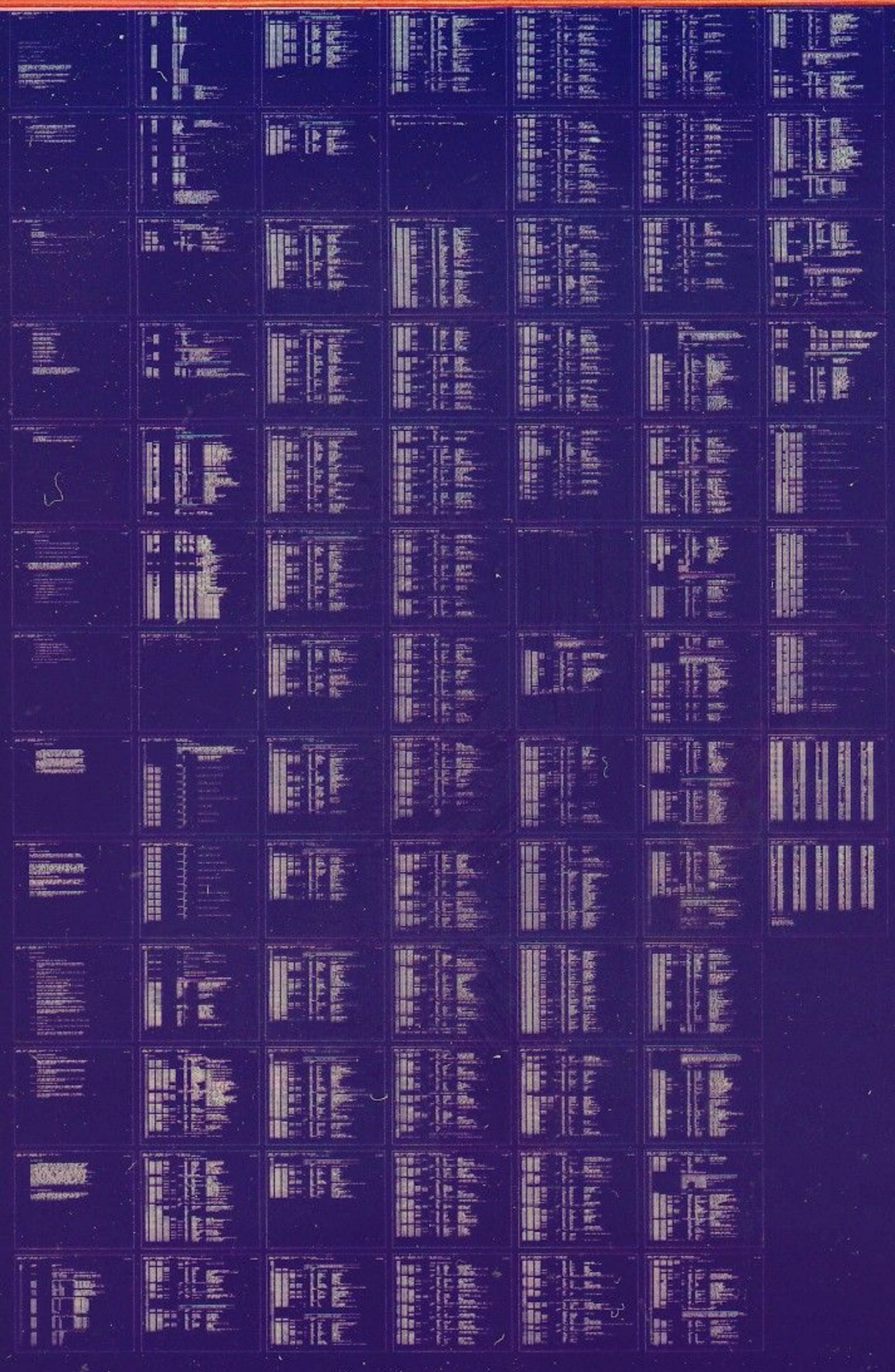


FICHE 1 OF 1

MAR 1979

digital

MADE IN USA



SEQ 0001

```
        .NLIST LOC,SEQ,BIN
        .REM_
```

PRODUCT CODE: AC-F169A-MC

PRODUCT NAME: CRDTAA0 DT07 UNIBUS SWITCH DIAG

DATE: 27 JANUARY 1979

AUTHOR(S): JAMES DUPRE

MAINTAINER: C.S.S. LVP GROUP

1 .0 ABSTRACT:

THIS DIAGNOSTIC IS DESIGNED TO TEST THE FUNCTIONS AND CAPABILITIES OF THE DT07 UNIBUS SWITCH. THE DIAGNOSTIC IS DIVIDED INTO THREE MAJOR SECTIONS AS FOLLOWS:

MONOPORT; THIS SECTION TESTS A SINGLE PORT WITHOUT INTERACTION WITH OTHER PORTS.

MULTIPORT; THIS SECTION TESTS THE DT07 PORT WITH THE OTHER PORTS IN THE SYSTEM.

MANUAL INTERVENTION; THIS SECTION TESTS CERTAIN FUNCTIONS WHICH REQUIRE OPERATOR ASSISTANCE.

2 .0 REQUIREMENTS:

2 .1 EQUIPMENT

PDP 11-COMPUTER
ASR-33 TELETYPE OR EQUIVALENT
DT07 PORT MODULE
MANUAL CONTROL PANEL (OPTIONAL BUT NEEDED FOR MANUAL INTERVENTION TEST!)

2 .2 PROGRAM STORAGE

PROGRAM REQUIRES 8 K WORDS OF MEMORY

2 .3 SOFTWARE

ABSOLUTE LOADER OR OTHER INPUT MEDIUM

3 .0 SWITCH REGISTER ASSIGNMENTS:

SW08(1)=LOOP ON TEST IN SWR<7:0>
SW08(0)=DO ALL TESTS IN SEQUENCE

SW09(1)=LOOP ON ERROR
SW09(0)=CONTINUE ON ERROR

SW10(1)=BELL ON ERROR
SW10(0)=NO BELL ON ERROR

SW11(1)=INHIBIT ITERATIONS
SW11(0)=ALLOW ITERATIONS

SW13(1)=INHIBIT ERROR TYPEOUTS
SW13(0)=ALLOW ERROR TYPEOUTS

SW14(1)=LOOP ON TEST
SW14(0)=DON'T LOOP ON TEST

SW15(1)=HALT ON ERROR
SW15(0)=CONTINUE ON ERROR

NOTE:

COMPUTERS WITHOUT A HARDWARE SWITCH REGISTER HAVE A SOFTWARE SWITCH REGISTER LOCATED IN MEMORY AT LOCATION 176 CALLED SWREG. THIS LOCATION CAN BE CHANGED EITHER MANUALLY OR BY TYPING THE CNTL+G KEYES TOGETHER THEN RESPONDING TO THE TERMINAL DIALOGUE.

4 .0 LOADING PROCEDURE:

THE STANDARD PROCEDURE FOR LOADING ABSOLUTE BINARY TAPES
SHOULD BE USED.
IF THE PROGRAM RESIDES ON A MASS STORAGE DEVICE SUCH AS
MAGTAPE OR DISK REFER TO THE XXDP USER'S GUIDE.

5 .0 STARTING PROCEDURE:

5 .1 STARTING ADDRESSES

200.INITIALIZES PROGRAM AND RUNS MONOPORT SECTION*

204.INITIALIZES PROGRAM AND RUNS MULTIPORT SECTION**

210.INPUTS PARAMETERS AND ALLOWS TEST SELECTION

214.INITIALIZES PROGRAM AND RUNS MANUAL INTERVENTION SECTION

* DEFAULT PARAMETERS WILL BE USED UNLESS AND UNTIL ACTUAL PARAMETERS
ARE INSERTED VIA ADDRESS 210 OR BY RUNNING THE MULTIPORT TEST.
** IF ACTUAL PARAMETERS HAVE NOT BEEN INSERTED AT LEAST ONCE THE
PROGRAM WILL JUMP TO THE PARAMETER INPUT ROUTINE FIRST; THEN
YOU MAY SELECT THE MULTIPORT SECTION.

5 .2 STARTING SEQUENCES

5 .2.1 STARTING SEQUENCE UNDER XXDP(CHAIN),APT OR ACT

START AT ADDRESS 200 ONLY. (DEFAULT PARAMETERS)

5 .2.2 NORMAL STARTING SEQUENCE

LOAD PROGRAM ACCORDING TO LOADING PROCEDURE

LOAD AND START ADDRESS 210

INPUT ACTUAL PARAMETERS AND SELECT TEST

5 .3 RESTART ADDRESSES

RESTART ADDRESSES ARE THE SAME AS STARTING ADDRESSES

6 .0 PRELIMINARY OPERATIONS:

6 .1 DEVICE ADDRESSES AND VECTOR ADDRESSES

THE STANDARD DEVICE ADDRESS IS 177420.

THE STANDARD DEVICE VECTOR ADDRESS IS 350.

THE STANDARD DEVICE PRIORITY LEVEL IS 7.

6 .2 DEVICE/OPTION SETUP

INSURE THAT THE DT07(S) ARE IN PROGRAMMABLE MODE.

6 .3 PRELIMINARY PROGRAMS NEEDED

NONE

7 .0 OPERATIONAL PROCEDURES:

NORMAL OPERATION WOULD BE TO LOAD THIS PROGRAM, INSURE THAT ALL DT07 PORTS ARE IN THE PROGRAMMABLE MODE, RUN THE MONOPORT TEST ON EACH PORT IN THE SYSTEM,THEN RUN THE MULTIPORT TEST ON ALL PORTS IN THE SYSTEM.
IF YOU HAVE A MANUAL CONTROL PANEL, RUN THE MANUAL INTERVENTION TEST.

THE PARAMETER INPUT ROUTINE IS TO BE USED WHENEVER YOU WISH TO INPUT OR CHANGE DIAGNOSTIC PARAMETERS INCLUDING DEVICE ADDRESS,VECTOR ADDRESS,INTERRUPT LEVEL, NUMBER AND I.D.#S OF OTHER PORTS TO BE TESTED.
THE PARAMETER INPUT ROUTINE WILL BE ENTERED AUTOMATICALY FROM THE MULTIPORT TEST IF NO PARAMETERS HAVE BEEN ENTERED PREVIOUSLY.

WHEN USING THE MULTIPORT SECTION, AFTER PARAMETERS HAVE BEEN ENTERED OR WHEN RESTARTING THE TEST, THE CPU WITH THE LOWEST NUMBERED DT07 PORT MUST BE STARTED LAST.

---

8.0 ERRORS:

8.1 MONOPORT SECTION ERRORS

ALL ERRORS IN THE MONOPORT SECTION HAVE THE GENERAL FORMAT;
ERROR PC,TEST#,GOOD DATA,BAD DATA,BAD DATA ADDRESS. CONTROL
OF ERROR PRINTOUT AND LOOPING IS BY THE SWITCH REGISTER OPTIONS.

8.2 MULTIPORT SECTION ERRORS

ERRORS IN THE MULTIPORT SECTION PRINT AN ERROR ID.#, A MESSAGE
BRIEFLY DESCRIBING THE ERROR AND THE CSR CONTENTS. ALL ERRORS
IN THIS SECTION ARE FATAL AND, THEREFORE, THE TEST MUST BE RESTARTED
ON ERROR. THERE IS NO ERROR LOOPING AND ERROR PRINTOUT IS AUTO-
MATIC. ERROR ID.#S 1A THRU 5A IDENTIFY ERRORS IN THE DT07 MODE
MULTIPORT TEST AND ERROR ID.#S 6 THRU 16 IDENTIFY ERRORS IN THE
DT03 MODE MULTIPORT TEST

8.3 MANUAL INTERVENTION SECTION ERRORS

ERRORS IN THIS SECTION PRINT AN ERROR ID.#, A BRIEF MESSAGE DES-
CRIBING THE ERROR, AND THE CSR CONTENTS AT ERROR TIME. THERE IS
NO ERROR LOOPING AND ERROR PRINTOUT IS AUTOMATIC; HOWEVER,
PRESSING CONTINUE, AFTER ERROR, WILL CAUSE THE FAILING SUBTEST TO
REPEAT EXECUTION.

8.4 MISCELLANEOUS ERRORS

THERE ARE TWO SUBROUTINES USED TO DETERMINE DEVICE MODE AND DEVICE
TIMING. ERRORS IN THESE ROUTINES ARE SELF-EXPLANATORY AND ARE
FATAL. THESE ERRORS USUALLY OCCUR BECAUSE ONE OF THE PORTS
WAS LEFT IN THE MANUAL MODE.

9.0 TEST DESCRIPTIONS:

9.1 MONOPORT TESTS

TEST1. TEST POST-RESET BIT PATTERN IN CSR.

THIS TEST CHECKS THE CONTROL STATUS REGISTER CONTENTS AFTER RESET TO INSURE THAT ANY RESET-CLEARABLE BITS ARE CLEARED.

TEST2. TEST READ/WRITE BITS IN CSR.

THIS TEST CHECKS THAT BITS 0 AND 6 CAN BE SET AND CLEARED IN THE CSR.

TEST3. TEST BYTE ACCESS IN CSR.

THIS TEST CHECKS THE BYTE ACCESS CAPABILITY OF THE CSR.

TEST4. CONNECT TEST WITHOUT INTERUPT ENABLE.

THIS TEST VERIFIES THE PROPER OPERATION OF THE CONNECT PROCESS WITH INTERUPTS DISABLED.

TEST5. RELEASE TEST WITHOUT INTERUPT ENABLE.

THIS TEST VERIFIES THE PROPER OPERATION OF THE RELEASE PROCESS WITH INTERUPTS DISABLED.

TEST6. CONNECT FAILURE TEST WITHOUT INTERUPT ENABLE.

THIS TEST CHECKS THE PROPER OPERATION OF THE DEVICE WHEN THE CONNECTION PROCESS CANNOT CONNECT TO THE SHARED BUS WITHOUT INTERUPTS ENABLED.

TEST7. RELEASE TEST WITHOUT BUS MASTERSHIP OR INTERUPT ENABLE.

THIS TEST CHECKS THAT THE DEVICE WILL RELEASE THE SHARED BUS WHEN UNIBUS MASTERSHIP CANNOT BE OBTAINED AND INTERUPTS ARE DISABLED.

TEST10. CONNECT TEST

THIS TEST VERIFIES THE CONNECT PROCESS UNDER INTERUPT CONTROL.

TEST11. RELEASE TEST

THIS TEST VERIFIES THE RELEASE PROCESS UNDER INTERUPT CONTROL.

TEST12. CONNECT FAILURE TEST

THIS TEST VERIFIES THE PROPER OPERATION OF THE CONNECT PROCESS,UNDER INTERUPT CONTROL,WHEN THE DEVICE CANNOT

OBTAIN BUS MASTERSHIP.

TEST13. RELEASE TEST WITHOUT BUS MASTERSHIP

THIS TEST VERIFIES THE THE PROPER OPERATION OF THE RELEASE PROCESS, UNDER INTERUPT CONTROL, WHEN THE DEVICE CANNOT OBTAIN BUS MASTERSHIP.

TEST14. RESET (RST) TEST

THIS TEST CHECKS THE OPERATION OF THE DEVICE RESET BIT (RST) IF AN EXTERNAL DEVICE REGISTER HAS BEEN INSERTED IN THE PARAMETER BLOCK.

TEST15. RESET HOLD (HLD) TEST

THIS TEST CHECKS THAT THE HLD BIT WILL HOLD THE SHARED BUS DURING AND AFTER A BUS RESET.

TEST16. HI-SPEED SWITCHING TEST

THIS TEST CAUSES THE DEVICE TO CONNECT AND DISCONNECT AT HIGH SPEED 1000 TIMES WHILE CHECKING FOR ERRORS.

TEST17. RESET-IN-NEUTRAL TEST

THIS TEST CHECKS FOR RESET-IN-NEUTRAL IF AN EXTERNAL DEVICE REGISTER IS PRESENT AND PRINTS OUT THE RESULT ON THE FIRST PASS.

9.2 MULTIPORT TEST

THE MULTIPORT TEST IS ACTUALLY TWO SEPARATE TESTS: ONE FOR THE DT07 IN DT03 MODE AND ONE FOR THE DT07 IN DT07 MODE. IN DT07 MODE THE PORT I.D.#S OF ALL THE PORTS TO BE TESTED ARE ARRANGED IN ASCENDING NUMERICAL ORDER THEN A MENU OF FUNCTIONS TO BE PERFORMED IS CREATED. EACH DT07 PORT WILL HAVE A TURN AS MASTER WHILE THE OTHER PORTS ARE SLAVES. THERE ARE TWO MASTER FUNCTIONS AND THREE SLAVE FUNCTIONS; THESE BEING: FUNCTION1 (MASTER) DETECT SELECTED SLAVE REQUEST AND REJECT IT, FUNCTION2 (MASTER) DETECT SELECTED SLAVE REQUEST AND IGNORE IT, FUNCTION3 (SLAVE) SEND REQUEST AND EXPECT REJECTION, FUNCTION4 (SLAVE) SEND REQUEST AND EXPECT CONNECTION, AND FUNCTION5 (SLAVE) OBSERVE REQUEST INDICATORS. IN DT03 MODE BOTH DT07S HAVE A TURN AS MASTER THEN AS SLAVE. THERE ARE TWO MASTER FUNCTIONS AND TWO SLAVE FUNCTIONS; THESE BEING: 1. RECEIVE REQUEST AND RELEASE THE SHARED BUS, 2. SEND REQUEST AND EXPECT REJECTION, 3. SEND REQUEST AND EXPECT CONNECTION, AND 4. DETECT REQUEST AND REJECT IT.

9.3 MANUAL INTERVENTION TEST

THE MANUAL INTERVENTION TEST VERIFIES THE PROPER OPERATION OF CERTAIN DEVICE FEATURES THAT REQUIRE MANUAL ASSISTANCE TO TEST. THIS TEST ALSO REQUIRES THE OPTIONAL MANUAL CONTROL PANEL FOR PROPER OPERATION. THE FEATURES TESTED ARE: MANUAL MODE, PORT POWER OK AND SWITCHED BUS POWER FAIL.

```
                        .SBTTL  BASIC DEFINITIONS
(1)
(1)                     ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)     001100          STACK=  1100
(1)                     .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
(1)                     .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)                     ;*MISCELLANEOUS DEFINITIONS
(1)     000011          HT=     11              ;;CODE FOR HORIZONTAL TAB
(1)     000012          LF=     12              ;;CODE FOR LINE FEED
(1)     000015          CR=     15              ;;CODE FOR CARRIAGE RETURN
(1)     000200          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)     177776          PS=     177776          ;;PROCESSOR STATUS WORD
(1)                     .EQUIV  PS,PSW
(1)     177774          STKLMT= 177774          ;;STACK LIMIT REGISTER
(1)     177772          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)     177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
(1)     177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
(1)
(1)                     ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)     000000          R0=     %0              ;;GENERAL REGISTER
(1)     000001          R1=     %1              ;;GENERAL REGISTER
(1)     000002          R2=     %2              ;;GENERAL REGISTER
(1)     000003          R3=     %3              ;;GENERAL REGISTER
(1)     000004          R4=     %4              ;;GENERAL REGISTER
(1)     000005          R5=     %5              ;;GENERAL REGISTER
(1)     000006          R6=     %6              ;;GENERAL REGISTER
(1)     000007          R7=     %7              ;;GENERAL REGISTER
(1)     000006          SP=R6                   ;;STACK POINTER
(1)     000007          PC=R7                   ;;PROGRAM COUNTER
(1)
(1)                     ;*PRIORITY LEVEL DEFINITIONS
(1)     000000          PR0=    0               ;;PRIORITY LEVEL 0
(1)     000040          PR1=    40              ;;PRIORITY LEVEL 1
(1)     000100          PR2=    100             ;;PRIORITY LEVEL 2
(1)     000140          PR3=    140             ;;PRIORITY LEVEL 3
(1)     000200          PR4=    200             ;;PRIORITY LEVEL 4
(1)     000240          PR5=    240             ;;PRIORITY LEVEL 5
(1)     000300          PR6=    300             ;;PRIORITY LEVEL 6
(1)     000340          PR7=    340             ;;PRIORITY LEVEL 7
(1)
(1)                     ;*"SWITCH REGISTER" SWITCH DEFINITIONS
(1)     100000          SW15=   100000
(1)     040000          SW14=   40000
(1)     020000          SW13=   20000
(1)     010000          SW12=   10000
(1)     004000          SW11=   4000
(1)     002000          SW10=   2000
(1)     001000          SW09=   1000
(1)     000400          SW08=   400
(1)     000200          SW07=   200
```

```
(1)             000100                  SW06=   100
(1)             000040                  SW05=   40
(1)             000020                  SW04=   20
(1)             000010                  SW03=   10
(1)             000004                  SW02=   4
(1)             000002                  SW01=   2
(1)             000001                  SW00=   1
(1)                                     .EQUIV  SW09,SW9
(1)                                     .EQUIV  SW08,SW8
(1)                                     .EQUIV  SW07,SW7
(1)                                     .EQUIV  SW06,SW6
(1)                                     .EQUIV  SW05,SW5
(1)                                     .EQUIV  SW04,SW4
(1)                                     .EQUIV  SW03,SW3
(1)                                     .EQUIV  SW02,SW2
(1)                                     .EQUIV  SW01,SW1
(1)                                     .EQUIV  SW00,SW0
(1)
(1)                                     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)             100000                  BIT15=  100000
(1)             040000                  BIT14=  40000
(1)             020000                  BIT13=  20000
(1)             010000                  BIT12=  10000
(1)             004000                  BIT11=  4000
(1)             002000                  BIT10=  2000
(1)             001000                  BIT09=  1000
(1)             000400                  BIT08=  400
(1)             000200                  BIT07=  200
(1)             000100                  BIT06=  100
(1)             000040                  BIT05=  40
(1)             000020                  BIT04=  20
(1)             000010                  BIT03=  10
(1)             000004                  BIT02=  4
(1)             000002                  BIT01=  2
(1)             000001                  BIT00=  1
(1)                                     .EQUIV  BIT09,BIT9
(1)                                     .EQUIV  BIT08,BIT8
(1)                                     .EQUIV  BIT07,BIT7
(1)                                     .EQUIV  BIT06,BIT6
(1)                                     .EQUIV  BIT05,BIT5
(1)                                     .EQUIV  BIT04,BIT4
(1)                                     .EQUIV  BIT03,BIT3
(1)                                     .EQUIV  BIT02,BIT2
(1)                                     .EQUIV  BIT01,BIT1
(1)                                     .EQUIV  BIT00,BIT0
(1)
(1)                                     ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)             000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)             000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)             000014                  TBITVEC=14              ;;"T" BIT
(1)             000014                  TRTVEC= 14              ;;TRACE TRAP
(1)             000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)             000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)             000024                  PWRVEC= 24              ;;POWER FAIL
(1)             000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)             000034                  TRAPVEC=34              ;;"TRAP" TRAP
```

```
(1)     000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)     000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
(1)     000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
385     177420                  ABASE=177420            ;BASE DT07 BUS ADDRESS EQUATE
386     160350                  AVECT1=160350           ;BASE DT07 PRIORITY & VECTOR ADDRESS EQUATE
387     001124                  GOOD=$GDDAT
388     001126                  BAD=$BDDAT
389     000001                  REQ=BIT0
390     001122                  BADA=$BDADR
391                             .SBTTL  MEMORY MANAGEMENT DEFINITIONS
(1)
(1)                             ;*KT11 VECTOR ADDRESS
(1)
(1)     000250                  MMVEC=  250
(1)
(1)                             ;*KT11 STATUS REGISTER ADDRESSES
(1)
(1)     177572                  SR0=    177572
(1)     177574                  SR1=    177574
(1)     177576                  SR2=    177576
(1)     172516                  SR3=    172516
(1)
(1)                             ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
(1)
(1)     172300                  KIPDR0= 172300
(1)     172302                  KIPDR1= 172302
(1)     172304                  KIPDR2= 172304
(1)     172306                  KIPDR3= 172306
(1)     172310                  KIPDR4= 172310
(1)     172312                  KIPDR5= 172312
(1)     172314                  KIPDR6= 172314
(1)     172316                  KIPDR7= 172316
(1)
(1)                             ;*KERNEL "I" PAGE ADDRESS REGISTERS
(1)
(1)     172340                  KIPAR0= 172340
(1)     172342                  KIPAR1= 172342
(1)     172344                  KIPAR2= 172344
(1)     172346                  KIPAR3= 172346
(1)     172350                  KIPAR4= 172350
(1)     172352                  KIPAR5= 172352
(1)     172354                  KIPAR6= 172354
(1)     172356                  KIPAR7= 172356
(1)
392                             .SBTTL  TRAP CATCHER
(1)
(1)     000000                          .=0
(1)                             ;*ALL UNUSED LOCATIONS OF THE VECTOR AREA CONTAIN
(1)                             ;*A ".+2, IOT" SEQUENCE TO CATCH AND PROCESS ILLEGAL
(1)                             ;*TRAPS AND INTERRUPTS THAT MIGHT OCCUR.
(1)                             ;*THE IOT TRAP WHICH IS TAKEN ON THE ILLEGAL TRAP/INT
(1)                             ;*TRAPS TO THE $SCOPE ROUTINE WHICH (IF THE RETURN PC IS
(1)                             ;*LESS THAN 1002) JUMPS TO THE $ERROR ROUTINE.
(1)                             ;*THE $ERROR ROUTINE WILL REPORT THE ERROR AS FOLLOWS:
(1)                             ;*      PC=YYYYYY UNEXPECTED TRAP TO XXX
(1)                             ;*AND RETURN TO THE PROGRAM AT PC=YYYYYY+2
```

TRAP CATCHER  SEQ 0016

```
(1)                                     ;*WHERE XXX=LOCATION OF ILLEGAL TRAP
(1)                                     ;*      YYYYYY=PC AT TIME OF TRAP
(1)                                     ;*NOTE: IF THE PROCESSOR IS NOT AN 11/05 THE PROGRAM
(1)                                     ;*      CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.
(1)
(1)  000000  000000                     $40CAT: HALT                    ;;HALT
(1)  000002  000737                             BR      .-100           ;;BRANCH TO 177700 & TIME OUT (NOT ON
(1)                                                                     ;;11/05)
(1)  000004  002454                             .WORD   ROUT1           ;;VECTOR TO STARTING ADDRESS
(1)  000006  000340                             .WORD   340             ;;WITH PRIORITY LEVEL 7
(1)          000174                             .=174
(1)  000174  000000                     DISPREG:.WORD   0               ;;SOFTWARE DISPLAY REGISTER
(1)  000176  000000                     SWREG:  .WORD   0               ;;SOFTWARE SWITCH REGISTER
(1)                                     .SBTTL  STARTING ADDRES(ES)
(1)  000200  000137  002454                     JMP     @#ROUT1 ;;GO TO START OF PROGRAM
393  000204  000137  002466                     JMP     ROUT2           ;GO TO MULTIPORT TEST
394  000210  000137  002500                     JMP     ROUT3           ;GO INPUT PARAMETERS
395  000214  000137  002516                     JMP     ROUT4           ;GO SET UP FOR MANUAL INTERVENTION TEST
```

```
                                .SBTTL  ACT11 HOOKS

                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
                                        $SVPC=.                 ;SAVE PC
000220                                  .=46
000046                                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000046  021220                         .=52
000052                                  .WORD   0               ;;2)SET LOC.52 TO ZERO
000052  000000                         .=$SVPC                 ;; RESTORE PC
000220                                  .=1000
001000                          .SBTTL  APT PARAMETER BLOCK

                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
001000                                  .$X=.   ;;SAVE CURRENT LOCATION
000024                                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                         200     ;;FOR APT START UP
000044                                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001000                         $APTHDR ;;POINT TO APT HEADER BLOCK
001000                                  .=.$X   ;;RESET LOCATION COUNTER
                                ;;******************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                 $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001206                 $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000017                 $TSTM:  .WORD   15.     ;;RUN TIM OF LONGEST TEST
001006  000074                 $PASTM: .WORD   60.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000000                 $UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000052                         .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
   400                                  .SBTTL  COMMON TAGS
   (1)
   (2)                                  ;;*****************************************************************
   (1)                                  ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
   (1)                                  ;*USED IN THE PROGRAM.
   (1)
   (1)          001100                          .=1100
   (1)  001100                          $CMTAG:                         ;;START OF COMMON TAGS
   (1)  001100  000000                          .WORD   0
   (1)  001102     000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
   (1)  001103     000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
   (1)  001104  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
   (1)  001106  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
   (1)  001110  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
   (1)  001112  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
   (1)  001114     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
   (1)  001115     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
   (1)  001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
   (1)  001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
   (1)  001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
   (1)  001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
   (1)  001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
   (1)  001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
   (1)  001132  000000                          .WORD   0
   (1)  001134     000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
   (1)  001135     000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
   (1)  001136  000000                          .WORD   0
   (1)  001140  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
   (1)  001142  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
   (1)  001144  177560                  $TKS:   177560                  ;;TTY KBD STATUS
   (1)  001146  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
   (1)  001150  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
   (1)  001152  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
   (1)  001154     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
   (1)  001155     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
   (1)  001156     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
   (1)  001157     000                  $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
   (3)  001160  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
   (3)  001162  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
   (3)  001164  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
   (3)  001166  000000                  $TMP3:  .WORD   0               ;;USER DEFINED
   (3)  001170  000000                  $TMP4:  .WORD   0               ;;USER DEFINED
   (1)  001172  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
   (1)  001174  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
   (1)  001176  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
   (1)  001202     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
   (1)  001203     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
   (1)  001204  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
   (2)                                  ;;*****************************************************************
   (2)                                  .SBTTL  APT MAILBOX-ETABLE
   (2)
   (3)                                  ;;*****************************************************************
   (2)                                  .EVEN
   (2)  001206                          $MAIL:                          ;;APT MAILBOX
   (2)  001206  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
   (2)  001210  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
```

```
(2)  001212  000000                $TESTN: .WORD   ATESTN  ;;TEST NUMBER
(2)  001214  000000                $PASS:  .WORD   APASS   ;;PASS COUNT
(2)  001216  000000                $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
(2)  001220  000000                $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
(2)  001222  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(2)  001224  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001226                        $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001226     000                $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001227     000                $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001230  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001232  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001234  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                ;*                      BITS 15-11=CPU TYPE
(2)                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                                ;*                              11/70=06,PDQ=07,Q=10
(2)                                ;*                      BIT 10=REAL TIME CLOCK
(2)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001236     000                $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001237     000                $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                ;*                              900 NSEC CORE=001
(2)                                ;*                              300 NSEC BIPOLAR=002
(2)                                ;*                              500 NSEC MOS=003
(2)  001240  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2)  001242     000                $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001243     000                $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001244  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001246     000                $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001247     000                $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001250  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001252     000                $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001253     000                $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001254  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001256  160350                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001260  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001262  177420                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001264  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001266  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001270  000000                $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
(2)  001272  000000                $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
(2)  001274  000000                $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
(2)  001276  000000                $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
(2)  001300  000000                $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
(2)  001302  000000                $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
(2)  001304  000000                $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
(2)  001306  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
(2)  001310  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
(2)  001312  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
(2)  001314  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
(2)  001316  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
(2)  001320  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
(2)  001322  000000                $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
(2)  001324  000000                $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
(2)  001326  000000                $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
```

```
   (2)  001330  000000                 $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
   (2)
   (2)
   (2)  001332                          $ETEND:
   (2)
```

```
(1)                                     .SBTTL  ERROR POINTER TABLE
(1)
(1)                                     ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1)                                     ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1)                                     ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1)                                     ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1)                                     ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1)                                     ;*      EM              ;;POINTS TO THE ERROR MESSAGE
(1)                                     ;*      DH              ;;POINTS TO THE DATA HEADER
(1)                                     ;*      DT              ;;POINTS TO THE DATA
(1)                                     ;*      DF              ;;POINTS TO THE DATA FORMAT
(1)
(1)
(1)  001332                             $ERRTB:
401                                     ;ERROR#1
402  001332  031510                             EM1             ;POST-RESET CONDITIONS NOT MET IN CSR
403  001334  032721                             DH1
404  001336  033176                             DT1
405  001340  000000                             0
406                                     ;ERROR#2
407  001342  031556                             EM2             ;READ/WRITE BIT FAILURE IN CSR
408  001344  032721                             DH1
409  001346  033176                             DT1
410  001350  000000                             0
411                                     ;ERROR#3
412  001352  031615                             EM3             ;BYTE ACCESS FAILURE IN CSR
413  001354  032721                             DH1
414  001356  033176                             DT1
415  001360  000000                             0
416                                     ;ERROR#4
417  001362  031651                             EM4             ;ERROR IN CONNECT PROCESS
418  001364  032721                             DH1
419  001366  033176                             DT1
420  001370  000000                             0
421                                     ;ERROR#5
422  001372  031703                             EM5             ;INTERUPT DETECTED WITH INTERUPT ENABLE CLEARED
423  001374  032721                             DH1
424  001376  033176                             DT1
425  001400  000000                             0
426                                     ;ERROR#6
427  001402  031763                             EM6             ;COULD NOT CONNECT
428  001404  032721                             DH1
429  001406  033176                             DT1
430  001410  000000                             0
431                                     ;ERROR#7
432  001412  032006                             EM7             ;ERROR IN RELEASE PROCESS
433  001414  032721                             DH1
434  001416  033176                             DT1
435  001420  000000                             0
436                                     ;ERROR#10
437  001422  032040                             EM10            ;NO INTERUPT DETECTED
438  001424  032721                             DH1
439  001426  033176                             DT1
440  001430  000000                             0
441                                     ;ERROR#11
```

```
442  001432  032066                  EM11            ;INTERUPT AT WRONG PRIORITY
443  001434  032721                  DH1
444  001436  033176                  DT1
445  001440  000000                  0
446                          ;ERROR#12
447  001442  032122                  EM12            ;CONNECT CONDITION FAILURE
448  001444  032721                  DH1
449  001446  033176                  DT1
450  001450  000000                  0
451                          ;ERROR#13
452  001452  032155                  EM13            ;ERROR IN CONNECT FAILURE PROCESS
453  001454  032721                  DH1
454  001456  033176                  DT1
455  001460  000000                  0
456                          ;ERROR#14
457  001462  032217                  EM14            ;ERROR IN RELEASE TEST W/O BUSS MASTERSHIP
458  001464  032721                  DH1
459  001466  033176                  DT1
460  001470  000000                  0
461                          ;ERROR#15
462  001472  032276                  EM15            ;RELEASE CONDITIONS NOT MET
463  001474  032721                  DH1
464  001476  033176                  DT1
465  001500  000000                  0
466                          ;ERROR#16
467  001502  032332                  EM16            ;TMO BIT NOT SET
468  001504  032721                  DH1
469  001506  033176                  DT1
470  001510  000000                  0
471                          ;ERROR#17
472  001512  032353                  EM17            ;COULDN'T READ/WRITE EXTERNAL DEVICE
473  001514  032764                  DH2
474  001516  033212                  DT2
475  001520  000000                  0
476                          ;ERROR#20
477  001522  032420                  EM20            ;RESET HOLD (HLD) DIDN'T DISABLE RESET
478  001524  032721                  DH1
479  001526  033176                  DT1
480  001530  000000                  0
481                          ;ERROR#21
482  001532  032467                  EM21            ;RESET (RST) DIDN'T WORK
483  001534  033035                  DH3
484  001536  033226                  DT3
485  001540  000000                  0
486                          ;ERROR#22
487  001542  032520                  EM22            ;RESET-IN-NEUTRAL OPERATION HAS CHANGED
488  001544  033116                  DH4
489  001546  033212                  DT2
490  001550  000000                  0
491
492
```

```
                                        ;
                                        ;       DT07 BUS REGISTER ADDRESS POINTERS
                                        ;
001552  177420                          CSR:    177420                  ;COMMAND/STATUS REGISTER
                                        ;
                                        ;       DT07 VECTOR ADDRESS POINTERS
                                        ;
001554  000350                          VEC0:   350                     ;NORMAL INTERRUPT VECTOR
001556  000352                          VEC2:   352                     ;
                                        ;
                                        ;       DT07 DEVICE LEVEL
                                        ;
001560  000340                          PRI:    340
                                        ;
                                        ;       COMMAND STATUS REGISTER BIT ASSIGNMENTS
                                        ;
                                        ;       COMMON PROGRAM LOCATION(S)
                                        ;
001562  000000                          TSTNUM: 0                       ;CONTAINS TEST NUMBER ON ERROR
                                        ;;*****************************************************************
                                        ;PROGRAM PARAMETER BLOCK
001564  000000                          DT03:   0000            ;DT03/DT07 MODE INDIACATER,0=DT07
001566  000000                          DEVAD:  0000            ;EXTERNAL DEVICE REGISTER ADDRESS
001570  000000                          DEVRW:  0000            ;READ/WRITE BITS IN ABOVE
001572  000000                          DEVRC:  0000            ;MASK OF RESET CLEARABLE BITS IN ABOVE
001574  177777                          PORTNO: -1              ;THIS PORT'S ID#
001576  177777                          EXPRT1: -1              ;EXTERNAL PORT ID (-1=NO PORT)
001600  177777                          EXPRT2: -1              ;     "     "  "
001602  177777                          EXPRT3: -1              ;     "     "  "
001604  177777                          EXPEND: -1              ;TABLE TERMINATOR
001606  000000                          EXPTCT: 0               ;# OF EXTERNAL PORTS
001610  177777                          SEQ:    -1              ;SEQUENCER
001612  177777                                  -1
001614  177777                                  -1
001616  177777                                  -1
001620  177777                                  -1
001622  001610                          SEQPTR: SEQ             ;SEQUENCER POINTER
001624  001612                          NEXENT: SEQ+2           ;NEXT ENTRY POINTER
001626  000011                          MENU:   .BLKW   9.      ;MENU STORAGE
001650  177777                                  -1              ;MENU TERMINATOR
001652  000000                          CURFUN: 0               ;CURRANT FUNCTION
001654  000000                          CURID:  0               ;CURRANT SLAVE ID
001656  000000                          CYCLE:  0               ;SUBPASS COUNT
001660  000000                          COVID:  0               ;CONVERTED SLAVE ID
001662  000005                          TEMP1:  .BLKW 5         ;TEMPORARY STORAGE
001674  000000                          ONOFF:  0               ;MASTER/SLAVE INDIACATOR
001676  000000                          TIMA:   0
001700  000000                          TIMB:   0
001702  000000                          KONST:  0
001704  000001                          DEVCON: 1               ;RESET-IN-NEUTRAL CONDITION FLAG
001706  177777                          TRIN:   -1              ;TEST RESET IN NEUTRAL FLAG
```

```
   549
   550  001710                                 .SBTTL PROGRAM START
   (1)                                 START:
   (1)                                 .SBTTL  INITIALIZE THE COMMON TAGS
   (1)                                 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
   (1)  001710  012706  001100                 MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
   (1)  001714  005026                         CLR     (R6)+           ;;CLEAR MEMORY LOCATION
   (1)  001716  022706  001140                 CMP     #SWR,R6 ;;DONE?
   (1)  001722  001374                         BNE     .-6             ;;LOOP BACK IF NO
   (1)  001724  012706  001100                 MOV     #STACK,SP       ;;SETUP THE STACK POINTER
   (1)                                 ;;INITIALIZE A FEW VECTORS
   (1)  001730  012737  030624  000020         MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
   (1)  001736  012737  000340  000022         MOV     #340,@#IOTVEC+2 ;;LEVEL 7
   (1)  001744  012737  030140  000030         MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
   (1)  001752  012737  000340  000032         MOV     #340,@#EMTVEC+2 ;;LEVEL 7
   (1)  001760  012737  031324  000034         MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
   (1)  001766  012737  000340  000036         MOV     #340,@#TRAPVEC+2;LEVEL 7
   (1)  001774  012737  031116  000024         MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
   (1)  002002  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;LEVEL 7
   (1)  002010  013737  021166  021160         MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
   (1)  002016  005037  001172                 CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
   (1)  002022  005037  001174                 CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
   (1)  002026  112737  000001  001115         MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
   (1)  002034  012737  002034  001106         MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
   (1)  002042  012737  002042  001110         MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
   (2)                                 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   (2)                                 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
   (2)  002050  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
   (2)  002054  012737  002110  000004         MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
   (2)  002062  012737  177570  001140         MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
   (2)  002070  012737  177570  001142         MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
   (2)  002076  022777  177777  177034         CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
   (2)  002104  001012                         BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
   (2)                                                                 ;;AND  THE HARDWARE SWR IS NOT = -1
   (2)  002106  000403                         BR      65$             ;;BRANCH IF NO TIMEOUT
   (2)  002110  012716  002116         64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
   (2)  002114  000002                         RTI
   (2)  002116  012737  000176  001140 65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
   (2)  002124  012737  000174  001142         MOV     #DISPREG,DISPLAY
   (2)  002132  012637  000004         66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
   (1)
   (2)  002136  005037  001214                 CLR     $PASS           ;;CLEAR PASS COUNT
   (2)  002142  132737  000200  001227         BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
   (2)  002150  001403                         BEQ     67$             ;;YES,USE NON-APT SWITCH
   (2)  002152  012737  001230  001140         MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
   (2)  002160                         67$:
   551  002160  012737  025772  000004 SETUP1: MOV     #TRAP4,@#ERRVEC         ;POINT TO TRAP ROUTINE
   552  002166  005000                         CLR     R0
   553  002170  013700  001256                 MOV     $VECT1,R0       ;GET PRIORITY LEVEL
   554  002174  000300                         SWAB    R0              ;PUT IN LEFT HALF
   555  002176  042700  177400                 BIC     #177400,R0      ;CLEAR UNWANTED BITS
   556  002202  010037  001560                 MOV     R0,PRI  ;SAVE DEVICE LEVEL
   557                                 ;
   558                                 ;       NOW SETUP BUS ADDRESS VALUES
   559                                 ;
   560  002206  013737  001262  001552 SETUP2: MOV     $BASE,CSR       ;STORE  BUS ADDRESS
   561                                 ;
```

```
                                        ;       NOW SETUP VECTOR ADDRESSES
                                        ;
002214  012700  001554                  MOV     #VEC0,R0        ;SETUP VECTOR POINTER
002220  013701  001256                  MOV     $VECT1,R1       ;GET BASE VECTOR ADDR
002224  042701  177400                  BIC     #-400,R1        ;CLEAR UNWANTED BITS
002230  010120                  SETUP3: MOV     R1,(R0)+        ;PUT ADDRS
002232  062701  000002                  ADD     #2,R1           ;INCR.TO NEXT LOC.
002236  022700  001560                  CMP     #VEC2+2,R0      ;DONE ALL LOCATIONS?
002242  001372                          BNE     SETUP3          ;BR,IF NOT DONE
002244  005737  000042                  TST     @#42            ;TEST CHAIN MODE UNDER XXDP
002250  001051                          BNE     1$              ;BR,IF CHAIN MODE
                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002252  005227  177777                  INC     #-1             ;;FIRST TIME?
002256  001046                          BNE     64$             ;;BRANCH IF NO
002260  022737  021220  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
002266  001442                          BEQ     64$             ;;BRANCH IF YES
002270  104401  002336                  TYPE    ,65$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002274  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
002300  001012                          BNE     66$             ;;BRANCH IF YES
002302  123727  001226  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
002310  001406                          BEQ     66$             ;;BRANCH IF YES
002312  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
002320  001005                          BNE     67$             ;;BRANCH IF NO
002322  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
002324  000403                          BR      67$
002326  112737  000001  001134  66$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
002334                          67$:
002334  000417                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <CRLF>/DECSPEC-11-Z077A  DT07 TEST/<CRLF>
002374                          64$:
002374  012777  022402  177152  1$:     MOV     #OOPS,@VEC0     ;LOAD UNEXPECTED INTERUPT RETURN
002402  012777  000340  177146          MOV     #340,@VEC2      ;SET PRIORITY 7 ON INTERUPT
002410  005077  177136                  CLR     @CSR            ;DETERMINE STATUS OF PWR OK BIT
002414  000005                          RESET                   ;RESET THE BUSS
002416  000240                          NOP
002420  000240                          NOP
002422  000240                          NOP
002424  005037  001702                  CLR     KONST
002430  017737  177116  001702          MOV     @CSR,KONST      ;SAVE CSR
002436  042737  173777  001702          BIC     #173777,KONST   ;MASK EVERYTHING EXCEPT PWR OK
002444  012706  001100          RESTRT: MOV     #STACK,SP       ;RESET STACK POINTER
002450  000177  000076                  JMP     @ROUTE          ;GO DO IT
002454  012737  002556  002552  ROUT1:  MOV     #MONPRT,ROUTE   ;SET UP FOR MONOPORT
002462  000137  001710                  JMP     START
002466  012737  007266  002552  ROUT2:  MOV     #MULPRT,ROUTE   ;SET UP FOR MULTIPORT
002474  000137  001710                  JMP     START
002500  005037  025754          ROUT3:  CLR     PARFLG  ;CLEAR PARAMETER FLAG
002504  012737  024334  002552          MOV     #REPAR,ROUTE    ;SET UP RETURN
002512  004737  023612                  JSR     PC,PARIN        ;GO GET PARAMETERS
002516  012737  002530  002552  ROUT4:  MOV     #1$,ROUTE       ;SET UP RETURN
002524  000137  001710                  JMP     START
002530  012737  014722  002552  1$:     MOV     #MANIN,ROUTE    ;SET UP FOR MANUAL INTERVENTION TEST
002536  004737  023120                  JSR     PC,DTIME        ;GO GET DEVICE TIMING
002542  004737  022556                  JSR     PC,DMODE        ;GO GET DEVICE MODE
```

```
                                        JMP     MAIN
                                ROUTE:  MONPRT                  ;MAIN TEST DISPATCHER
                                TEMP2:  0

                                MONPRT: JSR     PC,DMODE
                                        JSR     PC,DTIME
                                        TST     @#42            ;XXDP CHAINMODE OR APT/ACT?
                                        BNE     TST1            ;SKIP PRINTOUT IF ANYONE OF THE ABOVE
                                        TST     $PASS           ;0 PASSES?
                                        BNE     3$              ;BR=NO
                                        TST     DT03            ;WHAT MODE IS DEVICE?
                                        BEQ     2$              ;BR =DT07 MODE
                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <200>/THIS DT07 IS IN DT03 MODE./
                                64$:
                                        BR      3$              ;GO START TESTS
                                2$:
                                        TYPE    ,67$            ;;TYPE ASCIZ STRING
                                        BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  <200>/THIS DT07 IS PORT# /
                                66$:
                                        MOV     PORTNO,-(SP)    ;;SAVE PORTNO FOR TYPEOUT
                                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                3$:     JSR     PC,WAIT10       ;WAIT ONE SECOND


                                ;;**************************************************************
                                ;*TEST 1         TEST POST-RESET BIT PATTERN IN CSR.IDX1
                                ;;**************************************************************
                                TST1:   SCOPE
                                        MOV     #50.,$TIMES     ;;DO 50. ITERATIONS

                                        MOV     #340,PSW        ;RISE PRIORITY
                                        MOV     #101,@CSR       ;SET SOME BITS
                                        MOV     KONST,GOOD      ;STORE GOOD DATA
                                        RESET                   ;INIT. BUS
                                1$:     INC     R0              ;DELAY AWHILE
                                        BNE     1$
                                        MOV     @CSR,BAD        ;STORE CONTENTS OF CSR AND
                                        MOV     CSR,BADA        ;ADDRESS
                                        CMP     GOOD,BAD        ;COMPARE
                                        BEQ     2$              ;BR IF OK
                                        ERROR+1                 ;POST-RESET CONDITIONS NOT MET IN CSR
                                2$:                             ;GO TO NEXT TEST
```

```
                                        ;;*******************************************************************
                                        ;*TEST 2        TEST READ/WRITE BITS IN CSR.IDX2
                                        ;;*******************************************************************
003010  000004                  TST2:   SCOPE
003012  012737  000062  001172          MOV     #50.,$TIMES     ;;DO 50. ITERATIONS

003020  012737  000340  177776          MOV     #340,PSW        ;LOCKOUT INTERUPTS
003026  005077  176520                  CLR     @CSR            ;ZERO CSR
003032  013737  001702  001124  1$:     MOV     KONST,GOOD      ;STORE GOOD DATA
003040  062737  000001  001124          ADD     #1,GOOD
003046  012737  003032  001110          MOV     #1$,$LPERR      ;SET LOOP-ON-ERROR ADDRESS
003054  013777  001124  176470          MOV     GOOD,@CSR       ;SET BIT PATTERN
003062  017737  176464  001126          MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003070  042737  120600  001126          BIC     #120600,BAD     ;MASK THE CONNECT PROCESS INDIACATORS
003076  042737  000074  001126          BIC     #74,BAD         ;MASK RQ0-3
003104  023737  001124  001126          CMP     GOOD,BAD        ;COMPARE
003112  001401                          BEQ     3$              ;BR IF OK
003114  104002                          ERROR+2                 ;READ/WRITE BIT FAILURE IN CSR
003116  013737  001702  001124  3$:     MOV     KONST,GOOD      ;STORE GOOD DATA
003124  062737  000100  001124          ADD     #100,GOOD
003132  012737  003116  001110          MOV     #3$,$LPERR      ;SET LOOP-ON-ERROR ADDRESS
003140  013777  001124  176404          MOV     GOOD,@CSR       ;SET BIT PATTERN
003146  017737  176400  001126          MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003154  042737  120600  001126          BIC     #120600,BAD     ;MASK THE CONNECT PROCESS INDIACATORS
003162  023737  001124  001126          CMP     GOOD,BAD        ;COMPARE
003170  001401                          BEQ     4$              ;BR IF OK
003172  104002                          ERROR+2                 ;READ/WRITE BIT FAILURE IN CSR
003174  013737  001702  001124  4$:     MOV     KONST,GOOD      ;STORE GOOD DATA
003202  012737  003174  001110          MOV     #4$,$LPERR      ;SET LOOP-ON-ERROR ADDRESS
003210  013777  001124  176334          MOV     GOOD,@CSR       ;SET BIT PATTERN
003216  017737  176330  001126          MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003224  023737  001124  001126          CMP     GOOD,BAD        ;COMPARE
003232  001401                          BEQ     5$              ;BR IF OK
003234  104002                          ERROR+2                 ;READ/WRITE BIT FAILURE IN CSR
003236                          5$:                             ;GO TO NEXT TEST
```

```
                                ;;*****************************************************************
                                ;*TEST 3         TEST OF BYTE ACCESS IN CSR.IDX3
                                ;;*****************************************************************
                        TST3:   SCOPE
003240  012737  000012  001172          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS

003246  005077  176300                  CLR     @CSR            ;CLEAR CONTENTS OF CSR
003252  013700  001552                  MOV     CSR,R0          ;STORE ADDRESS OF CSR
003256  112710  000376                  MOVB    #376,@R0        ;TRY TO LOAD LOW ORDER BYTE
003262  013737  001702  001124          MOV     KONST,GOOD      ;STORE GOOD DATA
003270  062737  000100  001124          ADD     #100,GOOD
003276  011037  001126                  MOV     @R0,BAD         ;STORE BAD DATA
003302  023737  001124  001126          CMP     GOOD,BAD        ;COMPARE
003310  001401                          BEQ     2$              ;BR IF OK
003312  104003                          ERROR+3                 ;LOW BYTE ACCESS FAILURE IN CSR
003314  005077  176232          2$:     CLR     @CSR            ;CLEAR CSR
003320  013737  001702  001124          MOV     KONST,GOOD      ;STORE GOOD DATA (PWR OK)
003326  112760  000377  000001          MOVB    #377,1(R0)      ;TRY LOADING HI ORDER BYTE OF CSR
003334  011037  001126                  MOV     @R0,BAD         ;STORE CONTENTS OF CSR
003340  023737  001124  001126          CMP     GOOD,BAD        ;COMPARE
003346  001401                          BEQ     3$              ;BR IF OK
003350  104003                          ERROR+3                 ;HI-BYTE ACCESS FAILURE IN CSR
003352                          3$:                             ;GO TO NEXT TEST
```

```
                                        ;;*****************************************************************
                                        ;*TEST 4         CONNECT TEST WITHOUT INTERUPT ENABLE.IDX4
                                        ;;*****************************************************************
003352  000004                        TST4:   SCOPE
003354  012737  000012  001172                MOV     #10.,$TIMES     ;;DO 10. ITERATIONS

003362  005000                        1$:     CLR     R0              ;SET DELAY TIMER
003364  012737  000000  177776                MOV     #0,PSW          ;LOWER PRIORITY
003372  005077  176154                        CLR     @CSR            ;CLEAR CSR
003376  012777  003560  176150                MOV     #8$,@VEC0       ;SET RETURN
003404  012737  000340  177776                MOV     #340,PSW        ;RISE PRIORITY
003412  013737  001702  001124                MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,REQ)
003420  062737  020001  001124                ADD     #20001,GOOD
003426  005737  001564                        TST     DT03            ;DT03 MODE?
003432  001003                                BNE     11$             ;BR IF DT03
003434  062737  000400  001124                ADD     #400,GOOD       ;SET BRQ IF DT07
003442  012777  000001  176102  11$:          MOV     #REQ,@CSR       ;SET REQ (BIT0)
003450  017737  176076  001126  2$:           MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003456  042737  000074  001126                BIC     #74,BAD         ;MASK RQ0-3
003464  023737  001124  001126                CMP     GOOD,BAD        ;COMPARE
003472  001403                                BEQ     5$              ;BR IF OK
003474  005200                                INC     R0              ;DELAY
003476  001364                                BNE     2$              ;BR IF NOT DONE
003500  104004                                ERROR+4         ;SWB ACT (BIT13),BRQ OR REQ NOT SET
003502  012737  000000  177776  5$:           MOV     #0,PSW          ;LOWER PRIORITY
003510  013737  001702  001124                MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
003516  062737  020201  001124                ADD     #20201,GOOD
003524  005000                                CLR     R0              ;RESET DELAY
003526  017737  176020  001126  6$:           MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003534  023737  001124  001126                CMP     GOOD,BAD        ;COMPARE
003542  001403                                BEQ     7$              ;BR IF OK
003544  005200                                INC     R0              ;DELAY
003546  001367                                BNE     6$              ;BR IF NOT DONE
003550  104004                                ERROR+4                 ;CON(BIT7) NOT SET
003552  005077  175774          7$:           CLR     @CSR            ;CLEAR CSR AND DISCONNECT
003556  000406                                BR      9$              ;EXIT
003560                          8$:
003560  012600                                MOV     (SP)+,R0        ;;POP STACK INTO R0
003562  012600                                MOV     (SP)+,R0        ;;POP STACK INTO R0
003564  017737  175762  001126                MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003572  104005                                ERROR+5                 ;INTERUPT DETECTED WITH INTERUPT ENABLE CLEARED
003574  012777  022402  175752  9$:           MOV     #OOPS,@VEC0     ;RESET RETURN
                                                                      ;GO TO NEXT TEST
```

```
                                        ;;*****************************************************************
                                        ;*TEST 5         RELEASE TEST WITHOUT INTERUPT ENABLE.IDX5
                                        ;;*****************************************************************
003602  000004                        TST5:   SCOPE
003604  012737  000012  001172                MOV     #10.,$TIMES     ;;DO 10. ITERATIONS

003612  005000                        1$:     CLR     R0              ;SET DELAY TIMER
003614  012777  000001  175730                MOV     #1,@CSR         ;CONNECT
003622  013737  001702  001124                MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
003630  062737  020201  001124                ADD     #20201,GOOD
003636  012737  000000  177776                MOV     #0,PSW          ;DROP PRIORITY
003644  017737  175702  001126  2$:     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003652  023737  001124  001126                CMP     GOOD,BAD        ;COMPARE
003660  001404                              BEQ     3$              ;BR IF CONNECTED
003662  005200                              INC     R0              ;DELAY
003664  001367                              BNE     2$              ;BR IF NOT DONE
003666  104006                              ERROR+6                 ;COULD NOT CONNECT
003670  000461                              BR      7$              ;EXIT TEST
003672  012777  004036  175654  3$:     MOV     #8$,@VEC0       ;SET RETURN
003700  012737  000340  177776                MOV     #340,PSW        ;RISE PRIORITY
003706  005000                              CLR     R0              ;SET DELAY TIMER
003710  042777  000001  175634                BIC     #REQ,@CSR       ;CLEAR REQUEST
003716  013737  001702  001124                MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON)
003724  062737  020200  001124                ADD     #20200,GOOD
003732  005737  001564                        TST     DT03            ;DT03 MODE?
003736  001003                              BNE     4$              ;BR IF YES
003740  062737  000400  001124                ADD     #400,GOOD       ;ADD BRQ IF DT07
003746  017737  175600  001126  4$:     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
003754  023737  001124  001126                CMP     GOOD,BAD        ;COMPARE
003762  001403                              BEQ     5$              ;BR IF OK
003764  005200                              INC     R0              ;DELAY
003766  001367                              BNE     4$              ;BR IF NOT DONE
003770  104007                              ERROR+7                 ;BRQ NOT SET OR REQ NOT CLEARED
003772  012737  000000  177776  5$:     MOV     #0,PSW          ;DROP PRIORITY
004000  013737  001702  001124                MOV     KONST,GOOD      ;STORE GOOD DATA (PWR OK)
004006  005000                              CLR     R0              ;RESET DELAY
004010  017737  175536  001126  6$:     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
004016  023737  001124  001126                CMP     GOOD,BAD        ;COMPARE
004024  001403                              BEQ     7$              ;BR IF OK
004026  005200                              INC     R0              ;DELAY
004030  001367                              BNE     6$              ;BR IF NOT DONE
004032  104007                              ERROR+7                 ;CON OR SWB ACT NOT CLEARED
004034  000406                      7$:     BR      9$              ;EXIT
004036                              8$:
004036  012600                              MOV     (SP)+,R0        ;;POP STACK INTO R0
004040  012600                              MOV     (SP)+,R0        ;;POP STACK INTO R0
004042  017737  175504  001126              MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
004050  104005                              ERROR+5                 ;INTERUPPT DETECTED WITH INTERUPT ENABLE CLRD
004052  012777  022402  175474  9$:     MOV     #OOPS,@VEC0     ;RESET RETURN
                                                                    ;GO TO NEXT TEST
```

```
 772                                    ;;*****************************************************************
 (3)                                    ;*TEST 6        CONNECT FAILURE TEST WITHOUT INTERUPT ENABLE.IDX6
 (3)                                    ;;*****************************************************************
 (2) 004060  000004                     TST6:   SCOPE
 (1) 004062  012737  000012  001172             MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
 773
 774 004070  005000                     1$:     CLR     R0              ;SET DELAY TIMER
 775 004072  012737  000000  177776             MOV     #0,PSW          ;LOWER PRIORITY
 776 004100  005077  175446                     CLR     @CSR            ;CLEAR CSR
 777 004104  012777  004336  175442             MOV     #8$,@VEC0       ;SET RETURN
 778 004112  012737  000340  177776             MOV     #340,PSW        ;RISE PRIORITY
 779 004120  013737  001702  001124             MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,REQ)
 780 004126  062737  020001  001124             ADD     #20001,GOOD
 781 004134  005737  001564                     TST     DT03            ;DT03 MODE?
 782 004140  001003                             BNE     11$             ;BR IF DT03
 783 004142  062737  000400  001124             ADD     #400,GOOD       ;ADD BRQ IF DT07
 784 004150  012777  000001  175374  11$:       MOV     #REQ,@CSR       ;SET REQ (BIT0)
 785 004156  017737  175370  001126  2$:        MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 786 004164  042737  000074  001126             BIC     #74,BAD         ;MASK RQ0-3
 787 004172  023737  001124  001126             CMP     GOOD,BAD        ;COMPARE
 788 004200  001403                             BEQ     3$              ;BR IF OK
 789 004202  005200                             INC     R0              ;DELAY
 790 004204  001364                             BNE     2$              ;BR IF NOT DONE
 791 004206  104013                             ERROR+13                        ;SWB ACT (BIT13) OR REQ NOT SET
 792 004210  005737  001564             3$:     TST     DT03            ;DT03 MODE?
 793 004214  001024                             BNE     5$              ;BR IF YES
 794 004216  013737  001702  001124             MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,BRQ,REQ)
 795 004224  062737  020401  001124             ADD     #20401,GOOD
 796 004232  005000                             CLR     R0              ;RESET DELAY
 797 004234  017737  175312  001126  4$:        MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 798 004242  042737  000074  001126             BIC     #74,BAD         ;MASK RQ0-3
 799 004250  023737  001124  001126             CMP     GOOD,BAD        ;COMPARE
 800 004256  001403                             BEQ     5$              ;BR IF OK
 801 004260  005200                             INC     R0              ;DELAY
 802 004262  001364                             BNE     4$              ;BR IF NOT DONE
 803 004264  104013                             ERROR+13                        ;BRQ(BIT8) DID NOT SET
 804 004266  013737  001702  001124  5$:        MOV     KONST,GOOD      ;STORE GOOD DATA (TMO,PWR OK)
 805 004274  062737  100000  001124             ADD     #100000,GOOD
 806 004302  005000                             CLR     R0              ;RESET DELAY
 807 004304  017737  175242  001126  6$:        MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 808 004312  023737  001124  001126             CMP     GOOD,BAD        ;COMPARE
 809 004320  001403                             BEQ     7$              ;BR IF OK
 810 004322  005200                             INC     R0              ;DELAY
 811 004324  001367                             BNE     6$              ;BR IF NOT DONE
 812 004326  104013                             ERROR+13                        ;TMO(BIT15) NOT SET
 813 004330  005077  175216             7$:     CLR     @CSR            ;CLEAR CSR AND DISCONNECT
 814 004334  000407                             BR      9$              ;EXIT
 815 004336                             8$:
 (2) 004336  012600                             MOV     (SP)+,R0        ;;POP STACK INTO R0
 (2) 004340  012600                             MOV     (SP)+,R0        ;;POP STACK INTO R0
 816 004342  017737  175204  001126             MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 817 004350  104005                             ERROR+5                 ;INTERUPT DETECTED WITH INTERUPT ENABLE CLEARED
 818 004352  000766                             BR      7$              ;GO CLEAR CSR
 819 004354  012777  022402  175172  9$:        MOV     #OOPS,@VEC0     ;RESET RETURN
 820                                            ;GO TO NEXT TEST
```

```
                                        ;;*****************************************************************
                                        ;*TEST 7         RELEASE TEST WITHOUT BUSS MASTERSHIP OR INTERUPT ENABLE.IDX7
                                        ;;*****************************************************************
(2) 004362 000004                       TST7:   SCOPE
(1) 004364 012737 000012 001172                 MOV     #10.,$TIMES     ;;DO 10. ITERATIONS

004372 005000                           1$:     CLR     R0              ;SET DELAY TIMER
004374 012777 000001 175150                     MOV     #1,@CSR         ;CONNECT
004402 013737 001702 001124                     MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
004410 062737 020201 001124                     ADD     #20201,GOOD
004416 012737 000000 177776                     MOV     #0,PSW          ;DROP PRIORITY
004424 017737 175122 001126             2$:     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
004432 023737 001124 001126                     CMP     GOOD,BAD        ;COMPARE
004440 001404                                   BEQ     3$              ;BR IF CONNECTED
004442 005200                                   INC     R0              ;DELAY
004444 001367                                   BNE     2$              ;BR IF NOT DONE
004446 104006                                   ERROR+6                 ;COULD NOT CONNECT
004450 000461                                   BR      7$              ;EXIT TEST
004452 012777 004616 175074             3$:     MOV     #8$,@VEC0       ;SET RETURN
004460 012737 000340 177776                     MOV     #340,PSW        ;RISE PRIORITY
004466 005000                                   CLR     R0              ;SET DELAY TIMER
004470 042777 000001 175054                     BIC     #REQ,@CSR       ;CLEAR REQUEST
004476 013737 001702 001124                     MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON)
004504 062737 020200 001124                     ADD     #20200,GOOD
004512 005737 001564                            TST     DT03            ;DT03 MODE?
004516 001003                                   BNE     4$              ;BR IF YES
004520 062737 000400 001124                     ADD     #400,GOOD       ;ADD BRQ IF DT07
004526 017737 175020 001126             4$:     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
004534 023737 001124 001126                     CMP     GOOD,BAD        ;COMPARE
004542 001403                                   BEQ     5$              ;BR IF OK
004544 005200                                   INC     R0              ;DELAY
004546 001367                                   BNE     4$              ;BR IF NOT DONE
004550 104014                                   ERROR+14                        ;BRQ NOT SET OR REQ NOT CLEARED
004552 013737 001702 001124             5$:     MOV     KONST,GOOD      ;STORE GOOD DATA (TMO,PWR OK)
004560 062737 100000 001124                     ADD     #100000,GOOD
004566 005000                                   CLR     R0              ;RESET DELAY
004570 017737 174756 001126             6$:     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
004576 023737 001124 001126                     CMP     GOOD,BAD        ;COMPARE
004604 001403                                   BEQ     7$              ;BR IF OK
004606 005200                                   INC     R0              ;DELAY
004610 001367                                   BNE     6$              ;BR IF NOT DONE
004612 104014                                   ERROR+14                        ;TMO NOT SET
004614 000406                           7$:     BR      9$              ;EXIT
004616                                  8$:
(2) 004616 012600                               MOV     (SP)+,R0        ;;POP STACK INTO R0
(2) 004620 012600                               MOV     (SP)+,R0        ;;POP STACK INTO R0
004622 017737 174724 001126                     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
004630 104005                                   ERROR+5                 ;INTERUPPT DETECTED WITH INTERUPT ENABLE CLRD
004632 012777 022402 174714             9$:     MOV     #OOPS,@VEC0     ;RESET RETURN
                                                                        ;GO TO NEXT TEST
```

```
868
869                                     ;;*****************************************************************
(3)                                     ;*TEST 10        CONNECT TEST.IDX8
(3)                                     ;;*****************************************************************
(2) 004640 000004                       TST10:  SCOPE
(1) 004642 012737 000012 001172                 MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
870
871 004650 005077 174676                        CLR     @CSR            ;CLEAR CSR
872 004654 012737 000340 177776                 MOV     #340,PSW        ;RISE PRIORITY
873 004662 013737 001560 001124                 MOV     PRI,GOOD        ;STORE PRIORITY LEVEL
874 004670 012737 000340 001126                 MOV     #340,BAD        ;STORE ACTUAL PRIORITY INDIACATOR
875 004676 012777 004752 174650                 MOV     #2$,@VEC0       ;SET RETURN
876 004704 012777 000101 174640                 MOV     #101,@CSR       ;SET I.E. AND REQ
877 004712 032777 020000 174632                 BIT     #20000,@CSR     ;WAIT FOR SWB ACT TO APPEAR
878 004720 001774                               BEQ     .-6             ;BR IF NOT THERE
879 004722 162737 000040 177776 1$:             SUB     #40,PSW         ;LOWER PRIORITY UNTIL INTERUPTED
880 004730 000240                               NOP
881 004732 000240                               NOP
882 004734 000240                               NOP
883 004736 162737 000040 001126                 SUB     #40,BAD         ;  ''      ''     INDIACATOR
884 004744 001366                               BNE     1$              ;BR UNTIL DONE
885 004746 104010                               ERROR+10                ;NO INTERUPT DETECTED
886 004750 000425                               BR      4$              ;EXIT
887 004752                              2$:
(2) 004752 012600                               MOV     (SP)+,R0        ;;POP STACK INTO R0
(2) 004754 012600                               MOV     (SP)+,R0        ;;POP STACK INTO R0
888 004756 023737 001124 001126                 CMP     GOOD,BAD        ;CORRECT BREAK LEVEL?
889 004764 001401                               BEQ     3$              ;BR IF OK
890 004766 104011                               ERROR+11                ;INTERUPT AT WRONG PRIORITY
891 004770 017737 174556 001126 3$:             MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
892 004776 013737 001702 001124                 MOV     KONST,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON,IE,REQ)
893 005004 062737 020301 001124                 ADD     #20301,GOOD
894 005012 023737 001124 001126                 CMP     GOOD,BAD        ;COMPARE
895 005020 001401                               BEQ     4$              ;BR IF OK
896 005022 104012                               ERROR+12                ;CONNECT CONDITION FAILURE
897 005024 005077 174522                4$:     CLR     @CSR            ;CLEAR CSR
898 005030 005037 177776                        CLR     PSW             ;LOWER PRIORITY
899 005034 012777 022402 174512 5$:             MOV     #OOPS,@VEC0     ;RESET RETURN
900                                                                     ;GO TO NEXT TEST
```

```
902
903                                     ;;*****************************************************************
(3)                                     ;*TEST 11        RELEASE TEST .IDX9
(3)                                     ;;*****************************************************************
(2)  005042  000004                     TST11:  SCOPE
(1)  005044  012737  000012  001172             MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
904
905  005052  005000                     1$:     CLR     R0              ;SET DELAY TIMER
906  005054  012777  000001  174470             MOV     #1,@CSR         ;CONNECT
907  005062  013737  001702  001124             MOV     KONST,GOOD
908  005070  062737  020201  001124             ADD     #20201,GOOD     ;STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
909  005076  012737  000000  177776             MOV     #0,PSW          ;DROP PRIORITY
910  005104  017737  174442  001126  2$:      MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
911  005112  023737  001124  001126             CMP     GOOD,BAD        ;COMPARE
912  005120  001404                             BEQ     3$              ;BR IF CONNECTED
913  005122  005200                             INC     R0              ;DELAY
914  005124  001367                             BNE     2$              ;BR IF NOT DONE
915  005126  104006                             ERROR+6                 ;COULD NOT CONNECT
916  005130  000422                             BR      7$              ;EXIT TEST
917  005132  005737  001564             3$:     TST     DT03            ;DT03 MODE?
918  005136  001420                             BEQ     8$              ;BR=NO
919  005140  005000                             CLR     R0              ;SET DELAY TIMER
920  005142  042777  000001  174402             BIC     #REQ,@CSR       ;CLEAR REQUEST
921  005150  013737  001702  001124  5$:      MOV     KONST,GOOD      ;STORE GOOD DATA (PWR OK)
922  005156  017737  174370  001126  6$:      MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
923  005164  023737  001124  001126             CMP     GOOD,BAD        ;COMPARE
924  005172  001401                             BEQ     7$              ;BR IF OK
925  005174  104015                             ERROR+15                ;RELEASE CONDITIONS NOT MET
926  005176  000446                     7$:     BR      10$             ;GO TO NEXT TEST
927  005200  012777  005254  174346  8$:      MOV     #9$,@VEC0       ;SET RETURN
928  005206  052777  000100  174336             BIS     #100,@CSR       ;SET I.E.
929  005214  042777  000001  174330             BIC     #1,@CSR         ;CLR REQ.
930  005222  004737  021370                     JSR     PC,WAIT05       ;GO WAIT
931  005226  013737  001702  001124             MOV     KONST,GOOD      ;LOAD GOOD
932  005234  062737  000100  001124             ADD     #100,GOOD
933  005242  017737  174304  001126             MOV     @CSR,BAD
934  005250  104010                             ERROR+10                ;NO INTERUPT
935  005252  000420                             BR      10$             ;EXIT
936  005254  012706  001100             9$:     MOV     #STACK,SP       ;RESET STACK
937  005260  013737  001702  001124             MOV     KONST,GOOD      ;LOAD GOOD
938  005266  062737  000100  001124             ADD     #100,GOOD
939  005274  017737  174252  001126             MOV     @CSR,BAD
940  005302  023737  001124  001126             CMP     GOOD,BAD
941  005310  001401                             BEQ     10$
942  005312  104015                             ERROR+15
943  005314  005077  174232             10$:    CLR     @CSR            ;CLEAR AND DISCONNECT
944  005320  012777  022402  174226             MOV     #OOPS,@VEC0     ;RESET RETURN
```

```
                                        ;;*****************************************************************
                                        ;*TEST 12        CONNECT FAILURE TEST.IDX10
                                        ;;*****************************************************************
                                        TST12:  SCOPE
(1)  005330  012737  000012  001172            MOV     #10.,$TIMES     ;;DO 10. ITERATIONS

949  005336  005077  174210                  CLR     @CSR            ;CLEAR CSR
950  005342  012737  000340  177776          MOV     #340,PSW        ;RISE PRIORITY
951  005350  005000                          CLR     R0              ;SET DELAY TIMER
952  005352  012777  005464  174174          MOV     #2$,@VEC0       ;SET RETURN
953  005360  013737  001702  001124          MOV     KONST,GOOD
954  005366  062737  100100  001124          ADD     #100100,GOOD    ;STORE GOOD DATA (TMO,PWR OK,IE)
955  005374  005737  001564                  TST     DT03            ;DT03 MODE?
956  005400  001003                          BNE     6$              ;BR IF DT03
957  005402  062737  000400  001124          ADD     #400,GOOD       ;ADD BRQ IF DT07
958  005410  012777  000101  174134  6$:     MOV     #101,@CSR       ;SET I.E. AND REQ
959  005416  017737  174130  001126  1$:     MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
960  005424  023737  001124  001126          CMP     GOOD,BAD        ;COMPARE
961  005432  001404                          BEQ     11$             ;BR IF OK
962  005434  005200                          INC     R0              ;DELAY
963  005436  001367                          BNE     1$              ;BR IF NOT DONE
964  005440  104016                          ERROR+16                ;TMO NOT SET
965  005442  000414                          BR      5$              ;EXIT
966  005444  005000                  11$:    CLR     R0              ;RESET DELAY
967  005446  012737  000000  177776          MOV     #0,PSW          ;DROP PRIORITY
968  005454  005200                  22$:    INC     R0              ;DELAY
969  005456  001376                          BNE     22$             ;BR IF NOT DONE
970  005460  104010                          ERROR+10                ;NO INTERUPT DETECTED
971  005462  000404                          BR      5$              ;EXIT
972  005464                          2$:
(2)  005464  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(2)  005466  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
973  005470  005077  174056          4$:     CLR     @CSR            ;CLEAR CSR
974  005474  012777  022402  174052  5$:     MOV     #OOPS,@VEC0     ;RESET RETURN
975                                                                  ;GO TO NEXT TEST
```

```
 977
 978                                    ;;*****************************************************************
 (3)                                    ;*TEST 13       RELEASE TEST WITHOUT BUSS MASTERSHIP.IDX11
 (3)                                    ;;*****************************************************************
 (2) 005502 000004                      TST13:  SCOPE
 (1) 005504 012737 000012 001172                MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
 979
 980 005512 005000                      1$:     CLR     R0              ;SET DELAY TIMER
 981 005514 012777 000001 174030                MOV     #1,@CSR         ;CONNECT
 982 005522 013737 001702 001124                MOV     KONST,GOOD
 983 005530 062737 020201 001124                ADD     #20201,GOOD     ;STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
 984 005536 012737 000000 177776                MOV     #0,PSW          ;DROP PRIORITY
 985 005544 017737 174002 001126 2$:    MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 986 005552 023737 001124 001126                CMP     GOOD,BAD        ;COMPARE
 987 005560 001404                              BEQ     3$              ;BR IF CONNECTED
 988 005562 005200                              INC     R0              ;DELAY
 989 005564 001367                              BNE     2$              ;BR IF NOT DONE
 990 005566 104006                              ERROR+6                 ;COULD NOT CONNECT
 991 005570 000461                              BR      7$              ;EXIT TEST
 992 005572 012777 005730 173754 3$:    MOV     #6$,@VEC0       ;SET RETURN
 993 005600 012737 000340 177776                MOV     #340,PSW        ;RISE PRIORITY
 994 005606 005000                              CLR     R0              ;SET DELAY TIMER
 995 005610 012777 000100 173734                MOV     #100,@CSR       ;CLEAR REQUEST AND SET IE
 996 005616 017737 173730 001126 4$:    MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 997 005624 013737 001702 001124                MOV     KONST,GOOD
 998 005632 062737 100100 001124                ADD     #100100,GOOD    ;STORE GOOD DATA(TMO,PWR OK,IE)
 999 005640 005737 001564                       TST     DT03            ;DT03 MODE?
1000 005644 001003                              BNE     41$             ;BR IF DT03
1001 005646 062737 000400 001124                ADD     #400,GOOD       ;ADD BRQ IF DT07
1002 005654 023737 001124 001126 41$:   CMP     GOOD,BAD        ;COMPARE
1003 005662 001404                              BEQ     5$              ;BR IF OK
1004 005664 005200                              INC     R0              ;DELAY
1005 005666 001353                              BNE     4$              ;BR IF NOT DONE
1006 005670 104016                              ERROR+16                ;TMO NOT SET
1007 005672 000420                              BR      7$              ;EXIT
1008 005674 013737 001702 001124 5$:    MOV     KONST,GOOD
1009 005702 062737 100100 001124                ADD     #100100,GOOD    ;STORE GOOD DATA (TMO,PWR OK,IE)
1010 005710 005000                              CLR     R0              ;RESET DELAY
1011 005712 012737 000000 177776                MOV     #0,PSW          ;DROP PRIORITY
1012 005720 005200                      51$:    INC     R0              ;DELAY
1013 C05722 001376                              BNE     51$             ;BR IF NOT DONE
1014 005724 104010                              ERROR+10                ;NO INTERUPT
1015 005726 000402                              BR      7$              ;EXIT
1016 005730                             6$:
 (2) 005730 012600                              MOV     (SP)+,R0        ;;POP STACK INTO R0
 (2) 005732 012600                              MOV     (SP)+,R0        ;;POP STACK INTO R0
1017 005734 012777 022402 173612 7$:    MOV     #OOPS,@VEC0     ;RESET RETURN
1018                                            ;GO TO NEXT TEST
```

```
 1020
 1021
 1022                                   ;;*****************************************************************
  (3)                                   ;*TEST 14       RESET (RST) TEST.IDX12
  (3)                                   ;;*****************************************************************
  (2) 005742  000004                    TST14:  SCOPE
  (1) 005744  012737  000012  001172            MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
 1023
 1024 005752  005737  001214                    TST     $PASS           ;FIRST PASS?
 1025 005756  001551                            BEQ     5$              ;SKIP TEST ON FIRST PASS
 1026 005760  005737  001704                    TST     DEVCON          ;RESET IN NEUTRAL?
 1027 005764  100546                            BMI     5$              ;BR=NO RESET IN NEUTRAL
 1028 005766  005737  001566                    TST     DEVAD           ;IS A DEVICE REGISTER PRESENT?
 1029 005772  001543                            BEQ     5$              ;EXIT IF NOT
 1030 005774  005737  001572                    TST     DEVRC           ;RESETABLE BITS PRESENT?
 1031 006000  001540                            BEQ     5$              ;BR=NO SO EXIT
 1032 006002  012737  000000  177776            MOV     #0,PSW          ;DROP PRIORITY
 1033 006010  012777  000001  173534            MOV     #REQ,@CSR       ;CONNECT
 1034 006016  105777  173530          1$:       TSTB    @CSR            ;CONNECTED?
 1035 006022  100416                            BMI     2$              ;BR IF YES
 1036 006024  005777  173522                    TST     @CSR            ;TIMEOUT?
 1037 006030  100372                            BPL     1$              ;BR IF NOT
 1038 006032  017737  173514  001126            MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 1039 006040  013737  001702  001124            MOV     KONST,GOOD
 1040 006046  062737  020201  001124            ADD     #20201,GOOD     ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
 1041 006054  104006                            ERROR+6                 ;COULDN'T CONNECT
 1042 006056  000511                            BR      5$              ;EXIT
 1043 006060  013777  001570  173500  2$:       MOV     DEVRW,@DEVAD    ;SET READ/WRITE BITS IN EXTERNAL DEVICE REGISTER
 1044 006066  017737  173474  001126            MOV     @DEVAD,BAD      ;STORE CONTENTS OF DEVAD
 1045 006074  005137  001570                    COM     DEVRW           ;COMPLEMENT R/W BITS
 1046 006100  043737  001570  001126            BIC     DEVRW,BAD       ;CLEAR OUT BITS NOT UNDER TEST
 1047 006106  005137  001570                    COM     DEVRW           ;RESTORE ORIGINAL PATTERN
 1048 006112  023737  001570  001126            CMP     DEVRW,BAD       ;DID THEY SET?
 1049 006120  001402                            BEQ     3$              ;BR IF YES
 1050 006122  104017                            ERROR+17                ;COULDN'T SET READ/WRITE BITS IN DEVAD
 1051 006124  000466                            BR      5$              ;EXIT
 1052 006126  052777  001000  173416  3$:       BIS     #1000,@CSR      ;SET RST
 1053 006134  012737  000002  021442            MOV     #2,TOCK         ;SET UP FOR 22MSEC COUNT
 1054 006142  004737  021402                    JSR     PC,WAIT
 1055 006146  032777  001000  173376  31$:      BIT     #1000,@CSR      ;STILL SET?
 1056 006154  001430                            BEQ     4$              ;BR=NO
 1057 006156  104401  006164                    TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 006162  000424                            BR      64$             ;;GET OVER THE ASCIZ
  (1)                                   ;;65$:  .ASCIZ  <200>/ERROR! RST(BIT9) DIDN'T CLEAR IN TIME./
  (1) 006234                            64$:
 1058 006234  000000                            HALT
 1059 006236  017737  173324  001126  4$:       MOV     @DEVAD,BAD      ;STORE CONTENTS OF DEVAD
 1060 006244  033737  001572  001126            BIT     DEVRC,BAD       ;RESET CONDITIONS MET?
 1061 006252  001413                            BEQ     5$              ;BR IF YES
 1062 006254  005200                            INC     R0              ;WAIT AWHILE
 1063 006256  001367                            BNE     4$
 1064 006260  013737  001572  001124            MOV     DEVRC,GOOD      ;SAVE DEVRC
 1065 006266  005137  001124                    COM     GOOD
 1066 006272  043737  001124  001126            BIC     GOOD,BAD        ;HI LITE BAD BITS
 1067 006300  104021                            ERROR+21                ;RESET (RST) DIDN'T WORK
 1068 006302  005077  173244          5$:       CLR     @CSR            ;SHUT DOWN AND GO TO NEXT TEST
```

```
 1070
 1071                                   ;;*****************************************************************
  (3)                                   ;*TEST 15       RESET HOLD (HLD) TEST.IDX13
  (3)                                   ;;*****************************************************************
  (2)  006306  000004                   TST15:  SCOPE
  (1)  006310  012737  000012  001172           MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
 1072
 1073  006316  005737  001564                   TST     DT03            ;DT03 MODE
 1074  006322  001043                           BNE     3$              ;BR IF YES
 1075  006324  012737  000000  177776           MOV     #0,PSW          ;DROP PRIORITY
 1076  006332  012777  000001  173212           MOV     #REQ,@CSR       ;CONNECT
 1077  006340  105777  173206          1$:      TSTB    @CSR            ;CONNECTED?
 1078  006344  100416                           BMI     2$              ;BR IF YES
 1079  006346  005777  173200                   TST     @CSR            ;TIMEOUT?
 1080  006352  100372                           BPL     1$              ;BR IF NOT
 1081  006354  017737  173172  001126           MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 1082  006362  013737  001702  001124           MOV     KONST,GOOD
 1083  006370  062737  020201  001124           ADD     #20201,GOOD     ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
 1084  006376  104006                           ERROR+6                 ;COULDN'T CONNECT
 1085  006400  000414                           BR      3$              ;EXIT
 1086  006402  052777  000002  173142  2$:      BIS     #2,@CSR         ;SET HLD(BIT1)
 1087  006410  000005                           RESET                   ;DO RESET
 1088  006412  017737  173134  001126           MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 1089  006420  032737  000200  001126           BIT     #200,BAD        ;STILL CONNECTED?
 1090  006426  001001                           BNE     3$              ;BR IF YES
 1091  006430  104020                           ERROR+20                ;HLD DIDN'T DISABLE RESET
 1092  006432  005077  173114          3$:      CLR     @CSR            ;SHUTDOWN AND GO TO NEXT TEST
```

```
1094
1095
1096                                    ;;*****************************************************************
 (3)                                    ;*TEST 16        HI-SPEED SWITCHING TEST.IDX14
 (3)                                    ;;*****************************************************************
 (2) 006436 000004                      TST16:  SCOPE
 (1) 006440 012737 C00012 001172                MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1097
1098 006446 012701 001750               MOV     #1000.,R1       ;SET COUNTER
1099 006452 012737 000000 177776 11$:   MOV     #0,PSW          ;DROP PRIORITY
1100 006460 012777 000001 173064        MOV     #REQ,@CSR       ;CONNECT
1101 006466 105777 173060       1$:     TSTB    @CSR            ;CONNECTED?
1102 006472 100416                      BMI     2$              ;BR IF YES
1103 006474 005777 173052               TST     @CSR            ;TIMEOUT?
1104 006500 100372                      BPL     1$              ;BR IF NOT
1105 006502 017737 173044 001126        MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
1106 006510 013737 001702 001124        MOV     KONST,GOOD
1107 006516 062737 020201 001124        ADD     #20201,GOOD     ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
1108 006524 104006                      ERROR+6                 ;COULDN'T CONNECT
1109 006526 000422                      BR      5$              ;EXIT
1110 006530 042777 000001 173014 2$:    BIC     #1,@CSR         ;RELEASE CONNECTION
1111 006536 013737 001702 001124        MOV     KONST,GOOD      ;STORE GOOD DATA(PWR OK)
1112 006544 005000                      CLR     R0              ;SET DELAY TIMER
1113 006546 017737 173000 001126 3$:    MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
1114 006554 023737 001124 001126        CMP     GOOD,BAD        ;COMPARE
1115 006562 001402                      BEQ     4$              ;BR IF OK
1116 006564 104015                      ERROR+15                ;RELEASE CONDITIONS NOT MET
1117 006566 000402                      BR      5$              ;EXIT
1118 006570 005301              4$:     DEC     R1              ;COUNT A SUBPASS
1119 006572 001327                      BNE     11$             ;BR IF CONTINUE
1120 006574 005077 172752       5$:     CLR     @CSR            ;SHUTDOWN AND GO TO NEXT TEST
1121
1122
1123
1124
1125                                    ;;*****************************************************************
 (3)                                    ;*TEST 17        RESET-IN-NEUTRAL TEST.IDX15
 (3)                                    ;;*****************************************************************
 (2) 006600 000004                      TST17:  SCOPE
 (1) 006602 012737 000001 001172                MOV     #1,$TIMES       ;;DO 1 ITERATION
1126
1127 006610 005737 001566               TST     DEVAD           ;IS A DEVICE REGISTER PRESENT?
1128 006614 001002                      BNE     13$             ;CONTINUE TEST IF PRESENT
1129 006616 000137 007250               JMP     12$             ;GO EXIT
1130 006622 005737 001572       13$:    TST     DEVRC           ;ANY RESETABLE BITS?
1131 006626 001002                      BNE     14$             ;BR=YES
1132 006630 000137 007250               JMP     12$             ;GO EXIT
1133 006634 012737 000000 177776 14$:   MOV     #0,PSW          ;DROP PRIORITY
1134 006642 012777 000001 172702        MOV     #REQ,@CSR       ;CONNECT
1135 006650 105777 172674       1$:     TSTB    @CSR            ;CONNECTED?
1136 006654 100416                      BMI     2$              ;BR IF YES
1137 006656 005777 172670               TST     @CSR            ;TIMEOUT?
1138 006662 100372                      BPL     1$              ;BR IF NOT
1139 006664 017737 172662 001126        MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
1140 006672 013737 001702 001124        MOV     KONST,GOOD
1141 006700 062737 020201 001124        ADD     #20201,GOOD     ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
```

```
1142  006706  104006                        ERROR+6                 ;COULDN'T CONNECT
1143  006710  000557                        BR      12$             ;EXIT
1144  006712  013777  001570  172646  2$:   MOV     DEVRW,@DEVAD    ;SET READ/WRITE BITS IN EXTERNAL DEVICE REGISTER
1145  006720  017737  172642  001126        MOV     @DEVAD,BAD      ;STORE CONTENTS OF DEVAD
1146  006726  005137  001570                COM     DEVRW           ;COMPLEMENT R/W BITS
1147  006732  043737  001570  001126        BIC     DEVRW,BAD       ;CLEAR OUT BITS NOT UNDER TEST
1148  006740  005137  001570                COM     DEVRW           ;RESTORE ORIGINAL PATTERN
1149  006744  023737  001570  001126        CMP     DEVRW,BAD       ;DID THEY SET?
1150  006752  001402                        BEQ     3$              ;BR IF YES
1151  006754  104017                        ERROR+17                ;COULDN'T SET READ/WRITE BITS IN DEVAD
1152  006756  000534                        BR      12$             ;EXIT
1153  006760  005077  172566          3$:   CLR     @CSR            ;DISCONNECT
1154  006764  105777  172562          4$:   TSTB    @CSR            ;DONE?
1155  006770  100775                        BMI     4$              ;BR=NO
1156  006772  012737  000000  177776        MOV     #0,PSW          ;DROP PRIORITY
1157  007000  012777  000001  172544        MOV     #REQ,@CSR       ;CONNECT
1158  007006  105777  172540          5$:   TSTB    @CSR            ;CONNECTED?
1159  007012  100416                        BMI     6$              ;BR IF YES
1160  007014  005777  172532                TST     @CSR            ;TIMEOUT?
1161  007020  100372                        BPL     5$              ;BR IF NOT
1162  007022  017737  172524  001126        MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
1163  007030  013737  001702  001124        MOV     KONST,GOOD
1164  007036  062737  020201  001124        ADD     #20201,GOOD     ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
1165  007044  104006                        ERROR+6                 ;COULDN'T CONNECT
1166  007046  000500                        BR      12$             ;EXIT
1167  007050  017737  172512  001126  6$:   MOV     @DEVAD,BAD      ;STORE CONTENTS OF DEVAD
1168  007056  033737  001572  001126        BIT     DEVRC,BAD       ;RESET CONDITIONS MET?
1169  007064  001403                        BEQ     7$              ;BR=YES
1170  007066  012702  177777                MOV     #-1,R2          ;SET R2 FOR NO RESET
1171  007072  000401                        BR      8$
1172  007074  005002                7$:     CLR     R2              ;SET R2 FOR RESET
1173  007076  005737  001214          8$:   TST     $PASS           ;FIRST PASS?
1174  007102  001056                        BNE     11$             ;BR=NO
1175  007104  005702                        TST     R2              ;CHECK RESET CONDITION
1176  007106  001427                        BEQ     9$              ;BR=RESET
1177  007110  104401  007116                TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  007114  000423                        BR      64$             ;;GET OVER THE ASCIZ
 (1)                            ;;65$:     .ASCIZ  <200>/THIS DT07 INHIBITS RESET-IN-NEUTRAL./
 (1)  007164                    64$:
1178  007164  000422                        BR      10$
1179  007166                          9$:
 (1)  007166  104401  007174                TYPE    ,67$            ;;TYPE ASCIZ STRING
 (1)  007172  000417                        BR      66$             ;;GET OVER THE ASCIZ
 (1)                            ;;67$:     .ASCIZ  <200>/THIS DT07 RESETS IN NEUTRAL./
 (1)  007232                    66$:
1180  007232  010237  001704          10$:  MOV     R2,DEVCON       ;STORE CONDITION
1181  007236  000404                        BR      12$
1182  007240  020237  001704          11$:  CMP     R2,DEVCON       ;HAS IT CHANGED?
1183  007244  001401                        BEQ     12$             ;BR=NO
1184  007246  104022                        ERROR+22                ;RESET-IN-NEUTRAL OPERATION HAS CHANGED
1185  007250  005077  172276          12$:  CLR     @CSR            ;CLEAR AND DISCONNECT
1186
1187
```

```
 1189
 1190
 1191  007254  012737  002444  021232  MONEND: MOV     #RESTRT,$RTNAD  ;SET RETURN
 1192  007262  000137  021132                  JMP     $EOP            ;GO DO END PASS
 1193
```

```
1195
1196                                    .SBTTL  MULTIPORT TEST
1197
1198  007266  005737  025754            MULPRT: TST     PARFLG          ;CHECK FOR PARAMETERS
1199  007272  100402                            BMI     1$              ;BR=YES
1200  007274  004737  023612                    JSR     PC,PARIN        ;GO INPUT PARAMETERS
1201  007300  012737  001000  001656    1$:     MOV     #1000,CYCLE     ;SET COUNTER
1202  007306  005737  001704                    TST     DEVCON          ;RESET IN NEUTRAL OPTION USED?
1203  007312  001003                            BNE     2$              ;BR=NO
1204  007314  005037  001706                    CLR     TRIN            ;SET FLAG FOR TEST
1205  007320  000403                            BR      2$+6            ;CONTINUE
1206  007322  012737  177777  001706    2$:     MOV     #-1,TRIN        ;SET FLAG FOR NO TEST
1207  007330  005737  001564                    TST     DT03            ;DT03 MODE?
1208  007334  001402                            BEQ     3$              ;BR=DT07 MODE
1209  007336  000137  012422                    JMP     MULDT3          ;GO DO DT03 MULTIPORT
1210  007342  012777  022402  172204    3$:     MOV     #OOPS,@VEC0     ;SET RETURN
1211  007350  012777  000340  172200            MOV     #340,@VEC2      ;SET LEVEL
1212  007356  004737  021700                    JSR     PC,SEQMAK       ;GO CONSTRUCT SEQUENCER
1213  007362  004737  022246                    JSR     PC,SEQINT       ;INIT. SEQUENCER
1214  007366  012737  000000  177776            MOV     #0,PSW          ;DROP PRIORITY
1215  007374  023705  001574                    CMP     PORTNO,R5       ;MASTER OR SLAVE?
1216  007400  001066                            BNE     SLAVE           ;BR=SLAVE
1217  007402  012737  000632  021442            MOV     #632,TOCK       ;SET CLOCK FOR 5 SEC
1218  007410  004737  021402                    JSR     PC,WAIT
1219  007414  000405                            BR      MASTER          ;GO TO MASTER CODE AFTER SYNC WAIT
1220  007416  023705  001574            DISPAT: CMP     PORTNO,R5       ;MY TURN AS MASTER?
1221  007422  001402                            BEQ     MASTER          ;BR=YES
1222  007424  000137  007556                    JMP     SLAVE           ;GO BE SLAVE
1223  007430  004737  022010            MASTER: JSR     PC,MASMEN       ;GO MAKE MASTER MENU
1224  007434  012737  000001  021442            MOV     #1,TOCK
1225  007442  004737  021402                    JSR     PC,WAIT
1226  007446  004737  021254                    JSR     PC,CONNEC       ;GO CONNECT
1227  007452  005037  001652            1$:     CLR     CURFUN          ;INSURE NO EXTRANEOUS BITS PRESENT
1228  007456  116437  000001  001652            MOVB    1(R4),CURFUN    ;GET CURRANT FUNCTION
1229  007464  005037  001654                    CLR     CURID           ;CLEAN IT OUT
1230  007470  111437  001654                    MOVB    (R4),CURID      ;GET CURRANT ID
1231  007474  000241                            CLC
1232  007476  006137  001652                    ROL     CURFUN          ;DOUBLE IT
1233  007502  012703  012406                    MOV     #TABM-2,R3      ;LOAD BASE ADDRESS
1234  007506  063703  001652                    ADD     CURFUN,R3       ;ADD OFFSET
1235  007512  011303                            MOV     @R3,R3          ;GET ACTUAL ADDRESS
1236  007514  004713                            JSR     PC,@R3          ;GO DO IT
1237  007516  005724                            TST     (R4)+           ;STEP THRU MENU
1238  007520  022714  177777                    CMP     #-1,(R4)        ;DONE MENU?
1239  007524  001352                            BNE     1$              ;BR=NO
1240  007526  004737  022270                    JSR     PC,SEQSTP       ;GO STEP SEQUENCER
1241  007532  005337  001656                    DEC     CYCLE           ;COUNT
1242  007536  001402                            BEQ     2$              ;BR IF DONE
1243  007540  000137  007416                    JMP     DISPAT          ;GO START AGAIN
1244  007544  012737  007266  021232    2$:     MOV     #MULPRT,$RTNAD  ;SET UP RETURN
1245  007552  000137  021134                    JMP     $EOP+2          ;GO DO END PASS
1246  007556  004737  022076            SLAVE:  JSR     PC,SALMEN       ;GO MAKE SLAVE MENU
1247  007562  004737  021444                    JSR     PC,SYNCUP       ;GO SYNCUP! 2 MINUTE TIME OUT
1248  007566  005037  001652            1$:     CLR     CURFUN          ;CLEAN IT
1249  007572  116437  000001  001652            MOVB    1(R4),CURFUN    ;GET CURRANT FUNCTION
1250  007600  005037  001654                    CLR     CURID           ;CLEAN IT
```

```
1251  007604  111437  001654                  MOVB    (R4),CURID      ;GET CURRANT ID
1252  007610  000241                          CLC
1253  007612  006137  001652                  ROL     CURFUN          ;DOUBLE IT
1254  007616  012703  012406                  MOV     #TABM-2,R3      ;LOAD BASE ADDRESS
1255  007622  063703  001652                  ADD     CURFUN,R3       ;ADD OFFSET
1256  007626  011303                          MOV     @R3,R3          ;GET ACTUAL ADDRESS
1257  007630  004713                          JSR     PC,@R3          ;GO DO IT
1258  007632  005724                          TST     (R4)+           ;STEP THRU MENU
1259  007634  022714  177777                  CMP     #-1,(R4)        ;DONE MENU
1260  007640  001352                          BNE     1$              ;BR=NO
1261  007642  004737  022270                  JSR     PC,SEQSTP       ;GO STEP SEQUENCER
1262  007646  005337  001656                  DEC     CYCLE           ;COUNT
1263  007652  001402                          BEQ     2$              ;BRIF DONE
1264  007654  000137  007416                  JMP     DISPAT          ;GO START AGAIN
1265  007660  012737  007266  021232  2$:     MOV     #MULPRT,$RTNAD  ;SET UP RETURN
1266  007666  000137  021134                  JMP     $EOP+2          ;GO DO END PASS
1267
1268
1269                                          ;EXTERNAL CONNECT REQUEST REJECTED
1270
1271  007672  004737  021254          FUNCT1: JSR     PC,CONNEC       ;GO CONNECT
1272  007676  004737  021312                  JSR     PC,CONID        ;GO CONVERT SLAVE ID
1273  007702  012777  010042  171644          MOV     #1$,@VEC0       ;SET RETURN
1274  007710  112777  000101  171634          MOVB    #101,@CSR       ;ENABLE INTERUPT
1275  007716  005000                          CLR     R0              ;SET TIMER
1276  007720  005200                          INC     R0              ;TIME OUT
1277  007722  001376                          BNE     .-2             ;BR IF NOT DONE
1278  007724  104401  007732                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  007730  000434                          BR      64$             ;;GET OVER THE ASCIZ
 (1)                              ;;65$:      .ASCIZ  <200>/ERROR1A! NO EXTERNAL REQUEST INTERUPT DETECTED.  CSR= /
 (1)  010022                      64$:
1279  010022  017737  171524  001126          MOV     @CSR,BAD        ;SAVE CSR
1280  010030  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
 (1)  010034  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1281  010036  000000                          HALT
1282  010040  000776                          BR      .-2
1283  010042                           1$:
 (2)  010042  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 (2)  010044  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1284  010046  017737  171500  001126          MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
1285  010054  032737  010000  001126          BIT     #10000,BAD      ;MAKE SURE EXT INT IS SET
1286  010062  001036                          BNE     2$              ;BR = OK
1287  010064  104401  010072                  TYPE    ,67$            ;;TYPE ASCIZ STRING
 (1)  010070  000426                          BR      66$             ;;GET OVER THE ASCIZ
 (1)                              ;;67$:      .ASCIZ  <200>/ERROR1B! EXTERNAL INTERUPT NOT SET.  CSR= /
 (1)  010146                      66$:
1288  010146  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
 (1)  010152  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1289  010154  000000                          HALT
1290  010156  000776                          BR      .-2
1291  010160  033737  001660  001126  2$:     BIT     COVID,BAD       ;CORRECT SLAVE ID?
1292  010166  001042                          BNE     3$              ;BR=YES
1293  010170  104401  010176                  TYPE    ,69$            ;;TYPE ASCIZ STRING
 (1)  010174  000432                          BR      68$             ;;GET OVER THE ASCIZ
 (1)                              ;;69$:      .ASCIZ  <200>/ERROR1C! EXTERNAL REQUEST FROM WRONG PORT.  CSR= /
 (1)  010262                      68$:
```

```
 1294  010262  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  010266  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1295  010270  000000                          HALT
 1296  010272  000776                          BR      .-2
 1297  010274  012777  000101  171250  3$:     MOV     #101,@CSR       ;CLEAR EXT INT
 1298  010302  012777  022402  171244          MOV     #OOPS,@VEC0     ;RESET RETURN
 1299  010310  005037  177776                  CLR     PSW             ;LOWER PRIORITY
 1300  010314  000207                          RTS     PC
 1301
 1302
 1303                                  ;EXTERNAL REQUEST IGNORED
 1304
 1305  010316  004737  021254          FUNCT2: JSR     PC,CONNEC       ;GO CONNECT
 1306  010322  004737  021312                  JSR     PC,CONID        ;GO CONVERT SLAVE ID
 1307  010326  012777  010620  171220          MOV     #1$,@VEC0       ;SET RETURN
 1308  010334  005737  001706                  TST     TRIN            ;CAN WE TEST RESET IN NEUTRAL?
 1309  010340  001052                          BNE     6$              ;BR=NO
 1310  010342  013777  001570  171216          MOV     DEVRW,@DEVAD    ;LOAD READ/WRITE BITS
 1311  010350  023777  001570  171210          CMP     DEVRW,@DEVAD    ;LOADED?
 1312  010356  001443                          BEQ     6$              ;BR=YES
 1313  010360  104401  010366                  TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  010364  000432                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                                  ;;65$:  .ASCIZ  <200>/ERROR2F! COULDN'T LOAD EXTERNAL DEVICE REGISTER.../
  (1)  010452                          64$:
 1314  010452  013746  001566                  MOV     DEVAD,-(SP)     ;;SAVE DEVAD FOR TYPEOUT
  (1)  010456  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1315  010460  012737  177777  001706          MOV     #-1,TRIN        ;SET FLAG FOR NO TEST
 1316  010466  112777  000101  171056  6$:     MOVB    #101,@CSR       ;ENABLE INTERUPT
 1317  010474  005000                          CLR     R0              ;SET TIMER
 1318  010476  005200                          INC     R0              ;TIME OUT
 1319  010500  001376                          BNE     .-2             ;BR IF NOT DONE
 1320  010502  104401  010510                  TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  010506  000434                          BR      66$             ;;GET OVER THE ASCIZ
  (1)                                  ;;67$:  .ASCIZ  <200>/ERROR2A! NO EXTERNAL REQUEST INTERUPT DETECTED.  CSR= /
  (1)  010600                          66$:
 1321  010600  017737  170746  001126          MOV     @CSR,BAD        ;SAVE CSR
 1322  010606  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  010612  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1323  010614  000000                          HALT
 1324  010616  000776                          BR      .-2
 1325  010620                          1$:
  (2)  010620  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  (2)  010622  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 1326  010624  017737  170722  001126          MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 1327  010632  032737  010000  001126          BIT     #10000,BAD      ;MAKE SURE EXT INT IS SET
 1328  010640  001036                          BNE     2$              ;BR = OK
 1329  010642  104401  010650                  TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1)  010646  000426                          BR      68$             ;;GET OVER THE ASCIZ
  (1)                                  ;;69$:  .ASCIZ  <200>/ERROR2B! EXTERNAL INTERUPT NOT SET.  CSR= /
  (1)  010724                          68$:
 1330  010724  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  010730  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1331  010732  000000                          HALT
 1332  010734  000776                          BR      .-2
 1333  010736  033737  001660  001126  2$:     BIT     COVID,BAD       ;CORRECT SLAVE ID?
 1334  010744  001042                          BNE     3$              ;BR=YES
```

```
 1335  010746  104401  010754                  TYPE    ,71$            ;;TYPE ASCIZ STRING
  (1)  010752  000432                          BR      70$             ;;GET OVER THE ASCIZ
  (1)                                  ;;71$:  .ASCIZ  <200>/ERROR2C! EXTERNAL REQUEST FROM WRONG PORT.  CSR= /
  (1)  011040                          70$:
 1336  011040  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  011044  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1337  011046  000000                          HALT
 1338  011050  000776                          BR      .-2
 1339  011052  012777  011170  170474  3$:     MOV     #4$,@VEC0       ;SET RETURN
 1340  011060  005037  177776                  CLR     PSW             ;LOWER PRIORITY
 1341  011064  004737  021370                  JSR     PC,WAIT05       ;WAIT
 1342  011070  104401  011076                  TYPE    ,73$            ;;TYPE ASCIZ STRING
  (1)  011074  000430                          BR      72$             ;;GET OVER THE ASCIZ
  (1)                                  ;;73$:  .ASCIZ  <200>/ERROR2D! SLAVE DIDN'T TAKE SHARED BUSS. CSR= /
  (1)  011156                          72$:
 1343  011156  017746  170370                  MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  011162  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1344  011164  000000                          HALT
 1345  011166  000776                          BR      .-2
 1346  011170                          4$:
  (2)  011170  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  (2)  011172  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  (2)  011174  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 1347  011176  005777  170350                  TST     @CSR            ;TIMEOUT SET?
 1348  011202  100436                          BMI     5$              ;BR=YES
 1349  011204  104401  011212                  TYPE    ,75$            ;;TYPE ASCIZ STRING
  (1)  011210  000423                          BR      74$             ;;GET OVER THE ASCIZ
  (1)                                  ;;75$:  .ASCIZ  <200>/ERROR2E! NO TMO ON INTERUPT.  CSR= /
  (1)  011260                          74$:
 1350  011260  017737  170266  001126          MOV     @CSR,BAD        ;SAVE CSR
 1351  011266  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  011272  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1352  011274  000000                          HALT
 1353  011276  000776                          BR      .-2
 1354  011300  012777  022402  170246  5$:     MOV     #OOPS,@VEC0     ;RESET RETURN
 1355  011306  005077  170240                  CLR     @CSR            ;CLEAR CSR
 1356  011312  005037  177776                  CLR     PSW             ;LOWER PRIORITY
 1357  011316  000207                          RTS     PC              ;EXIT
 1358
 1359
 1360
 1361                                  ;SEND CONNECT REQUEST EXPECT REJECTION
 1362
 1363  011320  005000                  FUNCT3: CLR     R0              ;SET TIMERS
 1364  011322  012701  000400                  MOV     #400,R1
 1365  011326  032777  020000  170216  1$:     BIT     #20000,@CSR     ;SWITCHED BUSS ACTIVE?
 1366  011334  001051                          BNE     2$              ;BR=YES
 1367  011336  005200                          INC     R0              ;TIME OUT
 1368  011340  001372                          BNE     1$
 1369  011342  005301                          DEC     R1
 1370  011344  001370                          BNE     1$
 1371  011346  017737  170200  001126          MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 1372  011354  104401  011362                  TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  011360  000432                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                                  ;;65$:  .ASCIZ  <200>/ERROR3A! NO SWITCHED BUSS MASTER DETECTED. CSR = /
  (1)  011446                          64$:
```

```
 1373  011446  013746  001126                 MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  011452  104402                         TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1374  011454  000000                         HALT
 1375  011456  000776                         BR      .-2
 1376  011460  012777  011476  170066  2$:    MOV     #3$,@VEC0       ;SET RETURN
 1377  011466  012777  000101  170056         MOV     #101,@CSR       ;SET IE + REQ
 1378  011474  000001                         WAIT                    ;WAIT FOR INTERUPT
 1379  011476                        3$:
  (2)  011476  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
  (2)  011500  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
 1380  011502  005777  170044                 TST     @CSR            ;TIMEOUT?
 1381  011506  100435                         BMI     4$              ;BR=YES
 1382  011510  017737  170036  001126         MOV     @CSR,BAD        ;SAVE CSR
 1383  011516  104401  011524                 TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  011522  000422                         BR      66$             ;;GET OVER THE ASCIZ
  (1)                           ;;67$:  .ASCIZ  <200>/ERROR3B! NO TIMEOUT DETECTED. CSR=/
  (1)  011570                   66$:
 1384  011570  013746  001126                 MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  011574  104402                         TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1385  011576  000000                         HALT
 1386  011600  000776                         BR      .-2
 1387  011602  005077  167744         4$:     CLR     @CSR            ;CLEAR + DISCONNECT
 1388  011606  012777  022402  167740         MOV     #OOPS,@VEC0     ;RESET RETURN
 1389  011614  005037  177776                 CLR     PSW             ;LOWER PRIORITY
 1390  011620  000207                         RTS     PC              ;EXIT
 1391                                         ;SEND CONNECT REQUEST EXPECT CONNECTION
 1392  011622  005000                FUNCT4:  CLR     R0              ;SET TIMERS
 1393  011624  012701  000400                 MOV     #400,R1
 1394  011630  032777  020000  167714  1$:     BIT     #20000,@CSR     ;SWITCHED BUSS ACTIVE?
 1395  011636  001051                         BNE     2$              ;BR=YES
 1396  011640  005200                         INC     R0              ;TIME OUT
 1397  011642  001372                         BNE     1$
 1398  011644  005301                         DEC     R1
 1399  011646  001370                         BNE     1$
 1400  011650  017737  167676  001126         MOV     @CSR,BAD        ;STORE CONTENTS OF CSR
 1401  011656  104401  011664                 TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  011662  000432                         BR      64$             ;;GET OVER THE ASCIZ
  (1)                           ;;65$:  .ASCIZ  <200>/ERROR4A! NO SWITCHED BUSS MASTER DETECTED. CSR = /
  (1)  011750                   64$:
 1402  011750  013746  001126                 MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  011754  104402                         TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1403  011756  000000                         HALT
 1404  011760  000776                         BR      .-2
 1405  011762  013737  001676  001662  2$:     MOV     TIMA,TEMP1      ;SET TIMER
 1406  011770  006337  001662                 ASL     TEMP1
 1407  011774  032777  020000  167550  21$:    BIT     #20000,@CSR     ;SWB ACT SET?
 1408  012002  001707                         BEQ     FUNCT4  ;BR=NO
 1409  012004  005337  001662                 DEC     TEMP1
 1410  012010  001371                         BNE     21$
 1411  012012  012777  012030  167534         MOV     #3$,@VEC0       ;SET RETURN
 1412  012020  012777  000101  167524         MOV     #101,@CSR       ;SET IE + REQ
 1413  012026  000001                         WAIT                    ;WAIT FOR INTERUPT
 1414  012030                        3$:
  (2)  012030  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
  (2)  012032  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
 1415  012034  105777  167512                 TSTB    @CSR            ;CONNECTED?
```

```
                                                BMI     4$              ;BR=YES
                                                MOV     @CSR,BAD        ;SAVE CSR
                                                TYPE    ,67$            ;;TYPE ASCIZ STRING
                                                BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  <200>/ERROR4B! CONNECT REQUEST FAILED. CSR=/
                                        66$:
                                                MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
                                                TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                                HALT
                                                BR      .-2
                                        4$:     TST     TRIN            ;CAN WE TEST RESET IN NEUTRAL
                                                BNE     5$              ;BR=NO
                                                BIT     DEVRC,@DEVAD    ;RESET CONDITIONS MET?
                                                BEQ     5$              ;BR=YES
                                                TYPE    ,69$            ;;TYPE ASCIZ STRING
                                                BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <200>/ERROR4C! NO RESET IN NEUTRAL DETECTED./
                                        68$:
                                                MOV     #-1,TRIN        ;CANCEL FURTHER TESTING OF RESET IN NEUTRAL
                                        5$:     CLR     @CSR            ;CLEAR + DISCONNECT
                                                MOV     #OOPS,@VEC0     ;RESET RETURN
                                                CLR     PSW             ;LOWER PRIORITY
                                                RTS     PC              ;EXIT
                                                ;OBSERVE REQUEST INDICATORS
                                        FUNCT5: JSR     PC,CONID        ;GO CONVERT PORT ID
                                                CLR     R0              ;SET TIMERS
                                                MOV     #400,R1
                                        1$:     BIT     COVID,@CSR      ;LOOK FOR REQUEST
                                                BNE     2$              ;BR=FOUND IT
                                                INC     R0              ;TIME OUT
                                                BNE     1$
                                                DEC     R1
                                                BNE     1$
                                                TYPE    ,65$            ;;TYPE ASCIZ STRING
                                                BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <200>/ERROR5A! NO REQUEST ACTIVITY DETECTED./
                                        64$:
                                                HALT
                                                BR      .-2
                                        2$:     BIT     COVID,@CSR      ;DID IT GO AWAY?
                                                BNE     2$              ;BR=NOT YET
                                                RTS     PC              ;EXIT


                                        TABM:   FUNCT1                  ;DISPATCH TABLE FOR MULTIPORT TEST
                                                FUNCT2
                                                FUNCT3
                                                FUNCT4
                                                FUNCT5
```

```
1458
1459
1460
1461                                    .SBTTL  BIPORT TEST IN DT03 MODE
1462
1463  012422  012737  000000  177776  MULDT3: MOV     #0,PSW          ;LOWER PRIORITY
1464  012430  005077  167116          CLR     @CSR            ;START WITH A CLEAR REGISTER
1465  012434  012737  010000  001656  MOV     #10000,CYCLE    ;SET SUB COUNT
1466  012442  005037  021366          CLR     TRIG            ;CLEAR TRIGGER
1467  012446  032777  020000  167076  1$:     BIT     #20000,@CSR     ;ACTIVE?
1468  012454  001401                  BEQ     11$             ;BR=NO
1469  012456  000464                  BR      21$             ;GO BE SLAVE
1470  012460  012777  012604  167066  11$:    MOV     #2$,@VEC0       ;SET RETURN
1471  012466  012777  000101  167056          MOV     #101,@CSR       ;SEND REQUEST
1472  012474  004737  021354          JSR     PC,WAIT20       ;GO WAIT
1473  012500  104401  012506          TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  012504  000432                  BR      64$             ;;GET OVER THE ASCIZ
 (1)                                  ;;65$:  .ASCIZ  <200>/ERROR6! NO INTERUPT AFTER CONNECT REQUEST.  CSR= /
 (1)  012572                          64$:
1474  012572  017746  166754          MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  012576  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1475  012600  000000                  HALT
1476  012602  000776                  BR      .-2
1477  012604  012706  001100          2$:     MOV     #STACK,SP       ;RESET STACK
1478  012610  005037  177776          CLR     PSW             ;LOWER PRIORITY
1479  012614  005777  166732          TST     @CSR            ;TIMEOUT?
1480  012620  100007                  BPL     3$              ;BR=NO
1481  012622  000240                  NOP
1482  012624  000240                  NOP
1483  012626  000240                  NOP
1484  012630  005037  001674  21$:    CLR     ONOFF           ;INDIACATE FIRST SLAVE
1485  012634  000137  014014          JMP     SECT4           ;GO TO SECTION 4
1486  012640  012737  177777  001674  3$:     MOV     #-1,ONOFF       ;INDIACATE FIRST MASTER
1487
1488                                    ;SECTION 2 : WAIT FOR EXTERNAL REQUEST THEN RELEASE SWITCH
1489
1490  012646  005737  001706  SECT2:  TST     TRIN            ;CAN WE TEST RESET IN NEUTRAL
1491  012652  001052                  BNE     2$              ;BR=NO
1492  012654  013777  001570  166704  MOV     DEVRW,@DEVAD    ;LOAD DEVICE REGISTER
1493  012662  023777  001570  166676  CMP     DEVRW,@DEVAD    ;OK?
1494  012670  001443                  BEQ     2$              ;BR=YES
1495  012672  104401  012700          TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  012676  000432                  BR      64$             ;;GET OVER THE ASCIZ
 (1)                                  ;;65$:  .ASCIZ  <200>/ERROR7A! COULDN'T LOAD EXTERNAL DEVICE REGISTER../
 (1)  012764                          64$:
1496  012764  013746  001566          MOV     DEVAD,-(SP)     ;;SAVE DEVAD FOR TYPEOUT
 (1)  012770  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1497  012772  012737  177777  001706  MOV     #-1,TRIN        ;CANCEL FURTHER TESTING OF RESET IN NEUTRAL
1498  013000  012777  013124  166546  2$:     MOV     #1$,@VEC0       ;SET RETURN
1499  013006  012737  000632  021442          MOV     #632,TOCK       ;SET DELAY FOR 5 SECONDS
1500  013014  004737  021402          JSR     PC,WAIT
1501  013020  104401  013026          TYPE    ,67$            ;;TYPE ASCIZ STRING
 (1)  013024  000432                  BR      66$             ;;GET OVER THE ASCIZ
 (1)                                  ;;67$:  .ASCIZ  <200>/ERROR7! NO EXTERNAL INTERUPT REQUEST SEEN.  CSR= /
 (1)  013112                          66$:
1502  013112  017746  166434          MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
```

```
  (1) 013116 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1503  013120 000000                          HALT
1504  013122 000776                          BR      .-2
1505  013124 012706 001100          1$:      MOV     #STACK,SP       ;RESET STACK
1506  013130 013737 001702 001124           MOV     KONST,GOOD
1507  013136 062737 030301 001124           ADD     #30301,GOOD
1508  013144 023777 001124 166400           CMP     GOOD,@CSR       ;MAKE SURE IT'S AN EXTERNAL REQUEST
1509  013152 001435                          BEQ     SECT2A          ;BR=OK
1510  013154 104401 013162                   TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1) 013160 000425                          BR      68$             ;;GET OVER THE ASCIZ
  (1)                           ;;69$:   .ASCIZ  <200>/ERROR8! UNEXPECTED INTERUPT TYPE.  CSR= /
  (1) 013234                    68$:
1511  013234 017746 166312                   MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1) 013240 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1512  013242 000000                          HALT
1513  013244 000776                          BR      .-2
1514  013246 000401                SECT2A:  BR      1$              ;RELEASE TYPE SWITCHBACK
1515  013250 000420                          BR      SECT2B          ;GO DO PASSIVE RELEASE
1516  013252 012737 000240 013246   1$:      MOV     #240,SECT2A     ;TURN SWITCHBACK
1517  013260 005077 166266                   CLR     @CSR            ;DROP SWITCH ACTIVELY
1518  013264 005037 177776                   CLR     PSW             ;LOWER PRIORITY
1519  013270 105777 166256                   TSTB    @CSR            ;DROPPED?
1520  013274 100775                          BMI     .-4             ;BR=NO
1521  013276 032777 020000 166246           BIT     #20000,@CSR     ;PICKED UP?
1522  013304 001774                          BEQ     .-6             ;BR=NO
1523  013306 000137 013542                   JMP     SECT2C          ;GO TO EXIT
1524  013312 012737 000401 013246   SECT2B:  MOV     #401,SECT2A     ;TURN SWITCHBACK
1525  013320 012777 013434 166226           MOV     #1$,@VEC0       ;SET RETURN
1526  013326 005037 177776                   CLR     PSW             ;LOWER PRIORITY
1527  013332 004737 021354                   JSR     PC,WAIT20       ;GO WAIT FOR TIMEOUT
1528  013336 104401 013344                   TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 013342 000427                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                           ;;65$:   .ASCIZ  <200>/ERROR9! NO TIMEOUT INTERUPT DETECTED. CSR= /
  (1) 013422                    64$:
1529  013422 017746 166124                   MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1) 013426 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1530  013430 000000                          HALT
1531  013432 000776                          BR      .-2
1532  013434 012706 001100          1$:      MOV     #STACK,SP       ;RESET STACK
1533  013440 005777 166106                   TST     @CSR            ;MAKE SURE IT'S A TIMEOUT
1534  013444 100436                          BMI     SECT2C          ;BR=OK
1535  013446 104401 013454                   TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1) 013452 000426                          BR      66$             ;;GET OVER THE ASCIZ
  (1)                           ;;67$:   .ASCIZ  <200>/ERROR10! INTERUPT NOT CAUSED BY TMO. CSR= /
  (1) 013530                    66$:
1536  013530 017746 166016                   MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1) 013534 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1537  013536 000000                          HALT
1538  013540 000776                          BR      .-2
1539  013542 005077 166004         SECT2C:  CLR     @CSR            ;CLEAR CSR
1540  013546 005037 177776                   CLR     PSW             ;LOWER PRIORITY
1541
1542                                ;SECTION 3 : SEND REQUEST EXPECT REJECTION
1543
1544  013552 012777 013662 165774   SECT3:   MOV     #1$,@VEC0       ;SET RETURN
1545  013560 012777 000101 165764            MOV     #101,@CSR       ;SEND REQUEST
```

```
1546  013566  004737  021354          JSR     PC,WAIT20       ;GO WAIT FOR TIMEOUT
1547  013572  104401  013600          TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  013576  000424                  BR      64$             ;;GET OVER THE ASCIZ
 (1)                          ;;65$:  .ASCIZ  <200>/ERROR11! NO INTERUPT DETECTED.  CSR= /
 (1)  013650                  64$:
1548  013650  017746  165676          MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  013654  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1549  013656  000000                  HALT
1550  013660  000776                  BR      .-2
1551  013662  012706  001100  1$:     MOV     #STACK,SP       ;RESET STACK
1552  013666  005777  165660          TST     @CSR            ;TIMEOUT?
1553  013672  100437                  BMI     2$              ;BR=YES
1554  013674  104401  013702          TYPE    ,67$            ;;TYPE ASCIZ STRING
 (1)  013700  000427                  BR      66$             ;;GET OVER THE ASCIZ
 (1)                          ;;67$:  .ASCIZ  <200>/ERROR12! INTERUPT NOT CAUSED BY TMO.  CSR= /
 (1)  013760                  66$:
1555  013760  017746  165566          MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  013764  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1556  013766  000000                  HALT
1557  013770  000776                  BR      .-2
1558  013772  005077  165554  2$:     CLR     @CSR            ;CLEAR CSR
1559  013776  005037  177776          CLR     PSW             ;LOWER PRIORITY
1560  014002  005737  021366          TST     TRIG            ;IS TRIGGER SET?
1561  014006  001402                  BEQ     SECT4           ;BR=NO
1562  014010  000137  014660          JMP     MULEND          ;FINISHED A SUB PASS
1563
1564                                  ;SECTION 4 : SEND REQUEST EXPECT CONNECTION
1565
1566  014014  012777  014124  165532  SECT4:  MOV     #1$,@VEC0       ;SET RETURN
1567  014022  012777  000101  165522          MOV     #101,@CSR       ;SEND REQUEST
1568  014030  004737  021354                  JSR     PC,WAIT20       ;GO WAIT FOR TIMEOUT
1569  014034  104401  014042                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  014040  000424                  BR      64$             ;;GET OVER THE ASCIZ
 (1)                          ;;65$:  .ASCIZ  <200>/ERROR13! NO INTERUPT DETECTED.  CSR= /
 (1)  014112                  64$:
1570  014112  017746  165434          MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  014116  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1571  014120  000000                  HALT
1572  014122  000776                  BR      .-2
1573  014124  012706  001100  1$:     MOV     #STACK,SP       ;RESET STACK
1574  014130  013737  001702  001124  MOV     KONST,GOOD
1575  014136  062737  020301  001124  ADD     #20301,GOOD
1576  014144  023777  001124  165400  CMP     GOOD,@CSR       ;CONNECTED?
1577  014152  001432                  BEQ     SECT5           ;BR=YES
1578  014154  104401  014162          TYPE    ,67$            ;;TYPE ASCIZ STRING
 (1)  014160  000422                  BR      66$             ;;GET OVER THE ASCIZ
 (1)                          ;;67$:  .ASCIZ  <200>/ERROR14! COULD NOT CONNECT.  CSR= /
 (1)  014226                  66$:
1579  014226  017746  165320          MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  014232  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1580  014234  000000                  HALT
1581  014236  000776                  BR      .-2
1582
1583                                  ;SECTION 5 :DETECT EXTERNAL REQUEST AND REJECT IT
1584
1585  014240  005737  001706  SECT5:  TST     TRIN            ;CAN WE TEST RESET IN NEUTRAL?
```

```
                                                BNE     3$              ;BR=NO
                                                BIT     DEVRC,@DEVAD    ;RESET CONDITION MET
                                                BEQ     3$              ;BR=YES
                                                TYPE    ,65$            ;;TYPE ASCIZ STRING
                                                BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <200>/ERROR15A! COULDN'T DETECT RESET IN NEUTRAL./
                                        64$:
                                                MOV     #-1,TRIN        ;CANCEL FURTHER TESTING OF RESET IN NEUTRAL
                                        3$:     MOV     #1$,@VEC0       ;SET RETURN
                                                CLR     PSW             ;LOWER PRIORITY
                                                JSR     PC,WAIT20       ;GO WAIT
                                                TYPE    ,67$            ;;TYPE ASCIZ STRING
                                                BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  <200>/ERROR15! NO EXTERNAL REQUEST DETECTED.  CSR= /
                                        66$:
                                                MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
                                                TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                                HALT
                                                BR      .-2
                                        1$:     MOV     #STACK,SP       ;RESET STACK
                                                MOV     KONST,GOOD
                                                ADD     #30301,GOOD
                                                CMP     GOOD,@CSR       ;EXT. INT. ?
                                                BEQ     2$              ;BR=OK
                                                TYPE    ,69$            ;;TYPE ASCIZ STRING
                                                BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <200>/ERROR16! UNEXPECTED INTERUPT.  CSR= /
                                        68$:
                                                MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
                                                TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                                HALT
                                                BR      .-2
                                        2$:     MOV     #101,@CSR       ;CLEAR EXT. INT.
                                                MOV     #OOPS,@VEC0     ;RESET RETURN
                                                CLR     PSW             ;LOWER PRIORITY

                                        DT3SP:  DEC     CYCLE           ;COUNT A SUBPASS
                                                BEQ     1$              ;BR=DONE
                                                JMP     SECT2           ;CONTINUE
                                        1$:     TST     ONOFF           ;FIRST MASTER OR SLAVE
                                                BEQ     2$              ;BR=SLAVE
                                                BR      MULEND          ;MASTER REPORTS ENDPASS
                                        2$:     MOV     #-1,TRIG        ;SET TRIGGER FOR SLAVE
                                                JMP     SECT2           ;CONTINUE SLAVE AT SECT2

                                        MULEND: TST     ONOFF           ;WAS THIS FIRST SLAVE?
                                                BEQ     1$              ;BR=YES
                                                MOV     #MULDT3,$RTNAD  ;SET RETURN
                                                JMP     $EOP+2          ;GO DO ENDPASS
                                        1$:     MOV     #2$,$RTNAD      ;SET RETURN
                                                JMP     $EOP+2          ;GO DO ENDPASS
                                        2$:     JSR     PC,WAIT20       ;WAIT 2 SECONDS
                                                JMP     MULDT3          ;GO TO IT!
```

```
1629
1630                                    ;MANUAL INTERVENTION TEST
1631
1632  014722  005737  001564            MANIN:  TST     DT03            ;CHECK DT03 MODE
1633  014726  001402                            BEQ     MANMOD          ;BR=DT07 MODE
1634  014730  000137  016000                    JMP     PWROK           ;SKIP OVER MANUAL MODE TEST IF DT03
1635  014734  005077  164612            MANMOD: CLR     @CSR            ;CLEAR CSR
1636  014740  104401  014746                    TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  014744  000427                            BR      64$             ;;GET OVER THE ASCIZ
 (1)                                    ;;65$:  .ASCIZ  <200>/PUT THIS DT07 IN MANUAL MODE THEN TYPE <CR>./
 (1)  015024                            64$:
1637  015024  104410                            RDCHR
1638  015026  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
1639  015030  032777  002000  164514    1$:     BIT     #2000,@CSR      ;MANUAL MODE SET?
1640  015036  001040                            BNE     2$              ;BR=YES
1641  015040  104401  015046                    TYPE    ,67$            ;;TYPE ASCIZ STRING
 (1)  015044  000430                            BR      66$             ;;GET OVER THE ASCIZ
 (1)                                    ;;67$:  .ASCIZ  <200>/ERROR1!  MANUAL MODE (B T10) IS NOT SET. CSR= /
 (1)  015126                            66$:
1642  015126  017746  164420                    MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  015132  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1643  015134  000000                            HALT
1644  015136  000676                            BR      MANMOD          ;GO TRY AGAIN
1645  015140  012777  000001  164404    2$:     MOV     #1,@CSR         ;TRY TO SET REQ IN MAN.MOD.
1646  015146  032777  000001  164376            BIT     #1,@CSR         ;SET?
1647  015154  001443                            BEQ     3$              ;BR=NO
1648  015156  104401  015164                    TYPE    ,69$            ;;TYPE ASCIZ STRING
 (1)  015162  000433                            BR      68$             ;;GET OVER THE ASCIZ
 (1)                                    ;;69$:  .ASCIZ  <200>/ERROR2! REQ BIT NOT DISABLED IN MANUAL MODE.  CSR= /
 (1)  015252                            68$:
1649  015252  017746  164274                    MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  015256  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1650  015260  000000                            HALT
1651  015262  000726                            BR      2$              ;TRY AGAIN
1652  015264                            3$:
 (1)  015264  104401  015272                    TYPE    ,71$            ;;TYPE ASCIZ STRING
 (1)  015270  000426                            BR      70$             ;;GET OVER THE ASCIZ
 (1)                                    ;;71$:  .ASCIZ  <200>/CONNECT THIS DT07 MANUALY THEN TYPE <CR>./
 (1)  015346                            70$:
1653  015346  104410                            RDCHR
1654  015350  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
1655  015352  032777  000001  164172            BIT     #1,@CSR         ;IS REQ SET NOW?
1656  015360  001041                            BNE     4$              ;BR=YES
1657  015362  104401  015370                    TYPE    ,73$            ;;TYPE ASCIZ STRING
 (1)  015366  000431                            BR      72$             ;;GET OVER THE ASCIZ
 (1)                                    ;;73$:  .ASCIZ  <200>/ERROR3! REQ. BIT NOT SET WHEN CONNECTED.  CSR= /
 (1)  015452                            72$:
1658  015452  017746  164074                    MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)  015456  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1659  015460  000000                            HALT
1660  015462  000700                            BR      3$
1661  015464  000005                    4$:     RESET                   ;RESET THE BUSS
1662  015466  032777  000001  164056            BIT     #1,@CSR         ;REQ STILL THERE?
1663  015474  001034                            BNE     5$              ;BR=YES
1664  015476  104401  015504                    TYPE    ,75$            ;;TYPE ASCIZ STRING
 (1)  015502  000424                            BR      74$             ;;GET OVER THE ASCIZ
```

```
  (1)
  (1)  015554
1665   015554  017746  163772          ;;75$:  .ASCIZ  <200>/ERROR4! RESET CLEARED REQ. BIT.  CSR= /
 74$:
                       MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  015560  104402  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1666   015562  000000  HALT
1667   015564  000737  BR      4$
1668   015566
 5$:
  (1)  015566  104401  015574  TYPE    ,77$            ;;TYPE ASCIZ STRING
  (1)  015572  000432  BR      76$             ;;GET OVER THE ASCIZ
  (1)                 ;;77$:  .ASCIZ  <200>/PUT THIS DT07 IN PROGRAMMABLE MODE THEN TYPE <CR>./
  (1)  015660
 76$:
1669   015660  104410  RDCHR
1670   015662  012600  MOV     (SP)+,R0        ;;POP STACK INTO R0
1671   015664  032777  002000  163660  BIT     #2000,@CSR      ;MAN.MOD. GONE?
1672   015672  001442  BEQ     PWROK           ;BR=YES
1673   015674  104401  015702  TYPE    ,79$            ;;TYPE ASCIZ STRING
  (1)  015700  000432  BR      78$             ;;GET OVER THE ASCIZ
  (1)                 ;;79$:  .ASCIZ  <200>/ERROR5!  MANUAL MODE (BIT10) DIDN'T CLEAR.  CSR= /
  (1)  015766
 78$:
1674   015766  017746  163560  MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  015772  104402  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1675   015774  000000  HALT
1676   015776  000673  BR      5$
1677   016000  005077  163546  PWROK:  CLR     @CSR            ;CLEAR DEVICE
1678   016004  104401  016012  TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  016010  000422  BR      64$             ;;GET OVER THE ASCIZ
  (1)                 ;;65$:  .ASCIZ  <200>/POWER UP ALL DT07S THEN TYPE <CR>./
  (1)  016056
 64$:
1679   016056  104410  RDCHR
1680   016060  012600  MOV     (SP)+,R0        ;;POP STACK INTO R0
1681   016062  032777  004000  163462  1$:     BIT     #4000,@CSR      ;PWROK OK?
1682   016070  001034  BNE     2$              ;BR=YES
1683   016072  104401  016100  TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  016076  000424  BR      66$             ;;GET OVER THE ASCIZ
  (1)                 ;;67$:  .ASCIZ  <200>/ERROR6! PWR.OK (BIT11) NOT SET.  CSR= /
  (1)  016150
 66$:
1684   016150  017746  163376  MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  016154  104402  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1685   016156  000000  HALT
1686   016160  000707  BR      PWROK
1687   016162  005037  177776  2$:     CLR     PSW             ;LOWER PRIORITY
1688   016166  004737  021254  JSR     PC,CONNEC       ;GO CONNECT
1689   016172  104401  016200  TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1)  016176  000424  BR      68$             ;;GET OVER THE ASCIZ
  (1)                 ;;69$:  .ASCIZ  <200>/POWER DOWN A DT07 PORT THEN TYPE <CR>./
  (1)  016250
 68$:
1690   016250  104410  RDCHR
1691   016252  012600  MOV     (SP)+,R0        ;;POP STACK INTO R0
1692   016254  032777  004000  163270  BIT     #4000,@CSR      ;PWROK NOT OK?
1693   016262  001437  BEQ     3$              ;BR=OK
1694   016264  104401  016272  TYPE    ,71$            ;;TYPE ASCIZ STRING
  (1)  016270  000427  BR      70$             ;;GET OVER THE ASCIZ
  (1)                 ;;71$:  .ASCIZ  <200>/ERROR7! PWR. OK (BIT11) NOT CLEARED.  CSR= /
  (1)  016350
 70$:
1695   016350  017746  163176  MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  016354  104402  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
 1696  016356  000000                                    HALT
 1697  016360  000700                                    BR      2$
 1698  016362  032777  000200  163162  3$:               BIT     #200,@CSR       ;STILL CONNECTED?
 1699  016370  001032                                    BNE     4$              ;BR=YES
 1700  016372  104401  016400                            TYPE    ,73$            ;;TYPE ASCIZ STRING
  (1)  016376  000421                                    BR      72$             ;;GET OVER THE ASCIZ
  (1)                                           ;;73$:   .ASCIZ  <200>/ERROR8! LOST SHARED BUSS.  CSR= /
  (1)  016442                                   72$:
 1701  016442  017746  163104                            MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  016446  104402                                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1702  016450  000000                                    HALT
 1703  016452  000137  016000                            JMP     PWROK
 1704  016456                                4$:
  (1)  016456  104401  016464                            TYPE    ,75$            ;;TYPE ASCIZ STRING
  (1)  016462  000433                                    BR      74$             ;;GET OVER THE ASCIZ
  (1)                                           ;;75$:   .ASCIZ  <200>/RESTORE ALL DT07S TO ORIGINAL STATE THEN TYPE <CR>./
  (1)  016552                                   74$:
 1705  016552  104410                                    RDCHR
 1706  016554  012600                                    MOV     (SP)+,R0        ;;POP STACK INTO R0
 1707
 1708  016556  005077  162770                SWBPWF:CLR      @CSR            ;CLEAR DEVICE
 1709  016562  013746  000024                            MOV     @#PWRVEC,-(SP)          ;;PUSH @#PWRVEC ON STACK
 1710  016566  012737  016776  000024                    MOV     #11$,@#PWRVEC   ;SET RETURN
 1711  016574  012777  000100  162750                    MOV     #100,@CSR       ;SET I.E.
 1712  016602  005037  177776                1$:       CLR     PSW             ;LOWER PRIORITY
 1713  016606  104401  016614                            TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  016612  000426                                    BR      64$             ;;GET OVER THE ASCIZ
  (1)                                           ;;65$:   .ASCIZ  <200>/POWER DOWN THE SHARED BUSS THEN TYPE <CR>./
  (1)  016670                                   64$:
 1714  016670  104410                                    RDCHR
 1715  016672  012600                                    MOV     (SP)+,R0        ;;POP STACK INTO R0
 1716  016674  032777  040000  162650                    BIT     #40000,@CSR     ;SWB PWF SET?
 1717  016702  001112                                    BNE     2$              ;BR=YES
 1718  016704  104401  016712                            TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  016710  000425                                    BR      66$             ;;GET OVER THE ASCIZ
  (1)                                           ;;67$:   .ASCIZ  <200>/ERROR8!  SWB PWF (BIT14) NOT SET.  CSR= /
  (1)  016764                                   66$:
 1719  016764  017746  162562                            MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  016770  104402                                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1720  016772  000000                                    HALT
 1721  016774  000702                                    BR      1$
 1722  016776  012706  001100                11$:       MOV     #STACK,SP       ;RESET STACK
 1723  017002  005746                                    TST     -(SP)           ;POSITION STACK
 1724  017004  012637  000024                            MOV     (SP)+,@#PWRVEC          ;;POP STACK INTO @#PWRVEC
 1725  017010  104401  017016                            TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1)  017014  000440                                    BR      68$             ;;GET OVER THE ASCIZ
  (1)                                           ;;69$:   .ASCIZ  <200>/ERROR9!  TRAP THRU POWER FAIL VECTOR WHEN DISCONNECTED.  CSR= /
  (1)  017116                                   68$:
 1726  017116  017746  162430                            MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  017122  104402                                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1727  017124  000000                                    HALT
 1728  017126  000613                                    BR      SWBPWF
 1729  017130                                2$:
  (1)  017130  104401  017136                            TYPE    ,71$            ;;TYPE ASCIZ STRING
  (1)  017134  000423                                    BR      70$             ;;GET OVER THE ASCIZ
  (1)                                           ;;71$:   .ASCIZ  <200>/POWER UP SHARED BUSS THEN TYPE <CR>/
```

```
  (1) 017204                                    70$:
1730  017204  104410                                    RDCHR
1731  017206  012600                                    MOV     (SP)+,R0        ;;POP STACK INTO R0
1732  017210  032777  040000  162334                    BIT     #40000,@CSR     ;DID IT CLEAR?
1733  017216  001441                                    BEQ     3$              ;BR=YES
1734  017220  104401  017226                            TYPE    ,73$            ;;TYPE ASCIZ STRING
  (1) 017224  000430                                    BR      72$             ;;GET OVER THE ASCIZ
  (1)                                           ;;73$:  .ASCIZ  <200>/ERROR10!  SWB PWF (BIT14) DIDN'T CLEAR.  CSR= /
  (1) 017306                                    72$:
1735  017306  017746  162240                            MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1) 017312  104402                                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1736  017314  000000                                    HALT
1737  017316  000137  016602                            JMP     1$
1738              017322                    ERREN=.
1739  017322  005077  162224                3$:         CLR     @CSR    ;CLEAR IT
1740  017326  012737  017560  000024                    MOV     #PFVSEV,@#PWRVEC        ;SET RETURNS
1741  017334  012777  020404  162212                    MOV     #PWFSER,@VEC0
1742  017342  005037  177776                            CLR     PSW
1743  017346  004737  021254                            JSR     PC,CONNEC       ;GO CONNECT
1744  017352  052777  000100  162172                    BIS     #100,@CSR       ;SET I.E.
1745  017360  104401  017366                            TYPE    ,75$            ;;TYPE ASCIZ STRING
  (1) 017364  000423                                    BR      74$             ;;GET OVER THE ASCIZ
  (1)                                           ;;75$:  .ASCIZ  <200>/POWER THE SHARED BUSS DOWN THEN UP./
  (1) 017434                                    74$:
1746  017434  012737  011470  021442                    MOV     #11470,TOCK     ;SET TIMER FOR 1 MIN.
1747  017442  004737  021402                            JSR     PC,WAIT
1748  017446  104401  017454                            TYPE    ,77$            ;;TYPE ASCIZ STRING
  (1) 017452  000435                                    BR      76$             ;;GET OVER THE ASCIZ
  (1)                                           ;;77$:  .ASCIZ  <200>/ERROR11!  NO EVIDENCE OF POWER FAILURE DETECTED.  CSR= /
  (1) 017546                                    76$:
1749  017546  017746  162000                            MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1) 017552  104402                                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1750  017554  000000                                    HALT
1751  017556  000661                                    BR      3$
1752
1753  017560  017737  161766  001126  PFVSEV: MOV     @CSR,BAD        ;SAVE CSR CONTENTS
1754  017566  042737  000400  001126          BIC     #400,BAD        ;MASK BRQ IF PRESENT
1755  017574  042777  000100  161750          BIC     #100,@CSR       ;DROP I.E.
1756  017602  012737  017612  000024          MOV     #1$,@#PWRVEC    ;POINT TO POWER UP ROUTINE
1757  017610  000000                          HALT    ;IF YOU STOPPED HERE NO POWER UP TRAP OCCURED
1758  017612  017737  161734  002554  1$:     MOV     @CSR,TEMP2      ;STORE CSR
1759  01762C  013737  001702  001124          MOV     KONST,GOOD      ;ASSEMBLE GOOD DATA
1760  017626  062737  060301  001124          ADD     #60301,GOOD
1761  017634  023737  001124  001126          CMP     GOOD,BAD        ;CORRECT AT PWR DOWN?
1762  017642  001465                          BEQ     2$              ;BR=YES
1763  017644  104401  017652                  TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 017650  000433                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                                   ;;65$:  .ASCIZ  <200>/ERROR12!  CSR INCORRECT @ POWER DOWN TIME.  CSR WAS /
  (1) 017740                            64$:
1764  017740  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1) 017744  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1765  017746  104401  017754                  TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1) 017752  000410                          BR      66$             ;;GET OVER THE ASCIZ
  (1)                                   ;;67$:  .ASCIZ  <200>/IT SHOULD BE /
  (1) 017774                            66$:
1766  017774  013746  001124                  MOV     GOOD,-(SP)      ;;SAVE GOOD FOR TYPEOUT
```

```
  (1) 020000 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1767  020002 000000                          HALT
1768  020004 012706 001100          11$:     MOV     #STACK,SP       ;RESET STACK
1769  020010 005746                          TST     -(SP)           ;POSITION STACK
1770  020012 000137 017322                   JMP     ERREN
1771  020016 013737 002554 001126   2$:     MOV     TEMP2,BAD       ;GET PWR UP DATA
1772  020024 042737 000400 001126           BIC     #400,BAD        ;MASK BRQ IF PRESENT
1773  020032 162737 020301 001124           SUB     #20301,GOOD     ;CORRECT GOOD DATA
1774  020040 023737 001124 001126           CMP     GOOD,BAD        ;CORRECT @ PWR UP?
1775  020046 001460                          BEQ     3$              ;BR=YES
1776  020050 104401 020056                   TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1) 020054 000432                          BR      68$             ;;GET OVER THE ASCIZ
  (1)                           ;;69$:     .ASCIZ  <200>/ERROR13!  CSR INCORRECT @ POWER UP TIME. CSR WAS /
  (1) 020142                    68$:
1777  020142 013746 001126                   MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1) 020146 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1778  020150 104401 020156                   TYPE    ,71$            ;;TYPE ASCIZ STRING
  (1) 020154 000410                          BR      70$             ;;GET OVER THE ASCIZ
  (1)                           ;;71$:     .ASCIZ  <200>/IT SHOULD BE /
  (1) 020176                    70$:
1779  020176 013746 001124                   MOV     GOOD,-(SP)      ;;SAVE GOOD FOR TYPEOUT
  (1) 020202 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1780  020204 000000                          HALT
1781  020206 000676                          BR      11$
1782  020210 032777 040000 161334   3$:     BIT     #40000,@CSR     ;DID SWB PWF CLEAR?
1783  020216 001440                          BEQ     4$              ;BR=YES
1784  020220 104401 020226                   TYPE    ,73$            ;;TYPE ASCIZ STRING
  (1) 020224 000430                          BR      72$             ;;GET OVER THE ASCIZ
  (1)                           ;;73$:     .ASCIZ  <200>/ERROR14!  SWB PWF (BIT14) DIDN'T CLEAR.  CSR= /
  (1) 020306                    72$:
1785  020306 017746 161240                   MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1) 020312 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1786  020314 000000                          HALT
1787  020316 000632                          BR      11$
1788  020320                        4$:
  (1) 020320 104401 020326                   TYPE    ,75$            ;;TYPE ASCIZ STRING
  (1) 020324 000420                          BR      74$             ;;GET OVER THE ASCIZ
  (1)                           ;;75$:     .ASCIZ  <200>/THIS DT07 TRAPS ON POWER FAIL!/
  (1) 020366                    74$:
1789        020366              POSTSV=.
1790  020366 012706 001100                   MOV     #STACK,SP       ;RESET STACK
1791  020372 005746                          TST     -(SP)           ;POSITION STACK
1792  020374 012637 000024                   MOV     (SP)+,@#PWRVEC          ;;POP STACK INTO @#PWRVEC
1793  020400 000137 021060                   JMP     MANEND
1794
1795  020404 017737 161142 001126   PWFSER: MOV     @CSR,BAD        ;SAVE CSR
1796  020412 042737 000400 001126           BIC     #400,BAD        ;MASK BRQ IF PRESENT
1797  020420 013737 001702 001124           MOV     KONST,GOOD      ;ASSEMBLE GOOD DATA
1798  020426 062737 060301 001124           ADD     #60301,GOOD
1799  020434 023737 001124 001126           CMP     GOOD,BAD        ;CORRECT @PWR DOWN?
1800  020442 001457                          BEQ     1$              ;BR=YES
1801  020444 104401 020452                   TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 020450 000427                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                           ;;65$:     .ASCIZ  <200>/ERROR15!  CSR INCORRECT @ POWER DOWN.  WAS /
  (1) 020530                    64$:
1802  020530 013746 001126                   MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
```

```
  (1) 020534 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1803 020536 104401 020544                    TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1) 020542 000406                          BR      66$             ;;GET OVER THE ASCIZ
  (1)                                ;;67$:  .ASCIZ  <200>/SHOULD BE /
  (1) 020560                         66$:
1804 020560 013746 001124                    MOV     GOOD,-(SP)      ;;SAVE GOOD FOR TYPEOUT
  (1) 020564 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1805 020566 000000                           HALT
1806 020570 012706 001100          11$:      MOV     #STACK,SP       ;RESET STACK
1807 020574 005746                           TST     -(SP)           ;POSITION STACK
1808 020576 000137 017322                    JMP     ERREN
1809 020602 012737 000632 021442   1$:       MOV     #632,TOCK       ;SET TIMER FOR 5SEC.
1810 020610 004737 021402                    JSR     PC,WAIT ;GO WAIT
1811 020614 017737 160732 001126             MOV     @CSR,BAD        ;SAVE CSR
1812 020622 042737 000400 001126             BIC     #400,BAD        ;MASK BRQ IF PRESENT
1813 020630 162737 060201 001124             SUB     #60201,GOOD     ;CORRECT DATA
1814 020636 023737 001124 001126             CMP     GOOD,BAD        ;CORRECT NOW?
1815 020644 001455                           BEQ     2$              ;BR=YES
1816 020646 104401 020654                    TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1) 020652 000431                          BR      68$             ;;GET OVER THE ASCIZ
  (1)                                ;;69$:  .ASCIZ  <200>/ERROR16! CSR INCORRECT AFTER POWER UP. CSR WAS /
  (1) 020736                         68$:
1817 020736 017746 160610                    MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1) 020742 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1818 020744 104401 020752                    TYPE    ,71$            ;;TYPE ASCIZ STRING
  (1) 020750 000406                          BR      70$             ;;GET OVER THE ASCIZ
  (1)                                ;;71$:  .ASCIZ  <200>/SHOULD BE /
  (1) 020766                         70$:
1819 020766 013746 001124                    MOV     GOOD,-(SP)      ;;SAVE GOOD FOR TYPEOUT
  (1) 020772 104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1820 020774 000000                           HALT
1821 020776 000674                           BR      11$
1822 021000                        2$:
  (1) 021000 104401 021006                   TYPE    ,73$            ;;TYPE ASCIZ STRING
  (1) 021004 000423                          BR      72$             ;;GET OVER THE ASCIZ
  (1)                                ;;73$:  .ASCIZ  <200>/THIS DT07 INTERRUPTS ON POWER FAIL!/
  (1) 021054                         72$:
1823 021054 000137 020366                    JMP     POSTSV
1824
1825 021060                        MANEND:
  (1) 021060 104401 021066                   TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 021064 000420                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                                ;;65$:  .ASCIZ  <200>/MANUAL INTERVENTION TEST DONE./
  (1) 021126                         64$:
1826 021126 000000                           HALT
1827 021130 000776                           BR      .-2
```

```
1829                                    ;       NEWTST  <<INSERT NEXT TEST HERE>>
```

```
 1831                                   .SBTTL  END OF PASS ROUTINE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*INCREMENT THE PASS NUMBER ($PASS)
  (1)                                   ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
  (1)                                   ;*IF THERES A MONITOR GO TO IT
  (1)                                   ;*IF THERE ISN'T JUMP TO RESTRT
  (1)
  (1)  021132                           $EOP:
  (1)  021132  000004                           SCOPE
  (1)  021134  005037  001102                   CLR     $TSTNM          ;;ZERO THE TEST NUMBER
  (1)  021140  005037  001172                   CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
  (1)  021144  005237  001214                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
  (1)  021150  042737  100000  001214          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
  (1)  021156  005327                           DEC     (PC)+           ;;LOOP?
  (1)  021160  000001           $EOPCT: .WORD   1
  (1)  021162  003022                           BGT     $DOAGN          ;;YES
  (1)  021164  012737                           MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
  (1)  021166  000001           $ENDCT: .WORD   1
  (1)  021170  021160                           $EOPCT
  (1)  021172  104401  021237                   TYPE    ,$ENDMG         ;;TYPE "END PASS #"
  (2)  021176  013746  001214                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
  (2)  021202  104405                           TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
  (1)  021204  104401  021234                   TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
  (1)  021210  013700  000042           $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
  (1)  021214  001405                           BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
  (1)  021216  000005                           RESET                   ;;CLEAR THE WORLD
  (1)  021220  004710           $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
  (1)  021222  000240                           NOP                     ;;SAVE ROOM
  (1)  021224  000240                           NOP                     ;;FOR
  (1)  021226  000240                           NOP                     ;;ACT11
  (1)  021230                           $DOAGN:
  (1)  021230  000137                           JMP     @(PC)+          ;;RETURN
  (1)  021232  002444           $RTNAD: .WORD   RESTRT
  (1)  021234     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
  (1)  021237     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
  (1)  021244  050040  051501  020123
  (1)  021252  000043
```

```
                                        .SBTTL USER ROUTINES

021254  012737  000240  021270  CONNEC: MOV     #240,2$         ;SET SWITCH
021262  105777  160264          1$:     TSTB    @CSR            ;ALL READY CONNECTED?
021266  100410                          BMI     3$              ;BR = YES
021270  000774                  2$:     BR      1$              ;CONTINUE
021272  012777  000001  160252          MOV     #1,@CSR         ;CONNECT
021300  012737  000774  021270          MOV     #774,2$         ;RESTORE SWITCH
021306  000765                          BR      1$              ;BR UNTIL CONNECTED
021310  000207                  3$:     RTS     PC              ;EXIT

021312  012737  000004  001660  CONID:  MOV     #4,COVID        ;CONVERT SLAVE ID
021320  013700  001654                  MOV     CURID,R0        ;SAVE CURRANT ID
021324  001405                  1$:     BEQ     2$              ;BR IF DONE
021326  000241                          CLC
021330  006137  001660                  ROL     COVID           ;SHIFT ID
021334  005300                          DEC     R0              ;COUNT DOWN TO ZERO
021336  000772                          BR      1$
021340  000207                  2$:     RTS     PC              ;EXIT

021342  012737  000122  021442  WAIT10: MOV     #122,TOCK               ;SET CLOCK
021350  000137  021402                  JMP     WAIT
021354  012737  000244  021442  WAIT20: MOV     #244,TOCK               ;SET CLOCK
021362  000137  021402                  JMP     WAIT
021366  000000                  TRIG:   000
021370  012737  000051  021442  WAIT05: MOV     #51,TOCK                ;SET CLOCK
021376  000137  021402                  JMP     WAIT
021402  013737  001676  021440  WAIT:   MOV     TIMA,TICK       ;SET BASE TIMER
021410  005777  160136          1$:     TST     @CSR
021414  000400                          BR      .+2
021416  005337  021440                  DEC     TICK
021422  001372                          BNE     1$
021424  005337  021442                  DEC     TOCK
021430  001364                          BNE     WAIT
021432  005237  021442                  INC     TOCK
021436  000207                          RTS     PC
021440  000000                  TICK:   0
021442  000000                  TOCK:   0

021444  005000                  SYNCUP: CLR     R0              ;SET DELAY TIME
021446  012701  000400                  MOV     #400,R1
021452  032777  020000  160072  1$:     BIT     #20000,@CSR             ;BUSS ACTIVE?
021460  001434                          BEQ     2$              ;BR=NO
021462  005200                          INC     R0              ;TIME OUT
021464  001372                          BNE     1$
021466  005301                          DEC     R1
021470  001370                          BNE     1$
021472  104401  021500                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)    021476  000420                  BR      64$             ;;GET OVER THE ASCIZ
 (1)                            ;;65$:  .ASCIZ  <200>/SYNCRONIZATION FAILURE!  CSR= /
 (1)    021540                  64$:
021540  017746  160006                  MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
 (1)    021544  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
021546  000000                          HALT
021550  000776                          BR      .-2
```

```
 1885  021552  005000                    2$:     CLR     R0              ;RESET DELAY TIME
 1886  021554  012701  000000                    MOV     #0,R1
 1887  021560  032777  020000  157764    3$:     BIT     #20000,@CSR     ;BUSS ACTIVE?
 1888  021566  001043                            BNE     4$              ;BR=YES
 1889  021570  005200                            INC     R0              ;TIME OUT
 1890  021572  001372                            BNE     3$
 1891  021574  005301                            DEC     R1
 1892  021576  001370                            BNE     3$
 1893  021600  104401  021606                    TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  021604  000427                            BR      66$             ;;GET OVER THE ASCIZ
  (1)                                   ;;67$:  .ASCIZ  <200>/MASTER PORT DIDN'T TAKE SHARED BUSS!  CSR= /
  (1)  021664                           66$:
 1894  021664  017746  157662                    MOV     @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
  (1)  021670  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1895  021672  000000                            HALT
 1896  021674  000776                            BR      .-2
 1897  021676  000207                    4$:     RTS     PC              ;EXIT
 1898                                           ;CONSTRUCT SEQUENCER
 1899  021700  012701  001574            SEQMAK: MOV     #PORTNO,R1      ;STORE ADDRESS OF DATA
 1900  021704  012702  001662                    MOV     #TEMP1,R2       ;SET TEMPORARY STORAGE POINTER
 1901  021710  012703  001610                    MOV     #SEQ,R3         ;LOAD FIRST SEQUENCER ADDRESS
 1902  021714  005037  001606                    CLR     EXPTCT          ;CLEAR PORT COUNT
 1903  021720  012122                    1$:     MOV     (R1)+,(R2)+     ;FILL TEMP
 1904  021722  005237  001606                    INC     EXPTCT          ;COUNT THEM
 1905  021726  005711                            TST     @R1             ;LAST ONE
 1906  021730  100373                            BPL     1$              ;BR=NO
 1907  021732  012712  177777                    MOV     #-1,@R2         ;TERMINATE DATA
 1908  021736  005001                            CLR     R1              ;LOAD DETECTOR
 1909  021740  012702  001662            2$:     MOV     #TEMP1,R2       ;RESET DATA POINTER
 1910  021744  020112                    3$:     CMP     R1,(R2)         ;ID FOUND?
 1911  021746  001407                            BEQ     5$              ;BR=YES
 1912  021750  005722                    4$:     TST     (R2)+           ;STEP AND TEST FOR LAST
 1913  021752  100374                            BPL     3$              ;BR IF NOT DONE
 1914  021754  005201                            INC     R1              ;STEP DETECTOR
 1915  021756  022701  000004                    CMP     #4,R1           ;CHECK FOR DONE
 1916  021762  001366                            BNE     2$              ;BR IF NOT DONE
 1917  021764  000404                            BR      6$              ;FINISH UP
 1918  021766  011223                    5$:     MOV     @R2,(R3)+       ;LOAD SEQUENCER
 1919  021770  012712  000400                    MOV     #400,@R2        ;ERASE ENTRY
 1920  021774  000765                            BR      4$              ;CONTINUE
 1921  021776  005337  001606            6$:     DEC     EXPTCT          ;CORRECT EXTERNAL PORT COUNT
 1922  022002  012713  177777                    MOV     #-1,@R3         ;TERMINATE SEQUENCER
 1923  022006  000207                            RTS     PC              ;EXIT
 1924                                           ;ROUTINE TO ASSEMBLE MENUS
 1925  022010  012737  177777  001674  MASMEN:  MOV     #-1,ONOFF       ;SET MASTER/SLAVE INDICATOR
 1926  022016  004737  022504                    JSR     PC,FLUSHM       ;GO FLUSH MENU
 1927  022022  012701  001626                    MOV     #MENU,R1        ;SET POINTER
 1928  022026  117721  157572            1$:     MOVB    @NEXENT,(R1)+   ;LOAD MENU WITH SLAVE ID
 1929  022032  112721  000001                    MOVB    #1,(R1)+        ;LOAD FUNCTION #1
 1930  022036  004737  022344                    JSR     PC,STPNEX       ;GET NEXT SLAVE ID
 1931  022042  000771                            BR      1$              ;CONTINUE
 1932  022044  004737  022344            2$:     JSR     PC,STPNEX       ;GO GET NEXT ID
 1933  022050  000405                            BR      3$              ;BR=NOT FINISHED
 1934  022052  012711  177777                    MOV     #-1,@R1         ;TERMINATE MENU
 1935  022056  012704  001626                    MOV     #MENU,R4        ;STORE MENUS
 1936  022062  000207                            RTS     PC              ;EXIT
```

```
1937  022064  117721  157534          3$:     MOVB    @NEXENT,(R1)+
1938  022070  112721  000002                  MOVB    #2,(R1)+        ;LOAD FUNCTION #2
1939  022074  000763                          BR      2$              ;CONTINUE
1940  022076  005037  001674          SALMEN: CLR     ONOFF           ;CLR MASTER/SLAVE INDICATOR
1941  022102  004737  022504                  JSR     PC,FLUSHM       ;GO FLUSH MENU
1942  022106  012701  001626                  MOV     #MENU,R1        ;SET POINTER
1943  022112  027737  157506  001574  1$:     CMP     @NEXENT,PORTNO  ;IS THIS ME?
1944  022120  001010                          BNE     3$              ;BR=NO
1945  022122  117721  157476                  MOVB    @NEXENT,(R1)+   ;LOAD SLAVE ID
1946  022126  112721  000003                  MOVB    #3,(R1)+        ;LOAD FUNCTION 3
1947  022132  004737  022344          2$:     JSR     PC,STPNEX       ;GO GET NEXT ID
1948  022136  000765                          BR      1$              ;BR=NOT DONE
1949  022140  000405                          BR      4$              ;GO TO NEXT SET
1950  022142  117721  157456          3$:     MOVB    @NEXENT,(R1)+
1951  022146  112721  000005                  MOVB    #5,(R1)+        ;LOAD FUNCTION 5
1952  022152  000767                          BR      2$              ;CONTINUE
1953  022154  004737  022344          4$:     JSR     PC,STPNEX       ;GO GET NEXT ID
1954  022160  000405                          BR      5$              ;GO CONTINUE
1955  022162  012711  177777                  MOV     #-1,@R1         ;TERMINATE MENU
1956  022166  012704  001626                  MOV     #MENU,R4        ;STORE MENUS
1957  022172  000207                          RTS     PC              ;EXIT
1958  022174  027737  157424  001574  5$:     CMP     @NEXENT,PORTNO  ;IS THIS ME?
1959  022202  001014                          BNE     7$              ;BR=NO
1960  022204  117721  157414                  MOVB    @NEXENT,(R1)+
1961  022210  112721  000004                  MOVB    #4,(R1)+        ;LOAD FUNCTION 4
1962  022214  004737  022344          6$:     JSR     PC,STPNEX       ;GO GET NEXT ID
1963  022220  000765                          BR      5$              ;CONTINUE
1964  022222  012711  177777                  MOV     #-1,@R1         ;TERMINATE MENU
1965  022226  012704  001626                  MOV     #MENU,R4        ;STORE MENUS
1966  022232  000207                          RTS     PC              ;EXIT
1967  022234  117721  157364          7$:     MOVB    @NEXENT,(R1)+
1968  022240  112721  000005                  MOVB    #5,(R1)+        ;LOAD FUNCTION 5
1969  022244  000763                          BR      6$              ;CONTINUE
1970  022246  012737  001610  001622  SEQINT: MOV     #SEQ,SEQPTR     ;SET POINTER TO FIRST ENTRY
1971  022254  012737  001612  001624          MOV     #SEQ+2,NEXENT   ;GET NEXT ENTRY
1972  022262  017705  157334                  MOV     @SEQPTR,R5      ;LOAD R5
1973  022266  000207                          RTS     PC              ;EXIT
1974  022270  062737  000002  001622  SEQSTP: ADD     #2,SEQPTR       ;STEP SEQUENCER
1975  022276  005777  157320                  TST     @SEQPTR         ;CHECK FOR END
1976  022302  100761                          BMI     SEQINT          ;RELOAD IF DONE
1977  022304  017705  157312                  MOV     @SEQPTR,R5
1978  022310  013737  001622  001624          MOV     SEQPTR,NEXENT   ;STEP NEXT ENTRY
1979  022316  062737  000002  001624          ADD     #2,NEXENT
1980  022324  005777  157274                  TST     @NEXENT         ;OFF THE END?
1981  022330  100401                          BMI     2$              ;BR=YES
1982  022332  000207                  1$:     RTS     PC              ;NO! EXIT
1983  022334  012737  001610  001624  2$:     MOV     #SEQ,NEXENT     ;CORRECT POINTER
1984  022342  000773                          BR      1$              ;EXIT
1985  022344  062737  000002  001624  STPNEX: ADD     #2,NEXENT       ;STEP NEXT ENTRY
1986  022352  005777  157246                  TST     @NEXENT         ;OFF THE END?
1987  022356  100003                          BPL     1$              ;BR=NO
1988  022360  012737  001610  001624          MOV     #SEQ,NEXENT     ;RESET
1989  022366  027705  157232          1$:     CMP     @NEXENT,R5      ;BACK TO CURRENT DT07?
1990  022372  001002                          BNE     2$              ;BR=NO
1991  022374  062716  000002                  ADD     #2,@SP          ;SET UP FINISHED RETURN
1992  022400  000207                  2$:     RTS     PC              ;EXIT
```

```
 1993
 1994
 1995  022402  011637  001126          OOPS:   MOV     @SP,BAD         ;SAVE ERROR PC
 1996  022406  104401  022414                  TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  022412  000427                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                                  ;;65$:  .ASCIZ  <200>/UNEXPECTED INTERUPT FROM DT07 AT ADDRESS.../
  (1)  022472                          64$:
 1997  022472  013746  001126                  MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  022476  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1998  022500  000000                          HALT
 1999  022502  000776                          BR      .-2
 2000
 2001  022504  012700  001626          FLUSHM: MOV     #MENU,R0        ;SET POINTER
 2002  022510  012701  000013                  MOV     #11.,R1         ;SET FLUSH COUNT
 2003  022514  012720  177777          1$:     MOV     #-1,(R0)+       ;FLUSH WITH -1
 2004  022520  005301                          DEC     R1              ;COUNT
 2005  022522  001374                          BNE     1$              ;CONT.
 2006  022524  000207                          RTS     PC              ;EXIT
 2007  022526  012700  001564          FLUSHP: MOV     #DT03,R0        ;LOAD POINTER
 2008  022532  012701  000017                  MOV     #15.,R1         ;SET COUNT
 2009  022536  012720  177777                  MOV     #-1,(R0)+       ;FLUSH
 2010  022542  005020                          CLR     (R0)+
 2011  022544  012720  177777          1$:     MOV     #-1,(R0)+       ;FLUSH WITH -1
 2012  022550  005301                          DEC     R1              ;COUNT
 2013  022552  001374                          BNE     1$              ;CONT
 2014  022554  000207                          RTS     PC              ;EXIT
 2015
 2016
 2017  022556                          DMODE:
 2018  022556  012737  000340  177776  3$:     MOV     #340,PSW        ;RISE PRIORITY
 2019  022564  012777  000001  156760          MOV     #1,@CSR         ;CONNECT
 2020  022572  017737  156754  001126  4$:     MOV     @CSR,BAD        ;SAVE CSR
 2021  022600  032737  000400  001126          BIT     #400,BAD        ;LOOK FOR BRQ
 2022  022606  001010                          BNE     5$              ;BR=FOUND
 2023  022610  032737  100000  001126          BIT     #100000,BAD     ;DID WE TIME OUT?
 2024  022616  001765                          BEQ     4$              ;BR=NO
 2025  022620  012737  177777  001564          MOV     #-1,DT03        ;INDIACATE DT03 MODE
 2026  022626  000531                          BR      8$              ;EXIT
 2027  022630  005037  001564          5$:     CLR     DT03            ;INDIACATE DT07 MODE
 2028  022634  013700  001126                  MOV     BAD,R0          ;SAVE IMAGE
 2029  022640  042700  177703                  BIC     #177703,R0      ;SAVE ONLY RQ0-3
 2030  022644  000241                          CLC                     ;CONVERT RQ BIT
 2031  022646  006000                          ROR     R0
 2032  022650  006000                          ROR     R0
 2033  022652  005002                          CLR     R2
 2034  022654  032700  000001          6$:     BIT     #1,R0           ;BIT SET?
 2035  022660  001051                          BNE     7$              ;BR=YES
 2036  022662  005202                          INC     R2              ;STEP INDIACATOR
 2037  022664  000241                          CLC
 2038  022666  006000                          ROR     R0
 2039  022670  022702  000004                  CMP     #4,R2           ;TOO HIGH?
 2040  022674  101367                          BHI     6$              ;BR =NO
 2041  022676  104401  022704                  TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  022702  000433                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                                  ;;65$:  .ASCIZ  <200>/ERROR!  BRQ DETECTED WITHOUT ANY RQ. BIT SET.  CSR= /
  (1)  022772                          64$:
```

```
 2042  022772  013746  001126                MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  022776  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2043  023000  000000                        HALT                    ;FATAL ERROR BRQ SEEN BUT NO RQ
 2044  023002  000776                        BR      .-2
 2045  023004  010237  001574          7$:   MOV     R2,PORTNO       ;LOAD PORT ID#
 2046  023010  000241                        CLC
 2047  023012  006000                        ROR     R0              ;MAKE SURE ONLY ONE RQ IS PRESENT
 2048  023014  001436                        BEQ     8$              ;BR=OK
 2049  023016  104401  023024                TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  023022  000426                        BR      66$             ;;GET OVER THE ASCIZ
  (1)                             ;;67$:    .ASCIZ  <200>/ERROR!  MORE THAN ONE RQ. BIT SET.  CSR= /
  (1)  023100                     66$:
 2050  023100  013746  001126                MOV     BAD,-(SP)       ;;SAVE BAD FOR TYPEOUT
  (1)  023104  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2051  023106  000000                        HALT
 2052  023110  000776                        BR      .-2
 2053  023112  005077  156434          8$:   CLR     @CSR            ;CLEAR AND DISCONNECT AND EXIT
 2054  023116  000207                        RTS     PC
 2055
 2056
 2057  023120  012737  000340  177776  DTIME:  MOV     #340,PSW        ;LOCK OUT INTERUPTS
 2058  023126  005037  001676                CLR     TIMA            ;CLEAR TIMERS
 2059  023132  005077  156414                CLR     @CSR            ;CLEAR DEVICE
 2060  023136  105777  156410          1$:   TSTB    @CSR            ;MAKE SURE IT'S CLEAR
 2061  023142  100775                        BMI     1$
 2062  023144  005277  156402                INC     @CSR            ;SET REQ
 2063  023150  005777  156376          2$:   TST     @CSR            ;TMO SET?
 2064  023154  100424                        BMI     3$              ;BR=YES
 2065  023156  005237  001676                INC     TIMA            ;COUNT
 2066  023162  001372                        BNE     2$
 2067  023164  104401  023172                TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  023170  000415                        BR      64$             ;;GET OVER THE ASCIZ
  (1)                             ;;65$:    .ASCIZ  <200>/ERROR! TIMING TOO LONG!/
  (1)  023224                     64$:
 2068  023224  000000                        HALT
 2069  023226  005037  177776          3$:   CLR     PSW             ;LOWER PRIORITY
 2070  023232  012777  000001  156312        MOV     #1,@CSR         ;CONNECT
 2071  023240  105777  156306          4$:   TSTB    @CSR            ;CONNECTED?
 2072  023244  100375                        BPL     4$
 2073  023246  005037  001700                CLR     TIMB            ;CLAER WATCHDOG TIMER
 2074  023252  012737  000340  177776        MOV     #340,PSW        ;RISE PRIORITY
 2075  023260  005377  156266                DEC     @CSR            ;DROP REQ
 2076  023264  005777  156262          5$:   TST     @CSR            ;LOOK FOR TMO
 2077  023270  100430                        BMI     6$              ;BR=FOUND
 2078  023272  005237  001700                INC     TIMB
 2079  023276  001372                        BNE     5$
 2080  023300  104401  023306                TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  023304  000421                        BR      66$             ;;GET OVER THE ASCIZ
  (1)                             ;;67$:    .ASCIZ  <200>/ERROR! WATCHDOG TIMING TOO LONG!/
  (1)  023350                     66$:
 2081  023350  000000                        HALT
 2082  023352  013700  001676          6$:   MOV     TIMA,R0         ;COMPARE TIMING FOR PROPER RATIO
 2083  023356  013701  001700                MOV     TIMB,R1
 2084  023362  005002                        CLR     R2
 2085  023364  012703  000031                MOV     #25.,R3
 2086  023370  060002                  7$:   ADD     R0,R2           ;MULTIPLY TIMA BY 25
```

```
 2087  023372  005303                          DEC     R3
 2088  023374  001375                          BNE     7$                ;BR=NOT DONE
 2089  023376  160102                  8$:     SUB     R1,R2             ;DIVIDE TIMA BY TIMB
 2090  023400  005203                          INC     R3
 2091  023402  020102                          CMP     R1,R2             ;DONE?
 2092  023404  101774                          BLOS    8$                ;BR=NO
 2093  023406  020327  000033                  CMP     R3,#33            ;TOO LOW?
 2094  023412  103034                          BHIS    9$                ;BR=NO
 2095  023414  104401  023422                  TYPE    ,69$              ;;TYPE ASCIZ STRING
  (1)  023420  000427                          BR      68$               ;;GET OVER THE ASCIZ
  (1)                               ;;69$:     .ASCIZ  <200>%ERROR! WATCHDOG/REQUEST TIMER RATIO TOO LOW!%
  (1)  023500                       68$:
 2096  023500  000000                          HALT
 2097  023502  000776                          BR      .-2
 2098  023504  020327  000044          9$:     CMP     R3,#44            ;TOO HIGH?
 2099  023510  101435                          BLOS    11$               ;BR=NO
 2100  023512  104401  023520                  TYPE    ,71$              ;;TYPE ASCIZ STRING
  (1)  023516  000430                          BR      70$               ;;GET OVER THE ASCIZ
  (1)                               ;;71$:     .ASCIZ  <200>%ERROR! WATCHDOG/REQUEST TIMER RATIO TOO HIGH!%
  (1)  023600                       70$:
 2101  023600  000000                          HALT
 2102  023602  000776                          BR      .-2
 2103  023604  005077  155742          11$:    CLR     @CSR              ;CLEAR AND DISCONNECT
 2104  023610  000207                          RTS     PC
 2105
 2106
 2107                                          ;PROGRAM PARAMETER INPUT ROUTINE
 2108
 2109  023612  012737  177777  025754  PARIN:  MOV     #-1,PARFLG        ;SET PARFLG
 2110  023620  004737  022526                  JSR     PC,FLUSHP         ;GO FLUSH OUT PARAMETER BLOCK
 2111  023624  104401  023632                  TYPE    ,65$              ;;TYPE ASCIZ STRING
  (1)  023630  000433                          BR      64$               ;;GET OVER THE ASCIZ
  (1)                               ;;65$:     .ASCIZ  <200>/DO YOU WANT TO CHANGE THE DEVICE ADDRESSES?(Y OR N)/
  (1)  023720                       64$:
 2112  023720  104410                          RDCHR
 2113  023722  012600                          MOV     (SP)+,R0          ;STORE CHAR.
 2114  023724  110037  025756                  MOVB    R0,ECHOB
 2115  023730  104401  025756                  TYPE    ,ECHOB
 2116  023734  022700  000131                  CMP     #'Y,R0            ;YES?
 2117  023740  001173                          BNE     10$               ;BR=NO
 2118  023742  104401  023750                  TYPE    ,67$              ;;TYPE ASCIZ STRING
  (1)  023746  000444                          BR      66$               ;;GET OVER THE ASCIZ
  (1)                               ;;67$:     .ASCIZ  <200>/IN THE FOLLOWING THREE PRINTOUTS TYPE THE NEW DATA OR 0 IF NO CHAN
  (1)  024060                       66$:
 2119  024060  104401  024066                  TYPE    ,69$              ;;TYPE ASCIZ STRING
  (1)  024064  000411                          BR      68$               ;;GET OVER THE ASCIZ
  (1)                               ;;69$:     .ASCIZ  <200>/DEVICE ADDRESS= /
  (1)  024110                       68$:
 2120  024110  013746  001552                  MOV     CSR,-(SP)         ;;SAVE CSR FOR TYPEOUT
  (1)  024114  104402                          TYPOC                     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2121  024116  104401  032603                  TYPE    ,SPACE
 2122  024122  104412                          RDOCT
 2123  024124  012600                          MOV     (SP)+,R0          ;STORE INFO
 2124  024126  001402                          BEQ     11$
 2125  024130  010037  001262                  MOV     R0,$BASE
 2126  024134                            11$:
```

```
  (1) 024134 104401 024142          TYPE    ,71$            ;;TYPE ASCIZ STRING
  (1) 024140 000411                 BR      70$             ;;GET OVER THE ASCIZ
  (1)                         ;;71$: .ASCIZ  <200>/DEVICE VECTOR= /
  (1) 024164                  70$:
2127 024164 013700 001256           MOV     $VECT1,R0
2128 024170 042700 177400           BIC     #177400,R0
2129 024174 010046                  MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
  (1) 024176 104402                 TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2130 024200 104401 032603           TYPE    ,SPACE
2131 024204 104412                  RDOCT
2132 024206 012600                  MOV     (SP)+,R0
2133 024210 001402                  BEQ     12$
2134 024212 010001                  MOV     R0,R1           ;SAVE IT TEMPORARILY
2135 024214 000404                  BR      14$             ;JUMP OVER NEXT TWO LINES
2136 024216 013701 001256     12$:  MOV     $VECT1,R1       ;SAVE OLD VECTOR ADDRESS
2137 024222 042701 177400           BIC     #177400,R1      ;MASK OUT JUNK
2138 024226                   14$:
  (1) 024226 104401 024234          TYPE    ,73$            ;;TYPE ASCIZ STRING
  (1) 024232 000410                 BR      72$             ;;GET OVER THE ASCIZ
  (1)                         ;;73$: .ASCIZ  <200>/BREAK LEVEL= /
  (1) 024254                  72$:
2139 024254 013700 001256           MOV     $VECT1,R0
2140 024260 000300                  SWAB    R0
2141 024262 042700 177400           BIC     #177400,R0
2142 024266 010046                  MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
  (1) 024270 104402                 TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2143 024272 104401 032603           TYPE    ,SPACE
2144 024276 104412                  RDOCT
2145 024300 012600                  MOV     (SP)+,R0
2146 024302 001405                  BEQ     13$
2147 024304 000300                  SWAB    R0
2148 024306 060100                  ADD     R1,R0
2149 024310 010037 001256           MOV     R0,$VECT1
2150 024314 000405                  BR      10$
2151 024316 042737 000377 001256 13$: BIC   #377,$VECT1
2152 024324 060137 001256           ADD     R1,$VECT1
2153 024330 000137 001710     10$:  JMP     START
2154       024334                   REPAR=.
2155 024334 004737 022556           JSR     PC,DMODE        ;GO GET MODE AND ID#
2156 024340 004737 023120           JSR     PC,DTIME        ;GO GET DEVICE TIMING
2157 024344 005737 001564           TST     DT03            ;WHAT MODE?
2158 024350 001422                  BEQ     20$             ;BR=DT07
2159 024352 104401 024360           TYPE    ,75$            ;;TYPE ASCIZ STRING
  (1) 024356 000416                 BR      74$             ;;GET OVER THE ASCIZ
  (1)                         ;;75$: .ASCIZ  <200>/THIS DT07 IS IN DT03 MODE./
  (1) 024414                  74$:
2160 024414 000421                  BR      1$              ;CONTINUE AT 1$
2161 024416                   20$:
  (1) 024416 104401 024424          TYPE    ,77$            ;;TYPE ASCIZ STRING
  (1) 024422 000413                 BR      76$             ;;GET OVER THE ASCIZ
  (1)                         ;;77$: .ASCIZ  <200>/THIS DT07 IS PORT# /
  (1) 024452                  76$:
2162 024452 013746 001574           MOV     PORTNO,-(SP)    ;;SAVE PORTNO FOR TYPEOUT
  (1) 024456 104402                 TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2163 024460                   1$:
  (1) 024460 104401 024466          TYPE    ,79$            ;;TYPE ASCIZ STRING
```

```
  (1)  024464  000447                        BR      78$                ;;GET OVER THE ASCIZ
  (1)                                 ;;79$:  .ASCIZ  <200>/TYPE THE ADDRESS OF AN EXTERNAL DEVICE REGISTER ON THE SHARED BUSS
  (1)  024604                         78$:
 2164  024604  104412                        RDOCT
 2165  024606  012637  001566                MOV     (SP)+,DEVAD        ;STORE IN DEVAD
 2166  024612  001523                        BEQ     2$                 ;SKIP OVER NEXT TWO QUESTIONS IF NO DEVICE
 2167  024614  104401  024622                TYPE    ,81$               ;;TYPE ASCIZ STRING
  (1)  024620  000442                        BR      80$                ;;GET OVER THE ASCIZ
  (1)                                 ;;81$:  .ASCIZ  <200> % TYPE AN OCTAL MASK OF READ/WRITE BITS IN ABOVE DEVICE REGISTER..
  (1)  024726                         80$:
 2168  024726  104412                        RDOCT
 2169  024730  012637  001570                MOV     (SP)+,DEVRW        ;STORE BIT PATTERN
 2170  024734  104401  024742                TYPE    ,83$               ;;TYPE ASCIZ STRING
  (1)  024740  000445                        BR      82$                ;;GET OVER THE ASCIZ
  (1)                                 ;;83$:  .ASCIZ  <200> % TYPE AN OCTAL MASK OF RESET-CLEARABLE BITS IN ABOVE DEVICE REGIS
  (1)  025054                         82$:
 2171  025054  104412                        RDOCT
 2172  025056  012637  001572                MOV     (SP)+,DEVRC        ;STORE BIT PATTERN
 2173  025062  005737  001564        2$:     TST     DT03               ;NO NEED TO CONTINUE IF DT03 MODE
 2174  025066  001067                        BNE     7$                 ;BR=DT03
 2175  025070                         3$:4$:
  (1)  025070  104401  025076                TYPE    ,85$               ;;TYPE ASCIZ STRING
  (1)  025074  000427                        BR      84$                ;;GET OVER THE ASCIZ
  (1)                                 ;;85$:  .ASCIZ  <200>/HOW MANY OTHER PORTS ARE THERE TO BE TESTED?/
  (1)  025154                         84$:
 2176  025154  104412                        RDOCT
 2177  025156  012637  001606                MOV     (SP)+,EXPTCT       ;STORE EXTERNAL PORT COUNT
 2178  025162  001431                        BEQ     7$                 ;GO TO 7$
 2179  025164  012700  001576        5$:     MOV     #EXPRT1,R0         ;SET POINTER
 2180  025170  013701  001606                MOV     EXPTCT,R1
 2181  025174                         6$:
  (1)  025174  104401  025202                TYPE    ,87$               ;;TYPE ASCIZ STRING
  (1)  025200  000416                        BR      86$                ;;GET OVER THE ASCIZ
  (1)                                 ;;87$:  .ASCIZ  <200>/TYPE IN EXTERNAL PORT ID../
  (1)  025236                         86$:
 2182  025236  104412                        RDOCT
 2183  025240  012620                        MOV     (SP)+,(R0)+        ;STORE IT
 2184  025242  005301                        DEC     R1                 ;COUNT IT
 2185  025244  001353                        BNE     6$                 ;CONTINUE LOAD IF NOT DONE
 2186  025246                         7$:
  (1)  025246  104401  025254                TYPE    ,89$               ;;TYPE ASCIZ STRING
  (1)  025252  000422                        BR      88$                ;;GET OVER THE ASCIZ
  (1)                                 ;;89$:  .ASCIZ  <200>/SELECT TEST. (0 FOR HELP MESSAGE)/
  (1)  025320                         88$:
 2187  025320  104412                        RDOCT
 2188  025322  012600                        MOV     (SP)+,R0
 2189  025324  001003                        BNE     8$
 2190  025326  104401  032605                TYPE    ,SELMS
 2191  025332  000745                        BR      7$
 2192  025334  022700  000004        8$:     CMP     #4,R0   ;VALID TEST?
 2193  025340  101016                        BHI     9$
 2194  025342  104401  025350                TYPE    ,91$               ;;TYPE ASCIZ STRING
  (1)  025346  000412                        BR      90$                ;;GET OVER THE ASCIZ
  (1)                                 ;;91$:  .ASCIZ  <200>/INVALID SELECTION!/
  (1)  025374                         90$:
 2195  025374  000724                        BR      7$
```

```
2196  025376  022700  000002          9$:     CMP     #2,R0           ;MULTIPORT?
2197  025402  001044                          BNE     21$             ;BR=NO
2198  025404  005737  001606                  TST     EXPTCT          ;ONLY ONE PORT?
2199  025410  001041                          BNE     21$             ;BR=NO
2200  025412  005737  001564                  TST     DT03            ;DT03 MODE?
2201  025416  001036                          BNE     21$             ;BR=YES
2202  025420  104401  025426                  TYPE    ,93$            ;;TYPE ASCIZ STRING
 (1)  025424  000432                          BR      92$             ;;GET OVER THE ASCIZ
 (1)                                  ;;93$:  .ASCIZ  <200>/CAN NOT RUN THE MULTIPORT TEST WITH ONLY ONE PORT!/
 (1)  025512                          92$:
2203  025512  000655                          BR      7$
2204  025514  022700  000002          21$:    CMP     #2,R0           ;MULTIPORT TEST?
2205  025520  001057                          BNE     22$             ;BR=NO
2206  025522  104401  025530                  TYPE    ,95$            ;;TYPE ASCIZ STRING
 (1)  025526  000454                          BR      94$             ;;GET OVER THE ASCIZ
 (1)                                  ;;95$:  .ASCIZ  <200>/THE DT07 WITH THE LOWEST PORT# MUST BE STARTED LAST WHEN RUNNING/<
 (1)  025660                          94$:
2207  025660  000241                  22$:    CLC
2208  025662  006100                          ROL     R0
2209  025664  016016  025762                  MOV     TX(R0),(SP)
2210  025670  104401  025676                  TYPE    ,97$            ;;TYPE ASCIZ STRING
 (1)  025674  000424                          BR      96$             ;;GET OVER THE ASCIZ
 (1)                                  ;;97$:  .ASCIZ  <200>/TYPE <CR> WHEN READY TO CONTINUE TEST./
 (1)  025746                          96$:
2211  025746  104410                          RDCHR
2212  025750  012600                          MOV     (SP)+,R0
2213  025752  000207                          RTS     PC              ;EXIT
2214  025754  000000                  PARFLG: 000                     ;PARAMETERS PRESENT FLAG
2215  025756     000      200      000 ECHOB:  .BYTE   0,200,0         ;ECHO BYTE
2216          025762                          .EVEN
2217  025762  000000                  TX:     0
2218  025764  002454                          ROUT1
2219  025766  002466                          ROUT2
2220  025770  002516                          ROUT4
2221
2222
2223  025772                          TRAP4:
 (2)  025772  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2224  025774  104401  026002                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 (1)  026000  000422                          BR      64$             ;;GET OVER THE ASCIZ
 (1)                                  ;;65$:  .ASCIZ  <200>/UNEXPECTED TRAP TO VECTOR 4 FROM../
 (1)  026046                          64$:
2225  026046  010046                          MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
 (1)  026050  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2226  026052  000000                          HALT
2227  026054  000137  001710                  JMP     START
```

```
                                        .NLIST  MC,MD,CND
                                        .SBTTL  TYPE ROUTINE
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
 (1)                                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
 (1)                                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
 (1)                                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
 (1)                                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
 (1)                                    ;*
 (1)                                    ;*CALL:
 (1)                                    ;*1) USING A TRAP INSTRUCTION
 (1)                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
 (1)                                    ;*OR
 (1)                                    ;*      TYPE
 (1)                                    ;*      MESADR
 (1)                                    ;*
 (1)
 (1)  026060  105737  001157            $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
 (1)  026064  100002                            BPL     1$              ;;BR IF YES
 (1)  026066  000000                            HALT                    ;;HALT HERE IF NO TERMINAL
 (1)  026070  000430                            BR      3$              ;;LEAVE
 (1)  026072  010046                    1$:     MOV     R0,-(SP)        ;;SAVE R0
 (1)  026074  017600  000002                    MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
 (1)  026100  122737  000001  001226            CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 (1)  026106  001011                            BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
 (1)  026110  132737  000100  001227            BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
 (1)  026116  001405                            BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
 (1)  026120  010037  026130                    MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
 (1)  026124  004737  026350                    JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
 (1)  026130  000000                    61$:    .WORD   0               ;;MESSAGE ADDRESS
 (1)  026132  132737  000040  001227    62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
 (1)  026140  001003                            BNE     60$             ;;YES,SKIP TYPE OUT
 (1)  026142  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 (1)  026144  001005                            BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
 (1)  026146  005726                            TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
 (1)  026150  012600                    60$:    MOV     (SP)+,R0        ;;RESTORE R0
 (1)  026152  062716  000002            3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
 (1)  026156  000002                            RTI                     ;;RETURN
 (1)  026160  122716  000011            4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
 (1)  026164  001430                            BEQ     8$
 (1)  026166  122716  000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
 (1)  026172  001006                            BNE     5$
 (1)  026174  005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
 (1)  026176  104401                            TYPE                    ;;TYPE A CR AND LF
 (1)  026200  001203                            $CRLF
 (1)  026202  105037  026336                    CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
 (1)  026206  000755                            BR      2$              ;;GET NEXT CHARACTER
 (1)  026210  004737  026272            5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
 (1)  026214  123726  001156            6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
 (1)  026220  001350                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
 (1)  026222  013746  001154                    MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
 (1)                                            ;;AND THE NULL CHAR.
 (1)  026226  105366  000001            7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
 (1)  026232  002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
 (1)  026234  004737  026272                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
```

```
   (1)  026240  105337  026336                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
   (1)  026244  000770                        BR      7$              ;;LOOP
   (1)
   (1)                                ;HORIZONTAL TAB PROCESSOR
   (1)
   (1)  026246  112716  000040        8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
   (1)  026252  004737  026272        9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
   (1)  026256  132737  000007  026336        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
   (1)  026264  001372                        BNE     9$              ;;TAB STOP
   (1)  026266  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
   (1)  026270  000724                        BR      2$              ;;GET NEXT CHARACTER
   (1)  026272  105777  152652        $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
   (1)  026276  100375                        BPL     $TYPEC
   (1)  026300  116677  000002  152644        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
   (1)  026306  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
   (1)  026314  001003                        BNE     1$              ;;BRANCH IF NO
   (1)  026316  105037  026336                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
   (1)  026322  000406                        BR      $TYPEX          ;;EXIT
   (1)  026324  122766  000012  000002  1$:    CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
   (1)  026332  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
   (1)  026334  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
   (1)  026336  000000        $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
   (1)  026340  000207        $TYPEX: RTS     PC
   (1)
  2231                                .SBTTL  APT COMMUNICATIONS ROUTINE
   (1)
   (2)                                ;;**************************************************************
   (1)  026342  112737  000001  026606  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
   (1)  026350  112737  000001  026604  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
   (1)  026356  000403                        BR      $ATYC
   (1)  026360  112737  000001  026606  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
   (1)  026366                        $ATYC:
   (3)  026366  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
   (3)  026370  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
   (1)  026372  105737  026604                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
   (1)  026376  001450                        BEQ     5$              ;;IF NOT:  BR
   (1)  026400  122737  000001  001226        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
   (1)  026406  001031                        BNE     3$              ;;IF NOT:  BR
   (1)  026410  132737  000100  001227        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
   (1)  026416  001425                        BEQ     3$              ;;IF NOT:  BR
   (1)  026420  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
   (1)  026424  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
   (1)  026432  005737  001206        1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
   (1)  026436  001375                        BNE     1$              ;;IF NOT:  WAIT
   (1)  026440  010037  001222                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
   (1)  026444  105720                2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
   (1)  026446  001376                        BNE     2$
   (1)  026450  163700  001222                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
   (1)  026454  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
   (1)  026456  010037  001224                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
   (1)  026462  012737  000004  001206        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
   (1)  026470  000413                        BR      5$
   (1)  026472  017637  000004  026516  3$:    MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
   (1)  026500  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
   (3)  026506  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
   (1)  026512  004737  026060                JSR     PC,$TYPE        ;;CALL TYPE MACRO
```

```
(1)  026516  000000                    4$:     .WORD   0
(1)  026520                            5$:
(1)  026520  105737  026606           10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  026524  001416                            BEQ     12$             ;;IF NOT:  BR
(1)  026526  005737  001226                    TST     $ENV            ;;RUNNING UNDER APT?
(1)  026532  001413                            BEQ     12$             ;;IF NOT:  BR
(1)  026534  005737  001206           11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  026540  001375                            BNE     11$             ;;IF NOT:  WAIT
(1)  026542  017637  000004  001210           MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  026550  062766  000002  000004           ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  026556  005237  001206                    INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  026562  105037  026606           12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  026566  105037  026605                    CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  026572  105037  026604                    CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  026576  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  026600  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  026602  000207                            RTS     PC              ;;RETURN
(1)  026604     000           $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  026605     000           $LFLG:  .BYTE   0               ;;LOG FLAG
(1)  026606     000           $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          026610                    .EVEN
(1)          000200           APTSIZE=200
(1)          000001           APTENV=001
(1)          000100           APTSPOOL=100
(1)          000040           APTCSUP=040
2232                           .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)                            ;;******************************************************************
(1)                            ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)                            ;*CHANGE IT TO BINARY.
(1)                            ;*CALL:
(1)                            ;*      RDOCT                   ;;READ AN OCTAL NUMBER
(1)                            ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1)                            ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
(1)
(1)  026610  011646           $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1)  026612  016666  000004  000002    MOV     4(SP),2(SP)     ;;INPUT NUMBER
(3)  026620  010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  026622  010146                    MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  026624  010246                    MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(1)  026626  104411           1$:     RDLIN                   ;;READ AN ASCIZ LINE
(1)  026630  012600                    MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
(1)  026632  005001                    CLR     R1              ;;CLEAR DATA WORD
(1)  026634  005002                    CLR     R2
(1)  026636  112046           2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
(1)  026640  001412                    BEQ     3$              ;;IF ZERO GET OUT
(1)  026642  006301                    ASL     R1              ;;*2
(1)  026644  006102                    ROL     R2
(1)  026646  006301                    ASL     R1              ;;*4
(1)  026650  006102                    ROL     R2
(1)  026652  006301                    ASL     R1              ;;*8
(1)  026654  006102                    ROL     R2
(1)  026656  042716  177770            BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
(1)  026662  062601                    ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
(1)  026664  000764                    BR      2$              ;;LOOP
(1)  026666  005726           3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
```

```
(1)  026670  010166  000012                    MOV     R1,12(SP)       ;;SAVE THE RESULT
(1)  026674  010237  026710                    MOV     R2,$HIOCT
(3)  026700  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  026702  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  026704  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  026706  000002                            RTI                     ;;RETURN
(1)  026710  000000                $HIOCT: .WORD   0                   ;;HIGH ORDER BITS GO HERE
2233                                           .SBTTL  TTY INPUT ROUTINE
(1)
(2)                                            ;;**********************************************************************
(1)                                            .ENABL  LSB
(1)
(2)                                            ;;**********************************************************************
(1)                                            ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)                                            ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1)                                            ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1)                                            ;*WHEN OPERATING IN TTY FLAG MODE.
(1)  026712  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
(1)  026720  001074                            BNE     15$             ;;BRANCH IF NO
(1)  026722  105777  152216                    TSTB    @$TKS           ;;CHAR THERE?
(1)  026726  100071                            BPL     15$             ;;IF NO, DON'T WAIT AROUND
(1)  026730  117746  152212                    MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
(1)  026734  042716  177600                    BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
(1)  026740  022726  000007                    CMP     #7,(SP)+        ;;IS IT A CONTROL G?
(1)  026744  001062                            BNE     15$             ;;NO, RETURN TO USER
(1)  026746  123727  001134  000001            CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  026754  001456                            BEQ     15$             ;;BRANCH IF YES
(1)
(1)  026756  104401  027437                    TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  026762  104401  027444        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  026766  013746  000176                    MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  026772  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  026774  104401  027455                    TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  027000  005046                19$:     CLR     -(SP)           ;;CLEAR COUNTER
(1)  027002  005046                        CLR     -(SP)           ;;THE NEW SWR
(1)  027004  105777  152134        7$:     TSTB    @$TKS           ;;CHAR THERE?
(1)  027010  100375                        BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  027012  117746  152130                    MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  027016  042716  177600                    BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1)  027022  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  027026  001005                        BNE     10$             ;;BRANCH IF NOT
(1)  027030  104401  027432                    TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  027034  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  027040  000757                        BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  027042  021627  000015        10$:    CMP     (SP),#15        ;;IS IT A <CR>?
(1)  027046  001022                        BNE     16$             ;;BRANCH IF NO
(1)  027050  005766  000004                    TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  027054  001403                        BEQ     11$             ;;BRANCH IF YES
(1)  027056  016677  000002  152054            MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  027064  062706  000006        11$:    ADD     #6,SP           ;;CLEAR UP STACK
```

```
.MAIN.  MACY11 30A(1052)  29-JAN-79  13:38  PAGE 13-15
CRDTAA.P11    29-JAN-79 13:32          TTY INPUT ROUTINE                                SEQ 0073

    (1)  027070  104401  001203          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
    (1)  027074  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
    (1)  027102  001003                          BNE     15$             ;;BRANCH IF NOT
    (1)  027104  012777  000100  152032          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
    (1)  027112  000002                  15$:    RTI                     ;;RETURN
    (1)  027114  004737  026272          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
    (1)  027120  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
    (1)  027124  002420                          BLT     18$             ;;BRANCH IF YES
    (1)  027126  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
    (1)  027132  003015                          BGT     18$             ;;BRANCH IF YES
    (1)  027134  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
    (1)  027140  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
    (1)  027144  001403                          BEQ     17$             ;;BRANCH IF YES
    (1)  027146  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
    (1)  027150  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
    (1)  027152  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
    (1)  027154  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
    (1)  027160  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
    (1)  027164  000707                          BR      7$              ;;GET THE NEXT ONE
    (1)  027166  104401  001202          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
    (1)  027172  000720                          BR      20$             ;;SIMULATE CONTROL-U
    (1)                               .DSABL  LSB
    (1)
    (1)
    (2)                               ;;*****************************************************************
    (1)                               ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
    (1)                               ;*CALL:
    (1)                               ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
    (1)                               ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
    (1)                               ;*                              ;;WITH PARITY BIT STRIPPED OFF
    (1)                               ;
    (1)
    (1)  027174  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
    (1)  027176  016666  000004  000002  MOV     4(SP),2(SP)     ;;SAVE THE PS
    (1)  027204  105777  151734          1$:     TSTB    @$TKS           ;;WAIT FOR
    (1)  027210  100375                          BPL     1$              ;;A CHARACTER
    (1)  027212  117766  151730  000004  MOVB    @$TKB,4(SP)     ;;READ THE TTY
    (1)  027220  042766  177600  000004  BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
    (1)  027226  026627  000004  000023  CMP     4(SP),#23       ;;IS IT A CONTROL-S?
    (1)  027234  001013                  BNE     3$              ;;BRANCH IF NO
    (1)  027236  105777  151702          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
    (1)  027242  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
    (1)  027244  117746  151676                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
    (1)  027250  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
    (1)  027254  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
    (1)  027260  001366                          BNE     2$              ;;IF NOT DISCARD IT
    (1)  027262  000750                          BR      1$              ;;YES, RESUME
    (1)  027264  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
    (1)  027272  002407                          BLT     4$              ;;BRANCH IF YES
    (1)  027274  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
    (1)  027302  003003                          BGT     4$              ;;BRANCH IF YES
    (1)  027304  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
    (1)  027312  000002                  4$:     RTI                     ;;GO BACK TO USER
    (2)                               ;;*****************************************************************
    (1)                               ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
    (1)                               ;*CALL:
```

```
(1)                                     ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                     ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  027314  010346                     $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  027316  012703  027422             1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  027322  022703  027432             2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
(1)  027326  101405                             BLOS    4$              ;;BR IF YES
(1)  027330  104410                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  027332  112613                             MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  027334  122713  000177             10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  027340  001003                             BNE     3$              ;;SKIP IF NOT
(1)  027342  104401  001202             4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  027346  000763                             BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  027350  111337  027420             3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  027354  104401  027420                     TYPE    ,9$
(1)  027360  122723  000015                     CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  027364  001356                             BNE     2$              ;;LOOP IF NOT RETURN
(1)  027366  105063  177777                     CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  027372  104401  001204                     TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  027376  012603                             MOV     (SP)+,R3        ;;RESTORE R3
(1)  027400  011646                             MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  027402  016666  000004  000002             MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  027410  012766  027422  000004             MOV     #$TTYIN,4(SP)
(1)  027416  000002                             RTI                     ;;RETURN
(1)  027420     000                     9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  027421     000                             .BYTE   0               ;;TERMINATOR
(1)  027422  000010                     $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
(1)  027432  052536  005015     000     $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
(1)  027437     136  006507  000012     $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
(1)  027444  005015  053523  020122     $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  027452  020075     000
(1)  027455     040  047040  053505     $MNEW:  .ASCIZ  /  NEW = /
(1)  027462  036440  000040
2234                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)                                     ;;******************************************************************
(1)                                     ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                                     ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                                     ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPOS                   ;;CALL FOR TYPEOUT
(1)                                     ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                                     ;*      .BYTE   M               ;;M=1 OR 0
(1)                                     ;*                                      ;;1=TYPE LEADING ZEROS
(1)                                     ;*                                      ;;0=SUPPRESS LEADING ZEROS
(1)                                     ;*
(1)                                     ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                                     ;*$TYPOS OR $TYPOC
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPON                   ;;CALL FOR TYPEOUT
(1)                                     ;*
(1)                                     ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)                                     ;*CALL:
```

```
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)
(1)  027466  017646  000000            $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  027472  116637  000001  027711            MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  027500  112637  027713                    MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  027504  062716  000002                    ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  027510  000406                            BR      $TYPON
(1)  027512  112737  000001  027711  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  027520  112737  000006  027713            MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  027526  112737  000005  027710  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  027534  010346                            MOV     R3,-(SP)        ;;SAVE R3
(1)  027536  010446                            MOV     R4,-(SP)        ;;SAVE R4
(1)  027540  010546                            MOV     R5,-(SP)        ;;SAVE R5
(1)  027542  113704  027713                    MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  027546  005404                            NEG     R4
(1)  027550  062704  000006                    ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  027554  110437  027712                    MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  027560  113704  027711                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  027564  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  027570  005003                            CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  027572  006105                    1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)  027574  000404                            BR      3$              ;;GO DO MSB
(1)  027576  006105                    2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  027600  006105                            ROL     R5
(1)  027602  006105                            ROL     R5
(1)  027604  010503                            MOV     R5,R3
(1)  027606  006103                    3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  027610  105337  027712                    DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  027614  100016                            BPL     7$              ;;BR IF NO
(1)  027616  042703  177770                    BIC     #177770,R3      ;;GET RID OF JUNK
(1)  027622  001002                            BNE     4$              ;;TEST FOR 0
(1)  027624  005704                            TST     R4              ;;SUPPRESS THIS 0?
(1)  027626  001403                            BEQ     5$              ;;BR IF YES
(1)  027630  005204                    4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  027632  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  027636  052703  000040            5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  027642  110337  027706                    MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  027646  104401  027706                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  027652  105337  027710            7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  027656  003347                            BGT     2$              ;;BR IF MORE TO DO
(1)  027660  002402                            BLT     6$              ;;BR IF DONE
(1)  027662  005204                            INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  027664  000744                            BR      2$              ;;GO DO THE LAST DIGIT
(1)  027666  012605                    6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  027670  012604                            MOV     (SP)+,R4        ;;RESTORE R4
(1)  027672  012603                            MOV     (SP)+,R3        ;;RESTORE R3
(1)  027674  016666  000002  000004            MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  027702  012616                            MOV     (SP)+,(SP)
(1)  027704  000002                            RTI                     ;;RETURN
(1)  027706     000                    8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  027707     000                            .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  027710     000                    $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  027711     000                    $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  027712  000000                    $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
2235                                    .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
```

```
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)                                     ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)                                     ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)                                     ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)                                     ;*REPLACED WITH SPACES.
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
(1)                                     ;*      TYPDS                   ;;GO TO THE ROUTINE
(1)
(1)  027714                             $TYPDS:
(3)  027714  010046                             MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  027716  010146                             MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  027720  010246                             MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  027722  010346                             MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  027724  010546                             MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  027726  012746  020200                     MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  027732  016605  000020                     MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  027736  100004                             BPL     1$              ;;BR IF INPUT IS POS.
(1)  027740  005405                             NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  027742  112766  000055  000001             MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  027750  005000                     1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  027752  012703  030130                     MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  027756  112723  000040                     MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  027762  005002                     2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  027764  016001  030120                     MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  027770  160105                     3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
(1)  027772  002402                             BLT     4$              ;;BR IF DONE
(1)  027774  005202                             INC     R2              ;;INCREASE THE BCD DIGIT BY 1
(1)  027776  000774                             BR      3$
(1)  030000  060105                     4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  030002  005702                             TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  030004  001002                             BNE     5$              ;;FALL THROUGH IF 0
(1)  030006  105716                             TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  030010  100407                             BMI     7$              ;;BR IF YES
(1)  030012  106316                     5$:     ASLB    (SP)            ;;MSD?
(1)  030014  103003                             BCC     6$              ;;BR IF NO
(1)  030016  116663  000001  177777             MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  030024  052702  000060             6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  030030  052702  000040             7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  030034  110223                             MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  030036  005720                             TST     (R0)+           ;;JUST INCREMENTING
(1)  030040  020027  000010                     CMP     R0,#10          ;;CHECK THE TABLE INDEX
(1)  030044  002746                             BLT     2$              ;;GO DO THE NEXT DIGIT
(1)  030046  003002                             BGT     8$              ;;GO TO EXIT
(1)  030050  010502                             MOV     R5,R2           ;;GET THE LSD
(1)  030052  000764                             BR      6$              ;;GO CHANGE TO ASCII
(1)  030054  105726                     8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  030056  100003                             BPL     9$              ;;BR IF NO
(1)  030060  116663  177777  177776             MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  030066  105013                     9$:     CLRB    (R3)            ;;SET THE TERMINATOR
(3)  030070  012605                             MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  030072  012603                             MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  030074  012602                             MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  030076  012601                             MOV     (SP)+,R1        ;;POP STACK INTO R1
```

```
(3)  030100  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  030102  104401  030130                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  030106  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  030114  012616                        MOV     (SP)+,(SP)
(1)  030116  000002                        RTI                     ;;RETURN TO USER
(1)  030120  023420                $DTBL:  10000.
(1)  030122  001750                        1000.
(1)  030124  000144                        100.
(1)  030126  000012                        10.
(1)  030130  000004                $DBLK:  .BLKW   4
2236                               .SBTTL  ERROR HANDLER ROUTINE
(1)
(2)                                ;;**********************************************************************
(1)                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)                                ;*AND GO TO $ERRTYP ON ERROR
(1)                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                ;*SW15=1        HALT ON ERROR
(1)                                ;*SW13=1        INHIBIT ERROR TYPEOUTS
(1)                                ;*SW10=1        BELL ON ERROR
(1)                                ;*SW09=1        LOOP ON ERROR
(1)                                ;*CALL
(1)                                ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1)  030140                        $ERROR:
(1)  030140  104407                        CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(2)  030142  113737  001102  001562        MOVB    $TSTNM,TSTNUM   ;SET UP TEST # ON ERROR
(1)  030150  105237  001103        7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
(1)  030154  001775                        BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
(1)  030156  013777  001102  150756        MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  030164  032777  002000  150746        BIT     #BIT10,@SWR     ;;BELL ON ERROR?
(1)  030172  001402                        BEQ     1$              ;;NO - SKIP
(1)  030174  104401  001176                TYPE    ,$BELL          ;;RING BELL
(1)  030200  005237  001112        1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
(1)  030204  011637  001116                MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  030210  162737  000002  001116        SUB     #2,$ERRPC
(1)  030216  117737  150674  001114        MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  030224  032777  020000  150706        BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
(1)  030232  001055                        BNE     20$             ;;SKIP TYPEOUTS
(1)  030234  021627  001002                CMP     (SP),#1002      ;;IF RETURN PC LESS THAN 1002
(1)  030240  101046                        BHI     12$             ;;ERROR IS ILLEGAL TRAP
(1)                                ;;PROCESS UNEXPECTED TRAP OR INTERRUPT
(1)  030242  016637  000004  001116        MOV     4(SP),$ERRPC    ;;GET PC AT TIME OF FALSE TRAP
(1)  030250  162737  000002  001116        SUB     #2,$ERRPC       ;;ADJUST PC
(1)  030256  104401  030322                TYPE    ,10$            ;;TYPE HEADER
(2)  030262  013746  001116                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)  030266  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  030270  104401  030330                TYPE    ,11$
(1)  030274  162716  000004                SUB     #4,(SP)         ;;GET FALSE TRAP VECTOR ADDR
(1)  030300  011637  001116                MOV     (SP),$ERRPC
(2)  030304  013746  001116                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)  030310  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  030312  104401  001203                TYPE    ,$CRLF
(1)  030316  022626                        CMP     (SP)+,(SP)+     ;;POP FALSE TRAP VECTOR PC&ADDR
(1)  030320  000422                        BR      20$
(1)  030322  050200  036503  000040  10$:  .ASCIZ  <200>'PC= '
```

```
(1)  030330 020040 047125 054105 11$:    .ASCIZ  '  UNEXPECTED TRAP 10 '
(1)  030336 042520 052103 042105
(1)  030344 052040 040522 020120
(1)  030352 047524 000040
(1)                                      .EVEN
(1)  030356                      12$:
(1)  030356 004737 030470                JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(1)  030362 104401 001203                TYPE    ,$CRLF
(1)  030366                      20$:
(1)  030366 122737 000001 001226         CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  030374 001007                       BNE     2$              ;;NO,SKIP APT ERROR REPORT
(1)  030376 113737 001114 030410         MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1)  030404 004737 026360                JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
(1)  030410    000               21$:    .BYTE   0
(1)  030411    000                       .BYTE   0
(1)  030412 000777               22$:    BR      22$             ;;APT ERROR LOOP
(1)  030414 005777 150520        2$:     TST     @SWR            ;;HALT ON ERROR
(1)  030420 100002                       BPL     3$              ;;SKIP IF CONTINUE
(1)  030422 000000                       HALT                    ;;HALT ON ERROR!
(1)  030424 104407                       CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(1)  030426 032777 001000 150504 3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
(1)  030434 001402                       BEQ     4$              ;;BR IF NO
(1)  030436 013716 001110                MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
(1)  030442 005737 001174        4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
(1)  030446 001402                       BEQ     5$              ;;BR IF NONE
(1)  030450 013716 001174                MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  030454                      5$:
(1)  030454 022737 021220 000042         CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
(1)  030462 001001                       BNE     6$              ;;BRANCH IF NO
(1)  030464 000000                       HALT                    ;;YES
(1)  030466                      6$:
(1)  030466 000002                       RTI                     ;;RETURN
2237                             .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                              ;;**************************************************************
(1)                              ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1)                              ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1)                              ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  030470                      $ERRTYP:
(1)  030470 104401 001203                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  030474 010046                       MOV     R0,-(SP)        ;;SAVE R0
(1)  030476 005000                       CLR     R0              ;;PICKUP THE ITEM INDEX
(1)  030500 153700 001114                BISB    @#$ITEMB,R0
(1)  030504 001004                       BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                              ;;TYPE THE PC OF THE ERROR
(2)  030506 013746 001116                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                              ;;ERROR ADDRESS
(2)  030512 104402                       TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  030514 000426                       BR      6$              ;;GET OUT
(1)  030516 005300               1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1)  030520 006300                       ASL     R0              ;;      WORK FOR THE ERROR TABLE
(1)  030522 006300                       ASL     R0
(1)  030524 006300                       ASL     R0
(1)  030526 062700 001332                ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(1)  030532 012037 030542                MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
```

```
(1)  030536  001404                        BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1)  030540  104401                        TYPE                    ;;TYPE THE "ERROR MESSAGE"
(1)  030542  000000                 2$:    .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
(1)  030544  104401  001203               TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  030550  012037  030560         3$:    MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
(1)  030554  001404                        BEQ     5$              ;;SKIP TYPEOUT IF 0
(1)  030556  104401                        TYPE                    ;;TYPE THE "DATA HEADER"
(1)  030560  000000                 4$:    .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
(1)  030562  104401  001203               TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  030566  011000                 5$:    MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
(1)  030570  001004                        BNE     7$              ;;GO TYPE THE DATA
(1)  030572  012600                 6$:    MOV     (SP)+,R0        ;;RESTORE R0
(1)  030574  104401  001203               TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  030600  000207                        RTS     PC              ;;RETURN
(1)  030602                         7$:
(2)  030602  013046                        MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
(2)  030604  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  030606  005710                        TST     (R0)            ;;IS THERE ANOTHER NUMBER?
(1)  030610  001770                        BEQ     6$              ;;BR IF NO
(1)  030612  104401  030620               TYPE    ,8$             ;;TYPE TWO(2) SPACES
(1)  030616  000771                        BR      7$              ;;LOOP
(1)  030620  020040  000            8$:    .ASCIZ  /  /            ;;TWO(2) SPACES
(1)          030624                        .EVEN
2238                                .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)                                 ;;**********************************************************************
(1)                                 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)                                 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)                                 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                 ;*SW14=1        LOOP ON TEST
(1)                                 ;*SW11=1        INHIBIT ITERATIONS
(1)                                 ;*SW09=1        LOOP ON ERROR
(1)                                 ;*SW08=1        LOOP ON TEST IN SWR<7:0>
(1)                                 ;*CALL
(1)                                 ;*      SCOPE           ;;SCOPE=IOT
(1)
(1)  030624                         $SCOPE:
(1)  030624  104407                        CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
(1)                                 ;;GO TO ERROR ROUTINE IF RETURN PC LESS THAN 1002
(1)                                 ;;OTHERWISE CONTINUE
(1)  030626  021627  001002                CMP     (SP),#1002      ;;UNEXPECTED TRAP OR INTERRRUPT
(1)  030632  101002                        BHI     1$              ;;ARE TRAPPED HERE VIA IOT
(1)  030634  000137  030140                JMP     $ERROR          ;;GO PROCESS UNEXPECTED TRAP
(1)  030640  032777  040000  150272  1$:    BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
(1)  030646  001114                        BNE     $OVER           ;;YES IF SW14=1
(1)                                 ;#####START OF CODE FOR THE XOR TESTER#####
(1)  030650  000416                 $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
(1)                                                                 ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
(1)  030652  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  030656  012737  030676  000004        MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
(1)  030664  005737  177060                TST     @#177060        ;;TIME OUT ON XOR?
(1)  030670  012637  000004                MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  030674  000463                        BR      $SVLAD          ;;GO TO THE NEXT TEST
(1)  030676  022626                 5$:    CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
(1)  030700  012637  000004                MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
```

```
(1)  030704  000423                        BR      7$              ;;LOOP ON THE PRESENT TEST
(1)  030706                          6$:;#####END OF CODE FOR THE XOR TESTER#####
(1)  030706  032777  000400  150224        BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
(1)  030714  001404                        BEQ     2$              ;;BR IF NO
(1)  030716  127737  150216  001102        CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
(1)  030724  001465                        BEQ     $OVER           ;;BR IF YES
(1)  030726  105737  001103          2$:   TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
(1)  030732  001421                        BEQ     3$              ;;BR IF NO
(1)  030734  123737  001115  001103        CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  030742  101015                        BHI     3$              ;;BR IF NO
(1)  030744  032777  001000  150166        BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
(1)  030752  001404                        BEQ     4$              ;;BR IF NO
(1)  030754  013737  001110  001106  7$:   MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  030762  000446                        BR      $OVER
(1)  030764  105037  001103          4$:   CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
(1)  030770  005037  001172                CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  030774  000415                        BR      1$              ;;ESCAPE TO THE NEXT TEST
(1)  030776  032777  004000  150134  3$:   BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
(1)  031004  001011                        BNE     1$              ;;BR IF YES
(1)  031006  005737  001214                TST     $PASS           ;;IF FIRST PASS OF PROGRAM
(1)  031012  001406                        BEQ     1$              ;;      INHIBIT ITERATIONS
(1)  031014  005237  001104                INC     $ICNT           ;;INCREMENT ITERATION COUNT
(1)  031020  023737  001172  001104        CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)  031026  002024                        BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
(1)  031030  012737  000001  001104  1$:   MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
(1)  031036  013737  031114  001172        MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
(1)  031044  105237  001102          $SVLAD: INCB  $TSTNM          ;;COUNT TEST NUMBERS
(1)  031050  113737  001102  001212        MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1)  031056  011637  001106                MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1)  031062  011637  001110                MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  031066  005037  001174                CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  031072  112737  000001  001115        MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  031100  013777  001102  150034  $OVER: MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  031106  013716  001106                MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(1)  031112  000002                        RTI                     ;;FIXES PS
(1)  031114  003720                  $MXCNT: 2000.                 ;;MAX. NUMBER OF ITERATIONS
2239                                 .SBTTL  POWER DOWN AND UP ROUTINES
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;POWER DOWN ROUTINE
(1)  031116  012737  031262  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1)  031124  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
(3)  031132  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  031134  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  031136  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  031140  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  031142  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
(3)  031144  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(3)  031146  017746  147766                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
(1)  031152  010637  031266                  MOV     SP,$SAVR6       ;;SAVE SP
(1)  031156  012737  031170  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
(1)  031164  000000                          HALT
(1)  031166  000776                          BR      .-2             ;;HANG UP
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;POWER UP ROUTINE
```

```
   (1)  031170  012737  031262  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
   (1)  031176  013706  031266                  MOV     $SAVR6,SP        ;;GET SP
   (1)  031202  005037  031266                  CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
   (1)  031206  005237  031266          1$:     INC     $SAVR6           ;;WAIT FOR THE INC
   (1)  031212  001375                          BNE     1$               ;;OF  WORD
   (3)  031214  012677  147720                  MOV     (SP)+,@SWR       ;;POP STACK INTO @SWR
   (3)  031220  012605                          MOV     (SP)+,R5         ;;POP STACK INTO R5
   (3)  031222  012604                          MOV     (SP)+,R4         ;;POP STACK INTO R4
   (3)  031224  012603                          MOV     (SP)+,R3         ;;POP STACK INTO R3
   (3)  031226  012602                          MOV     (SP)+,R2         ;;POP STACK INTO R2
   (3)  031230  012601                          MOV     (SP)+,R1         ;;POP STACK INTO R1
   (3)  031232  012600                          MOV     (SP)+,R0         ;;POP STACK INTO R0
   (1)  031234  012737  031116  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
   (1)  031242  012737  000340  000026          MOV     #340,@#PWRVEC+2  ;;PRIO:7
   (1)  031250  104401                          TYPE                     ;;REPORT THE POWER FAILURE
   (1)  031252  031270                  $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
   (1)  031254  012716                          MOV     (PC)+,(SP)       ;;RESTART AT RESTRT
   (1)  031256  002444                  $PWRAD: .WORD   RESTRT           ;;RESTART ADDRESS
   (1)  031260  000002                          RTI
   (1)  031262  000000                  $ILLUP: HALT                     ;;THE POWER UP SEQUENCE WAS STARTED
   (1)  031264  000776                          BR      .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
   (1)  031266  000000                  $SAVR6: 0                        ;;PUT THE SP HERE
  2240  031270  005015  042522  052123  PWRMSG: .ASCIZ  <15><12>/RESTARTED FROM POWER FAIL/
        031276  051101  042524  020104
        031304  051106  046517  050040
        031312  053517  051105  043040
        031320  044501  000114
  2241                                  .EVEN
  2242                                  .SBTTL  TRAP DECODER
   (1)
   (2)                                  ;;******************************************************************
   (1)                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
   (1)                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
   (1)                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
   (1)                                  ;*GO TO THAT ROUTINE.
   (1)
   (1)  031324  010046                  $TRAP:  MOV     R0,-(SP)         ;;SAVE R0
   (1)  031326  016600  000002                  MOV     2(SP),R0         ;;GET TRAP ADDRESS
   (1)  031332  005740                          TST     -(R0)            ;;BACKUP BY 2
   (1)  031334  111000                          MOVB    (R0),R0          ;;GET RIGHT BYTE OF TRAP
   (1)  031336  006300                          ASL     R0               ;;POSITION FOR INDEXING
   (1)  031340  016000  031360                  MOV     $TRPAD(R0),R0    ;;INDEX TO TABLE
   (1)  031344  000200                          RTS     R0               ;;GO TO ROUTINE
   (1)
   (1)
   (1)                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
   (1)
   (1)  031346  011646                  $TRAP2: MOV     (SP),-(SP)       ;;MOVE THE PC DOWN
   (1)  031350  016666  000004  000002          MOV     4(SP),2(SP)      ;;MOVE THE PSW DOWN
   (1)  031356  000002                          RTI                      ;;RESTORE THE PSW
   (1)
   (3)                                  .SBTTL  TRAP TABLE
   (3)
   (3)                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
   (3)                                  ;*BY THE "TRAP" INSTRUCTION.
   (3)
```

```
                                        ;       ROUTINE
                                        ;       -------
(3)  031360  031346                     $TRPAD: .WORD   $TRAP2
(3)  031362  026060                             $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3)  031364  027512                             $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)  031366  027466                             $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)  031370  027526                             $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)  031372  027714                             $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
(1)
(3)  031374  026762                             $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
(1)
(3)  031376  026712                             $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
(3)  031400  027174                             $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3)  031402  027314                             $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3)  031404  026610                             $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
2243                                    .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
(1)
(2)                                     ;;**************************************************************
(1)                                     ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
(1)                                     ;*WITH A RANGE OF 0 TO 2(+33)-1.
(1)                                     ;*CALL:
(1)                                     ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
(1)                                     ;*      RETURN                  ;;RETURN HERE THE RANDOM
(1)                                     ;*                              ;;NUMBER WILL BE IN
(1)                                     ;*                              ;;$HINUM,$LONUM
(1)
(1)  031406                             $RAND:
(3)  031406  010046                             MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  031410  010146                             MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  031412  010246                             MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(1)  031414  013700  031506                     MOV     $LONUM,R0       ;;SET R0 WITH LOW
(1)  031420  013701  031504                     MOV     $HINUM,R1       ;;SET R1 WITH HIGH
(1)  031424  012702  177771                     MOV     #-7,R2          ;;SET SHIFT COUNT
(1)  031430  006300                     1$:     ASL     R0              ;;SHIFT R0 LEFT AND
(1)  031432  006101                             ROL     R1              ;;ROTATE CARRY INTO R1 AND
(1)  031434  005202                             INC     R2              ;;CHECK FOR DONE
(1)  031436  001374                             BNE     1$              ;;CONTINUE SHIFT LOOP
(1)  031440  063700  031506                     ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
(1)  031444  005501                             ADC     R1              ;;PROPOGATE CARRY
(1)  031446  063701  031504                     ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
(1)  031452  062700  001057                     ADD     #1057,R0        ;;ADD LOW CONSTANT
(1)  031456  005501                             ADC     R1              ;;PROPOGATE CARRY
(1)  031460  062701  047401                     ADD     #47401,R1       ;;ADD HIGH CONSTANT
(1)  031464  010037  031506                     MOV     R0,$LONUM       ;;SAVE R0
(1)  031470  010137  031504                     MOV     R1,$HINUM       ;;SAVE R1
(3)  031474  012602                             MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  031476  012601                             MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  031500  012600                             MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  031502  000207                             RTS     PC              ;;RETURN
(1)  031504  176543                     $HINUM: .WORD   176543
(1)  031506  123456                     $LONUM: .WORD   123456
```

```
                                                        .SBTTL  ASCII MESSAGES
                                                
031510 047520 052123 051055  EM1:    .ASCIZ  /POST-RESET CONDITIONS NOT MET IN CSR./
031516 051505 052105 041440
031524 047117 044504 044524
031532 047117 020123 047516
031540 020124 042515 020124
031546 047111 041440 051123
031554 000056
031556 042522 042101 053457  EM2:    .ASCIZ  %READ/WRITE BIT FAILURE IN CSR.%
031564 044522 042524 041040
031572 052111 043040 044501
031600 052514 042522 044440
031606 020116 051503 027122
031614    000
031615    102 052131 020105  EM3:    .ASCIZ  /BYTE ACCESS FAILURE IN CSR./
031622 041501 042503 051523
031630 043040 044501 052514
031636 042522 044440 020116
031644 051503 027122    000
031651    105 051122 051117  EM4:    .ASCIZ  /ERROR IN CONNECT PROCESS./
031656 044440 020116 047503
031664 047116 041505 020124
031672 051120 041517 051505
031700 027123    000
031703    111 052116 051105  EM5:    .ASCIZ  /INTERUPT DETECTED WITH INTERUPT ENABLE CLEARED./
031710 050125 020124 042504
031716 042524 052103 042105
031724 053440 052111 020110
031732 047111 042524 052522
031740 052120 042440 040516
031746 046102 020105 046103
031754 040505 042522 027104
031762    000
031763    103 052517 042114  EM6:    .ASCIZ  /COULD NOT CONNECT./
031770 047040 052117 041440
031776 047117 042516 052103
032004 000056
032006 051105 047522 020122  EM7:    .ASCIZ  /ERROR IN RELEASE PROCESS./
032014 047111 051040 046105
032022 040505 042523 050040
032030 047522 042503 051523
032036 000056
032040 047516 044440 052116  EM10:   .ASCIZ  /NO INTERUPT DETECTED./
032046 051105 050125 020124
032054 042504 042524 052103
032062 042105 000056
032066 047111 042524 052522  EM11:   .ASCIZ  /INTERUPT AT WRONG PRIORITY./
032074 052120 040440 020124
032102 051127 047117 020107
032110 051120 047511 044522
032116 054524 000056
032122 047503 047116 041505  EM12:   .ASCIZ  /CONNECT CONDITION FAILURE./
032130 020124 047503 042116
032136 052111 047511 020116
032144 040506 046111 051125
```

```
                032152  027105     000
2257            032155     105  051122  051117  EM13:   .ASCIZ  /ERROR IN CONNECT FAILURE PROCESS./
                032162  044440  020116  047503
                032170  047116  041505  020124
                032176  040506  046111  051125
                032204  020105  051120  041517
                032212  051505  027123     000
2258            032217     105  051122  051117  EM14:   .ASCIZ  /ERROR IN RELEASE TEST WITHOUT BUSS MASTERSHIP./
                032224  044440  020116  042522
                032232  042514  051501  020105
                032240  042524  052123  053440
                032246  052111  047510  052125
                032254  041040  051525  020123
                032262  040515  052123  051105
                032270  044123  050111  000056
2259            032276  042522  042514  051501  EM15:   .ASCIZ  /RELEASE CONDITIONS NOT MET./
                032304  020105  047503  042116
                032312  052111  047511  051516
                032320  047040  052117  046440
                032326  052105  000056
2260            032332  046524  020117  044502  EM16:   .ASCIZ  /TMO BIT NOT SET./
                032340  020124  047516  020124
                032346  042523  027124     000
2261            032353     103  052517  042114  EM17:   .ASCIZ  %COULDN'T READ/WRITE EXTERNAL DEVICE.%
                032360  023516  020124  042522
                032366  042101  053457  044522
                032374  042524  042440  052130
                032402  051105  040516  020114
                032410  042504  044526  042503
                032416  000056
2262            032420  042522  042523  020124  EM20:   .ASCIZ  /RESET HOLD (HLD) DIDN'T DISABLE RESET./
                032426  047510  042114  024040
                032434  046110  024504  042040
                032442  042111  023516  020124
                032450  044504  040523  046102
                032456  020105  042522  042523
                032464  027124     000
2263            032467     122  051505  052105  EM21:   .ASCIZ  /RESET (RST) DIDN'T WORK./
                032474  024040  051522  024524
                032502  042040  042111  023516
                032510  020124  047527  045522
                032516  000056
2264            032520  044124  020105  050117  EM22:   .ASCIZ  /THE OPERATION OF THE RESET-IN-NEUTRAL HAS CHANGED./
                032526  051105  052101  047511
                032534  020116  043117  052040
                032542  042510  051040  051505
                032550  052105  044455  026516
                032556  042516  052125  040522
                032564  020114  040510  020123
                032572  044103  047101  042507
                032600  027104     000
2265            032603     040     000          SPACE:  .ASCIZ  / /
2266            032605     200  036460  044124  SELMS:  .ASCII  <200>/0=THIS MESSAGE/
                032612  051511  046440  051505
                032620  040523  042507
2267            032624  030600  046475  047117          .ASCII  <200>/1=MONOPORT TEST/
```

```
                032632  050117  051117  020124
                032640  042524  052123
 2268           032644  031200  046475  046125          .ASCII  <200>/2=MULTIPORT TEST/
                032652  044524  047520  052122
                032660  052040  051505     124
 2269           032665     200  036463  040515          .ASCIZ  <200>/3=MANUAL INTERVENTION TEST/
                032672  052516  046101  044440
                032700  052116  051105  042526
                032706  052116  047511  020116
                032714  042524  052123     000
 2270           032721     105  051122  051117  DH1:    .ASCII  /ERROR  TEST#   SHOULD  WAS     FROM/<200>
                032726  052011  051505  021524
                032734  051411  047510  046125
                032742  004504  040527  004523
                032750  051106  046517     200
 2271           032755     120  004503  041011          .ASCIZ  /PC             BE/
                032762  000105
 2272           032764  051105  047522  004522  DH2:    .ASCII  /ERROR  TEST#   WROTE   READ    DEVICE ADDRESS/<200>
                032772  042524  052123  004443
                033000  051127  052117  004505
                033006  042522  042101  042011
                033014  053105  041511  020105
                033022  042101  051104  051505
                033030  100123
 2273           033032  041520     000                  .ASCIZ  /PC/
 2274           033035     105  051122  051117  DH3:    .ASCII  /ERROR  TEST#   BITS FAILING RESET      DEVICE ADDRESS/<200>
                033042  052011  051505  021524
                033050  041011  052111  020123
                033056  040506  046111  047111
                033064  020107  042522  042523
                033072  004524  042504  044526
                033100  042503  040440  042104
                033106  042522  051523     200
 2275           033113     120  000103                  .ASCIZ  /PC/
 2276           033116  051105  047522  004522  DH4:    .ASCII  /ERROR  TEST#   RESET   READ    DEVICE ADDRESS/<200>
                033124  042524  052123  004443
                033132  042522  042523  004524
                033140  042522  042101  042011
                033146  053105  041511  020105
                033154  042101  051104  051505
                033162  100123
 2277           033164  041520  004411  044502          .ASCIZ  /PC             BITS/
                033172  051524     000
 2278                   033176                          .EVEN
 2279           033176  001116  001562  001124  DT1:    $ERRPC,TSTNUM,GOOD,BAD,BADA,0
                033204  001126  001122  000000
 2280           033212  001116  001562  001570  DT2:    $ERRPC,TSTNUM,DEVRW,BAD,DEVAD,0
                033220  001126  001566  000000
 2281           033226  001116  001562  001126  DT3:    $ERRPC,TSTNUM,BAD,DEVAD,0
                033234  001566  000000
 2282                   000001                          .END
```

.MAIN. MACY11 30A(1052) 29-JAN-79 13:38 PAGE 14
CRDTAA.P11 29-JAN-79 13:32 SYMBOL TABLE SEQ 0086

```
ABASE = 177420
ACDW1 = 000000
ACDW2 = 000000
ACPUOP= 000000
ADDW0 = 000000
ADDW1 = 000000
ADDW10= 000000
ADDW11= 000000
ADDW12= 000000
ADDW13= 000000
ADDW14= 000000
ADDW15= 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT= 000000
ADEVM = 000000
AENV  = 000000
AENVM = 000000
AFATAL= 000000
AMADR1= 000000
AMADR2= 000000
AMADR3= 000000
AMADR4= 000000
AMAMS1= 000000
AMAMS2= 000000
AMAMS3= 000000
AMAMS4= 000000
AMSGAD= 000000
AMSGLG= 000000
AMSGTY= 000000
AMTYP1= 000000
AMTYP2= 000000
AMTYP3= 000000
AMTYP4= 000000
APASS = 000000
APRIOR= 000000
APTCSU= 000040
APTENV= 000001
APTSIZ= 000200
APTSPO= 000100
ASWREG= 000000
ATESTN= 000000
AUNIT = 000000
AUSWR = 000000
AVECT1= 160350
AVECT2= 000000
BAD   = 001126
BADA  = 001122
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BPTVEC= 000014
CKSWR = 104407
CONID   021312
CONNEC  021254
COVID   001660
CR    = 000015
CRLF  = 000200
CSR     001552
CURFUN  001652
CURID   001654
CYCLE   001656
DDISP = 177570
DEVAD   001566
DEVCON  001704
DEVRC   001572
DEVRW   001570
DH1     032721
DH2     032764
DH3     033035
DH4     033116
DISPAT  007416
DISPLA  001142
DISPRE  000174
DMODE   022556
DSWR  = 177570
DTIME   023120
DT03    001564
DT1     033176
DT2     033212
DT3     033226
DT3SP   014624
ECHOB   025756
EMTVEC= 000030
EM1     031510
EM10    032040
EM11    032066
EM12    032122
EM13    032155
EM14    032217
EM15    032276
EM16    032332
EM17    032353
EM2     031556
EM20    032420
EM21    032467
EM22    032520
EM3     031615
EM4     031651
EM5     031703
EM6     031763
EM7     032006
ERREN = 017322
ERRVEC= 000004
EXPEND  001604
EXPRT1  001576
EXPRT2  001600
EXPRT3  001602
EXPTCT  001606
FLUSHM  022504
FLUSHP  022526
FUNCT1  007672
FUNCT2  010316
FUNCT3  011320
FUNCT4  011622
FUNCT5  012262
GOOD  = 001124
GTSWR = 104406
HT    = 000011
IOTVEC= 000020
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KONST   001702
LF    = 000012
MANEND  021060
MANIN   014722
MANMOD  014734
MASMEN  022010
MASTER  007430
MENU    001626
MMVEC = 000250
MONEND  007254
MONPRT  002556
MULDT3  012422
MULEND  014660
MULPRT  007266
NEXENT  001624
ONOFF   001674
OOPS    022402
PARFLG  025754
PARIN   023612
PFVSEV  017560
PIRQ  = 177772
PIRQVE= 000240
PORTNO  001574
POSTSV= 020366
PRI     001560
PR0   = 000000
PR1   = 000040
PR2   = 000100
PR3   = 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
PWFSER  020404
PWRMSG  031270
PWROK   016000
PWRVEC= 000024
RDCHR = 104410
RDLIN = 104411
RDOCT = 104412
REPAR = 024334
REQ   = 000001
RESTRT  002444
RESVEC= 000010
ROUTE   002552
ROUT1   002454
ROUT2   002466
ROUT3   002500
ROUT4   002516
R6    =%000006
R7    =%000007
SALMEN  022076
SECT2   012646
SECT2A  013246
SECT2B  013312
SECT2C  013542
SECT3   013552
SECT4   014014
SECT5   014240
SELMS   032605
SEQ     001610
SEQINT  022246
SEQMAK  021700
SEQPTR  001622
SEQSTP  022270
SETUP1  002160
SETUP2  002206
SETUP3  002230
SLAVE   007556
SPACE   032603
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STACK = 001100
START   001710
STKLMT= 177774
STPNEX  022344
SWBPWF  016556
SWR     001140
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
```

```
SW15  = 100000        TYPOC = 104402        $DDW7   001310        $LPERR  001110        $SVLAD  031044
SW2   = 000004        TYPON = 104404        $DDW8   001312        $MADR1  001240        $SVPC = 000220
SW3   = 000010        TYPOS = 104403        $DDW9   001314        $MADR2  001244        $SWR  = 167400
SW4   = 000020        VEC0    001554        $DEVCT  001216        $MADR3  001250        $SWREG  001230
SW5   = 000040        VEC2    001556        $DEVM   001264        $MADR4  001254        $SWRMK= 000000
SW6   = 000100        WAIT    021402        $DOAGN  021230        $MAIL   001206        $TESTN  001212
SW7   = 000200        WAIT05  021370        $DTBL   030120        $MAMS1  001236        $TIMES  001172
SW8   = 000400        WAIT10  021342        $ENDAD  021220        $MAMS2  001242        $TKB    001146
SW9   = 001000        WAIT20  021354        $ENDCT  021166        $MAMS3  001246        $TKS    001144
SYNCUP  021444        $APTHD  001000        $ENDMG  021237        $MAMS4  001252        $TMP0   001160
TABM    012410        $ATYC   026366        $ENULL  021234        $MBADR  001002        $TMP1   001162
TBITVE= 000014        $ATY1   026342        $ENV    001226        $MFLG   026604        $TMP2   001164
TEMP1   001662        $ATY3   026350        $ENVM   001227        $MNEW   027455        $TMP3   001166
TEMP2   002554        $ATY4   026360        $EOP    021132        $MSGAD  001222        $TMP4   001170
TICK    021440        $AUTOB  001134        $EOPCT  021160        $MSGLG  001224        $TN   = 000020
TIMA    001676        $BASE   001262        $ERFLG  001103        $MSGTY  001206        $TPB    001152
TIMB    001700        $BDADR  001122        $ERMAX  001115        $MSWR   027444        $TPFLG  001157
TKVEC = 000060        $BDDAT  001126        $ERROR  030140        $MTYP1  001237        $TPS    001150
TOCK    021442        $BELL   001176        $ERRPC  001116        $MTYP2  001243        $TRAP   031324
TPVEC = 000064        $CDW1   001266        $ERRTB  001332        $MTYP3  001247        $TRAP2  031346
TRAPVE= 000034        $CDW2   001270        $ERRTY  030470        $MTYP4  001253        $TRP  = 000013
TRAP4   025772        $CHARC  026336        $ERTTL  001112        $MXCNT  031114        $TRPAD  031360
TRIG    021366        $CKSWR  026712        $ESCAP  001174        $NULL   001154        $TSTM   001004
TRIN    001706        $CMTAG  001100        $ETABL  001226        $NWTST= 000001        $TSTNM  001102
TRTVEC= 000014        $CM3  = 000000        $ETEND  001332        $OCNT   027710        $TTYIN  027422
TSTNUM  001562        $CM4  = 000005        $FATAL  001210        $OMODE  027712        $TYPDS  027714
TST1    002722        $CNTLG  027437        $FFLG   026606        $OVER   031100        $TYPE   026060
TST10   004640        $CNTLU  027432        $FILLC  001156        $PASS   001214        $TYPEC  026272
TST11   005042        $CPUOP  001234        $FILLS  001155        $PASTM  001006        $TYPEX  026340
TST12   005326        $CRLF   001203        $GDADR  001120        $PWRAD  031256        $TYPOC  027512
TST13   005502        $DBLK   030130        $GDDAT  001124        $PWRDN  031116        $TYPON  027526
TST14   005742        $DDW0   001272        $GET42  021210        $PWRMG  031252        $TYPOS  027466
TST15   006306        $DDW1   001274        $GTSWR  026762        $PWRUP  031170        $UNIT   001220
TST16   006436        $DDW10  001316        $HIBTS  001000        $QUES   001202        $UNITM  001010
TST17   006600        $DDW11  001320        $HINUM  031504        $RAND   031406        $USWR   001232
TST2    003010        $DDW12  001322        $HIOCT  026710        $RDCHR  027174        $VECT1  001256
TST3    003236        $DDW13  001324        $ICNT   001104        $RDLIN  027314        $VECT2  001260
TST4    003352        $DDW14  001326        $ILLUP  031262        $RDOCT  026610        $XTSTR  030650
TST5    003602        $DDW15  001330        $INTAG  001135        $RDSZ = 000010        $$GET4= 000000
TST6    004060        $DDW2   001276        $ITEMB  001114        $RTNAD  021232        $OFILL  027711
TST7    004362        $DDW3   001300        $LF     001204        $SAVR6  031266        $40CAT  000000
TX      025762        $DDW4   001302        $LFLG   026605        $SCOPE  030624        .     = 033240
TYPDS = 104405        $DDW5   001304        $LONUM  031506        $SETUP= 000137        .$X   = 001000
TYPE  = 104401        $DDW6   001306        $LPADR  001106        $STUP = 177777

. ABS.  033240      000

 ERRORS DETECTED:  0

 CRDTAA,CRDTAA_CRDTAA.P11
 RUN-TIME: 80 %6 1 SECONDS
 RUN-TIME RATIO: 361/!29=2.7
 CORE USED:  26K  (51 PAGES)
```