

DT07

DT07 UNIBUS SW DIAG  
CRDTABO

AH-F170B-MC  
FICHE 1 OF 1

APR 1982  
COPYRIGHT © 79-82  
MADE IN USA



The main body of the document contains a grid of approximately 15 columns and 15 rows of technical data. Each cell in the grid contains a small, dense table or diagram, likely representing a component of a software diagnostic routine. The text is too small to be legible, but the layout suggests a structured data format. The right half of the page is mostly blank, with some faint, illegible markings.

1

.NLIST LOC,SEQ,BIN  
.REM\_

PRODUCT CODE: AC-F169B-MC

PRODUCT NAME: CRDTAB0 UNIBUS SW DIAG

DATE: 27 JANUARY 1979

UPDATE: 01 NOVEMBER 1981

AUTHOR(S): JAMES DUPRE

UPDATE AUTHOR(S): VIJAY ANANDWALA

MAINTAINER: CSS PERIPHERALS AND GRAPHICS GROUP

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

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

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

COPYRIGHT: 1979,1980,1982 DIGITAL EQUIPMENT CORP.  
MAYNARD MASS.

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.

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

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

NPR TEST; THIS TEST CHECKS OUT THE NPR LINE ON THE SWITCH BUS.

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!)  
UBE <UNIBUS EXERCISER> (OPTIONAL BUT NEEDED FOR NPR 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 KEYS 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.INPUT PARAMETERS AND ALLOWS TEST SELECTION
  
  
- 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 200  
                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 MULTIPOST TEST ON ALL PORTS IN THE SYSTEM.  
IF YOU HAVE A MANUAL CONTROL PANEL, RUN THE MANUAL INTER-  
VENTION TEST. IF YOU HAVE UBE ON THE SWITCH BUS, RUN THE  
NPR TEST.

THE PARAMETER INPUT ROUTINE IS TO BE USED WHEN-  
EVER 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 AUTOMATICALLY  
FROM THE MULTIPOST TEST IF NO PARAMETERS HAVE BEEN ENTERED  
PREVIOUSLY.

WHEN USING THE MULTIPOST 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 MULTIPOINT SECTION ERRORS

ERRORS IN THE MULTIPOINT 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  
MULTIPOINT TEST AND ERROR ID.#S 6 THRU 16 IDENTIFY ERRORS IN THE  
DT03 MODE MULTIPOINT 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 NPR TEST

ERRORS IN THIS TEST PRINT A BRIEF MESSAGE DESCRIBING THE ERROR AND  
ERROR PC.

8.5 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 INTERRUPT ENABLE.

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

TEST5. RELEASE TEST WITHOUT INTERRUPT ENABLE.

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

TEST6. CONNECT FAILURE TEST WITHOUT INTERRUPT ENABLE.

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

TEST7. RELEASE TEST WITHOUT BUS MASTERSHIP OR INTERRUPT ENABLE.

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

TEST10. CONNECT TEST

THIS TEST VERIFIES THE CONNECT PROCESS UNDER INTERRUPT CONTROL.

TEST11. RELEASE TEST

THIS TEST VERIFIES THE RELEASE PROCESS UNDER INTERRUPT CONTROL.

TEST12. CONNECT FAILURE TEST

THIS TEST VERIFIES THE PROPER OPERATION OF THE CONNECT PROCESS, UNDER INTERRUPT 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 INTERRUPT 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 MULTIPOINT TEST

THE MULTIPOINT 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.

## 9.4 NPR TEST

THE NPR TEST CHECKS OUT THE NPR LINE ON THE SWITCH BUS.

```
384 - 167400 $SWR=167400
385 000001 $TN=1
386
387 .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 IGT,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= X0 ;;GENERAL REGISTER
(1) 000001 R1= X1 ;;GENERAL REGISTER
(1) 000002 R2= X2 ;;GENERAL REGISTER
(1) 000003 R3= X3 ;;GENERAL REGISTER
(1) 000004 R4= X4 ;;GENERAL REGISTER
(1) 000005 R5= X5 ;;GENERAL REGISTER
(1) 000006 R6= X6 ;;GENERAL REGISTER
(1) 000007 R7= X7 ;;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

```

;\*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

```

;\*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

```

.MAIN. MACY11 30A(1052) 16-NOV-81 17:26 PAGE 12-2  
 CRDTAB.P11 16-NOV-81 17:16 BASIC DEFINITIONS

SEQ 0015

```

(1)      000060      TKVEC= 60          ;;TTY KEYBOARD VECTOR
(1)      000064      TPVEC= 64          ;;TTY PRINTER VECTOR
(1)      000240      PIRQVEC=240       ;;PROGRAM INTERRUPT REQUEST VECTOR
388      177420      ABASE=177420     ;BASE DT07 BUS ADDRESS EQUATE
389      160350      AVECT1=160350    ;BASE DT07 PRIORITY & VECTOR ADDRESS EQUATE
390      001124      GOOD=$GDDAT
391      001126      BAD=$BDDAT
392      000001      REQ=BIT0
393      001122      BADA=$BDADR
394      .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)
395      .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

```



```

(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 002612 .WORD ROUT3 ;;VECTOR TO STARTING ADDRESS
(1) 000006 000340 .WORD 340 ;;WITH PRIORITY LEVEL 7
(1) 000174 000174 .=174
(1) 000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
(1) .SBTTL STARTING ADDRESS(ES)
(1) 000200 000137 002612 JMP @#ROUT3 ;;GO TO START OF PROGRAM

```

```

397          .SBTTL ACT11 HOOKS
(1)
(2)          ::*****
(1)          ;HOOKS REQUIRE BY ACT11
(1)          $SVPC=.          ;SAVE PC
(1)          .=46
(1) 000046 000046          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)          .=52
(1) 000052 000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)          .=$SVPC          ;; RESTORE PC
398          .=1000
399          .SBTTL APT PARAMETER BLOCK
(1)
(2)          ::*****
(1)          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)          ::*****
(1)          .SX=.          ;;SAVE CURRENT LOCATION
(1)          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200          200          ;;FOR APT START UP
(1)          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000          $APTHDR          ;;POINT TO APT HEADER BLOCK
(1)          .=$X          ;;RESET LOCATION COUNTER
(2)          ::*****
(1)          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)          ;INTERFACE SPEC.
(1)
(1) 001000          $APTHD:
(1) 001000 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001206          $MBADR: .WORD $MAIL          ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000017          $STMT: .WORD 15.          ;;RUN TIM OF LONGEST TEST
(1) 001006 000074          $PASTM: .WORD 60.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000000          $UNITM: .WORD 0          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 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          000000          $CMTAG:          ;;START OF COMMON TAGS
(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	\$DEVN:	.WORD	ADEVN	::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

.MAIN. MACY11 30A(1052) 16-NOV-81 17:26 PAGE 12-7  
CRDTAB.P11 16-NOV-81 17:16 APT MAILBOX-ETABLE

SEQ 0020

(2) 001330 000000  
(2)  
(2)  
(2) 001332  
(2)

\$DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15

\$ETEND:

```
(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) $ERRTB:
(1) 001332 ;ERROR#1
401 ;EM1 ;:POST-RESET CONDITIONS NOT MET IN CSR
402 001332 033554 ;DH1
403 001334 035005 ;DT1
404 001336 035356 ;0
405 001340 000000 ;ERROR#2
406 ;EM2 ;:READ/WRITE BIT FAILURE IN CSR
407 001342 033622 ;DH1
408 001344 035005 ;DT1
409 001346 035356 ;0
410 001350 000000 ;ERROR#3
411 ;EM3 ;:BYTE ACCESS FAILURE IN CSR
412 001352 033661 ;DH1
413 001354 035005 ;DT1
414 001356 035356 ;0
415 001360 000000 ;ERROR#4
416 ;EM4 ;:ERROR IN CONNECT PROCESS
417 001362 033715 ;DH1
418 001364 035005 ;DT1
419 001366 035356 ;0
420 001370 000000 ;ERROR#5
421 ;EM5 ;:INTERUPT DETECTED WITH INTERUPT ENABLE CLEARED
422 001372 033747 ;DH1
423 001374 035005 ;DT1
424 001376 035356 ;0
425 001400 000000 ;ERROR#6
426 ;EM6 ;:COULD NOT CONNECT
427 001402 034027 ;DH1
428 001404 035005 ;DT1
429 001406 035356 ;0
430 001410 000000 ;ERROR#7
431 ;EM7 ;:ERROR IN RELEASE PROCESS
432 001412 034052 ;DH1
433 001414 035005 ;DT1
434 001416 035356 ;0
435 001420 000000 ;ERROR#10
436 ;EM10 ;:NO INTERUPT DETECTED
437 001422 034104 ;DH1
438 001424 035005 ;DT1
439 001426 035356 ;0
440 001430 000000 ;ERROR#11
441
```

442	001432	034132	EM11	;INTERUPT AT WRONG PRIORITY
443	001434	035005	DH1	
444	001436	035356	DT1	
445	001440	000000	0	
446			;ERROR#12	
447	001442	034166	EM12	;CONNECT CONDITION FAILURE
448	001444	035005	DH1	
449	001446	035356	DT1	
450	001450	000000	0	
451			;ERROR#13	
452	001452	034221	EM13	;ERROR IN CONNECT FAILURE PROCESS
453	001454	035005	DH1	
454	001456	035356	DT1	
455	001460	000000	0	
456			;ERROR#14	
457	001462	034263	EM14	;ERROR IN RELEASE TEST W/C BUSS MASTERSHIP
458	001464	035005	DH1	
459	001466	035356	DT1	
460	001470	000000	0	
461			;ERROR#15	
462	001472	034342	EM15	;RELEASE CONDITIONS NOT MET
463	001474	035005	DH1	
464	001476	035356	DT1	
465	001500	000000	0	
466			;ERROR#16	
467	001502	034376	EM16	;TMO BIT NOT SET
468	001504	035005	DH1	
469	001506	035356	DT1	
470	001510	000000	0	
471			;ERROR#17	
472	001512	034417	EM17	;COULDN'T READ/WRITE EXTERNAL DEVICE
473	001514	035050	DH2	
474	001516	035372	DT2	
475	001520	000000	0	
476			;ERROR#20	
477	001522	034464	EM20	;RESET HOLD (HLD) DIDN'T DISABLE RESET
478	001524	035005	DH1	
479	001526	035356	DT1	
480	001530	000000	0	
481			;ERROR#21	
482	001532	034533	EM21	;RESET (RST) DIDN'T WORK
483	001534	035121	DH3	
484	001536	035406	DT3	
485	001540	000000	0	
486			;ERROR#22	
487	001542	034564	EM22	;RESET-IN-NEUTRAL OPERATION HAS CHANGED
488	001544	035202	DH4	
489	001546	035372	DT2	
490	001550	000000	0	
491			;ERROR+23	
492	001552	027474	EM23	;ERROR:NPR DATO NOT DONE
493	001554	035274	DH5	
494	001556	035420	DT4	
495	001560	000000	0	
496			;ERROR+24	
497	001562	027524	EM24	;ERROR:NPR DID NOT SET RDY

498 001564 035274  
499 001566 035420  
500 001570 000000  
501  
502 001572 027556  
503 001574 035307  
504 001576 035424  
505 001600 000000  
506  
507 001602 027634  
508 001604 035274  
509 001606 035420  
510 001610 000000  
511  
512  
513 001612 027726  
514 001614 035274  
515 001616 035420  
516 001620 000000  
517  
518  
519  
520

DH5  
DT4  
0  
;ERROR+25  
EM25  
DH6  
DT5  
0  
;ERROR+26  
EM26  
DH5  
DT4  
0  
;ERROR+27  
EM27  
DH5  
DT4  
0

;ERROR:UBE DID NOT INTERRUPT WHEN NPR FINISHED

;ERROR:TWO LOC WRITTEN WHEN ONE NPR AND INT ON DONE ENABLE

;ERROR:DEVICE DOESN'T EXIST ON SWITCH BUS



```

522      :
523      :      DT07 BUS REGISTER ADDRESS POINTERS
524      :
525 001622 177420 CSR: 177420 ;COMMAND/STATUS REGISTER
526      :
527      :      DT07 VECTOR ADDRESS POINTERS
528      :
529 001624 000350 VEC0: 350 ;NORMAL INTERRUPT VECTOR
530 001626 000352 VEC2: 352 ;
531      :
532      :      DT07 DEVICE LEVEL
533      :
534 001630 000340 PRI: 340
535      :
536      :      COMMAND STATUS REGISTER BIT ASSIGNMENTS
537      :
538      :      COMMON PROGRAM LOCATION(S)
539      :
540 001632 000000 TSTNUM: 0 ;CONTAINS TEST NUMBER ON ERROR
541      :*****
542      :PROGRAM PARAMETER BLOCK
543 001634 000000 DT03: 0000 ;DT03/DT07 MODE INDICATOR,0=DT07
544 001636 000000 DEVAD: 0000 ;EXTERNAL DEVICE REGISTER ADDRESS
545 001640 000000 DEVRW: 0000 ;READ/WRITE BITS IN ABOVE
546 001642 000000 DEVRC: 0000 ;MASK OF RESET CLEARABLE BITS IN ABOVE
547 001644 177777 PORTNO: -1 ;THIS PORT'S ID#
548 001646 177777 EXPRT1: -1 ;EXTERNAL PORT ID (-1=NO PORT)
549 001650 177777 EXPRT2: -1 ;
550 001652 177777 EXPRT3: -1 ;
551 001654 177777 EXPEND: -1 ;TABLE TERMINATOR
552 001656 000000 EXPTCT: 0 ;# OF EXTERNAL PORTS
553 001660 177777 SEQ: -1 ;SEQUENCER
554 001662 177777 -1
555 001664 177777 -1
556 001666 177777 -1
557 001670 177777 -1
558 001672 001660 SEQPTR: SEQ ;SEQUENCER POINTER
559 001674 001662 NEXENT: SEQ+2 ;NEXT ENTRY POINTER
560 001676 000011 MENU: .BLKW 9. ;MENU STORAGE
561 001720 177777 -1 ;MENU TERMINATOR
562 001722 000000 CURFUN: 0 ;CURREANT FUNCTION
563 001724 000000 CURID: 0 ;CURREANT SLAVE ID
564 001726 000000 CYCLE: 0 ;SUBPASS COUNT
565 001730 000000 COVID: 0 ;CONVERTED SLAVE ID
566 001732 000005 TEMP1: .BLKW 5 ;TEMPORARY STORAGE
567 001744 000000 ONOFF: 0 ;MASTER/SLAVE INDICATOR
568 001746 000000 TIMA: 0
569 001750 000000 TIMB: 0
570 001752 000000 KONST: 0
571 001754 000001 DLVCON: 1 ;RESET-IN-NEUTRAL CONDITION FLAG
572 001756 177777 TRIN: -1 ;TEST RESET IN NEUTRAL FLAG
573 001760 000000 USEPAR: 0 ;USING PARIN FLAG 0=NOT USING
574
575

```

```

578          .SBTTL PROGRAM START
579 001762   START:
(1)          .SBTTL INITIALIZE THE COMMON TAGS
(1)          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001762  012706 001100   MOV    # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 001766  005026          CLR    (R6)+           ;;CLEAR MEMORY LOCATION
(1) 001770  022706 001140   CMP    #SWR,R6      ;;DONE?
(1) 001774  001374          BNE    #-6           ;;LOOP BACK IF NO
(1) 001776  012706 001100   MOV    #STACK,SP    ;;SETUP THE STACK POINTER
(1)          ;;INITIALIZE A FEW VECTORS
(1) 002002  012737 032670 000020   MOV    # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 002010  012737 000340 000022   MOV    #340,@#IOTVEC+2 ;;LEVEL 7
(1) 002016  012737 032204 000030   MOV    # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 002024  012737 000340 000032   MOV    #340,@#EMTVEC+2 ;;LEVEL 7
(1) 002032  012737 033370 000034   MOV    # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 002040  012737 000340 000036   MOV    #340,@#TRAPVEC+2;LEVEL 7
(1) 002046  012737 033162 000024   MOV    # $PURDN,@#PURVEC ;;POWER FAILURE VECTOR
(1) 002054  012737 000340 000026   MOV    #340,@#PURVEC+2 ;;LEVEL 7
(1) 002062  013737 021606 021600   MOV    $ENDCT,$EOPCT  ;;SETUP END-OF-PROGRAM COUNTER
(1) 002070  005037 001172          CLR    $TIMES        ;;INITIALIZE NUMBER OF ITERATIONS
(1) 002074  005037 001174          CLR    $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002100  112737 000001 001115   MOVB  #', $ERMAX     ;;ALLOW ONE ERROR PER TEST
(1) 002106  012737 002106 001106   MOV    #', $LPADR    ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002114  012737 002114 001110   MOV    #', $LPERR    ;;SETUP THE ERROR LOOP ADDRESS
(2)          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)          ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002122  013746 000004          MOV    @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 002126  012737 002162 000004   MOV    #64$,@#ERRVEC ;;SET UP ERROR VECTOR
(2) 002134  012737 177570 001140   MOV    #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 002142  012737 177570 001142   MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 002150  022777 177777 176762   CMP    #-1,@SWR     ;;TRY TO REFERENCE HARDWARE SWR
(2) 002156  001012          BNE    66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)          ;;AND THE HARDWARE SWR IS NOT = -1
(2) 002160  000403          BR     65$          ;;BRANCH IF NO TIMEOUT
(2) 002162  012716 002170          64$: MOV    #65$, (SP)   ;;SET UP FOR TRAP RETURN
(2) 002166  000002          RTI
(2) 002170  012737 000176 001140   65$: MOV    #SWREG,SWR   ;;POINT TO SOFTWARE SWR
(2) 002176  012737 000174 001142   MOV    #DISPREG,DISPLAY
(2) 002204  012637 000004          66$: MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)          CLR    $PASS      ;;CLEAR PASS COUNT
(2) 002210  005037 001214          BITB  #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 002214  132737 000200 J01227   BEQ    67$          ;;YES,USE NON-APT SWITCH
(2) 002222  001403          MOV    # $SWREG,SWR  ;;NO,USE APT SWITCH REGISTER
(2) 002224  012737 001230 001140   67$:
(2) 002232          SETUP1: MOV    #TRAP4,@#ERRVEC ;POINT TO TRAP ROUTINE
580 002232  012737 026444 000004   CLR    R0
581 002240  005000          MOV    $VECT1,R0    ;GET PRIORITY LEVEL
582 002242  013700 001256          BIC   #777,R0       ;CLEAR UNWANTED BITS
583 002246  042700 000777          SWAB  R0
584 002252  000300          MOV    R0,PRI      ;SAVE DEVICE LEVEL
585 002254  010037 001630
586          :
587          :
588          :
589 002260  013737 001262 001622   SETUP2: MOV    $BASE,CSR ;STORE BUS ADDRESS
590          :

```

```
591      :      NOW SETUP VECTOR ADDRESSES
592      :
593 002266 012700 001624      MOV      #VECO,R0      ;SETUP VECTOR POINTER
594 002272 013701 001256      MOV      $VECT1,R1     ;GET BASE VECTOR ADDR
595 002276 042701 177000      BIC      #177000,R1    ;CLEAR UNWANTED BITS
596 002302 010120      SETUP3: MOV      R1,(R0)+  ;PUT ADDR
597 002304 062701 000002      ADD      #2,R1        ;INCR.TO NEXT LOC.
598 002310 022700 001630      CMP      #VEC2+2,R0   ;DONE ALL LOCATIONS?
599 002314 001372      BNE      SETUP3       ;BR,IF NOT DONE
600 002316 005737 000042      TST      @#42         ;TEST CHAIN MODE UNDER XXDP
601 002322 001052      BNE      1$           ;BR,IF CHAIN MODE
602      .SBTTL TYPE PROGRAM NAME
(1)      ::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002324 005227 177777      INC      #-1          ;:FIRST TIME?
(1) 002330 001047      BNE      64$          ;:BRANCH IF NO
(1) 002332 022737 021640 000042  CMP      #SENDAD,@#42 ;:ACT-11?
(1) 002340 001443      BEQ      64$          ;:BRANCH IF YES
(1) 002342 104401 002410      TYPE     .65$         ;:TYPE ASCIZ STRING
(2)      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002346 005737 000042      TST      @#42         ;:ARE WE RUNNING UNDER XXDP/ACT?
(2) 002352 001012      BNE      66$          ;:BRANCH IF YES
(2) 002354 123727 001226 000001  CMPB     $ENV,#1      ;:ARE WE RUNNING UNDER APT?
(2) 002362 001406      BEQ      66$          ;:BRANCH IF YES
(2) 002364 023727 001140 000176  CMP      SWR,#SWREG   ;:SOFTWARE SWITCH REG SELECTED?
(2) 002372 001005      BNE      67$          ;:BRANCH IF NO
(2) 002374 104406      GTSWR     ;:GET SOFT-SWR SETTINGS
(2) 002376 000403      BR       67$
(2) 002400 112737 000001 001134 66$: MOVB     #1,$AUTOB    ;:SET AUTO-MODE INDICATOR
(2) 002406      67$:
(1) 002406 000420      BR       64$         ;:GET OVER THE ASCIZ
(1)      ::65$: .ASCIZ <CRLF>/MAINDEC-11-CRDTAB DT07 TEST/<CRLF>
(1) 002450      64$:
603 002450 005737 001760      1$: TST      USEPAR     ;:PARIN IN USE?
604 002454 001024      BNE      RESTRT      ;:BR=YES; SKIP OVER DEVICE ADDRESS DEPENDENT CODE
605 002456 012777 023022 177140  MOV      #OOPS,@VECO  ;:LOAD UNEXPECTED INTERRUPT RETURN
606 002464 012777 000340 177134  MOV      #340,@VEC2   ;:SET PRIORITY 7 ON INTERRUPT
607 002472 005077 177124      CLR      @CSR        ;:DETERMINE STATUS OF PWR OK BIT
608 002476 000005      RESET     ;:RESET THE BUSS
609 002500 000240      NOP
610 002502 000240      NOP
611 002504 000240      NOP
612 002506 005037 001752      CLR      KONST
613 002512 017737 177104 001752  MOV      @CSR,KONST   ;:SAVE CSR
614 002520 042737 173777 001752  BIC      #173777,KONST ;:MASK EVERYTHING EXCEPT PWR OK
615 002526 012706 001100      RESTRT: MOV      #STACK,SP ;:RESET STACK POINTER
616 002532 012737 000002 000512  MOV      #2,@#512     ;:PUT RTI IN USE VECTORS
617 002540 012737 000002 000516  MOV      #2,@#516     ;:DITTO
618 002546 012737 000002 000522  MOV      #2,@#522     ;:DITTO
619 002554 012737 000002 000526  MOV      #2,@#526     ;:DITTO
620 002562 000177 000104      JMP      @ROUTE      ;:GO DO IT
621 002566 012737 002676 002672  ROUT1: MOV      #MONPRT,ROUTE ;:SET UP FOR MONOPOINT
622 002574 000137 001762      JMP      START
623 002600 012737 007556 002672  ROUT2: MOV      #MULPRT,ROUTE ;:SET UP FOR MULTIPOINT
624 002606 000137 001762      JMP      START
625 002612 012737 000001 001760  ROUT3: MOV      #1,USEPAR ;:SET FLAG
626 002620 012737 024756 002672  MOV      #PARIN,ROUTE ;:SET RETURN
```

.MAIN. MACY11 30A(1052) 16-NOV-81 17:26 PAGE 12-14  
 CRDTAB.P11 16-NOV-81 17:16 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0027

```

627 002626 005037 026426      CLR      PARFLG ;CLEAR PARAMETER FLAG
628 002632 000137 001762      JMP      START
629 002636 012737 002650 002672 ROUT4: MOV      #1$,ROUTE ;SET UP RETURN
630 002644 000137 001762      JMP      START
631 002650 012737 015300 002672 1$:  MOV      #MANIN,ROUTE ;SET UP FOR MANUAL INTERVENTION TEST
632 002656 004737 023540      JSR      PC,DTIME ;GO GET DEVICE TIMING
633 002662 004737 023176      JSR      PC,DMODE ;GO GET DEVICE MODE
634 002666 000137 015300      JMP      MANIN
635 002672 002676      ROUTE: MONPRT ;MAIN TEST DISPATCHER
636 002674 000000      TEMP2: 0
637
638 002676      MONPRT:
639 002676 004737 026620      JSR      PC,CNTLC ;IS IT CONTROL C ?
640 002702 005737 026426      TST      PARFLG
641 002706 100402      BMI      1$
642 002710 004737 024232      JSR      PC,GETADR
643 002714 004737 023176 1$:  JSR      PC,DMODE
644 002720 004737 023540      JSR      PC,DTIME
645 002724 005737 000042      TST      @#42 ;XXDP CHAINMODE OR APT/ACT?
646 002730 001053      BNE      TST1 ;SKIP PRINTOUT IF ANYONE OF THE ABOVE
647 002732 005737 001214      TST      $PASS ;0 PASSES?
648 002736 001046      BNE      3$ ;BR=NO
649 002740 005737 001634      TST      DT03 ;WHAT MODE IS DEVICE?
650 002744 001422      BEQ      2$ ;BR =DT07 MODE
651 002746 104401 002754      TYPE      ,65$ ;:TYPE ASCIZ STRING
(1) 002752 000416      BR       64$ ;:GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>/THIS DT07 IS IN DT03 MODE./
(1) 64$:
652 003010 000421      BR       3$ ;GO START TESTS
653 003012 2$:
(1) 003012 104401 003020      TYPE      ,67$ ;:TYPE ASCIZ STRING
(1) 003016 000413      BR       66$ ;:GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <200>/THIS DT07 IS PORT# /
(1) 66$:
654 003046 013746 001644      MOV      PORTNO,-(SP) ;:SAVE PORTNO FOR TYPEOUT
(1) 003052 104402      TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
655 003054 004737 021762 3$:  JSR      PC,WAIT10 ;WAIT ONE SECOND
656
657
658

```

```

660
661
662
(3)
(3)
(2) 003060 000004
(1) 003062 012737 000062 001172
663 003070 004737 026620
664 003074 012737 000340 177776
665 003102 012777 000101 176512
666 003110 013737 001752 001124
667 003116 000005
668 003120 005200
669 003122 001376
670 003124 017737 176472 001126
671 003132 013737 001622 001122
672 003140 023737 001124 001126
673 003146 001401
674 003150 104001
675 003152

```

.SBTTL

```

:*****
:*TEST 1 TEST POST-RESET BIT PATTERN IN CSR.IDX1
:*****
TST1: SCOPE
MOV #50.,$TIMES ;DO 50. ITERATIONS
JSR PC,CNTLC ;IS IT CONTROL C?
MOV #340,PSW ;RISE PRIORITY
MOV #101,@CSR ;SET SOME BITS
MOV KONST,GOOD ;STORE GOOD DATA
RESET ;INIT. BUS
1$: 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

```

```

677      ::*****
(3)      ::*TEST 2      TEST READ/WRITE BITS IN CSR.IDX2
(3)      ::*****
(2) 003152 000004 TST2: SCOPE
(1) 003154 012737 000062 001172 MOV #50, $TIMES      ;;DO 50. ITERATIONS
678 003162 ^04737 026620 JSR PC, CNTLC      ;;IS IT CONTROL C?
679
680 003166 012737 000340 177776 MOV #340, PSW      ;LOCKOUT INTERUPTS
681 003174 005077 176422 CLR @CSR          ;ZERO CSR
682 003200 013737 001752 001124 1$: MOV KONST, GOOD    ;STORE GOOD DATA
683 003206 062737 000001 001124 ADD #1, GOOD
684 003214 012737 003200 001110 MOV #1$, $LPERR   ;SET LOOP-ON-ERROR ADDRESS
685 003222 013777 001124 176372 MOV GOOD, @CSR    ;SET BIT PATTERN
686 003230 017737 176366 001126 MOV @CSR, BAD     ;STORE CONTENTS OF CSR
687 003236 042737 120600 001126 BIC #120600, BAD ;MASK THE CONNECT PROCESS INDIACATORS
688 003244 042737 000074 001126 BIC #74, BAD     ;MASK RQ0-3
689 003252 023737 001124 001126 CMP GOOD, BAD    ;COMPARE
690 003260 001401 BEQ 3$          ;BR IF OK
691 003262 104002 ERROR+2      ;READ/WRITE BIT FAILURE IN CSR
692 003264 013737 001752 001124 3$: MOV KONST, GOOD    ;STORE GOOD DATA
693 003272 062737 000100 001124 ADD #100, GOOD
694 003300 012737 003264 001110 MOV #3$, $LPERR   ;SET LOOP-ON-ERROR ADDRESS
695 003306 013777 001124 176306 MOV GOOD, @CSR    ;SET BIT PATTERN
696 003314 017737 176302 001126 MOV @CSR, BAD     ;STORE CONTENTS OF CSR
697 003322 042737 120600 001126 BIC #120600, BAD ;MASK THE CONNECT PROCESS INDIACATORS
698 003330 023737 001124 001126 CMP GOOD, BAD    ;COMPARE
699 003336 001401 BEQ 4$          ;BR IF OK
700 003340 104002 ERROR+2      ;READ/WRITE BIT FAILURE IN CSR
701 003342 013737 001752 001124 4$: MOV KONST, GOOD    ;STORE GOOD DATA
702 003350 012737 003342 001110 MOV #4$, $LPERR   ;SET LOOP-ON-ERROR ADDRESS
703 003356 013777 001124 176236 MOV GOOD, @CSR    ;SET BIT PATTERN
704 003364 017737 176232 001126 MOV @CSR, BAD     ;STORE CONTENTS OF CSR
705 003372 023737 001124 001126 CMP GOOD, BAD    ;COMPARE
706 003400 001401 BEQ 5$          ;BR IF OK
707 003402 104002 ERROR+2      ;READ/WRITE BIT FAILURE IN CSR
708 003404 5$: GO TO NEXT TEST

```

710  
711  
(3)  
(3)  
(2)  
(1)  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730

003404 000004  
003406 012737 000012 001172  
003414 004737 026620  
  
003420 005077 176176  
003424 013700 001622  
003430 112710 000376  
003434 013737 001752 001124  
003442 062737 000100 001124  
003450 011037 001126  
003454 023737 001124 001126  
003462 001401  
003464 104003  
003466 005077 176130  
003472 013737 001752 001124  
003500 112760 000377 000001  
003506 011037 001126  
003512 023737 001124 001126  
003520 001401  
003522 104003  
003524

```

:*****
:*TEST 3 TEST OF BYTE ACCESS IN CSR.IDX3
:*****
TST3: SCOPE
MOV #10,$TIMES ;;DO 10. ITERATIONS
JSR PC,CNTLC ;;IS IT CONTRL C?

CLR @CSR ;CLEAR CONTENTS OF CSR
MOV CSR,R0 ;STORE ADDRESS OF CSR
MOVB #376,@R0 ;TRY TO LOAD LOW ORDER BYTE
MOV KONST,GOOD ;STORE GOOD DATA
ADD #100,GOOD
MOV @R0,BAD ;STORE BAD DATA
CMP GOOD,BAD ;COMPARE
BEQ 2$ ;BR IF OK
ERROR+3 ;LOW BYTE ACCESS FAILURE IN CSR
2$: CLR @CSR ;CLEAR CSR
MOV KONST,GOOD ;STORE GOOD DATA (PWR OK)
MOVB #377,1(R0) ;TRY LOADING HI ORDER BYTE OF CSR
MOV @R0,BAD ;STORE CONTENTS OF CSR
CMP GOOD,BAD ;COMPARE
BEQ 3$ ;BR IF OK
ERROR+3 ;HI-BYTE ACCESS FAILURE IN CSR
3$: ;GO TO NEXT TEST

```

```
732
733
(3)
(3)
(2) 003524 000004
(1) 003526 012737 000012 001172
734 003534 004737 026620
735
736 003540 005000
737 003542 012737 000000 177776
738 003550 005077 176046
739 003554 012777 003736 176042
740 003562 012737 000340 177776
741 003570 013737 001752 001124
742 003576 062737 020001 001124
743 003604 005737 001634
744 003610 001003
745 003612 062737 000400 001124
746 003620 012777 000001 175774
747 003626 017737 175770 001126
748 003634 042737 000074 001126
749 003642 023737 001124 001126
750 003650 001403
751 003652 005200
752 003654 001364
753 003656 104004
754 003660 012737 000000 177776
755 003666 013737 001752 001124
756 003674 062737 020201 001124
757 003702 005000
758 003704 017737 175712 001126
759 003712 023737 001124 001126
760 003720 001403
761 003722 005200
762 003724 001367
763 003726 104004
764 003730 005077 175666
765 003734 000406
766 003736
(2) 003736 012600
(2) 003740 012600
767 003742 017737 175654 001126
768 003750 104005
769 003752 012777 023022 175644
770
```

```
*****
*TEST 4 CONNECT TEST WITHOUT INTERRUPT ENABLE.IDX4
*****
TST4: SCOPE
MOV #10,$TIMES ;:DO 10. ITERATIONS
JSR PC,CNTLC ;:IS IT CONTROL C?

1$: CLR R0 ;:SET DELAY TIMER
MOV #0,PSW ;:LOWER PRIORITY
CLR @CSR ;:CLEAR CSR
MOV #8$,@VECO ;:SET RETURN
MOV #340,PSW ;:RISE PRIORITY
MOV KONST,GOOD ;:STORE GOOD DATA (SWB ACT,PWR OK,REQ)
ADD #20001,GOOD
TST DT03 ;:DT03 MODE?
BNE 11$ ;:BR IF DT03
ADD #400,GOOD ;:SET BRQ IF DT07
MOV #REQ,@CSR ;:SET REQ (BIT0)
2$: MOV @CSR,BAD ;:STORE CONTENTS OF CSR
BIC #74,BAD ;:MASK R00-3
CMP GOOD,BAD ;:COMPARE
BEQ 5$ ;:BR IF OK
INC R0 ;:DELAY
BNE 2$ ;:BR IF NOT DONE
ERROR+4 ;:SWB ACT (BIT13),BRQ OR REQ NOT SET
5$: MOV #0,PSW ;:LOWER PRIORITY
MOV KONST,GOOD ;:STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
ADD #20201,GOOD
CLR R0 ;:RESET DELAY
6$: MOV @CSR,BAD ;:STORE CONTENTS OF CSR
CMP GOOD,BAD ;:COMPARE
BEQ 7$ ;:BR IF OK
INC R0 ;:DELAY
BNE 6$ ;:BR IF NOT DONE
ERROR+4 ;:CON(BIT7) NOT SET
7$: CLR @CSR ;:CLEAR CSR AND DISCONNECT
BR 9$ ;:EXIT

8$: MOV (SP)+,R0 ;:POP STACK INTO R0
MOV (SP)+,R0 ;:POP STACK INTO R0
MOV @CSR,BAD ;:STORE CONTENTS OF CSR
ERROR+5 ;:INTERUPT DETECTED WITH INTERRUPT ENABLE CLEARED
9$: MOV #OOPS,@VECO ;:RESET RETURN
;:GO TO NEXT TEST
```



```

772
773      ;:*****
(3)      ;*TEST 5      RELEASE TEST WITHOUT INTERRUPT ENABLE.IDX5
(3)      ;:*****
(2) 003760 000004      TST5:  SCOPE
(1) 003762 012737 000012 001172      MOV      #10.,$TIMES      ;;DO 10. ITERATIONS
774 003770 004737 026620      JSR      PC,CNTLC      ;:IS IT CONTROL C?
775
776 003774 005000      1$:  CLR      R0      ;:SET DELAY TIMER
777 003776 012777 000001 175616      MOV      #1,@CSR      ;:CONNECT
778 004004 013737 001752 001124      MOV      KONST,GOOD      ;:STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
779 004012 062737 020201 001124      ADD      #20201,GOOD
780 004020 012737 000000 177776      MOV      #0,PSW      ;:DROP PRIORITY
781 004026 017737 175570 001126      2$:  MOV      @CSR,BAD      ;:STORE CONTENTS OF CSR
782 004034 023737 001124 001126      CMP      GOOD,BAD      ;:COMPARE
783 004042 001404      BEQ      3$      ;:BR IF CONNECTED
784 004044 005200      INC      R0      ;:DELAY
785 004046 001367      BNE      2$      ;:BR IF NOT DONE
786 004050 104006      ERROR+6      ;:COULD NOT CONNECT
787 004052 000461      BR      7$      ;:EXIT TEST
788 004054 012777 004220 175542      3$:  MOV      #8$,@VECO      ;:SET RETURN
789 004062 012737 000340 177776      MOV      #340,PSW      ;:RISE PRIORITY
790 004070 005000      CLR      R0      ;:SET DELAY TIMER
791 004072 042777 000001 175522      BIC      #REQ,@CSR      ;:CLEAR REQUEST
792 004100 013737 001752 001124      MOV      KONST,GOOD      ;:STORE GOOD DATA (SWB ACT,PWR OK,CON)
793 004106 062737 020200 001124      ADD      #20200,GOOD
794 004114 005737 001634      TST      DT03      ;:DT03 MODE?
795 004120 001003      BNE      4$      ;:BR IF YES
796 004122 062737 000400 001124      ADD      #400,GOOD      ;:ADD BRQ IF DT07
797 004130 017737 175466 001126      4$:  MOV      @CSR,BAD      ;:STORE CONTENTS OF CSR
798 004136 023737 001124 001126      CMP      GOOD,BAD      ;:COMPARE
799 004144 001403      BEQ      5$      ;:BR IF OK
800 004146 005200      INC      R0      ;:DELAY
801 004150 001367      BNE      4$      ;:BR IF NOT DONE
802 004152 104007      ERROR+7      ;:BRQ NOT SET OR REQ NOT CLEARED
803 004154 012737 000000 177776      5$:  MOV      #0,PSW      ;:DROP PRIORITY
804 004162 013737 001752 001124      MOV      KONST,GOOD      ;:STORE GOOD DATA (PWR OK)
805 004170 005000      CLR      R0      ;:RESET DELAY
806 004172 017737 175424 001126      6$:  MOV      @CSR,BAD      ;:STORE CONTENTS OF CSR
807 004200 023737 001124 001126      CMP      GOOD,BAD      ;:COMPARE
808 004206 001403      BEQ      7$      ;:BR IF OK
809 004210 005200      INC      R0      ;:DELAY
810 004212 001367      BNE      6$      ;:BR IF NOT DONE
811 004214 104007      ERROR+7      ;:CON OR SWB ACT NOT CLEARED
812 004216 000406      7$:  BR      9$
813 004220      8$:
(2) 004220 012600      MOV      (SP)+,R0      ;:POP STACK INTO R0
(2) 004222 012600      MOV      (SP)+,R0      ;:POP STACK INTO R0
814 004224 017737 175372 001126      MOV      @CSR,BAD      ;:STORE CONTENTS OF CSR
815 004232 104005      ERROR+5      ;:INTERUPT DETECTED WITH INTERRUPT ENABLE CLRD
816 004234 012777 023022 175362      9$:  MOV      #OOPS,@VECO      ;:RESET RETURN
817      ;GO TO NEXT TEST

```

```

819      ;:*****
(3)      ;*TEST 6      CONNECT FAILURE TEST WITHOUT INTERRUPT ENABLE.IDX6
(3)      ;:*****
(2) 004242 000004
(1) 004244 012737 000012 001172
820 004252 004737 026620
821
822 004256 005000
823 004260 012737 000000 177776
824 004266 005077 175330
825 004272 012777 004524 175324
826 004300 012737 000340 177776
827 004306 013737 001752 001124
828 004314 062737 020001 001124
829 004322 005737 001634
830 004326 001003
831 004330 062737 000400 001124
832 004336 012777 000001 175256
833 004344 017737 175252 001126
834 004352 042737 000074 001126
835 004360 023737 001124 001126
836 004366 001403
837 004370 005200
838 004372 001364
839 004374 104013
840 004376 005737 001634
841 004402 001024
842 004404 013737 001752 001124
843 004412 062737 020401 001124
844 004420 005000
845 004422 017737 175174 001126
846 004430 042737 000074 001126
847 004436 023737 001124 001126
848 004444 001403
849 004446 005200
850 004450 001364
851 004452 104013
852 004454 013737 001752 001124
853 004462 062737 100000 001124
854 004470 005000
855 004472 017737 175124 001126
856 004500 023737 001124 001126
857 004506 001403
858 004510 005200
859 004512 001367
860 004514 104013
861 004516 005077 175100
862 004522 000407
863 004524
(2) 004524 012600
(2) 004526 012600
864 004530 017737 175066 001126
865 004536 104005
866 004540 000766
867 004542 012777 023022 175054
868

```

```

TST6: SCOPE
MOV #10, $TIMES      ;;DO 10. ITERATIONS
JSR PC, CNTLC       ;IS IT CONTROL C?

1$: CLR R0           ;SET DELAY TIMER
MOV #0, PSW         ;LOWER PRIORITY
CLR @CSR           ;CLEAR CSR
MOV #8$, @VECO     ;SET RETURN
MOV #340, PSW      ;RISE PRIORITY
MOV KONST, GOOD    ;STORE GOOD DATA (SWB ACT, PWR OK, REQ)
ADD #20001, GOOD
TST DT03           ;DT03 MODE?
BNE 11$           ;BR IF DT03
ADD #400, GOOD     ;ADD BRQ IF DT07
11$: MOV #REQ, @CSR ;SET REQ (BIT0)
2$: MOV @CSR, BAD   ;STORE CONTENTS OF CSR
BIC #74, BAD       ;MASK RQ0-3
CMP GOOD, BAD      ;COMPARE
BEQ 3$            ;BR IF OK
INC R0             ;DELAY
BNE 2$            ;BR IF NOT DONE
ERROR+13          ;SWB ACT (BIT13) OR REQ NOT SET
3$: TST DT03       ;DT03 MODE?
BNE 5$            ;BR IF YES
MOV KONST, GOOD  ;STORE GOOD DATA (SWB ACT, PWR OK, BRQ, REQ)
ADD #20401, GOOD
CLR R0           ;RESET DELAY
4$: MOV @CSR, BAD ;STORE CONTENTS OF CSR
BIC #74, BAD     ;MASK RQ0-3
CMP GOOD, BAD   ;COMPARE
BEQ 5$          ;BR IF OK
INC R0          ;DELAY
BNE 4$          ;BR IF NOT DONE
ERROR+13       ;BRQ (BIT8) DID NOT SET
5$: MOV KONST, GOOD ;STORE GOOD DATA (TMO, PWR OK)
ADD #100000, GOOD
CLR R0         ;RESET DELAY
6$: MOV @CSR, BAD ;STORE CONTENTS OF CSR
CMP GOOD, BAD ;COMPARE
BEQ 7$        ;BR IF OK
INC R0        ;DELAY
BNE 6$        ;BR IF NOT DONE
ERROR+13     ;TMO (BIT15) NOT SET
7$: CLR @CSR  ;CLEAR CSR AND DISCONNECT
BR 9$        ;EXIT
8$: MOV (SP)+, R0 ;;POP STACK INTO R0
MOV (SP)+, R0 ;;POP STACK INTO R0
MOV @CSR, BAD ;STORE CONTENTS OF CSR
ERROR+5      ;INTERUPT DETECTED WITH INTERRUPT ENABLE CLEARED
BR 7$        ;GO CLEAR CSR
9$: MOV #OOPS, @VECO ;RESET RETURN
;GO TO NEXT TEST

```

```
870
871      ;:*****
(3)      ;*TEST 7      RELEASE TEST WITHOUT BUS MASTERSHIP OR INTERRUPT ENABLE.IDX7
(3)      ;:*****
(2) 004550 000004      TST7: SCOPE
(1) 004552 012737 000012 001172      MOV #10.,$TIMES      ;:DO 10. ITERATIONS
872 004560 004737 026620      JSR PC,CNTLC      ;:IS IT CONTROL C?
873
874 004564 005000      1$: CLR R0      ;:SET DELAY TIMER
875 004566 012777 000001 175026      MOV #1,@CSR      ;:CONNECT
876 004574 013737 001752 001124      MOV KONST,GOOD      ;:STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
877 004602 062737 020201 001124      ADD #20201,GOOD
878 004610 012737 000000 177776      MOV #0,PSW      ;:DROP PRIORITY
879 004616 017737 175000 001126      2$: MOV @CSR,BAD      ;:STORE CONTENTS OF CSR
880 004624 023737 001124 001126      CMP GOOD,BAD      ;:COMPARE
881 004632 001404      BEQ 3$      ;:BR IF CONNECTED
882 004634 005200      INC R0      ;:DELAY
883 004636 001367      BNE 2$      ;:BR IF NOT DONE
884 004640 104006      ERROR+6      ;:COULD NOT CONNECT
885 004642 000461      BR 7$      ;:EXIT TEST
886 004644 012777 005010 174752      3$: MOV #8$,@VECO      ;:SET RETURN
887 004652 012737 000340 177776      MOV #340,PSW      ;:RISE PRIORITY
888 004660 005000      CLR R0      ;:SET DELAY TIMER
889 004662 042777 000001 174732      BIC #REQ,@CSR      ;:CLEAR REQUEST
890 004670 013737 001752 001124      MOV KONST,GOOD      ;:STORE GOOD DATA (SWB ACT,PWR OK,CON)
891 004676 062737 020200 001124      ADD #20200,GOOD
892 004704 005737 001634      TST DT03      ;:DT03 MODE?
893 004710 001003      BNE 4$      ;:BR IF YES
894 004712 062737 000400 001124      ADD #400,GOOD      ;:ADD BRQ IF DT07
895 004720 017737 174676 001126      4$: MOV @CSR,BAD      ;:STORE CONTENTS OF CSR
896 004726 023737 001124 001126      CMP GOOD,BAD      ;:COMPARE
897 004734 001403      BEQ 5$      ;:BR IF OK
898 004736 005200      INC R0      ;:DELAY
899 004740 001367      BNE 4$      ;:BR IF NOT DONE
900 004742 104014      ERROR+14      ;:BRQ NOT SET OR REQ NOT CLEARED
901 004744 013737 001752 001124      5$: MOV KONST,GOOD      ;:STORE GOOD DATA (TMO,PWR OK)
902 004752 062737 100000 001124      ADD #100000,GOOD
903 004760 005000      CLR R0      ;:RESET DELAY
904 004762 017737 174634 001126      6$: MOV @CSR,BAD      ;:STORE CONTENTS OF CSR
905 004770 023737 001124 001126      CMP GOOD,BAD      ;:COMPARE
906 004776 001403      BEQ 7$      ;:BR IF OK
907 005000 005200      INC R0      ;:DELAY
908 005002 001367      BNE 6$      ;:BR IF NOT DONE
909 005004 104014      ERROR+14      ;:TMO NOT SET
910 005006 000406      7$: BR 9$      ;:EXIT
911 005010      8$:
(2) 005010 012600      MOV (SP)+,R0      ;:POP STACK INTO R0
(2) 005012 012600      MOV (SP)+,R0      ;:POP STACK INTO R0
912 005014 017737 174602 001126      MOV @CSR,BAD      ;:STORE CONTENTS OF CSR
913 005022 104005      ERROR+5      ;:INTERUPPT DETECTED WITH INTERRUPT ENABLE CLRD
914 005024 012777 023022 174572      9$: MOV #OOPS,@VECO      ;:RESET RETURN
915      ;:GO TO NEXT TEST
```

```
917
918
(3)
(3)
(2) 005032 000004
(1) 005034 012737 000012 001172
919 005042 004737 026620
920
921 005046 005077 174550
922 005052 012737 000340 177776
923 005060 013737 001630 001124
924 005066 012737 000340 001126
925 005074 012777 005150 174522
926 005102 012777 000101 174512
927 005110 032777 020000 174504
928 005116 001774
929 005120 162737 000040 177776 1$:
930 005126 000240
931 005130 000240
932 005132 000240
933 005134 162737 000040 001126
934 005142 001366
935 005144 104010
936 005146 000425
937 005150 2$:
(2) 005150 012600
(2) 005152 012600
938 005154 023737 001124 001126
939 005162 001401
940 005164 104011
941 005166 017737 174430 001126 3$:
942 005174 013737 001752 001124
943 005202 062737 020301 001124
944 005210 023737 001124 001126
945 005216 001401
946 005220 104012
947 005222 005077 174374 4$:
948 005226 005037 177776
949 005232 012777 023022 174364 5$:
950
```

```
*****
:*TEST 10 CONNECT TEST.IDX8
*****
TST10: SCOPE
MOV #10, $TIMES ;;DO 10. ITERATIONS
JSR PC, CNTLC ;;IS IT CONTROL C?
CLR @CSR ;;CLEAR CSR
MOV #340, PSW ;;RISE PRIORITY
MOV PRI, GOOD ;;STORE PRIORITY LEVEL
MOV #340, BAD ;;STORE ACTUAL PRIORITY INDICATOR
MOV #2$, @VECO ;;SET RETURN
MOV #101, @CSR ;;SET I.E. AND REQ
BIT #20000, @CSR ;;WAIT FOR SWB ACT TO APPEAR
BEQ #-6 ;;BR IF NOT THERE
SUB #40, PSW ;;LOWER PRIORITY UNTIL INTERUPTED
NOP
NOP
SUB #40, BAD ; " " INDICATOR
BNE 1$ ;;BR UNTIL DONE
ERROR+10 ;;NO INTERUPT DETECTED
BR 4$ ;;EXIT
2$:
MOV (SP)+, R0 ;;POP STACK INTO R0
MOV (SP)+, R0 ;;POP STACK INTO R0
CMP GOOD, BAD ;;CORRECT BREAK LEVEL?
BEQ 3$ ;;BR IF OK
ERROR+11 ;;INTERUPT AT WRONG PRIORITY
MOV @CSR, BAD ;;STORE CONTENTS OF CSR
MOV KONST, GOOD ;;STORE GOOD DATA (SWB ACT, PWR OK, CON, IE, REQ)
ADD #20301, GOOD
CMP GOOD, BAD ;;COMPARE
BEQ 4$ ;;BR IF OK
ERROR+12 ;;CONNECT CONDITION FAILURE
CLR @CSR ;;CLEAR CSR
CLR PSW ;;LOWER PRIORITY
MOV #OOPS, @VECO ;;RESET RETURN
;;GO TO NEXT TEST
```

```

952
953      ;*****
(3)      ;*TEST 11      RELEASE TEST .IDX9
(3)      ;*****
(2) 005240 000004      TST11: SCOPE
(1) 005242 012737 000012 001172      MOV      #10, $TIMES      ;;DO 10. ITERATIONS
954 005250 004737 026620      JSR      PC,CNTLC      ;IS IT CONTROL C?
955
956 005254 005000      1$: CLR      R0      ;SET DELAY TIMER
957 005256 012777 000001 174336      MOV      #1,@CSR      ;CONNECT
958 005264 013737 001752 001124      MOV      KONST,GOOD
959 005272 062737 020201 001124      ADD      #20201,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
960 005300 012737 000000 177776      MOV      #0,PSW      ;DROP PRIORITY
961 005306 017737 174310 001126      2$: MOV      @CSR,BAD      ;STORE CONTENTS OF CSR
962 005314 023737 001124 001126      CMP      GOOD,BAD      ;COMPARE
963 005322 001404      BEQ      3$      ;BR IF CONNECTED
964 005324 005200      INC      R0      ;DELAY
965 005326 001367      BNE      2$      ;BR IF NOT DONE
966 005330 104006      ERROR+6      ;COULD NOT CONNECT
967 005332 000422      BR      7$      ;EXIT TEST
968 005334 005737 001634      3$: TST      DT03      ;DT03 MODE?
969 005340 001420      BEQ      8$      ;BR=NO
970 005342 005000      CLR      R0      ;SET DELAY TIMER
971 005344 042777 000001 174250      BIC      #REQ,@CSR      ;CLEAR REQUEST
972 005352 013737 001752 001124      5$: MOV      KONST,GOOD      ;STORE GOOD DATA (PWR OK)
973 005360 017737 174236 001126      6$: MOV      @CSR,BAD      ;STORE CONTENTS OF CSR
974 005366 023737 001124 001126      CMP      GOOD,BAD      ;COMPARE
975 005374 001401      BEQ      7$      ;BR IF OK
976 005376 104015      ERROR+15      ;RELEASE CONDITIONS NOT MET
977 005400 000446      BR      10$      ;GO TO NEXT TEST
978 005402 012777 005456 174214      7$: MOV      #9$,@VECO      ;SET RETURN
979 005410 052777 000100 174204      8$: BIS      #100,@CSR      ;SET I.E.
980 005416 042777 000001 174176      BIC      #1,@CSR      ;CLR REQ.
981 005424 004737 022010      JSR      PC,WAIT05      ;GO WAIT
982 005430 013737 001752 001124      MOV      KONST,GOOD      ;LOAD GOOD
983 005436 062737 000100 001124      ADD      #100,GOOD
984 005444 017737 174152 001126      MOV      @CSR,BAD
985 005452 104010      ERROR+10      ;NO INTERRUPT
986 005454 000420      BR      10$      ;EXIT
987 005456 012706 001100      9$: MOV      #STACK,SP      ;RESET STACK
988 005462 013737 001752 001124      MOV      KONST,GOOD      ;LOAD GOOD
989 005470 062737 000100 001124      ADD      #100,GOOD
990 005476 017737 174120 001126      MOV      @CSR,BAD
991 005504 023737 001124 001126      CMP      GOOD,BAD
992 005512 001401      BEQ      10$
993 005514 104015      ERROR+15
994 005516 005077 174100      10$: CLR      @CSR      ;CLEAR AND DISCONNECT
995 005522 012777 023022 174074      MOV      #OOPS,@VECO      ;RESET RETURN
  
```

```
997
998
(3)
(3)
(2) 005530 000004
(1) 005532 012737 000012 001172
999 005540 004737 026620
1000
1001 005544 005077 174052
1002 005550 012737 000340 177776
1003 005556 005000
1004 005560 012777 005672 174036
1005 005566 013737 001752 001124
1006 005574 062737 100100 001124
1007 005602 005737 001634
1008 005606 001003
1009 005610 062737 000400 001124
1010 005616 012777 000101 173776 6$:
1011 005624 017737 173772 001126 1$:
1012 005632 023737 001124 001126
1013 005640 001404
1014 005642 005200
1015 005644 001367
1016 005646 104016
1017 005650 000414
1018 005652 005000
1019 005654 012737 000000 177776 11$:
1020 005662 005200 22$:
1021 005664 001376
1022 005666 104010
1023 005670 000404
1024 005672 2$:
(2) 005672 012600
(2) 005674 012600
1025 005676 005077 173720 4$:
1026 005702 012777 023022 173714 5$:
1027
```

```
*****
*TEST 12 CONNECT FAILURE TEST.IDX10
*****
TST12: SCOPE
MOV #10, $TIMES ;;DO 10. ITERATIONS
JSR PC,CNTLC ;IS IT CONTROL C?
CLR @CSR ;CLEAR CSR
MOV #340,PSW ;RISE PRIORITY
CLR RO ;SET DELAY TIMER
MOV #2$,@VECO ;SET RETURN
MOV KONST,GOOD
ADD #100100,GOOD ;STORE GOOD DATA (TMO,PWR OK,IE)
TST DT03 ;DT03 MODE?
BNE 6$ ;BR IF DT03
ADD #400,GOOD ;ADD BRQ IF DT07
MOV #101,@CSR ;SET I.E. AND REQ
MOV @CSR,BAD ;STORE CONTENTS OF CSR
CMP GOOD,BAD ;COMPARE
BEQ 11$ ;BR IF OK
INC RO ;DELAY
BNE 1$ ;BR IF NOT DONE
ERROR+16 ;TMO NOT SET
BR 5$ ;EXIT
CLR RO ;RESET DELAY
MOV #0,PSW ;DROP PRIORITY
INC RO ;DELAY
BNE 22$ ;BR IF NOT DONE
ERROR+10 ;NO INTERRUPT DETECTED
BR 5$ ;EXIT
MOV (SP)+,RO ;;POP STACK INTO RO
MOV (SP)+,RO ;;POP STACK INTO RO
CLR @CSR ;CLEAR CSR
MOV #0UPS,@VECO ;RESET RETURN
;GO TO NEXT TEST
```

```

1029
1030      ;:*****
(3)      ;*TEST 13      RELEASE TEST WITHOUT BUSS MASTERSHIP.IDX11
(3)      ;:*****
(2) 005710 000004      TST13: SCOPE
(1) 005712 012737 000012 001172      MOV      #10.,$TIMES      ;;DO 10. ITERATIONS
1031 005720 004737 026620      JSR      PC,CNTLC      ;IS IT CONTROL C?
1032
1033 005724 005000      1$: CLR      R0      ;SET DELAY TIMER
1034 005726 012777 000001 173666      MOV      #1,@CSR      ;CONNECT
1035 005734 013737 001752 001124      MOV      KONST,GOOD
1036 005742 062737 020201 001124      ADD      #20201,GOOD      ;STORE GOOD DATA (SWB ACT,PWR OK,CON,REQ)
1037 005750 012737 000000 177776      MOV      #0,PSW      ;DROP PRIORITY
1038 005756 017737 173640 001126      2$: MOV      @CSR,BAD      ;STORE CONTENTS OF CSR
1039 005764 023737 001124 001126      CMP      GOOD,BAD      ;COMPARE
1040 005772 001404      BEQ      3$      ;BR IF CONNECTED
1041 005774 005200      INC      R0      ;DELAY
1042 005776 001367      BNE      2$      ;BR IF NOT DONE
1043 006000 104006      ERROR+6      ;COULD NOT CONNECT
1044 006002 000461      BR      7$      ;EXIT TEST
1045 006004 012777 006142 173612      3$: MOV      #6$,@VECO      ;SET RETURN
1046 006012 012737 000340 177776      MOV      #340,PSW      ;RISE PRIORITY
1047 006020 005000      CLR      R0      ;SET DELAY TIMER
1048 006022 012777 000100 173572      MOV      #100,@CSR      ;CLEAR REQUEST AND SET IE
1049 006030 017737 173566 001126      4$: MOV      @CSR,BAD      ;STORE CONTENTS OF CSR
1050 006036 013737 001752 001124      MOV      KONST,GOOD
1051 006044 062737 100100 001124      ADD      #100100,GOOD      ;STORE GOOD DATA(TMO,PWR OK,IE)
1052 006052 005737 001634      TST      DT03      ;DT03 MODE?
1053 006056 001003      BNE      41$      ;BR IF DT03
1054 006060 062737 000400 001124      ADD      #400,GOOD      ;ADD BRQ IF DT07
1055 006066 023737 001124 001126      41$: CMP      GOOD,BAD      ;COMPARE
1056 006074 001404      BEQ      5$      ;BR IF OK
1057 006076 005200      INC      R0      ;DELAY
1058 006100 001353      BNE      4$      ;BR IF NOT DONE
1059 006102 104016      ERROR+16      ;TMO NOT SET
1060 006104 000420      BR      7$      ;EXIT
1061 006106 013737 001752 001124      5$: MOV      KONST,GOOD
1062 006114 062737 100100 001124      ADD      #100100,GOOD      ;STORE GOOD DATA (TMO,PWR OK,IE)
1063 006122 005000      CLR      R0      ;RESET DELAY
1064 006124 012737 000000 177776      MOV      #0,PSW      ;DROP PRIORITY
1065 006132 005200      51$: INC      R0      ;DELAY
1066 006134 001376      BNE      51$      ;BR IF NOT DONE
1067 006136 104010      ERROR+10      ;NO INTERRUPT
1068 006140 000402      BR      7$      ;EXIT
1069 006142      6$:
(2) 006142 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
(2) 006144 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
1070 006146 012777 023022 173450      7$: MOV      #OOPS,@VECO      ;RESET RETURN
1071      ;GO TO NEXT TEST
  
```

```

1073
1074
1075
(3)
(3)
(2) 006154 000004
(1) 006156 012737 000012 001172
1076 006164 004737 026620
1077
1078 006170 005737 001214
1079 006174 001562
1080 006176 005737 001754
1081 006202 100557
1082 006204 005737 001636
1083 006210 001554
1084 006212 005737 001642
1085 006216 001551
1086 006220 012737 000000 177776
1087 006226 012777 000001 173366
1088 006234 105777 173362 1$:
1089 006240 100417
1090 006242 005777 173354
1091 006246 100372
1092 006250 017737 173346 001126
1093 006256 013737 001752 001124
1094 006264 062737 020201 001124
1095 006272 104006
1096 006274 000137 006542
1097
1098 006300 004737 026562 2$:
1099 006304 005737 026560
1100 006310 001403
1101 006312 104027
1102 006314 000137 006542
1103 006320 20$:
1104 006320 013777 001640 173310
1105 006326 017737 173304 001126
1106 006334 005137 001640
1107 006340 043737 001640 001126
1108 006346 005137 001640
1109 006352 023737 001640 001126
1110 006360 001402
1111 006362 104017
1112 006364 000466
1113 006366 052777 001000 173226 3$:
1114 006374 012737 000002 022062
1115 006402 004737 022022
1116 006406 032777 001000 173206 31$:
1117 006414 001430
1118 006416 104401 006424
(1) 006422 000424
(1)
(1) 006474
1119 006474 000000
1120 006476 017737 173134 001126 4$:
1121 006504 033737 001642 001126

```

```

*****
: *TEST 14      RESET (RST) TEST.IDX12
*****
TST14: SCOPE
MOV      #10, $TIMES      ;;DO 10. ITERATIONS
JSR      PC, CNTLC       ;IS IT CONTROL C?

TST      $PASS           ;FIRST PASS?
BEQ      5$              ;SKIP TEST ON FIRST PASS
TST      DEVCON          ;RESET IN NEUTRAL?
BMI      5$              ;BR=NO RESET IN NEUTRAL
TST      DEVAD           ;IS A DEVICE REGISTER PRESENT?
BEQ      5$              ;EXIT IF NOT
TST      DEVRC          ;RESETABLE BITS PRESENT?
BEQ      5$              ;BR=NO SO EXIT
MOV      #0, PSW         ;DROP PRIORITY
MOV      #REQ, @CSR      ;CONNECT
TSTB    @CSR             ;CONNECTED?
BMI      2$              ;BR IF YES
TST      @CSR            ;TIMEOUT?
BPL      1$              ;BR IF NOT
MOV      @CSR, BAD       ;STORE CONTENTS OF CSR
MOV      KONST, GOOD     ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
ADD      #20201, GOOD    ;COULDN'T CONNECT
JMP      5$

2$:      JSR      PC, CKADR ;GO CHECK DEVICE ADDRESS EXIST?
TST      FLAG           ;IF FLAG=1, TRAPPED
BEQ      20$            ;
ERROR+27
JMP      5$              ;EXIT

20$:     MOV      ., DEVRW, @DEVAD ;SET READ/WRITE BITS IN EXTERNAL DEVICE REGISTER
MOV      @DEVAD, BAD    ;STORE CONTENTS OF DEVAD
COM      DEVRW          ;COMPLEMENT R/W BITS
BIC      DEVRW, BAD     ;CLEAR OUT BITS NOT UNDER TEST
COM      DEVRW          ;RESTORE ORIGINAL PATTERN
CMP      DEVRW, BAD     ;DID THEY SET?
BEQ      3$              ;BR IF YES
ERROR+17
BR       5$              ;COULDN'T SET READ/WRITE BITS IN DEVAD
;EXIT

3$:      BIS      #1000, @CSR ;SET RST
MOV      #2, TOCK       ;SET UP FOR 22MSEC COUNT
JSR      PC, WAIT

31$:    BIT      #1000, @CSR ;STILL SET?
BEQ      4$              ;BR=NO
TYPE     , 55$          ;;TYPE ASCIZ STRING
BR       64$            ;;GET OVER THE ASCIZ

65$:    .ASCIZ <200>/ERROR! RST(BIT9) DIDN'T CLEAR IN TIME./
64$:

4$:      HALT
MOV      @DEVAD, BAD    ;STORE CONTENTS OF DEVAD
BIT      DEVRC, BAD     ;RESET CONDITIONS MET?

```



.MAIN. MACY11 30A(1052) 16-NOV-81 17:26 PAGE 12-27  
CRDTAB.P11 16-NOV-81 17:16 T14 RESET (RST) TEST.IDX12

SEQ 0040

1122	006512	001413			BEQ	5\$		:BR IF YES
1123	006514	005200			INC	R0		:WAIT AWHILE
1124	006516	001367			BNE	4\$		
1125	006520	013737	001642	001124	MOV	DEVRC,GOOD		:SAVE DEVRC
1126	006526	005137	001124		COM	GOOD		
1127	006532	043737	001124	001126	BIC	GOOD,BAD		:HI LITE BAD BITS
1128	006540	104021			ERROR+21			:RESET (RST) DIDN'T WORK
1129	006542	005077	173054		CLR	@CSR		:SHUT DOWN AND GO TO NEXT TEST

5\$:



1156

1157

1158

(3)

(3)

(2)

(1)

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

(3)

(3)

(2)

(1)

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

006702	000004		
006704	012737	000012	001172
006712	004737	026620	
006716	012701	001750	
006722	012737	000000	177776
006730	012777	000001	172664
006736	105777	172660	
006742	100416		
006744	005777	172652	
006750	100372		
006752	017737	172644	001126
006760	013737	001752	001124
006766	062737	020201	001124
006774	104006		
006776	000422		
007000	042777	000001	172614
007006	013737	001752	001124
007014	005000		
007016	017737	172600	001126
007024	023737	001124	001126
007032	001402		
007034	104015		
007036	000402		
007040	005301		
007042	001327		
007044	005077	172552	
007050	000004		
007052	012737	000001	001172
007060	005737	001636	
007064	001012		
007066	000137	007540	
007072			
007072	004737	026562	
007076	005737	026560	
007102	001437		
007104	104027		
007106	000137	007540	
007112	005737	001642	
007116	001002		
007120	000137	007540	
007124	012737	000000	177776
007132	012777	000001	172462

```

:*****
:*TEST 16      HI-SPEED SWITCHING TEST.IDX14
:*****

```

```

TST16:  SCOPE
        MOV      #10.,$TIMES      ;;DO 10. ITERATIONS
        JSR      PC,CNTLC        ;IS IT CONTROL C?

        MOV      #1000.,R1       ;SET COUNTER
        MOV      #0,PSW          ;DROP PRIORITY
        MOV      #REQ,@CSR       ;CONNECT
        TSTB     @CSR            ;CONNECTED?
        BMI      2$              ;BR IF YES
        TST      @CSR            ;TIMEOUT?
        BPL      1$              ;BR IF NOT
        MOV      @CSR,BAD        ;STORE CONTENTS OF CSR
        MOV      KONST,GOOD      ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
        ADD      #20201,GOOD     ;COULDN'T CONNECT
        ERROR+6
        BR       5$              ;EXIT
        BIC      #1,@CSR         ;RELEASE CONNECTION
        MOV      KONST,GOOD      ;STORE GOOD DATA(PWR OK)
        CLR      R0              ;SET DELAY TIMER
        MOV      @CSR,BAD        ;STORE CONTENTS OF CSR
        CMP      GOOD,BAD        ;COMPARE
        BEQ      4$              ;BR IF OK
        ERROR+15
        BR       5$              ;RELEASE CONDITIONS NOT MET
        BR       5$              ;EXIT
        DEC      R1              ;COUNT A SUBPASS
        BNE      11$             ;BR IF CONTINUE
        CLR      @CSR            ;SHUTDOWN AND GO TO NEXT TEST

```

```

:*****
:*TEST 17      RESET-IN-NEUTRAL TEST.IDX15
:*****

```

```

TST17:  SCOPE
        MOV      #1,$TIMES      ;;DO 1 ITERATION

        TST      DEVAD           ;IS A DEVICE REGISTER PRESENT?
        BNE      15$             ;CONTINUE TEST IF PRESENT
        JMP      12$             ;GO EXIT

        JSR      PC,CKADR        ;CHECK IF DEVICE ADDRESS EXIST?
        TST      FLAG            ;IF FLAG=1;TRAP OCCURED
        BEQ      2$              ;BR:NO TRAP
        ERROR+27
        JMP      12$             ;GO EXIT
        TST      DEVRC           ;ANY RESETABLE BITS?
        BNE      14$             ;BR=YES
        JMP      12$             ;GO EXIT
        MOV      #0,PSW          ;DROP PRIORITY
        MOV      #REQ,@CSR       ;CONNECT

```

```

1204 007140 105777 172456 1$: TSTB @CSR ;CONNECTED?
1205 007144 100752 BMI 13$ ;BR IF YES
1206 007146 005777 172450 TST @CSR ;TIMEOUT?
1207 007152 100372 BPL 1$ ;BR IF NOT
1208 007154 017737 172442 001126 MOV @CSR,BAD ;STORE CONTENTS OF CSR
1209 007162 013737 001752 001124 MOV KONST,GOOD
1210 007170 062737 020201 001124 ADD #20201,GOOD ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
1211 007176 104006 ERROR+6 ;COULDN'T CONNECT
1212 007200 000557 BR 12$ ;EXIT
1213 007202 2$:
1214 007202 013777 001640 172426 MOV DEVRW,@DEVAD ;SET READ/WRITE BITS IN EXTERNAL DEVICE REGISTER
1215 007210 017737 172422 001126 MOV @DEVAD,BAD ;STORE CONTENTS OF DEVAD
1216 007216 005137 001640 COM DEVRW ;COMPLEMENT R/W BITS
1217 007222 043737 001640 001126 BIC DEVRW,BAD ;CLEAR OUT BITS NOT UNDER TEST
1218 007230 005137 001640 COM DEVRW ;RESTORE ORIGINAL PATTERN
1219 007234 023737 001640 001126 CMP DEVRW,BAD ;DID THEY SET?
1220 007242 001402 BEQ 3$ ;BR IF YES
1221 007244 104017 ERROR+17 ;COULDN'T SET READ/WRITE BITS IN DEVAD
1222 007246 000534 BR 12$ ;EXIT
1223 007250 005077 172346 3$: CLR @CSR ;DISCONNECT
1224 007254 105777 172342 4$: TSTB @CSR ;DONE?
1225 007260 100775 BMI 4$ ;BR=NO
1226 007262 012737 000000 177776 MOV #0,PSW ;DROP PRIORITY
1227 007270 012777 000001 172324 MOV #REQ,@CSR ;CONNECT
1228 007276 105777 172320 5$: TSTB @CSR ;CONNECTED?
1229 007302 100416 BMI 6$ ;BR IF YES
1230 007304 005777 172312 TST @CSR ;TIMEOUT?
1231 007310 100372 BPL 5$ ;BR IF NOT
1232 007312 017737 172304 001126 MOV @CSR,BAD ;STORE CONTENTS OF CSR
1233 007320 013737 001752 001124 MOV KONST,GOOD
1234 007326 062737 020201 001124 ADD #20201,GOOD ;STORE GOOD DATA(SWB ACT,PWR OK,CON,REQ)
1235 007334 104006 ERROR+6 ;COULDN'T CONNECT
1236 007336 000500 BR 12$ ;EXIT
1237 007340 017737 172272 001126 6$: MOV @DEVAD,BAD ;STORE CONTENTS OF DEVAD
1238 007346 033737 001642 001126 BIT DEVRC,BAD ;RESET CONDITIONS MET?
1239 007354 001403 BEQ 7$ ;BR=YES
1240 007356 012702 177777 MOV #-1,R2 ;SET R2 FOR NO RESET
1241 007362 000401 BR 8$
1242 007364 005002 7$: CLR R2 ;SET R2 FOR RESET
1243 007366 005737 001214 8$: TST $PASS ;FIRST PASS?
1244 007372 001056 BNE 11$ ;BR=NO
1245 007374 005702 TST R2 ;CHECK RESET CONDITION
1246 007376 001427 BEQ 9$ ;BR=RESET
1247 007400 104401 007406 TYPE ,65$ ;:TYPE ASCIZ STRING
(1) 007404 000423 BR 64$ ;:GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>/THIS DT07 INHIBITS RESET-IN-NEUTRAL./
(1) 64$:
1248 007454 000422 BR 10$
1249 007456 9$:
(1) 007456 104401 007464 TYPE ,67$ ;:TYPE ASCIZ STRING
(1) 007462 000417 BR 66$ ;:GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <200>/THIS DT07 RESETS IN NEUTRAL./
(1) 66$:
1250 007522 010237 001754 10$: MOV R2,DEVCON ;STORE CONDITION
1251 007526 000404 BR 12$
1252 007530 020237 001754 11$: CMP R2,DEVCON ;HAS IT CHANGED?

```

.MAIN. MACY11 30A(1052) 16-NOV-81 17:26 PAGE 12-31  
CRDTAB.P11 16-NOV-81 17:16 T17 RESET-IN-NEUTRAL TEST.IDX15

SEQ 0044

1253	007534	001401		BEQ	12\$	:BR=NO
1254	007536	104022		ERROR+22		:RESET-IN-NEUTRAL OPERATION HAS CHANGED
1255	007540	005077	172056	12\$: CLR	@CSR	:CLEAR AND DISCONNECT
1256						
1257						

```
1259  
1260  
1261 007544 012737 002526 021652 MONEND: MOV #RESTRT,$RTNAD ;SET RETURN  
1262 007552 000137 021552 JMP $EOP ;GO DO END PASS  
1263
```

```

1265          .SBTTL
1266
1267          .SBTTL  MULTIPORT TEST
1268
1269          MULPRT:
1270 007556 004737 026620      JSR    PC,CNTLC      ;IS IT CONTROL C?
1271 007562 005737 026426      TST    PARFLG       ;CHECK FOR PARAMETERS
1272 007566 100404              BMI    1$           ;BR=YES
1273 007570 004737 026620      JSR    PC,CNTLC     ;IS IT CONTROL C?
1274 007574 004737 024756      JSR    PC,PARIN     ;GO INPUT PARAMETERS
1275 007600 012737 001000 001726 1$:  MOV    #1000,CYCLE  ;SET COUNTER
1276 007606 005737 001754      TST    DEVCON       ;RESET IN NEUTRAL OPTION USED?
1277 007612 001003              BNE    2$           ;BR=NO
1278 007614 005037 001756      CLR    TRIN         ;SET FLAG FOR TEST
1279 007620 000403              BR     2$+6         ;CONTINUE
1280 007622 012737 177777 001756 2$:  MOV    #-1,TRIN     ;SET FLAG FOR NO TEST
1281 007630 005737 001634      TST    DT03         ;DT03 MODE?
1282 007634 001404              BEQ    3$           ;BR=DT07 MODE
1283 007636 004737 026620      JSR    PC,CNTLC     ;IS IT CONTROL C?
1284 007642 000137 012762      JMP    MULDT3       ;GO DO DT03 MULTIPORT
1285 007646 012777 023022 171750 3$:  MOV    #00PS,@VECO  ;SET RETURN
1286 007654 012777 000340 171744      MOV    #340,@VEC2   ;SET LEVEL
1287 007662 004737 022320      JSR    PC,SEQMAK    ;GO CONSTRUCT SEQUENCER
1288 007666 004737 022666      JSR    PC,SEQINT    ;INIT. SEQUENCER
1289 007672 012737 000000 177776      MOV    #0,PSW       ;DROP PRIORITY
1290 007700 023705 001644      CMP    PORTNO,R5    ;MASTER OR SLAVE?
1291 007704 001072              BNE    SLAVE        ;BR=SLAVE
1292 007706 012737 000632 022062      MOV    #632,TOCK    ;SET CLOCK FOR 5 SEC
1293 007714 004737 022022      JSR    PC,WAIT
1294 007720 000407              BR     MASTER       ;GO TO MASTER CODE AFTER SYNC WAIT
1295 007722 023705 001644      DISPAT: CMP    PORTNO,R5 ;MY TURN AS MASTER?
1296 007726 001404              BEQ    MASTER       ;BR=YES
1297 007730 004737 026620      JSR    PC,CNTLC     ;IS IT CONTROL C?
1298 007734 000137 010072      JMP    SLAVE        ;GO BE SLAVE
1299
1300          MASTER:
1300 007740 004737 026620      JSR    PC,CNTLC     ;IS IT CONTROL C?
1301 007744 004737 022430      JSR    PC,MASMEN    ;GO MAKE MASTER MENU
1302 007750 012737 000001 022062      MOV    #1,TOCK
1303 007756 004737 022022      JSR    PC,WAIT
1304 007762 004737 021674      JSR    PC,CONNEC    ;GO CONNECT
1305 007766 005037 001722 1$:  CLR    CURFUN       ;INSURE NO EXTRANEIOUS BITS PRESENT
1306 007772 116437 000001 001722      MOVB  1(R4),CURFUN ;GET CURRANT FUNCTION
1307 010000 005037 001724      CLR    CURID        ;CLEAN IT OUT
1308 010004 111437 001724      MOVB  (R4),CURID   ;GET CURRANT ID
1309 010010 000241              CLC
1310 010012 006137 001722      ROL    CURFUN       ;DOUBLE IT
1311 010016 012703 012746      MOV    #TABM-2,R3   ;LOAD BASE ADDRESS
1312 010022 063703 001722      ADD    CURFUN,R3    ;ADD OFFSET
1313 010026 011303              MOV    @R3,R3       ;GET ACTUAL ADDRESS
1314 010030 004713              JSR    PC,@R3       ;GO DO IT
1315 010032 005724              TST    (R4)+        ;STEP THRU MENU
1316 010034 022714 177777      CMP    #-1,(R4)     ;DONE MENU?
1317 010040 001352              BNE    1$           ;BR=NO
1318 010042 004737 022710      JSR    PC,SEQSTP    ;GO STEP SEQUENCER
1319 010046 005337 001726      DEC    CYCLE        ;COUNT
1320 010052 001402              BEQ    2$           ;BR IF DONE
  
```

```
1321 010054 000137 007722      JMP      DISPATCH      :GO START AGAIN
1322 010060 012737 007556 021652 2$:  MOV      #MULPRT,$RTNAD :SET UP RETURN
1323 010066 000137 021554      JMP      $EOP+2        :GO DO END PASS
1324 010072 004737 022516      SLAVE:  JSR      PC,SALMEN :GO MAKE SLAVE MENU
1325 010076 004737 022064      JSR      PC,SYNCUP     :GO SYNCUP! 2 MINUTE TIME OUT
1326 010102 005037 001722      1$:    CLR      CURFUN      :CLEAN IT
1327 010106 116437 000001 001722  MOVB    1(R4),CURFUN   :GET CURRANT FUNCTION
1328 010114 005037 001724      CLR      CURID        :CLEAN IT
1329 010120 111437 001724      MOVB    (R4),CURID    :GET CURRANT ID
1330 010124 000241      CLC
1331 010126 006137 001722      ROL      CURFUN      :DOUBLE IT
1332 010132 012703 012746      MOV      #TABM-2,R3   :LOAD BASE ADDRESS
1333 010136 063703 001722      ADD     CURFUN,R3     :ADD OFFSET
1334 010142 011303      MOV     @R3,R3        :GET ACTUAL ADDRESS
1335 010144 004713      JSR     PC,@R3        :GO DO IT
1336 010146 005724      TST     (R4)+         :STEP THRU MENU
1337 010150 022714 177777      CMP     #-1,(R4)     :DONE MENU
1338 010155 001352      BNE     1$           :BR=NO
1339 010156 004737 022710      JSR     PC,SEQSTP    :GO STEP SEQUENCER
1340 010162 005337 001726      DEC     CYCLE        :COUNT
1341 010166 001402      BEQ     2$           :BRIF DONE
1342 010170 000137 007722      JMP     DISPATCH     :GO START AGAIN
1343 010174 012737 007556 021652 2$:  MOV      #MULPRT,$RTNAD :SET UP RETURN
1344 010202 000137 021554      JMP     $EOP+2        :GO DO END PASS
1345
1346
1347      :EXTERNAL CONNECT REQUEST REJECTED
1348
1349 010206      FUNCT1:
1350 010206 004737 026620      JSR     PC,CNTLC     :IS IT CONTROL C?
1351 010212 004737 021674      JSR     PC,CONNEC   :GO CONNECT
1352 010216 004737 021732      JSR     PC,CONID    :GO CONVERT SLAVE ID
1353 010222 012777 010362 171374  MOV     #1$,@VECO    :SET RETURN
1354 010230 112777 000101 171364  MOVB    #101,@CSR    :ENABLE INTERUPT
1355 010236 005000      CLR     R0           :SET TIMER
1356 010240 005200      INC     R0           :TIME OUT
1357 010242 001376      BNE     -2           :BR IF NOT DONE
1358 010244 104401 010252      TYPE    ,65$        :TYPE ASCIZ STRING
(1) 010250 000434      BR      64$         :GET OVER THE ASCIZ
(1)      ::65$: .ASCIZ <200>/ERROR1A! NO EXTERNAL REQUEST INTERUPT DETECTED. CSR= /
(1) 64$:
1359 010342 017737 171254 001126  MOV     @CSR,BAD     :SAVE CSR
1360 010350 013746 001126  MOV     BAD,-(SP)    :SAVE BAD FOR TYPEOUT
(1) 010354 104402      TYPOC      :GO TYPE--OCTAL ASCII(ALL DIGITS)
1361 010356 000000      HALT
1362 010360 000776      BR      -2
1363 010362      1$:
(2) 010362 012600      MOV     (SP)+,R0     :POP STACK INTO R0
(2) 010364 012600      MOV     (SP)+,R0     :POP STACK INTO R0
1364 010366 017737 171230 001126  MOV     @CSR,BAD     :STORE CONTENTS OF CSR
1365 010374 032737 010000 001126  BIT     #10000,BAD   :MAKE SURE EXT INT IS SET
1366 010402 001036      BNE     2$           :BR = OK
1367 010404 104401 010412      TYPE    ,67$        :TYPE ASCIZ STRING
(1) 010410 000426      BR      66$         :GET OVER THE ASCIZ
(1)      ::67$: .ASCIZ <200>/ERROR1B! EXTERNAL INTERUPT NOT SET. CSR= /
(1) 66$:
```



```
1368 010466 013746 001126      MOV      BAD,-(SP)      ;;SAVE BAD FOR TYPEOUT
(1) 010472 104402                TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1369 010474 000000                HALT
1370 010476 000776                BR      .-2
1371 010500 033737 001730 001126 2$:  BIT      COVID,BAD      ;CORRECT SLAVE ID?
1372 010506 001042                BNE     3$            ;BR=YES
1373 010510 104401 010516                TYPE     ,69$        ;;TYPE ASCIZ STRING
(1) 010514 000432                BR      68$        ;;GET OVER THE ASCIZ
(1)                                ;;69$: .ASCIZ <200>/ERROR1C! EXTERNAL REQUEST FROM WRONG PORT. CSR= /
(1)                                68$:
1374 010602 013746 001126      MOV      BAD,-(SP)      ;;SAVE BAD FOR TYPEOUT
(1) 010606 104402                TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1375 010610 000000                HALT
1376 010612 000776                BR      .-2
1377 010614 012777 000101 171000 3$:  MOV      #101,@CSR      ;CLEAR EXT INT
1378 010622 012777 023022 170774                MOV      #OOPS,@VECO   ;RESET RETURN
1379 010630 005037 177776                CLR      PSW          ;LOWER PRIORITY
1380 010634 000207                RTS      PC
1381
1382
1383                                ;EXTERNAL REQUEST IGNORED
1384
1385                                FUNCT2:
1386 010636 004737 026620                JSR      PC,CNTLC      ;IS IT CONTROL C?
1387 010642 004737 021674                JSR      PC,CONNEC     ;GO CONNECT
1388 010646 004737 021732                JSR      PC,CONID      ;GO CONVERT SLAVE ID
1389 010652 012777 011144 170744                MOV      #1$,@VECO     ;SET RETURN
1390 010660 005737 001756                TST     TRIN          ;CAN WE TEST RESET IN NEUTRAL?
1391 010664 001052                BNE     6$            ;BR=NO
1392 010666 013777 001640 170742                MOV      DEVRW,@DEVAD  ;LOAD READ/WRITE BITS
1393 010674 023777 001640 170734                CMP     DEVRW,@DEVAD   ;LOADED?
1394 010702 001443                BEQ     6$            ;BR=YES
1395 010704 104401 010712                TYPE     ,65$        ;;TYPE ASCIZ STRING
(1) 010710 000432                BR      64$        ;;GET OVER THE ASCIZ
(1)                                ;;65$: .ASCIZ <200>/ERROR2F! COULDN'T LOAD EXTERNAL DEVICE REGISTER.../
(1)                                64$:
1396 010776 013746 001636      MOV      DEVAD,-(SP)    ;;SAVE DEVAD FOR TYPEOUT
(1) 011002 104402                TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1397 011004 012737 177777 001756                MOV      #-1,TRIN      ;SET FLAG FOR NO TEST
1398 011012 112777 000101 170602 6$:  MOV     #101,@CSR      ;ENABLE INTERRUPT
1399 011020 005000                CLR     R0            ;SET TIMER
1400 011022 005200                INC     R0            ;TIME OUT
1401 011024 001376                BNE     .-2          ;BR IF NOT DONE
1402 011026 104401 011034                TYPE     ,67$        ;;TYPE ASCIZ STRING
(1) 011032 000434                BR      66$        ;;GET OVER THE ASCIZ
(1)                                ;;67$: .ASCIZ <200>/ERROR2A! NO EXTERNAL REQUEST INTERRUPT DETECTED. CSR= /
(1)                                66$:
1403 011124 017737 170472 001126                MOV     @CSR,BAD      ;SAVE CSR
1404 011132 013746 001126                MOV     BAD,-(SP)     ;;SAVE BAD FOR TYPEOUT
(1) 011136 104402                TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1405 011140 000000                HALT
1406 011142 000776                BR      .-2
1407 011144                                1$:
(2) 011144 012600                MOV     (SP)+,R0      ;;POP STACK INTO R0
(2) 011146 012600                MOV     (SP)+,R0      ;;POP STACK INTO R0
1408 011150 017737 170446 001126                MOV     @CSR,BAD      ;STORE CONTENTS OF CSR
```

```
1409 011156 032737 010000 001126      BIT    #10000,BAD      ;MAKE SURE EXT INT IS SET
1410 011164 001036                BNE    2$             ;BR = OK
1411 011166 104401 011174                TYPE   ,69$          ;:TYPE ASCIZ STRING
(1) 011172 000426                BR     68$           ;:GET OVER THE ASCIZ
(1)                                ;:69$: .ASCIZ <200>/ERROR2B! EXTERNAL INTERUPT NOT SET. CSR= /
(1)                                ;:68$:
1412 011250 013746 001126                MOV    BAD,-(SP)     ;:SAVE BAD FOR TYPEOUT
(1) 011254 104402                TYPOC                ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1413 011256 000000                HALT
1414 011260 000776                BR     .-2
1415 011262 033737 001730 001126 2$:    BIT    COVID,BAD     ;CORRECT SLAVE ID?
1416 011270 001042                BNE    3$             ;BR=YES
1417 011272 104401 011300                TYPE   ,71$          ;:TYPE ASCIZ STRING
(1) 011276 000432                BR     70$           ;:GET OVER THE ASCIZ
(1)                                ;:71$: .ASCIZ <200>/ERROR2C! EXTERNAL REQUEST FROM WRONG PORT. CSR= /
(1)                                ;:70$:
1418 011364 013746 001126                MOV    BAD,-(SP)     ;:SAVE BAD FOR TYPEOUT
(1) 011370 104402                TYPOC                ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1419 011372 000000                HALT
1420 011374 000776                BR     .-2
1421 011376 012777 011514 170220 3$:    MOV    #4$,@VECO     ;SET RETURN
1422 011404 005037 177776                CLR    PSW           ;LOWER PRIORITY
1423 011410 004737 022010                JSR    PC,WAIT05     ;WAIT
1424 011414 104401 011422                TYPE   ,73$          ;:TYPE ASCIZ STRING
(1) 011420 000430                BR     72$           ;:GET OVER THE ASCIZ
(1)                                ;:73$: .ASCIZ <200>/ERROR2D! SLAVE DIDN'T TAKE SHARED BUSS. CSR= /
(1)                                ;:72$:
1425 011502 017746 170114                MOV    @CSR,-(SP)    ;:SAVE @CSR FOR TYPEOUT
(1) 011506 104402                TYPOC                ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1426 011510 000000                HALT
1427 011512 000776                BR     .-2
1428 011514                4$:
(2) 011514 012600                MOV    (SP)+,R0      ;:POP STACK INTO R0
(2) 011516 012600                MOV    (SP)+,R0      ;:POP STACK INTO R0
(2) 011520 012600                MOV    (SP)+,R0      ;:POP STACK INTO R0
1429 011522 005777 170074                TST    @CSR          ;TIMEOUT SET?
1430 011526 100436                BMI    5$             ;BR=YES
1431 011530 104401 011536                TYPE   ,75$          ;:TYPE ASCIZ STRING
(1) 011534 000423                BR     74$           ;:GET OVER THE ASCIZ
(1)                                ;:75$: .ASCIZ <200>/ERROR2E! NO TMO ON INTERUPT. CSR= /
(1)                                ;:74$:
1432 011604 017737 170012 001126                MOV    @CSR,BAD      ;SAVE CSR
1433 011612 013746 001126                MOV    BAD,-(SP)    ;SAVE BAD FOR TYPEOUT
(1) 011616 104402                TYPOC                ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1434 011620 000000                HALT
1435 011622 000776                BR     .-2
1436 011624 012777 023022 167772 5$:    MOV    #OOPS,@VECO   ;RESET RETURN
1437 011632 005077 167764                CLR    @CSR          ;CLEAR CSR
1438 011636 005037 177776                CLR    PSW           ;LOWER PRIORITY
1439 011642 000207                RTS    PC            ;EXIT
1440
1441
1442
1443                ;SEND CONNECT REQUEST EXPECT REJECTION
1444
1445 011644                FUNCT3:
```

```

1446 011644 004737 026620      JSR    PC,CNTLC      ;IS IT CONTROL C?
1447 011650 005000              CLR    RO            ;SET TIMERS
1448 011652 012701 000400      MOV    #400,R1
1449 011656 032777 020000 167736 1$:  BIT    #20000,@CSR   ;SWITCHED BUSS ACTIVE?
1450 011664 001051              BNE    2$           ;BR=YES
1451 011666 005200              INC    RO            ;TIME OUT
1452 011670 001372              BNE    1$
1453 011672 005301              DEC    R1
1454 011674 001370              BNE    1$
1455 011676 017737 167720 001126  MOV    @CSR,BAD      ;STORE CONTENTS OF CSR
1456 011704 104401 011712      TYPE   ,65$         ;:TYPE ASCIZ STRING
(1) 011710 000432              BR     64$          ;:GET OVER THE ASCIZ
(1)                               ;:65$: .ASCIZ <200>/ERROR3A! NO SWITCHED BUSS MASTER DETECTED. CSR = /
(1)                               64$:
1457 011776 013746 001126      MOV    BAD,-(SP)    ;:SAVE BAD FOR TYPEOUT
(1) 012002 104402              TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1458 012004 000000              HALT
1459 012006 000776              BR     -2
1460 012010 012777 012026 167606 2$:  MOV    #3$,@VECO    ;SET RETURN
1461 012016 012777 000101 167576  MOV    #101,@CSR    ;SET IE + REQ
1462 012024 000001              WAIT ;WAIT FOR INTERRUPT
1463 012026 000000              3$:
(2) 012026 012600              MOV    (SP)+,RO     ;:POP STACK INTO RO
(2) 012030 012600              MOV    (SP)+,RO     ;:POP STACK INTO RO
1464 012032 005777 167564      TST   @CSR          ;:TIMEOUT?
1465 012036 100435              BMI   4$           ;BR=YES
1466 012040 017737 167556 001126  MOV    @CSR,BAD      ;SAVE CSR
1467 012046 104401 012054      TYPE   ,67$         ;:TYPE ASCIZ STRING
(1) 012052 000422              BR     66$          ;:GET OVER THE ASCIZ
(1)                               ;:67$: .ASCIZ <200>/ERROR3B! NO TIMEOUT DETECTED. CSR=/
(1)                               66$:
1468 012120 013746 001126      MOV    BAD,-(SP)    ;:SAVE BAD FOR TYPEOUT
(1) 012124 104402              TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1469 012126 000000              HALT
1470 012130 000776              BR     -2
1471 012132 005077 167464 4$:  CLR    @CSR          ;CLEAR + DISCONNECT
1472 012136 012777 023022 167460  MOV    #0OPS,@VECO  ;RESET RETURN
1473 012144 005037 177776      CLR    PSW          ;LOWER PRIORITY
1474 012150 000207              RTS   PC            ;EXIT
1475                               ;SEND CONNECT REQUEST EXPECT CONNECTION
1476 012152              FUNCT4:
1477 012152 004737 026620      JSR    PC,CNTLC      ;IS IT CONTROL C?
1478 012156 005000              CLR    RO            ;SET TIMERS
1479 012160 012701 000400      MOV    #400,R1
1480 012164 032777 020000 167430 1$:  BIT    #20000,@CSR   ;SWITCHED BUSS ACTIVE?
1481 012172 001051              BNE    2$           ;BR=YES
1482 012174 005200              INC    RO            ;TIME OUT
1483 012176 001372              BNE    1$
1484 012200 005301              DEC    R1
1485 012202 001370              BNE    1$
1486 012204 017737 167412 001126  MOV    @CSR,BAD      ;STORE CONTENTS OF CSR
1487 012212 104401 012220      TYPE   ,65$         ;:TYPE ASCIZ STRING
(1) 012216 000432              BR     64$          ;:GET OVER THE ASCIZ
(1)                               ;:65$: .ASCIZ <200>/ERROR4A! NO SWITCHED BUSS MASTER DETECTED. CSR = /
(1)                               64$:
1488 012304 013746 001126      MOV    BAD,-(SP)    ;:SAVE BAD FOR TYPEOUT

```

```

(1) 012310 104402          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1489 012312 000000        HALT
1490 012314 000776        BR
1491 012316 013737 001746 001732 2$: MOV      -2          ;SET TIMER
1492 012324 006337 001732 ASL      TEMP1
1493 012330 032777 020000 167264 21$: BIT    #20000,@CSR ;SWB ACT SET?
1494 012336 001705        BEQ     FUNCT4 ;BR=NO
1495 012340 005337 001732 DEC     TEMP1
1496 012344 001371        BNE    21$
1497 012346 012777 012364 167250 MOV    #3$,@VECO ;SET RETURN
1498 012354 012777 000101 167240 MOV    #101,@CSR ;SET IE + REQ
1499 012362 000001        WAIT      ;WAIT FOR INTERUPT
1500 012364          3$:
(2) 012364 012600        MOV    (SP)+,RO ;POP STACK INTO RO
(2) 012366 012600        MOV    (SP)+,RO ;POP STACK INTO RO
1501 012370 105777 167226 TSTB   @CSR ;CONNECTED?
1502 012374 100437        BMI    4$ ;BR=YES
1503 012376 017737 167220 001126 MOV    @CSR,BAD ;SAVE CSR
1504 012404 104401 012412 TYPE   ,67$ ;TYPE ASCIIZ STRING
(1) 012410 000424        BR     66$ ;GET OVER THE ASCIIZ
(1)          ;;67$: .ASCIIZ <200>/ERROR4B! CONNECT REQUEST FAILED. CSR=/
(1) 012462          66$:
1505 012462 013746 001126 MOV    BAD,-(SP) ;SAVE BAD FOR TYPEOUT
(1) 012466 104402          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1506 012470 000000        HALT
1507 012472 000776        BR
1508 012474 005737 001756 4$: TST    TRIN ;CAN WE TEST RESET IN NEUTRAL
1509 012500 001036        BNE    5$ ;BR=NO
1510 012502 033777 001642 167126 BIT    DEVRC,@DEVAD ;RESET CONDITIONS MET?
1511 012510 001432        BEQ    5$ ;BR=YES
1512 012512 104401 012520 TYPE   ,69$ ;TYPE ASCIIZ STRING
(1) 012516 000424        BR     68$ ;GET OVER THE ASCIIZ
(1)          ;;69$: .ASCIIZ <200>/ERROR4C! NO RESET IN NEUTRAL DETECTED./
(1) 012570          68$:
1513 012570 012737 177777 001756 MOV    #-1,TRIN ;CANCEL FURTHER TESTING OF RESET IN NEUTRAL
1514 012576 005077 167020 5$: CLR    @CSR ;CLEAR + DISCONNECT
1515 012602 012777 023022 167014 MOV    #OOPS,@VECO ;RESET RETURN
1516 012610 005037 177776 CLR    PSW ;LOWER PRIORITY
1517 012614 000207        RTS     PC ;EXIT
1518          ;OBSERVE REQUEST INDICATORS
1519 012616          FUNCT5:
1520 012616 004737 026620 JSR    PC,CNTLC ;IS IT CONTROL C?
1521 012622 004737 021732 JSR    PC,CONID ;GO CONVERT PORT ID
1522 012626 005000        CLR    RO ;SET TIMERS
1523 012630 012701 000400 MOV    #400,R1
1524 012634 033777 001730 166760 1$: BIT    COVID,@CSR ;LOOK FOR REQUEST
1525 012642 001035        BNE    2$ ;BR=FOUND IT
1526 012644 005200        INC    RO ;TIME OUT
1527 012646 001372        BNE    1$
1528 012650 005301        DEC    R1
1529 012652 001370        BNE    1$
1530 012654 104401 012662 TYPE   ,65$ ;TYPE ASCIIZ STRING
(1) 012660 000424        BR     64$ ;GET OVER THE ASCIIZ
(1)          ;;65$: .ASCIIZ <200>/ERROR5A! NO REQUEST ACTIVITY DETECTED./
(1) 012732          64$:
1531 012732 000000        HALT

```

```
1532 012734 000776 BR .-2  
1533 012736 033777 001730 166656 2$: BIT COVID,@CSR ;DID IT GO AWAY?  
1534 012744 001374 BNE 2$ ;BR=NOT YET  
1535 012746 000207 RTS PC ;EXIT  
1536  
1537  
1538 012750 010206 TABM: FUNCT1 ;DISPATCH TABLE FOR MULTIPOINT TEST  
1539 012752 010636 FUNCT2  
1540 012754 011644 FUNCT3  
1541 012756 012152 FUNCT4  
1542 012760 012616 FUNCT5  
1543  
1544
```

```
1546
1547
1548
1549
1550      .SBTTL
1551      .SBTTL BIOPORT TEST IN DT03 MODE
1552      .SBTTL
1553      MULDT3:
1554 012762 004737 026620      JSR    PC,CNTLC      ;IS IT CONTROL C?
1555 012766 012737 000000 177776      MOV    #0,PSW      ;LOWER PRIORITY
1556 012774 005077 166622      CLR    @CSR        ;START WITH A CLEAR REGISTER
1557 013000 012737 002000 001726      MOV    #2000,CYCLE ;SET SUB COUNT
1558 013006 005037 022006      CLR    TRIG        ;CLEAR TRIGGER
1559 013012 032777 020000 166602 1$:    BIT    #20000,@CSR ;ACTIVE?
1560 013020 001401      BEQ    11$         ;BR=NO
1561 013022 000464      BR     21$         ;GO BE SLAVE
1562 013024 012777 013150 166572 11$:    MOV    #2$,@VECO   ;SET RETURN
1563 013032 012777 000101 166562      MOV    #101,@CSR   ;SEND REQUEST
1564 013040 004737 021774      JSR    PC,WAIT20   ;GO WAIT
1565 013044 104401 013052      TYPE   ,65$       ;:TYPE ASCIZ STRING
(1) 013050 000432      BR     64$        ;:GET OVER THE ASCIZ
(1)      ;:65$: .ASCIZ <200>/ERROR6! NO INTERUPT AFTER CONNECT REQUEST. CSR= /
(1)      64$:
1566 013136 017746 166460      MOV    @CSR,-(SP)  ;:SAVE @CSR FOR TYPEOUT
(1) 013142 104402      TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1567 013144 000000      HALT
1568 013146 000776      BR     .-2
1569 013150 012706 001100 2$:    MOV    #STACK,SP   ;RESET STACK
1570 013154 005037 177776      CLR    PSW        ;LOWER PRIORITY
1571 013160 005777 166436      TST   @CSR        ;TIMEOUT?
1572 013164 100007      BPL   3$          ;BR=NO
1573 013166 000240      NOP
1574 013170 000240      NOP
1575 013172 000240      NOP
1576 013174 005037 001744 21$:    CLR    ONOFF      ;INDIACATE FIRST SLAVE
1577 013200 000137 014372      JMP   SECT4       ;GO TO SECTION 4
1578 013204 012737 177777 001744 3$:    MOV    #-1,ONOFF  ;INDIACATE FIRST MASTER
1579
1580      ;SECTION 2 : WAIT FOR EXTERNAL REQUEST THEN RELEASE SWITC I
1581
1582 013212 005737 001756  SECT2: TST    TRIN        ;CAN WE TEST RESET IN NEUTRAL
1583 013216 001052      BNE   2$          ;BR=NO
1584 013220 013777 001640 166410      MOV    DEVRW,@DEVAD ;LOAD DEVICE REGISTER
1585 013226 023777 001640 166402      CMP   DEVRW,@DEVAD ;OK?
1586 013234 001443      BEQ   2$          ;BR=YES
1587 013236 104401 013244      TYPE   ,65$       ;:TYPE ASCIZ STRING
(1) 013242 000432      BR     64$        ;:GET OVER THE ASCIZ
(1)      ;:65$: .ASCIZ <200>/ERROR7A! COULDN'T LOAD EXTERNAL DEVICE REGISTER../
(1)      64$:
1588 013330 013746 001636      MOV    DEVAD,-(SP) ;:SAVE DEVAD FOR TYPEOUT
(1) 013334 104402      TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1589 013336 012737 177777 001756 2$:    MOV    #-1,TRIN   ;CANCEL FURTHER TESTING OF RESET IN NEUTRAL
1590 013344 012777 013470 166252      MOV    #1$,@VECO  ;SET RETURN
1591 013352 012737 000632 022062      MOV    #632,TOCK  ;SET DELAY FOR 5 SECONDS
1592 013360 004737 022022      JSR   PC,WAIT
1593 013364 104401 013372      TYPE   ,67$       ;:TYPE ASCIZ STRING
```

```
(1) 013370 000432 BR 66$ ;;GET OVER THE ASCIZ
(1) ;;67$: .ASCIZ <200>/ERROR7! NO EXTERNAL INTERRUPT REQUEST SEEN. CSR= /
(1) 013456 66$:
1594 013456 017746 166140 MOV @CSR,-(SP) ;;SAVE @CSR FOR TYPEOUT
(1) 013462 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1595 013464 000000 HALT
1596 013466 000776 BR -2
1597 013470 012706 001100 1$: MOV #STACK,SP ;RESET STACK
1598 013474 013737 001752 001124 MOV KONST,GOOD
1599 013502 062737 030301 001124 ADD #30301,GOOD
1600 013510 023777 001124 166104 CMP GOOD,@CSR ;MAKE SURE IT'S AN EXTERNAL REQUEST
1601 013516 001435 BEQ SECT2A ;BR=OK
1602 013520 104401 013526 TYPE ,69$ ;;TYPE ASCIZ STRING
(1) 013524 000425 BR 68$ ;;GET OVER THE ASCIZ
(1) ;;69$: .ASCIZ <200>/ERROR8! UNEXPECTED INTERRUPT TYPE. CSR= /
(1) 68$:
1603 013600 017746 166016 MOV @CSR,-(SP) ;;SAVE @CSR FOR TYPEOUT
(1) 013604 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1604 013606 000000 HALT
1605 013610 000776 BR -2
1606 013612 000401 SECT2A: BR 1$ ;RELEASE TYPE SWITCHBACK
1607 013614 000420 BR SECT2B ;GO DO PASSIVE RELEASE
1608 013616 012737 000240 013612 1$: MOV #240,SECT2A ;TURN SWITCHBACK
1609 013624 005077 165772 SECT2D: CLR @CSR ;DROP SWITCH ACTIVELY
1610 013630 005037 177776 CLR PSW ;LOWER PRIORITY
1611 013634 105777 165762 TSTB @CSR ;DROPPED?
1612 013640 100775 BMI -4 ;BR=NO
1613 013642 032777 020000 165752 BIT #20000,@CSR ;PICKED UP?
1614 013650 001774 BEQ -6 ;BR=NO
1615 013652 000137 014106 JMP SECT2C ;GO TO EXIT
1616 013656 012737 000401 013612 SECT2B: MOV #401,SECT2A ;TURN SWITCHBACK
1617 013664 012777 014000 165732 MOV #1$,@VECO ;SET RETURN
1618 013672 005037 177776 CLR PSW ;LOWER PRIORITY
1619 013676 004737 021774 JSR PC,WAIT20 ;GO WAIT FOR TIMEOUT
1620 013702 104401 013710 TYPE ,65$ ;;TYPE ASCIZ STRING
(1) 013706 000427 BR 64$ ;;GET OVER THE ASCIZ
(1) ;;65$: .ASCIZ <200>/ERROR9! NO TIMEOUT INTERRUPT DETECTED. CSR= /
(1) 64$:
1621 013766 017746 165630 MOV @CSR,-(SP) ;;SAVE @CSR FOR TYPEOUT
(1) 013772 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1622 013774 000000 HALT
1623 013776 000776 BR -2
1624 014000 012706 001100 1$: MOV #STACK,SP ;RESET STACK
1625 014004 005777 165612 TST @CSR ;MAKE SURE IT'S A TIMEOUT
1626 014010 100705 BMI SECT2D ;BR=OK
1627 014012 104401 014020 TYPE ,67$ ;;TYPE ASCIZ STRING
(1) 014016 000426 BR 66$ ;;GET OVER THE ASCIZ
(1) ;;67$: .ASCIZ <200>/ERROR10! INTERRUPT NOT CAUSED BY TMO. CSR= /
(1) 66$:
1628 014074 017746 165522 MOV @CSR,-(SP) ;;SAVE @CSR FOR TYPEOUT
(1) 014100 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1629 014102 000000 HALT
1630 014104 000776 BR -2
1631 014106 005077 165510 SECT2C: CLR @CSR ;CLEAR CSR
1632 014112 005037 177776 CLR PSW ;LOWER PRIORITY
1633
```

```
1634 ;SECTION 3 : SEND REQUEST EXPECT REJECTION
1635
1636 014116 012737 000001 022062 SECT3: MOV #1,TOCK ;SET MINIMUM TIME
1637 014124 004737 022022 JSR PC,WAIT ;GO WAIT
1638 014130 012777 014240 165466 MOV #1$,@VECO ;SET RETURN
1639 014136 012777 000101 165456 MOV #101,@CSR ;SEND REQUEST
1640 014144 004737 021774 JSR PC,WAIT20 ;GO WAIT FOR TIMEOUT
1641 014150 104401 014156 TYPE ,65$ ;TYPE ASCIZ STRING
(1) 014154 000424 BR ,64$ ;GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>/ERROR11! NO INTERRUPT DETECTED. CSR= /
(1) 64$:
1642 014226 017746 165370 MOV @CSR,-(SP) ;:SAVE @CSR FOR TIMEOUT
(1) 014232 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1643 014234 000000 HALT
1644 014236 000776 BR ,.-2
1645 014240 012706 001100 1$: MOV #STACK,SP ;RESET STACK
1646 014244 005777 165352 TST @CSR ;TIMEOUT?
1647 014250 100437 BMI 2$ ;BR=YES
1648 014252 104401 014260 TYPE ,67$ ;TYPE ASCIZ STRING
(1) 014256 000427 BR ,66$ ;GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <200>/ERROR12! INTERRUPT NOT CAUSED BY TMO. CSR= /
(1) 66$:
1649 014336 017746 165260 MOV @CSR,-(SP) ;:SAVE @CSR FOR TIMEOUT
(1) 014342 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1650 014344 000000 HALT
1651 014346 000776 BR ,.-2
1652 014350 005077 165246 2$: CLR @CSR ;CLEAR CSR
1653 014354 005037 177776 CLR PSW ;LOWER PRIORITY
1654 014360 005737 022006 TST TRIG ;IS TRIGGER SET?
1655 014364 001402 BEQ SECT4 ;BR=NO
1656 014366 000137 015236 JMP MULEND ;FINISHED A SUB PASS
1657
1658 ;SECTION 4 : SEND REQUEST EXPECT CONNECTION
1659
1660 014372 012777 014502 165224 SECT4: MOV #1$,@VECO ;SET RETURN
1661 014400 012777 000101 165214 MOV #101,@CSR ;SEND REQUEST
1662 014406 004737 021774 JSR PC,WAIT20 ;GO WAIT FOR TIMEOUT
1663 014412 104401 014420 TYPE ,65$ ;TYPE ASCIZ STRING
(1) 014416 000424 BR ,64$ ;GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>/ERROR13! NO INTERRUPT DETECTED. CSR= /
(1) 64$:
1664 014470 017746 165126 MOV @CSR,-(SP) ;:SAVE @CSR FOR TIMEOUT
(1) 014474 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1665 014476 000000 HALT
1666 014500 000776 BR ,.-2
1667 014502 012706 001100 1$: MOV #STACK,SP ;RESET STACK
1668 014506 013737 001752 001124 MOV KONST,GOOD
1669 014514 062737 020301 001124 ADD #20301,GOOD
1670 014522 023777 001124 165072 CMP GOOD,@CSR ;CONNECTED?
1671 014530 001432 BEQ SECT5 ;BR=YES
1672 014532 104401 014540 TYPE ,67$ ;TYPE ASCIZ STRING
(1) 014536 000422 BR ,66$ ;GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <200>/ERROR14! COULD NOT CONNECT. CSR= /
(1) 66$:
1673 014604 017746 165012 MOV @CSR,-(SP) ;:SAVE @CSR FOR TIMEOUT
(1) 014610 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
```



```

1674 014612 000000          HALT
1675 014614 000776          BR      .-2
1676
1677          ;SECTION 5 :DETECT EXTERNAL REQUEST AND REJECT IT
1678
1679 014616 005737 001756    SECT5:  TST      TRIN          ;CAN WE TEST RESET IN NEUTRAL?
1680 014622 001041          BNE      3$              ;BR=NO
1681 014624 033777 001642 165004  BIT      DEVRC,@DEVAD    ;RESET CONDITION MET
1682 014632 001435          BEQ      3$              ;BR=YES
1683 014634 104401 014642    TYPE     ,65$           ;:TYPE ASCIZ STRING
(1) 014640 000427          BR      64$             ;:GET OVER THE ASCIZ
(1)          ;:65$: .ASCIZ <200>/ERROR15A! COULDN'T DETECT RESET IN NEUTRAL./
(1) 014720          64$:
1684 014720 012737 177777 001756  MOV      #-1,TRIN        ;CANCEL FURTHER TESTING OF RESET IN NEUTRAL
1685 014726 012777 015044 164670 3$:  MOV      #1$,@VECO      ;SET RETURN
1686 014734 005037 177776          CLR      PSW            ;LOWER PRIORITY
1687 014740 004737 021774          JSR     PC,WAIT20       ;GO WAIT
1688 014744 104401 014752    TYPE     ,67$           ;:TYPE ASCIZ STRING
(1) 014750 000430          BR      66$             ;:GET OVER THE ASCIZ
(1)          ;:67$: .ASCIZ <200>/ERROR15! NO EXTERNAL REQUEST DETECTED. CSR= /
(1) 015032          66$:
1689 015032 017746 164564    MOV      @CSR,-(SP)      ;:SAVE @CSR FOR TYPEOUT
(1) 015036 104402          TYPOC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1690 015040 000000          HALT
1691 015042 000776          BR      .-2
1692 015044 012706 001100    1$:  MOV      #STACK,SP      ;RESET STACK
1693 015050 013737 001752 001124  MOV      KONST,GOOD
1694 015056 062737 030301 001124  ADD      #30301,GOOD
1695 015064 023777 001124 164530  CMP      GOOD,@CSR      ;EXT. INT. ?
1696 015072 001433          BEQ      2$              ;BR=OK
1697 015074 104401 015102    TYPE     ,69$           ;:TYPE ASCIZ STRING
(1) 015100 000423          BR      68$             ;:GET OVER THE ASCIZ
(1)          ;:69$: .ASCIZ <200>/ERROR16! UNEXPECTED INTERUPT. CSR= /
(1) 015150          68$:
1698 015150 017746 164446    MOV      @CSR,-(SP)      ;:SAVE @CSR FOR TYPEOUT
(1) 015154 104402          TYPOC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1699 015156 000000          HALT
1700 015160 000776          BR      .-2
1701 015162 012777 000101 164432 2$:  MOV      #101,@CSR      ;CLEAR EXT. INT.
1702 015170 012777 023022 164426  MOV      #OOPS,@VECO    ;RESET RETURN
1703 015176 005037 177776          CLR      PSW            ;LOWER PRIORITY
1704
1705 015202 005337 001726    DT3SP: DEC      CYCLE      ;COUNT A SUBPASS
1706 015206 001402          BCC     1$              ;BR=DONE
1707 015210 000137 013212    JMP     SECT2           ;CONTINUE
1708 015214 005737 001744    1$:  TST      ONOFF         ;FIRST MASTER OR SLAVE
1709 015220 001401          BEQ      2$              ;BR=SLAVE
1710 015222 000405          BR      MULEND          ;MASTER REPORTS ENDPASS
1711 015224 012737 177777 022006 2$:  MOV      #-1,TRIG      ;SET TRIGGER FOR SLAVE
1712 015232 000137 013212    JMP     SECT2           ;CONTINUE SLAVE AT SECT2
1713
1714 015236 005737 001744    MULEND: TST     ONOFF     ;WAS THIS FIRST SLAVE?
1715 015242 001405          BEQ      1$              ;BR=YES
1716 015244 012737 012762 021652  MOV      #MULDT3,$RTNAD ;SET RETURN
1717 015252 000137 021554          JMP     $EOP+2         ;GO DO ENDPASS
1718 015256 012737 015270 021652 1$:  MOV      #2$, $RTNAD    ;SET RETURN

```

1719	015264	000137	021554		JMP	\$EOP+2	:GO DO ENDPASS
1720	015270	004737	021774	2\$:	JSR	PC, WAIT20	:WAIT 2 SECONDS
1721	015274	000137	012762		JMP	MULDT3	:GO TO IT!

```
1723
1724 .SBTTL
1725 .SBTTL MANUAL INTERVENTION TEST
1726
1727 015300 MANIN:
1728 015300 004737 026620 JSR PC,CNTLC ;IS IT CONTROL C?
1729 015304 005737 001634 TST DT03 ;CHECK DT03 MODE
1730 015310 001402 BEQ MANMOD ;BR=DT07 MODE
1731 015312 000137 016376 JMP PWROK ;SKIP OVER MANUAL MODE TEST IF DT03
1732 015316 MANMOD:
1733 015316 004737 026620 JSR PC,CNTLC ;IS IT CONTROL C?
1734 015322 005077 164274 CLR @CSR ;CLEAR CSR
1735 015326 104401 015334 TYPE ,65$ ;TYPE ASCIZ STRING
(1) 015332 000427 BR ,64$ ;GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>/PUT THIS DT07 IN MANUAL MODE THEN TYPE <CR>./
(1) 64$:
1736 015412 104410 RDCHR
1737 015414 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
1738 015416 032777 002000 164176 1$: BIT #2000,@CSR ;MANUAL MODE SET?
1739 015424 001042 BNE 2$ ;BR=YES
1740 015426 104401 015434 TYPE ,67$ ;TYPE ASCIZ STRING
(1) 015432 000430 BR ,66$ ;GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <200>/ERROR1! MANUAL MODE (BIT10) IS NOT SET. CSR= /
(1) 66$:
1741 015514 017746 164102 MOV @CSR,-(SP) ;:SAVE @CSR FOR TYPEOUT
(1) 015520 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1742 015522 000000 HALT
1743 015524 004737 026620 JSR PC,CNTLC ;IS IT CONTROL C?
1744 015530 000672 BR MANMOD ;GO TRY AGAIN
1745 015532 012777 000001 164062 2$: MOV #1,@CSR ;TRY TO SET REQ IN MAN.MOD.
1746 015540 032777 000001 164054 BIT #1,@CSR ;SET?
1747 015546 001445 BEQ 3$ ;BR=NO
1748 015550 104401 015556 TYPE ,69$ ;TYPE ASCIZ STRING
(1) 015554 000433 BR ,68$ ;GET OVER THE ASCIZ
(1) ;:69$: .ASCIZ <200>/ERROR2! REQ BIT NOT DISABLED IN MANUAL MODE. CSR= /
(1) 68$:
1749 015644 017746 163752 MOV @CSR,-(SP) ;:SAVE @CSR FOR TYPEOUT
(1) 015650 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1750 015652 000000 HALT
1751 015654 004737 026620 JSR PC,CNTLC ;IS IT CONTROL C?
1752 015660 000724 BR 2$ ;TRY AGAIN
1753 015662 3$:
(1) 015662 104401 015670 TYPE ,71$ ;TYPE ASCIZ STRING
(1) 015666 000426 BR ,70$ ;GET OVER THE ASCIZ
(1) ;:71$: .ASCIZ <200>/CONNECT THIS DT07 MANUALLY THEN TYPE <CR>./
(1) 70$:
1754 015744 104410 RDCHR
1755 015746 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
1756 015750 032777 000001 163644 BIT #1,@CSR ;IS REQ SET NOW?
1757 015756 001041 BNE 4$ ;BR=YES
1758 015760 104401 015766 TYPE ,73$ ;TYPE ASCIZ STRING
(1) 015764 000431 BR ,72$ ;GET OVER THE ASCIZ
(1) ;:73$: .ASCIZ <200>/ERROR3! REQ. BIT NOT SET WHEN CONNECTED. CSR= /
(1) 72$:
1759 016050 017746 163546 MOV @CSR,-(SP) ;:SAVE @CSR FOR TYPEOUT
(1) 016054 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
1760 016056 000000          HALT
1761 016060 000700          BR          3$
1762 016062 000005          4$: RESET          ;RESET THE BUSS
1763 016064 032777 000001 163530 BIT          #1,@CSR ;REQ STILL THERE?
1764 016072 001034          BNE          5$      ;BR=YES
1765 016074 104401 016102 TYPE          75$     ;:TYPE ASCIZ STRING
(1) 016100 000424          BR          74$     ;:GET OVER THE ASCIZ
(1) 016152          ;:75$: .ASCIZ <200>/ERROR4! RESET CLEARED REQ. BIT. CSR= /
(1) 016152 017746 163444 74$: MOV          @CSR,-(SP) ;:SAVE @CSR FOR TYPEOUT
(1) 016156 104402          TYPOC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1767 016160 000000          HALT
1768 016162 000737          BR          4$
1769 016164          5$: TYPE          77$     ;:TYPE ASCIZ STRING
(1) 016164 104401 016172 BR          76$     ;:GET OVER THE ASCIZ
(1) 016170 000432          ;:77$: .ASCIZ <200>/PUT THIS DT07 IN PROGRAMMABLE MODE THEN TYPE <CR>./
(1) 016256          76$: RDCHR
1770 016256 104410          MOV          (SP)+,R0 ;:POP STACK INTO R0
1771 016260 012600          BIT          #2000,@CSR ;:MAN.MOD. GONE?
1772 016262 032777 002000 163332 BEQ          PWROK    ;BR=YES
1773 016270 001442          TYPE          79$     ;:TYPE ASCIZ STRING
1774 016272 104401 016300 BR          78$     ;:GET OVER THE ASCIZ
(1) 016276 000432          ;:79$: .ASCIZ <200>/ERROR5! MANUAL MODE (BIT10) DIDN'T CLEAR. CSR= /
(1) 016364          78$: MOV          @CSR,-(SP) ;:SAVE @CSR FOR TYPEOUT
1775 016364 017746 163232 TYPOC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 016370 104402          HALT
1776 016372 000000          BR          5$
1777 016374 000673          PWROK: JSR          PC,CNTLC ;IS IT CONTROL C?
1778 016376          CLR          @CSR    ;CLEAR DEVICE
1779 016376 004737 026620 TYPE          65$     ;:TYPE ASCIZ STRING
1780 016402 005077 163214 BR          64$     ;:GET OVER THE ASCIZ
1781 016406 104401 016414 ;:65$: .ASCIZ <200>/POWER UP ALL DT07S THEN TYPE <CR>./
(1) 016412 000422          64$: RDCHR
1782 016460 104410          MOV          (SP)+,R0 ;:POP STACK INTO R0
1783 016462 012600          BIT          #4000,@CSR ;:PWROK OK?
1784 016464 032777 004000 163130 1$: BNE          2$      ;BR=YES
1785 016472 001034          TYPE          67$     ;:TYPE ASCIZ STRING
1786 016474 104401 016502 BR          66$     ;:GET OVER THE ASCIZ
(1) 016500 000424          ;:67$: .ASCIZ <200>/ERROR6! PWR.OK (BIT11) NOT SET. CSR= /
(1) 016552          66$: MOV          @CSR,-(SP) ;:SAVE @CSR FOR TYPEOUT
1787 016552 017746 163044 TYPOC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 016556 104402          HALT
1788 016560 000000          BR          PWROK
1789 016562 000705          CLR          PSW      ;LOWER PRIORITY
1790 016564 005037 177776 2$: JSR          PC,CONN ;GO CONNECT
1791 016570 004737 021674 TYPE          69$     ;:TYPE ASCIZ STRING
1792 016574 104401 016602 BR          68$     ;:GET OVER THE ASCIZ
(1) 016600 000423          ;:69$: .ASCIZ <200>/POWER DOWN OTHER CPU THEN TYPE <CR>./
(1) 016650          68$: RDCHR
1793 016650 104410
```

```

1794 016652 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
1795 016654 032777 004000 162740 BIT      #4000,@CSR      ;;PWROK NOT OK?
1796 016662 001441          BEQ      3$              ;;BR=OK
1797 016664 104401 016672    TYPE     ,71$           ;;TYPE ASCIZ STRING
(1) 016670 000427          BR       70$           ;;GET OVER THE ASCIZ
(1)                          ;;71$: .ASCIZ <200>/ERROR7! PWR. OK (BIT11) NOT CLEARED. CSR= /
(1)                          70$:
1798 016750 017746 162646    MOV      @CSR,-(SP)     ;;SAVE @CSR FOR TYPEOUT
(1) 016754 104402          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1799 016756 000000          HALT
1800 016760 004737 026620    JSR     PC,CNTLC       ;IS IT CONTROL C?
1801 016764 000677          BR       2$
1802 016766 032777 000200 162626 3$: BIT      #200,@CSR      ;STILL CONNECTED?
1803 016774 001032          BNE     4$              ;BR=YES
1804 016776 104401 017004    TYPE     ,73$           ;;TYPE ASCIZ STRING
(1) 017002 000421          BR       72$           ;;GET OVER THE ASCIZ
(1)                          ;;73$: .ASCIZ <200>/ERROR8! LOST SHARED BUSS. CSR= /
(1)                          72$:
1805 017046 017746 162550    MOV      @CSR,-(SP)     ;;SAVE @CSR FOR TYPEOUT
(1) 017052 104402          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1806 017054 000000          HALT
1807 017056 000137 016376    JMP     PWROK
1808 017062          4$:
(1) 017062 104401 017070    TYPE     ,75$           ;;TYPE ASCIZ STRING
(1) 017066 000433          BR       74$           ;;GET OVER THE ASCIZ
(1)                          ;;75$: .ASCIZ <200>/RESTORE ALL DT07S TO ORIGINAL STATE THEN TYPE <CR>./
(1)                          74$:
1809 017156 104410          RDCHR
1810 017160 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
1811
1812 017162          SWBPWF:
1813 017162 004737 026620    JSR     PC,CNTLC       ;IS IT CONTROL C?
1814 017166 005077 162430    CLR     @CSR           ;CLEAR DEVICE
1815 017172 013746 000024    MOV     @#PWRVEC,-(SP) ;;PUSH @#PWRVEC ON STACK
1816 017176 012737 017406 000024 MOV     #11,@#PWRVEC   ;SET RETURN
1817 017204 012777 000100 162410 MOV     #100,@CSR      ;SET I.E.
1818 017212 005037 177776 1$: CLR     PSW           ;LOWER PRIORITY
1819 017216 104401 017224    TYPE     ,65$           ;;TYPE ASCIZ STRING
(1) 017222 000426          BR       64$           ;;GET OVER THE ASCIZ
(1)                          ;;65$: .ASCIZ <200>/POWER DOWN THE SHARED BUSS THEN TYPE <CR>./
(1)                          64$:
1820 017300          RDCHR
1821 017302 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
1822 017304 032777 040000 162310 BIT      #40000,@CSR    ;;SWB PWF SET?
1823 017312 001112          BNE     2$              ;BR=YES
1824 017314 104401 017322    TYPE     ,67$           ;;TYPE ASCIZ STRING
(1) 017320 000425          BR       66$           ;;GET OVER THE ASCIZ
(1)                          ;;67$: .ASCIZ <200>/ERROR8! SWB PWF (BIT14) NOT SET. CSR= /
(1)                          66$:
1825 017374 017746 162222    MOV      @CSR,-(SP)     ;;SAVE @CSR FOR TYPEOUT
(1) 017400 104402          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1826 017402 000000          HALT
1827 017404 000702          BR       1$
1828 017406 012706 001100 11$: MOV     #STACK,SP      ;RESET STACK
1829 017412 005746          TST     -(SP)           ;POSITION STACK
1830 017414 012637 000024    MOV     (SP)+,@#PWRVEC ;;POP STACK INTO @#PWRVEC

```

```

1831 017420 104401 017426          TYPE      ,69$      ;;TYPE ASCIZ STRING
(1) 017424 000440          BR          68$      ;;GET OVER THE ASCIZ
(1) 017526          ;;69$: .ASCIZ <200>/ERROR9! TRAP THRU POWER FAIL VECTOR WHEN DISCONNECTED. CSR= /
(1) 017526          68$:          MOV @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
1832 017526 017746 162070          TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 017532 104402          HALT
1833 017534 000000          BR          SWBPWF
1834 017536 000611          2$:
1835 017540          TYPE      ,71$      ;;TYPE ASCIZ STRING
(1) 017540 104401 017546          BR          70$      ;;GET OVER THE ASCIZ
(1) 017544 000423          ;;71$: .ASCIZ <200>/POWER UP SHARED BUSS THEN TYPE <CR>/
(1) 017614          70$:          RDCHR
1836 017614 104410          MOV (SP)+,R0      ;;POP STACK INTO R0
1837 017616 012600          BIT #40000,@CSR  ;;DID IT CLEAR?
1838 017620 032777 040000 161774          BEQ 3$          ;;BR=YES
1839 017626 001441          TYPE      ,73$      ;;TYPE ASCIZ STRING
1840 017630 104401 017636          BR          72$      ;;GET OVER THE ASCIZ
(1) 017634 000430          ;;73$: .ASCIZ <200>/ERROR10! SWB PWF (BIT14) DIDN'T CLEAR. CSR= /
(1) 017716          72$:          MOV @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
1841 017716 017746 161700          TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 017722 104402          HALT
1842 017724 000000          JMP 1$
1843 017726 000137 017212          ERREN=.
1844 017732 017732          3$:          CLR @CSR          ;CLEAR IT
1845 017732 005077 161664          MOV #PFVSEV,@#PWRVEC ;SET RETURNS
1846 017736 012737 020170 000024          MOV #PWFSEV,@VECO
1847 017744 012777 021024 161652          CLR PSW
1848 017752 005037 177776          JSR PC,CONNEC      ;GO CONNECT
1849 017756 004737 021674          BIS #100,@CSR      ;SET I.E.
1850 017762 052777 000100 161632          TYPE      ,75$      ;;TYPE ASCIZ STRING
1851 017770 104401 017776          BR          74$      ;;GET OVER THE ASCIZ
(1) 017774 000423          ;;75$: .ASCIZ <200>/POWER THE SHARED BUSS DOWN THEN UP./
(1) 020044          74$:          MOV #11470,TOCK      ;SET TIMER FOR 1 MIN.
1852 020044 012737 011470 022062          JSR PC,WAIT
1853 020052 004737 022022          TYPE      ,77$      ;;TYPE ASCIZ STRING
1854 020056 104401 020064          BR          76$      ;;GET OVER THE ASCIZ
(1) 020062 000435          ;;77$: .ASCIZ <200>/ERROR11! NO EVIDENCE OF POWER FAILURE DETECTED. CSR= /
(1) 020156          76$:          MOV @CSR,-(SP)      ;;SAVE @CSR FOR TYPEOUT
1855 020156 017746 161440          TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 020162 104402          HALT
1856 020164 000000          BR          3$
1857 020166 000661          PFVSEV:
1858          JSR PC,CNTLC      ;IS IT CONTROL C?
1859 020170          MOV @CSR,BAD      ;SAVE CSR CONTENTS
1860 020170 004737 026620          BIC #400,BAD      ;MASK BRQ IF PRESENT
1861 020174 017737 161422 001126          BIC #100,@CSR      ;DROP I.E.
1862 020202 042737 000400 001126          MOV #1$,@#PWRVEC  ;POINT TO POWER UP ROUTINE
1863 020210 042777 000100 161404          HALT ;IF YOU STOPPED HERE NO POWER UP TRAP OCCURED
1864 020216 012737 020226 000024          1$:          MOV @CSR,TEMP2      ;STORE CSR
1865 020224 000000          MOV KONST,GOOD     ;ASSEMBLE GOOD DATA
1866 020226 017737 161370 002674
1867 020234 013737 001752 001124

```

1868	020242	062737	060301	001124		ADD	#60301,GOOD	
1869	020250	023737	001124	001126		CMP	GOOD,BAD	:CORRECT AT PWR DOWN?
1870	020256	001465				BEQ	2\$	:BR=YES
1871	020260	104401	020266			TYPE	.65\$	::TYPE ASCIZ STRING
(1)	020264	000433				BR	64\$	::GET OVER THE ASCIZ
(1)					::65\$:	.ASCIZ	<200>/ERROR12!	CSR INCORRECT @ POWER DOWN TIME. CSR WAS /
(1)	020354				64\$:			
1872	020354	013746	001126			MOV	BAD,-(SP)	::SAVE BAD FOR TYPEOUT
(1)	020360	104402				TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
1873	020362	104401	020370			TYPE	.67\$	::TYPE ASCIZ STRING
(1)	020366	000410				BR	66\$	::GET OVER THE ASCIZ
(1)					::67\$:	.ASCIZ	<200>/IT SHOULD BE /	
(1)	020410				66\$:			
1874	020410	013746	001124			MOV	GOOD,-(SP)	::SAVE GOOD FOR TYPEOUT
(1)	020414	104402				TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
1875	020416	000000				HALT		
1876	020420	012706	001100		11\$:	MOV	#STACK,SP	:RESET STACK
1877	020424	005746				TST	-(SP)	:POSITION STACK
1878	020426	000137	017732			JMP	ERREN	
1879	020432	013737	002674	001126	2\$:	MOV	TEMP2,BAD	:GET PWR UP DATA
1880	020440	042737	000400	001126		BIC	#400,BAD	:MASK BRQ IF PRESENT
1881	020446	162737	020301	001124		SUB	#20301,GOOD	:CORRECT GOOD DATA
1882	020454	023737	001124	001126		CMP	GOOD,BAD	:CORRECT @ PWR UP?
1883	020462	001460				BEQ	3\$	:BR=YES
1884	020464	104401	020472			TYPE	.69\$	::TYPE ASCIZ STRING
(1)	020470	000432				BR	68\$	::GET OVER THE ASCIZ
(1)					::69\$:	.ASCIZ	<200>/ERROR13!	CSR INCORRECT @ POWER UP TIME. CSR WAS /
(1)	020556				68\$:			
1885	020556	013746	001126			MOV	BAD,-(SP)	::SAVE BAD FOR TYPEOUT
(1)	020562	104402				TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
1886	020564	104401	020572			TYPE	.71\$	::TYPE ASCIZ STRING
(1)	020570	000410				BR	70\$	::GET OVER THE ASCIZ
(1)					::71\$:	.ASCIZ	<200>/IT SHOULD BE /	
(1)	020612				70\$:			
1887	020612	013746	001124			MOV	GOOD,-(SP)	::SAVE GOOD FOR TYPEOUT
(1)	020616	104402				TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
1888	020620	000000				HALT		
1889	020622	000676				BR	11\$	
1890	020624	032777	040000	160770	3\$:	BIT	#40000,@CSR	:DID SWB PWF CLEAR?
1891	020632	001442				BEQ	4\$	:BR=YES
1892	020634	104401	020642			TYPE	.73\$	::TYPE ASCIZ STRING
(1)	020640	000430				BR	72\$	::GET OVER THE ASCIZ
(1)					::73\$:	.ASCIZ	<200>/ERROR14!	SWB PWF (BIT14) DIDN'T CLEAR. CSR= /
(1)	020722				72\$:			
1893	020722	017746	160674			MOV	@CSR,-(SP)	::SAVE @CSR FOR TYPEOUT
(1)	020726	104402				TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
1894	020730	000000				HALT		
1895	020732	004737	026620			JSR	PC,CN^LC	:IS IT CONTROL C?
1896	020736	000630				BR	11\$	
1897	020740				4\$:			
(1)	020740	104401	020746			TYPE	.75\$	::TYPE ASCIZ STRING
(1)	020744	000420				BR	74\$	::GET OVER THE ASCIZ
(1)					::75\$:	.ASCIZ	<200>/THIS DT07	TRAPS ON POWER FAIL! /
(1)	021006				74\$:			
1898		021006			POSTSV=.			
1899	021006	012706	001100			MOV	#STACK,SP	:RESET STACK

```

1900 021012 005746          TST      -(SP)          ;POSITION STACK
1901 021014 012637 000024  MOV      (SP)+,@#PWRVEC ;:POP STACK INTO @#PWRVEC
1902 021020 000137 021500  JMP      MANEND
1903
1904 021024 017737 160572 001126 PWF SER: MOV      @CSR,BAD      ;SAVE CSR
1905 021032 042737 000400 001126 BIC      #400,BAD      ;MASK BRQ IF PRESENT
1906 021040 013737 001752 001124 MOV      KONST,GOOD    ;ASSEMBLE GOOD DATA
1907 021046 062737 060301 001124 ADD      #60301,GOOD
1908 021054 023737 001124 001124 CMP      GOOD,BAD      ;CORRECT @PWR DOWN?
1909 021062 001457          BEQ      1$            ;BR=YES
1910 021064 104401 021072  TYPE     ,65$         ;:TYPE ASCIZ STRING
(1) 021070 000427          BR       64$         ;:GET OVER THE ASCIZ
(1)          ;:65$: .ASCIZ <200>/ERROR15! CSR INCORRECT @ POWER DOWN. WAS /
(1) 021150          64$:
1911 021150 013746 001126  MOV      BAD,-(SP)     ;:SAVE BAD FOR TYPEOUT
(1) 021154 104402          TYPOC    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1912 021156 104401 021164  TYPE     ,67$         ;:TYPE ASCIZ STRING
(1) 021162 000406          BR       66$         ;:GET OVER THE ASCIZ
(1)          ;:67$: .ASCIZ <200>/SHOULD BE /
(1) 021200          66$:
1913 021200 013746 001124  MOV      GOOD,-(SP)   ;:SAVE GOOD FOR TYPEOUT
(1) 021204 104402          TYPOC    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1914 021206 000000          HALT
1915 021210 012706 001100  11$:  MOV      #STACK,SP    ;RESET STACK
1916 021214 005746          TST      -(SP)          ;POSITION STACK
1917 021216 000137 017732  JMP      ERREN
1918 021222 012737 000632 022062 1$:  MOV      #632,TOCK     ;SET TIMER FOR 5SEC.
1919 021230 004737 022022  JSR      PC,WAIT ;GO WAIT
1920 021234 017737 160362 001126 MOV      @CSR,BAD      ;SAVE CSR
1921 021242 042737 000400 001126 BIC      #400,BAD      ;MASK BRQ IF PRESENT
1922 021250 162737 060201 001124 SUB      #60201,GOOD    ;CORRECT DATA
1923 021256 023737 001124 001124 CMP      GOOD,BAD      ;CORRECT NOW?
1924 021264 001455          BEQ      2$            ;BR=YES
1925 021266 104401 021274  TYPE     ,69$         ;:TYPE ASCIZ STRING
(1) 021272 000431          BR       68$         ;:GET OVER THE ASCIZ
(1)          ;:69$: .ASCIZ <200>/ERROR16! CSR INCORRECT AFTER POWER UP. CSR WAS /
(1) 021356          68$:
1926 021356 017746 160240  MOV      @CSR,-(SP)   ;:SAVE @CSR FOR TYPEOUT
(1) 021362 104402          TYPOC    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1927 021364 104401 021372  TYPE     ,71$         ;:TYPE ASCIZ STRING
(1) 021370 000406          BR       70$         ;:GET OVER THE ASCIZ
(1)          ;:71$: .ASCIZ <200>/SHOULD BE /
(1) 021406          70$:
1928 021406 013746 001124  MOV      GOOD,-(SP)   ;:SAVE GOOD FOR TYPEOUT
(1) 021412 104402          TYPOC    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1929 021414 000000          HALT
1930 021416 000674          BR       11$
1931 021420          2$:
(1) 021420 104401 021426  TYPE     ,73$         ;:TYPE ASCIZ STRING
(1) 021424 000423          BR       72$         ;:GET OVER THE ASCIZ
(1)          ;:73$: .ASCIZ <200>/THIS DT07 INTERRUPTS ON POWER FAIL!/
(1) 021474          72$:
1932 021474 000137 021006  JMP      POSTSV
1933
1934 021500          MANEND:
(1) 021500 104401 021506  TYPE     ,65$         ;:TYPE ASCIZ STRING
  
```



MAIN. MACY11 30A(1052) 16-NOV-81 17:26  
CRDTAB.P11 16-NOV-81 17:16

PAGE 12-51  
MANUAL INTERVENTION TEST

SEQ 0064

(1) 021504 000420  
(1)  
(1) 021546  
1935 021546 000000  
1936 021550 000776

BR 64\$  
64\$: .ASCIZ <200>/MANUAL INTERVENTION TEST DONE./  
64\$:  
HALT  
BR .-2  
::GET OVER THE ASCIZ

.MAIN. MACY11 30A(1052) 16-NOV-81 17:26  
CRDTAB.P11 16-NOV-81 17:16

PAGE 13

MANUAL INTERVENTION TEST

SEQ 0065

1938

: NEWTST <<INSERT NEXT TEST HERE>>



```

1942                                     .SBTTL USER ROUTINES
1943
1944
1945 021674 012737 000240 021710 CONNEC: MOV #240,2$ ;SET SWITCH
1946 021702 105777 157714 1$: TSTB @CSR ;ALL READY CONNECTED?
1947 021706 100410 BMI 3$ ;BR = YES
1948 021710 000774 2$: BR 1$ ;CONTINUE
1949 021712 012777 000001 157702 MOV #1,@CSR ;CONNECT
1950 021720 012737 000774 021710 MOV #774,2$ ;RESTORE SWITCH
1951 021726 000765 BR 1$ ;BR UNTIL CONNECTED
1952 021730 000207 3$: RTS PC ;EXIT
1953
1954 021732 012737 000004 001730 CONID: MOV #4,COVID ;CONVERT SLAVE ID
1955 021740 013700 001724 MOV CURID,RO ;SAVE CURRANT ID
1956 021744 001405 1$: BEQ 2$ ;BR IF DONE
1957 021746 000241 CLC
1958 021750 006137 001730 ROL COVID ;SHIFT ID
1959 021754 005300 DEC RO ;COUNT DOWN TO ZERO
1960 021756 000772 BR 1$
1961 021760 000207 2$: RTS PC ;EXIT
1962
1963 021762 012737 000122 022062 WAIT10: MOV #122,TOCK ;SET CLOCK
1964 021770 000137 022022 JMP WAIT
1965 021774 012737 000244 022062 WAIT20: MOV #244,TOCK ;SET CLOCK
1966 022002 000137 022022 JMP WAIT
1967 022006 000000 TRIG: 000
1968 022010 012737 000051 022062 WAIT05: MOV #51,TOCK ;SET CLOCK
1969 022016 000137 022022 JMP WAIT
1970 022022 013737 001746 022060 WAIT: MOV TIMA,TICK ;SET BASE TIMER
1971 022030 005777 157566 1$: TST @CSR
1972 022034 000400 BR +2
1973 022036 005337 022060 DEC TICK
1974 022042 001372 BNE 1$
1975 022044 005337 022062 DEC TOCK
1976 022050 001364 BNE WAIT
1977 022052 005237 022062 INC TOCK
1978 022056 000207 RTS PC
1979 022060 000000 TICK: 0
1980 022062 000000 TOCK: 0
1981
1982
1983
1984
1985 022064 005000 SYNCUP: CLR RO ;SET DELAY TIME
1986 022066 012701 000400 MOV #400,R1
1987 022072 032777 020000 157522 1$: BIT #20000,@CSR ;BUSS ACTIVE?
1988 022100 001434 BEQ 2$ ;BR=NO
1989 022102 005200 INC RO ;TIME OUT
1990 022104 001372 BNE 1$
1991 022106 005301 DEC R1
1992 022110 001370 BNE 1$
1993 022112 104401 022120 TYPE ,65$ ;:TYPE ASCIZ STRING
(1) 022116 000420 BR 64$ ;:GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>/SYNCRONIZATION FAILURE! CSR= /
(1) 64$:
1994 022160 017746 157436 MOV @CSR,-(SP) ;:SAVE @CSR FOR TYPEOUT

```

```

(1) 022164 104402          TYPCC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1995 022166 000000          HALT
1996 022170 000776          BR          -2
1997 022172 005000          2$: CLR      R0          ;RESET DELAY TIME
1998 022174 012701 000000  MOV      #0,R1
1999 022200 032777 020000 157414 3$: BIT      #20000,@CSR  ;BUSS ACTIVE?
2000 022206 001043          BNE      4$          ;BR=YES
2001 022210 005200          INC      R0          ;TIME OUT
2002 022212 001372          BNE      3$
2003 022214 005301          DEC      R1
2004 022216 001370          BNE      3$
2005 022220 104401 022226  TYPE      ,67$          ;;TYPE ASCIZ STRING
(1) 022224 000427          BR      66$          ;;GET OVER THE ASCIZ
(1) 022304          ;;67$: .ASCIZ <200>/MASTER PORT DIDN'T TAKE SHARED BUSS! CSR= /
(1) 022304 017746 157312  MOV      @CSR,-(SP)  ;;SAVE @CSR FOR TYPEOUT
(1) 022310 104402          TYPCC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2007 022312 000000          HALT
2008 022314 000776          BR      -2
2009 022316 000207          4$: RTS      PC          ;EXIT
2010
2011
2012
2013          ;CONSTRUCT SEQUENCER
2014 022320 012701 001644  SEQMAK: MOV      #PORTNO,R1  ;STORE ADDRESS OF DATA
2015 022324 012702 001732  MOV      #TEMP1,R2      ;SET TEMPORARY STORAGE POINTER
2016 022330 012703 001660  MOV      #SEQ,R3        ;LOAD FIRST SEQUENCER ADDRESS
2017 022334 005037 001656  CLR      EXPTCT          ;CLEAR PORT COUNT
2018 022340 012122          1$: MOV      (R1)+,(R2)+  ;FILL TEMP
2019 022342 005237 001656  INC      EXPTCT          ;COUNT THEM
2020 022346 005711          TST      @R1            ;LAST ONE
2021 022350 100373          BPL      1$            ;BR=NO
2022 022352 012712 177777  MOV      #-1,@R2        ;TERMINATE DATA
2023 022356 005001          CLR      R1            ;LOAD DETECTOR
2024 022360 012702 001732  2$: MOV      #TEMP1,R2  ;RESET DATA POINTER
2025 022364 020112          3$: CMP      R1,(R2)    ;ID FOUND?
2026 022366 001407          BEQ      5$            ;BR=YES
2027 022370 005722          4$: TST      (R2)+      ;STEP AND TEST FOR LAST
2028 022372 100374          BPL      3$            ;BR IF NOT DONE
2029 022374 005201          INC      R1            ;STEP DETECTOR
2030 022376 022701 000004  CMP      #4,R1          ;CHECK FOR DONE
2031 022402 001366          BNE      2$            ;BR IF NOT DONE
2032 022404 000404          BR      6$            ;FINISH UP
2033 022406 011223          5$: MOV      @R2,(R3)+  ;LOAD SEQUENCER
2034 022410 012712 000400  MOV      #400,@R2      ;ERASE ENTRY
2035 022414 000765          BR      4$            ;CONTINUE
2036 022416 005337 001656  6$: DEC      EXPTCT      ;CORRECT EXTERNAL PORT COUNT
2037 022422 012713 177777  MOV      #-1,@R3        ;TERMINATE SEQUENCER
2038 022426 000207          RTS      PC          ;EXIT
2039
2040
2041
2042          ;ROUTINE TO ASSEMBLE MENUS
2043 022430 012737 177777 001744  MASMEN: MOV      #-1,ONOFF  ;SET MASTER/SLAVE INDICATOR
2044 022436 004737 023124  JSR      PC,FLUSHM      ;GO FLUSH MENU
2045 022442 012701 001676  MOV      #MENU,R1      ;SET POINTER

```

```

2046 022446 117721 157222      1$:  MOVB  @NEXENT,(R1)+  ;LOAD MENU WITH SLAVE ID
2047 022452 112721 000001      MOVB  #1,(R1)+      ;LOAD FUNCTION #1
2048 022456 004737 022764      JSR   PC,STPNEX    ;GET NEXT SLAVE ID
2049 022462 000771          BR    1$          ;CONTINUE
2050 022464 004737 022764      2$:  JSR   PC,STPNEX    ;GO GET NEXT ID
2051 022470 000405          BR    3$          ;BR=NOT FINISHED
2052 022472 012711 177777      MOV   #-1,@R1     ;TERMINATE MENU
2053 022476 012704 001676      MOV   #MENU,R4    ;STORE MENUS
2054 022502 000207          RTS   PC          ;EXIT
2055 022504 117721 157164      3$:  MOVB  @NEXENT,(R1)+
2056 022510 112721 000002      MOVB  #2,(R1)+    ;LOAD FUNCTION #2
2057 022514 000763          BR    2$          ;CONTINUE
2058
2059
2060 022516 005037 001744      SALMEN: CLR  ONOFF    ;CLR MASTER/SLAVE INDICATOR
2061 022522 004737 023124      JSR   PC,FLUSHM   ;GO FLUSH MENU
2062 022526 012701 001676      MOV   #MENU,R1    ;SET POINTER
2063 022532 027737 157136 001644 1$:  CMP   @NEXENT,PORTNO ;IS THIS ME?
2064 022540 001010          BNE   3$          ;BR=NO
2065 022542 117721 157126      MOVB  @NEXENT,(R1)+ ;LOAD SLAVE ID
2066 022546 112721 000003      MOVB  #3,(R1)+    ;LOAD FUNCTION 3
2067 022552 004737 022764      2$:  JSR   PC,STPNEX    ;GO GET NEXT ID
2068 022556 000765          BR    1$          ;BR=NOT DONE
2069 022560 000405          BR    4$          ;GO TO NEXT SET
2070 022562 117721 157106      3$:  MOVB  @NEXENT,(R1)+
2071 022566 112721 000005      MOVB  #5,(R1)+    ;LOAD FUNCTION 5
2072 022572 000767          BR    2$          ;CONTINUE
2073 022574 004737 022764      4$:  JSR   PC,STPNEX    ;GO GET NEXT ID
2074 022600 000405          BR    5$          ;GO CONTINUE
2075 022602 012711 177777      MOV   #-1,@R1     ;TERMINATE MENU
2076 022606 012704 001676      MOV   #MENU,R4    ;STORE MENUS
2077 022612 000207          RTS   PC          ;EXIT
2078 022614 027737 157054 001644 5$:  CMP   @NEXENT,PORTNO ;IS THIS ME?
2079 022622 001014          BNE   7$          ;BR=NO
2080 022624 117721 157044      MOVB  @NEXENT,(R1)+
2081 022630 112721 000004      MOVB  #4,(R1)+    ;LOAD FUNCTION 4
2082 022634 004737 022764      6$:  JSR   PC,STPNEX    ;GO GET NEXT ID
2083 022640 000765          BR    5$          ;CONTINUE
2084 022642 012711 177777      MOV   #-1,@R1     ;TERMINATE MENU
2085 022646 012704 001676      MOV   #MENU,R4    ;STORE MENUS
2086 022652 000207          RTS   PC          ;EXIT
2087 022654 117721 157014      7$:  MOVB  @NEXENT,(R1)+
2088 022660 112721 000005      MOVB  #5,(R1)+    ;LOAD FUNCTION 5
2089 022664 000763          BR    6$          ;CONTINUE
2090
2091
2092          ;ROUTINE TO INIT SEQUENCER
2093
2094 022666 012737 001660 001672 SEQINT: MOV   #SEQ,SEQPTR  ;SET POINTER TO FIRST ENTRY
2095 022674 012737 001662 001674      MOV   #SEQ+2,NEXENT ;GET NEXT ENTRY
2096 022702 017705 156764      MOV   @SEQPTR,R5   ;LOAD R5
2097 022706 000207          RTS   PC          ;EXIT
2098 022710 062737 000002 001672 SEQSTP: ADD  #2,SEQPTR  ;STEP SEQUENCER
2099 022716 005777 156750      TST   @SEQPTR      ;CHECK FOR END
2100 022722 100761          BMI   SEQINT      ;RELOAD IF DONE
2101 022724 017705 156742      MOV   @SEQPTR,R5

```

```

2102 022730 013737 001672 001674      MOV      SEQPTR,NEXENT      ;STEP NEXT ENTRY
2103 022736 062737 000002 001674      ADD      #2,NEXENT
2104 022744 005777 156724                    TST      @NEXENT           ;OFF THE END?
2105 022750 100401                    BMI      2$                ;BR=YES
2106 022752 000207                    PC       PC                ;NO! EXIT
2107 022754 012737 001660 001674 1$:      MOV      #SEQ,NEXENT      ;CORRECT POINTER
2108 022762 000773                    BR       1$                ;EXIT
2109 022764 062737 000002 001674 STPNEX: ADD      #2,NEXENT      ;STEP NEXT ENTRY
2110 022772 005777 156676                    TST      @NEXENT           ;OFF THE END?
2111 022776 100003                    BPL      1$                ;BR=NO
2112 023000 012737 001660 001674      MOV      #SEQ,NEXENT      ;RESET
2113 023006 027705 156662 1$:      CMP      @NEXENT,R5        ;BACK TO CURRENT DT07?
2114 023012 001002                    BNE      2$                ;BR=NO
2115 023014 062716 000002                    ADD      #2,@SP
2116 023020 000207                    RTS      PC                ;SET UP FINISHED RETURN
2117                                     ;EXIT
2118
2119 023022 011637 001126      OOPS:   MOV      @SP,BAD      ;SAVE ERROR PC
2120 023026 104401 023034      TYPE    ,65$                ;:TYPE ASCIZ STRING
(1) 023032 000427      BR       64$                ;:GET OVER THE ASCIZ
(1)                                     ;:65$: .ASCIZ <200>/UNEXPECTED INTERRUPT FROM DT07 AT ADDRESS.../
(1) 64$:
2121 023112 013746 001126      MOV      BAD,-(SP)          ;:SAVE BAD FOR TYPEOUT
(1) 023116 104402      TYPOC   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2122 023120 000000      HALT
2123 023122 000776      BR       -2
2124
2125 023124 012700 001676      FLUSHM: MOV      #MENU,R0      ;SET POINTER
2126 023130 012701 000013      MOV      #11,R1           ;SET FLUSH COUNT
2127 023134 012720 177777 1$:      MOV      #-1,(R0)+        ;FLUSH WITH -1
2128 023140 005301      DEC     R1                ;COUNT
2129 023142 001374      BNE     1$                ;CONT.
2130 023144 000207      RTS     PC                ;EXIT
2131 023146 012700 001634      FLUSHP: MOV      #DT03,R0      ;LOAD POINTER
2132 023152 012701 000017      MOV      #15,R1           ;SET COUNT
2133 023156 012720 177777      MOV      #-1,(R0)+        ;FLUSH
2134 023162 005020      CLR     (R0)+
2135 023164 012720 177777 1$:      MOV      #-1,(R0)+        ;FLUSH WITH -1
2136 023170 005301      DEC     R1                ;COUNT
2137 023172 001374      BNE     1$                ;CONT
2138 023174 000207      RTS     PC                ;EXIT
2139
2140
2141 023176                                     DMODE:
2142 023176 012737 000340 177776 3$:      MOV      #340,PSW         ;RISE PRIORITY
2143 023204 012777 000001 156410      MOV      #1,@CSR          ;CONNECT
2144 023212 017737 156404 001126 4$:      MOV      @CSR,BAD         ;SAVE CSR
2145 023220 032737 000400 001126      BIT      #400,BAD         ;LOOK FOR BRQ
2146 023226 001010      BNE     5$                ;BR=FOUND
2147 023230 032737 100000 001126      BIT      #100000,BAD      ;DID WE TIME OUT?
2148 023236 001765      BEQ     4$                ;BR=NO
2149 023240 012737 177777 001634      MOV      #-1,DT03         ;INDIACATE DT03 MODE
2150 023246 000531      BR       8$                ;EXIT
2151 023250 005037 001634 5$:      CLR     DT03              ;INDIACATE DT07 MODE
2152 023254 013700 001126      MOV      BAD,R0           ;SAVE IMAGE
2153 023260 042700 177703      BIC     #177703,R0        ;SAVE ONLY RQ0-3

```

```

2154 023264 000241          CLC          ;CONVERT RQ BIT
2155 023266 006000          ROR          R0
2156 023270 006000          ROR          R0
2157 023272 005002          CLR          R2
2158 023274 032700 000001    6$:  BIT        #1,R0      ;BIT SET?
2159 023300 001051          BNE          7$          ;BR=YES
2160 023302 005202          INC          R2          ;STEP INDIACATOR
2161 023304 000241          CLC
2162 023306 006000          ROR          R0
2163 023310 022702 000004    CMP        #4,R2      ;TOO HIGH?
2164 023314 101367          BHI        6$          ;BR =NO
2165 023316 104401 023324    TYPE      ,65$        ;:TYPE ASCIZ STRING
(1) 023322 000433          BR         64$        ;:GET OVER THE ASCIZ
(1)                                ;:BRQ DETECTED WITHOUT ANY RQ. BIT SET.  CSR= /
(1) 023412          ;:65$: .ASCIZ <200>/ERROR!
2166 023412 013746 001126    64$:  MOV        BAD,-(SP) ;:SAVE BAD FOR TYPEOUT
(1) 023416 104402          TYPOC
2167 023420 000000          HALT        ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2168 023422 000776          BR         -2        ;:FATAL ERROR BRQ SEEN BUT NO RQ
2169 023424 010237 001644    7$:  MOV        R2,PORTNO ;:LOAD PORT ID#
2170 023430 000241          CLC
2171 023432 006000          ROR          R0          ;MAKE SURE ONLY ONE RQ IS PRESENT
2172 023434 001436          BEQ        8$          ;BR=OK
2173 023436 104401 023444    TYPE      ,67$        ;:TYPE ASCIZ STRING
(1) 023442 000426          BR         66$        ;:GET OVER THE ASCIZ
(1)                                ;:MORE THAN ONE RQ. BIT SET.  CSR= /
(1) 023520          66$:  MOV        BAD,-(SP) ;:SAVE BAD FOR TYPEOUT
2174 023520 013746 001126    TYPOC        ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 023524 104402          HALT
2175 023526 000000          BR         -2
2176 023530 000776          CLR        @CSR
2177 023532 005077 156064    8$:  RTS        PC          ;CLEAR AND DISCONNECT AND EXIT
2178 023536 000207
2179
2180
2181 023540 012737 000340 177776 DTIME: MOV        #340,PSW ;:LOCK OUT INTERUPTS
2182 023546 005037 001746          CLR        TIMA      ;:CLEAR TIMERS
2183 023552 005077 156044          CLR        @CSR     ;:CLEAR DEVICE
2184 023556 105777 156040    1$:  TSTB       @CSR     ;:MAKE SURE IT'S CLEAR
2185 023562 100775          BMI        1$
2186 023564 005277 156032          INC        @CSR     ;:SET REQ
2187 023570 005777 156026    2$:  TST        @CSR     ;:TMO SET?
2188 023574 100424          BMI        3$          ;:BR=YES
2189 023576 005237 001746          INC        TIMA     ;:COUNT
2190 023602 001372          BNE        2$
2191 023604 104401 023612    TYPE      ,65$        ;:TYPE ASCIZ STRING
(1) 023610 000415          BR         64$        ;:GET OVER THE ASCIZ
(1)                                ;:TIMING TOO LONG!/
(1) 023644          ;:65$: .ASCIZ <200>/ERROR!
2192 023644 000000          HALT
2193 023646 005037 177776    3$:  CLR        PSW      ;:LOWER PRIORITY
2194 023652 012777 000001 155742    MOV        #1,@CSR   ;:CONNECT
2195 023660 105777 155736    4$:  TSTB       @CSR     ;:CONNECTED?
2196 023664 100375          BPL        4$
2197 023666 005037 001750          CLR        TIMB     ;:CLAER WATCHDOG TIMER
2198 023672 012737 000340 177776    MOV        #340,PSW  ;:RISE PRIORITY
  
```



```

2199 023700 005377 155716          DEC      @CSR      ;DROP REQ
2200 023704 005777 155712          5$:    TST      @CSR      ;LOOK FOR TMO
2201 023710 100430                   BMI      6$        ;BR=FOUND
2202 023712 005237 001750          INC      TIMB
2203 023716 001372                   BNE      5$
2204 023720 104401 023726          TYPE    ,67$      ;;TYPE ASCIZ STRING
(1) 023724 000421                   BR      66$      ;;GET OVER THE ASCIZ
(1)                                ;;67$: .ASCIZ <200>/ERROR! WATCHDOG TIMING TOO LONG!/
(1)                                66$:
2205 023770 000000                   HALT
2206 023772 013700 001746          6$:    MOV      TIMA,R0      ;COMPARE TIMING FOR PROPER RATIO
2207 023776 013701 001750          MOV      TIMB,R1
2208 024002 005002                   CLR      R2
2209 024004 012703 000031          MOV      #25,R3
2210 024010 060002          7$:    ADD      R0,R2      ;MULTIPLY TIMA BY 25
2211 024012 005303                   DEC      R3
2212 024014 001375                   BNE      7$        ;BR=NOT DONE
2213 024016 160102          8$:    SUB      R1,R2      ;DIVIDE TIMA BY TIMB
2214 024020 005203                   INC      R3
2215 024022 020102                   CMP      R1,R2      ;DONE?
2216 024024 101774                   BLOS    8$        ;BR=NO
2217 024026 020327 000033          CMP      R3,#33     ;TOO LOW?
2218 024032 103034                   BHIS    9$        ;BR=NO
2219 024034 104401 024042          TYPE    ,69$      ;;TYPE ASCIZ STRING
(1) 024040 000427                   BR      68$      ;;GET OVER THE ASCIZ
(1)                                ;;69$: .ASCIZ <200>%ERROR! WATCHDOG/REQUEST TIMER RATIO TOO LOW!%
(1)                                68$:
2220 024120 000000                   HALT
2221 024122 000776                   BR      -2
2222 024124 020327 000044          9$:    CMP      R3,#44     ;TOO HIGH?
2223 024130 101435                   BLOS    11$      ;BR=NO
2224 024132 104401 024140          TYPE    ,71$      ;;TYPE ASCIZ STRING
(1) 024136 000430                   BR      70$      ;;GET OVER THE ASCIZ
(1)                                ;;71$: .ASCIZ <200>%ERROR! WATCHDOG/REQUEST TIMER RATIO TOO HIGH!%
(1)                                70$:
2225 024220 000000                   HALT
2226 024222 000776                   BR      -2
2227 024224 005077 155372          11$:   CLR      @CSR      ;CLEAR AND DISCONNECT
2228 024230 000207                   RTS      PC
2229
2230
2231                                ;PROGRAM PARAMETER INPUT ROUTINE
2232
2233 024232 012737 177777 026426 GETADR: MOV      #-1,PARFLG  ;SET PARFLG
2234 024240 005037 001760          CLR      USEPAR     ;CLEAR FLAG
2235 024244 004737 023146          JSR      PC,FLUSHP  ;GO FLUSH OUT PARAMETER BLOCK
2236 024250 104401 024256          TYPE    ,65$      ;;TYPE ASCIZ STRING
(1) 024254 000433                   BR      64$      ;;GET OVER THE ASCIZ
(1)                                ;;65$: .ASCIZ <200>/DO YOU WANT TO CHANGE THE DEVICE ADDRESSES?(Y OR N)/
(1)                                64$:
2237 024344 104410                   RDCHR
2238 024346 012600                   MOV      (SP)+,R0   ;STORE CHAR.
2239 024350 110037 026430          MOV      R0,ECHOB
2240 024354 104401 026430          TYPE    ,ECHOB
2241 024360 022700 000131          CMP      #'Y,R0     ;YES?
2242 024364 001173                   BNE     10$        ;BR=NO

```

```

2243 024366 104401 024374      TYPE      ,67$      ::TYPE ASCIZ STRING
(1) 024372 000444      BR      66$      ::GET OVER THE ASCIZ
(1) 024504      ::67$: .ASCIZ <200>/IN THE FOLLOWING THREE PRINTOUTS TYPE THE NEW DATA OR 0 IF NO CHAN
(1) 024504 104401 024512      66$: TYPE      ,69$      ::TYPE ASCIZ STRING
(1) 024510 000411      BR      68$      ::GET OVER THE ASCIZ
(1) 024534      ::69$: .ASCIZ <200>/DEVICE ADDRESS= /
(1) 024534 013746 001622      68$: MOV      CSR,-(SP)      ::SAVE CSR FOR TYPEOUT
(1) 024540 104402      TYPOC      ::GO TYPE--OCTAL ASCII(ALL DIGITS)
2246 024542 104401 034647      TYPE      ,SPACE
2247 024546 104412      RDOCT
2248 024550 012600      MOV      (SP)+,R0      ;STORE INFO
2249 024552 001402      BEQ      11$
2250 024554 010037 001262      MOV      R0,$BASE
2251 024560      11$:
(1) 024560 104401 024566      TYPE      ,71$      ::TYPE ASCIZ STRING
(1) 024564 000411      BR      70$      ::GET OVER THE ASCIZ
(1) 024610      ::71$: .ASCIZ <200>/DEVICE VECTOR= /
(1) 024610 013700 001256      70$: MOV      $VECT1,R0
2253 024614 042700 177000      BIC      #177000,R0
2254 024620 010046      MOV      R0,-(SP)      ::SAVE R0 FOR TYPEOUT
(1) 024622 104402      TYPOC      ::GO TYPE--OCTAL ASCII(ALL DIGITS)
2255 024624 104401 034647      TYPE      ,SPACE
2256 024630 104412      RDOCT
2257 024632 012600      MOV      (SP)+,R0
2258 024634 001402      BEQ      12$
2259 024636 010001      MOV      R0,R1      ;SAVE IT TEMPORARILY
2260 024640 000404      BR      14$      ;JUMP OVER NEXT TWO LINES
2261 024642 013701 001256      12$: MOV      $VECT1,R1      ;SAVE OLD VECTOR ADDRESS
2262 024646 042701 177000      BIC      #177000,R1      ;MASK OUT JUNK
2263 024652      14$:
(1) 024652 104401 024660      TYPE      ,73$      ::TYPE ASCIZ STRING
(1) 024656 000410      BR      72$      ::GET OVER THE ASCIZ
(1) 024700      ::73$: .ASCIZ <200>/BREAK LEVEL= /
(1) 024700 013700 001256      72$: MOV      $VECT1,R0
2265 024704 042700 000777      BIC      #777,R0
2266 024710 000300      SWAB      R0
2267 024712 010046      MOV      R0,-(SP)      ::SAVE R0 FOR TYPEOUT
(1) 024714 104402      TYPOC      ::GO TYPE--OCTAL ASCII(ALL DIGITS)
2268 024716 104401 034647      TYPE      ,SPACE
2269 024722 104412      RDOCT
2270 024724 012600      MOV      (SP)+,R0
2271 024726 001405      BEQ      13$
2272 024730 000300      SWAB      R0
2273 024732 060100      ADD      R1,R0
2274 024734 010037 001256      MOV      R0,$VECT1
2275 024740 000405      BR      10$
2276 024742 042737 000777 001256 13$: BIC      #777,$VECT1
2277 024750 060137 001256      ADD      R1,$VECT1
2278 024754      10$:
2279 024754 000207      RTS      PC      ;RETURN
2280 024756      PARIN:
2281 024756 004737 024232      JSR      PC,GETADR
  
```

```

2282 024762 012737 024774 002672      MOV      #REPAR,ROUTE      ;SET RETURN
2283
2284 024770 000137 001762      JMP      START
2285      024774      REPAR=.
2286 024774 004737 023176      JSR      PC,DMODE          ;GO GET MODE AND ID#
2287 025000 004737 023540      JSR      PC,DTIME          ;GO GET DEVICE TIMING
2288 025004 005737 001634      TST      DT03              ;WHAT MODE?
2289 025010 001422      BEQ      20$              ;BR=DT07
2290 025012 104401 025020      TYPE     ,65$             ;;TYPE ASCIZ STRING
(1) 025016 000416      BR       64$             ;;GET OVER THE ASCIZ
(1)      ;;65$: .ASCIZ <200>/THIS DT07 IS IN DT03 MODE./
(1) 025054 64$:
2291 025054 000421      BR       1$              ;CONTINUE AT 1$
2292 025056 20$:
(1) 025056 104401 025064      TYPE     ,67$             ;;TYPE ASCIZ STRING
(1) 025062 000413      BR       66$             ;;GET OVER THE ASCIZ
(1)      ;;67$: .ASCIZ <200>/THIS DT07 IS PORT# /
(1) 025112 66$:
2293 025112 013746 001644      MOV      PORTNO,-(SP)     ;;SAVE PORTNO FOR TYPEOUT
(1) 025116 104402      TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2294 025120 1$:
(1) 025120 104401 025126      TYPE     ,69$             ;;TYPE ASCIZ STRING
(1) 025124 000447      BR       68$             ;;GET OVER THE ASCIZ
(1)      ;;69$: .ASCIZ <200>/TYPE THE ADDRESS OF AN EXTERNAL DEVICE REGISTER ON THE SHARED BUSS
(1) 025244 68$:
2295 025244 104412      RDOCT
2296 025246 012637 001636      MOV      (SP)+,DEVAD      ;STORE IN DEVAD
2297 025252 001523      BEQ      2$              ;SKIP OVER NEXT TWO QUESTIONS IF NO DEVICE
2298 025254 104401 025262      TYPE     ,71$             ;;TYPE ASCIZ STRING
(1) 025260 000442      BR       70$             ;;GET OVER THE ASCIZ
(1)      ;;71$: .ASCIZ <200> % TYPE AN OCTAL MASK OF READ/WRITE BITS IN ABOVE DEVICE REGISTER..
(1) 025366 70$:
2299 025366 104412      RDOCT
2300 025370 012637 001640      MOV      (SP)+,DEV RW     ;STORE BIT PATTERN
2301 025374 104401 025402      TYPE     ,73$             ;;TYPE ASCIZ STRING
(1) 025400 000445      BR       72$             ;;GET OVER THE ASCIZ
(1)      ;;73$: .ASCIZ <200> % TYPE AN OCTAL MASK OF RESET-CLEARABLE BITS IN ABOVE DEVICE REGIS
(1) 025514 72$:
2302 025514 104412      RDOCT
2303 025516 012637 001642      MOV      (SP)+,DEV RC     ;STORE BIT PATTERN
2304 025522 005737 001634      TST      DT03            ;NO NEED TO CONTINUE IF DT03 MODE
2305 025526 001067      BNE      7$              ;BR=DT03
2306 025530 3$:4$:
(1) 025530 104401 025536      TYPE     ,75$             ;;TYPE ASCIZ STRING
(1) 025534 000427      BR       74$             ;;GET OVER THE ASCIZ
(1)      ;;75$: .ASCIZ <200>/HOW MANY OTHER PORTS ARE THERE TO BE TESTED?/
(1) 025614 74$:
2307 025614 104412      RDOCT
2308 025616 012637 001656      MOV      (SP)+,EXPTCT     ;STORE EXTERNAL PORT COUNT
2309 025622 001431      BEQ      7$              ;GO TO 7$
2310 025624 012700 001646      MOV      #EXPT1,R0        ;SET POINTER
2311 025630 013701 001656      MOV      EXPTCT,R1
2312 025634 6$:
(1) 025634 104401 025642      TYPE     ,77$             ;;TYPE ASCIZ STRING
(1) 025640 000416      BR       76$             ;;GET OVER THE ASCIZ
(1)      ;;77$: .ASCIZ <200>/TYPE IN EXTERNAL PORT ID../

```

```

(1) 025676          76$:
2313 025676 104412      RDOCT
2314 025700 012620      MOV      (SP)+,(R0)+      :STORE IT
2315 025702 005301      DEC      R1              :COUNT IT
2316 025704 001353      BNE      6$             :CONTINUE LOAD IF NOT DONE
2317 025706          7$:
(1) 025706 104401 025714  TYPE      79$           ;;TYPE ASCIZ STRING
(1) 025712 000422      BR      78$           ;;GET OVER THE ASCIZ
(1)          ;;79$: .ASCIZ <200>/SELECT TEST. (0 FOR HELP MESSAGE)/
(1) 025760          78$:
2318 025760 104412      RDOCT
2319 025762 012600      MOV      (SP)+,R0
2320 025764 001003      BNE      8$
2321 025766 104401 034651  TYPE      SELMS
2322 025772 000745      BR      7$
2323 025774 022700 000004  8$:      CMP      #4,R0          ;TEST FOR NPR?
2324 026000 001002      BNE      18$          ;NO:BR
2325 026002 000137 026652  JMP      TSTNPR        ;GO CHECK FOR NPR LINE
2326 026006 022700 000004  18$:     CMP      #4,R0          ;VALID TEST?
2327 026012 101016      BHI      9$
2328 026014 104401 026022  TYPE      81$           ;;TYPE ASCIZ STRING
(1) 026020 000412      BR      80$           ;;GET OVER THE ASCIZ
(1)          ;;81$: .ASCIZ <200>/INVALID SELECTION!/
(1) 026046          80$:
2329 026046 000717      BR      7$
2330 026050 022700 000002  9$:      CMP      #2,R0          ;MULTIPOINT?
2331 026054 001044      BNE      21$          ;BR=NO
2332 026056 005737 001656  TST      EXPTCT        ;ONLY ONE PORT?
2333 026062 001041      BNE      21$          ;BR=NO
2334 026064 005737 001634  TST      DT03          ;DT03 MODE?
2335 026070 001036      BNE      21$          ;BR=YES
2336 026072 104401 026100  TYPE      83$           ;;TYPE ASCIZ STRING
(1) 026076 000432      BR      82$           ;;GET OVER THE ASCIZ
(1)          ;;83$: .ASCIZ <200>/CAN NOT RUN THE MULTIPOINT TEST WITH ONLY ONE PORT!/
(1) 026164          82$:
2337 026164 000650      BR      7$
2338 026166 022700 000002  21$:     CMP      #2,R0          ;MULTIPOINT TEST?
2339 026172 001057      BNE      22$          ;BR=NO
2340 026174 104401 026202  TYPE      85$           ;;TYPE ASCIZ STRING
(1) 026200 000454      BR      84$           ;;GET OVER THE ASCIZ
(1)          ;;85$: .ASCIZ <200>/THE DT07 WITH THE LOWEST PORT# MUST BE STARTED LAST WHEN RUNNING/<
(1) 026332          84$:
2341 026332 000241      CLC
2342 026334 006100      ROL      R0
2343 026336 016016 026434  MOV      TX(R0),(SP)
2344 026342 104401 026350  TYPE      87$           ;;TYPE ASCIZ STRING
(1) 026346 000424      BR      86$           ;;GET OVER THE ASCIZ
(1)          ;;87$: .ASCIZ <200>/TYPE <CR> WHEN READY TO CONTINUE TEST./
(1) 026420          86$:
2345 026420 104410      RDCHR
2346 026422 012600      MOV      (SP)+,R0
2347 026424 000207      RTS      PC
2348 026426 000000      PARFLG: 000          ;EXIT
2349 026430      000      200      000  ECHOB: .BYTE 0,200,0 ;PARAMETERS PRESENT FLAG
2350          026434      .EVEN          ;ECHO BYTE
2351 026434 000000      TX:      0
  
```

```

2352 026436 002566 ROUT1
2353 026440 002600 ROUT2
2354 026442 002636 ROUT4
2355
2356
2357 026444 TRAP4:
(2) 026444 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
2358 026446 104401 026454 TYPE ,65$ ;;TYPE ASCIZ STRING
(1) 026452 000422 RR 64$ ;;GET OVER THE ASCIZ
(1) ;;65$: .ASCIZ <200>/UNEXPECTED TRAP TO VECTOR 4 FROM../
(1) 64$:
2359 026520 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
(1) 026522 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2360 026524 000000 HALT
2361 026526 000137 001762 JMP START
2362
2363 026532 ERRROUT:
2364 026532 016637 000004 000004 MOV 4(SP),@#ERRVEC ;ADJUST STACK
2365 026540 005237 026560 INC FLAG
2366 026544 016666 000002 000004 MOV 2(SP),4(SP) ;PS
2367 026552 011666 000002 MOV (SP),2(SP) ;PC
2368 026556 000002 RTI
2369
2370 026560 000000 FLAG: 0
2371
2372 026562 010246 CKADR: MOV R2,-(SP) ;SAVE R2
2373 026564 013702 001636 MOV DEVAD,R2 ;DEVICE ADDRESS IN R2
2374 026570 013746 000004 MOV @#ERRVEC,-(SP) ;SAVE OLD TRAP ADDRESS
2375 026574 012737 026532 000004 MOV @#ERRROUT,@#ERRVEC ;NEW TRAP ADDRESS
2376 026602 005037 026560 CLR FLAG ;INITIALIZE FLAG
2377 026606 005712 TST (R2) ;IS ADDRESS ADDRESSABLE?
2378 026610 000240 NOP
2379 026612 005726 TST (SP)+ ;RESTORE OLD TRAP ADDRESS
2380 026614 010226 MOV R2,(SP)+ ;RESTORE R2
2381 026616 000207 RTS PC
2382
2383 026620 CNTLC:
2384 026620 105777 152320 TSTB @#TKS ;IS CHAR THERE?
2385 026624 100011 BPL 1$ ;NO RETURN
2386 026626 117746 152314 MOVB @#TKB,-(SP) ;PUT CHAR ON STACK
2387 026632 042716 177600 BIC #^C177,(SP) ;STRIP OFF UNWANTED BITS
2388 026636 022726 000003 CMP #3,(SP)+ ;IS IT CONTROL C?
2389 026642 001002 BNE 1$ ;NO RETURN
2390 026644 000137 002612 JMP ROUT3 ;GO TO START AGAIN
2391 026650 000207 1$: RTS PC
2392

```

```

2394
2395      ;CHECK OUT NPR LINE ON THE SWITCH BUS
2396
2397 026652      ;TSTNPR:
2398 026652 005037 027404      CLR PASS      ;CLEAR PASS COUNT
2399 026656 005037 027410      CLR NUM      ;CLEAR NUM
2400 026662 104401 026670      TYPE ,65$    ;TYPE ASCIZ STRING
      (1) 026666 000417      BR 64$      ;GET OVER THE ASCIZ
      (1)      ;:65$: .ASCIZ <200>/TEST NPR LINE ON SWITCH BUS/
      (1) 026726      64$:
2401 026726 104401 026734      TYPE ,67$    ;TYPE ASCIZ STRING
      (1) 026732 000415      BR 66$      ;GET OVER THE ASCIZ
      (1)      ;:67$: .ASCIZ <200>/TYPE CONTROL C TO EXIT/<200>
      (1) 026766      66$:
2402 026766      NPR:
2403 026766 004737 026620      JSR PC,CNTLC  ;IS IT CONTROL C?
2404 026772 005237 027410      INC NUM      ;COUNT NUMBER OF ITERATION
2405 026776 012737 000003 027406  MOV #3,BRCNT  ;COUNTER FOR BR LEVEL
2406 027004 012702 027460      MOV #BRLV,R2 ;POINTER TO BRLEVEL TABLE
2407 027010 012706 001100      MOV #STACK,SP ;RESTORE STACK
2408 027014 012737 000000 177776  MOV #0,PSW   ;DROP PRIORITY
2409 027022 012777 000001 152572  MOV #REQ,@CSR ;CONNECT
2410 027030 105777 152566      7$: TSTB @CSR  ;CONNECTED?
2411 027034 100421      BMI 8$      ;BR IF YES
2412 027036 005777 152560      TST @CSR    ;TIME OUT?
2413 027042 100372      BPL 7$      ;BR IF NOT
2414 027044 017737 152552 001126  MOV @CSR,BAD ;STORE CONTENT OF CSR
2415 027052 013737 001752 001124  MOV KONST,GOOD ;
2416 027060 062737 020201 001124  ADD #20201,GOOD ;STORE GOOD DATA
2417 027066 104006      ERROR+6    ;COUNTN'T CONNECT
2418 027070 005077 152526      CLR @CSR    ;CLEAR CSR
2419 027074 000137 002612      JMP ROUT3
2420
2421 027100      8$:
2422 027100 004737 026620      JSR PC,CNTLC  ;
2423 027104 022702 027470      CMP #BREND,R2 ;IS IT END OF TABLE?
2424 027112 012706 001100      BEQ 5$      ;YES:BR
2425 027116 012737 000340 177776  MOV #STACK,SP ;RESTORE STACK
2426 027124 004737 030020      MOV #340,@#PSW ;LOCK OUT INTERRUPT
2427 027130 005037 027434      JSR PC,CLRREG ;CLEAR ALL UBE REGISTER
2428 027134 012777 177777 000250  CLR BUFF1    ;CLEAR BUFF LOCATION
2429 027142 012777 027434 000246  MOV #177777,@BEBD ;LOAD UBE DATA REG WITH TEST DATA
2430 027150 012777 177777 000236  MOV #BUFF1,@BEBA ;LOAD UBE ADDRESS REG WITH BUFF ADR
2431 027156 012777 003041 000234  MOV #177777,@BECC ;SET UBE TO DO ONE CYCLE
2432 027164 000240      MOV #3041,@BECCR1 ;HAVE UBE SEND DATA VIA NPR
2433 027166 004737 030074      NOP        ;ALLOW UBE TO SET BUS
2434 027172 005704      JSR PC,CRDY  ;CHECK READY SET
2435 027174 001037      TST R4      ;DID RDY SET?
2436 027176 005737 027434      BNE 1$      ;BRANCH TO ERROR IF RDY DIDN'T SET
2437 027202 001445      TST BUFF1   ;WAS DATA DONE?
2438 027204 005037 027434      BEQ 2$      ;BRANCH TO ERROR IF NPR NOT DONE
2439 027210 005037 027436      CLR BUFF1   ;CLEAR BUF LOCATION
2440 027214 012777 027262 000204  CLR BUFF1+2 ;CLEAR BUF LOCATION PLUS 2
2441 027222 012777 027434 000166  MOV #3$,@INTVEC ;SETUP FOR INTERRUPT
2442 027230 012777 177777 000156  MOV #BUFF1,@BEBA ;LOAD TEST ADDRESS
2443 027236 005237 027406      MOV #177777,@BECC ;SETUP UBE TO DO 1 CYCLE
      INC BRCNT ;COUNT BR LEVEL
  
```



```
2500 027466 003161          3161          ;DATO AT BR7
2501 027470 000000          BREND: 0
2502 027472 000000          0
2503
2504 027474 051105 047522 035122 EM23: .ASCIZ /ERROR:NPR DATO NOT DONE/
      027502 050116 020122 040504
      027510 047524 047040 052117
      027516 042040 047117 000105
2505 027524 051105 047522 035122 EM24: .ASCIZ /ERROR:NPR DID NOT SET RDY/
      027532 050116 020122 044504
      027540 020104 047516 020124
      027546 042523 020124 042122
      027554 000131
2506 027556 051105 047522 035122 EM25: .ASC'Z /ERROR:UBE DID NOT INTERRUPT WHEN NPR FINISHED/
      027564 041125 020105 044504
      027572 020104 047516 020124
      027600 047111 042524 051122
      027606 050125 020124 044127
      027614 047105 047040 051120
      027622 043040 047111 051511
      027630 042510 000104
2507 027634 051105 047522 035122 EM26: .ASCIZ /ERROR:TWO LOC WRITTEN WHEN ONE NPR AND INT ON DONE ENABLE/
      027642 053524 020117 047514
      027650 020103 051127 052111
      027656 042524 020116 044127
      027664 047105 047440 042516
      027672 047040 051120 040440
      027700 042116 044440 052116
      027706 047440 020116 047504
      027714 042516 042440 040516
      027722 046102 000105
2508 027726 051105 047522 035122 EM27: .ASCIZ /ERROR:DEVICE DOESN'T EXIST ON SWITCH BUS/
      027734 042504 044526 042503
      027742 042040 042517 047123
      027750 052047 042440 044530
      027756 052123 047440 020116
      027764 053523 052111 044103
      027772 041040 051525 000
2509
2510 027777 377 377 000 NULL: .BYTE -1,-1,0 ;NULL CHARACTER STRING
2511 030002 005015 047105 020104 ENDMG: .ASCIZ <15><12>/END PASS #/
      030010 040520 051523 021440
      030016 000
2512
2513 030020 .EVEN
2514
2515 ;SUBROUTINE TO TYPE PC OF ERROR MESSAGE
2516
2517 ;TERRPC:BIT #SW13,@SWR ;INHIBIT ERROR PRINT OUT
2518 ; BNE 1$ ;BRANCH IF YES
2519 ; TYPE MSG15 ;TYPE PC OF ERROR MESSAGE
2520 ; MOV $ERRPC,-(SP) ;SAVE $ERRPC FOR TYPEOUT
2521 ; TYPC
2522 ;1$: RTS PC
2523
2524 ;SUBROUTINE TO CLEAR ALL UBE REG
```



```
2525  
2526 030020  
2527 030020 005077 177400  
2528 030024 005077 177372  
2529 030030 005077 177364  
2530 030034 005077 177356  
2531 030040 005077 177350  
2532 030044 005077 177342  
2533 030050 000207  
2534  
2535  
2536  
2537 030052 010546  
2538 030054 013705 027426  
2539 030060 005725  
2540 030062 010577 177340  
2541 030066 005015  
2542 030070 012605  
2543 030072 000207  
2544  
2545  
2546  
2547 030074 005004  
2548 030076 005005  
2549 030100 005205  
2550 030102 105777 177312  
2551 030106 100405  
2552 030110 032705 000200  
2553 030114 001771  
2554 030116 012704 000001  
2555 030122 000207  
2556  
2557  
2558
```

```
CLRREG: CLR @BERE ;CLEAR ERROR CONDITION  
CLR @BECR2 ;CLEAR BECR2 REG  
CLR @BECR1 ;CLEAR BECR1 REG  
CLR @BEBA ;CLEAR BEBA REG  
CLR @BECC ;CLEAR BECC REG  
CLR @BEDD ;CLEAR BEDD REG  
RTS PC
```

```
;SUBROUTINE TO RESTORE TRAP CATCHER TO UBE VECTOR AREA  
RCATCH: MOV R5, -(SP) ;SAVE R5 ON STACK  
MOV INTVEC, R5 ;GET INTVEC  
TST (R5)+ ;CALCULATE INTVEC+2  
MOV R5, @INTVEC ;PUT INTVEC+2 IN INTVEC  
CLR (R5) ;PUT HALT IN INTVEC+2  
MOV (SP)+, R5 ;RESTORE R5  
RTS PC
```

```
;SUBROUTINE TO CHECK RDY BIT SET  
CRDY: CLR R4  
CLR R5  
2$: INC R5 ;UPDATE COUNTER  
TSTB @BECR1 ;SEE IF RDY SET  
BMI 1$ ;BRANCH IF SET  
BIT #200, R5 ;WAITED > 100MICROSEC  
BEQ 2$ ;CONTINUE TO LOCK FOR RDY IF R5 NOT=128  
MOV #1, R4 ;SET R4=1 TO INDICATE ERROR  
1$: RTS PC ;RETURN
```



```
(1) 030304 105337 030402          DECB      $CHARCNT      ;;DO NOT COUNT AS A COUNT
(1) 030310 000770          BR        7$          ;;LOOP
(1)
(1)          ;HORIZONTAL TAB PROCESSOR
(1) 030312 112716 000040      8$:      MOVB      #' (SP)      ;;REPLACE TAB WITH SPACE
(1) 030316 004737 030336      9$:      JSR        PC,$TYPEC      ;;TYPE A SPACE
(1) 030322 132737 000007 030402  BITB      #7,$CHARCNT      ;;BRANCH IF NOT AT
(1) 030330 001372          BNE      9$          ;;TAB STOP
(1) 030332 005726          TST      (SP)+        ;;POP SPACE OFF STACK
(1) 030334 000724          BR        2$          ;;GET NEXT CHARACTER
(1) 030336 105777 150606      $TYPEC: TSTB      @$TPS      ;;WAIT UNTIL PRINTER IS READY
(1) 030342 100375          BPL      $TYPEC
(1) 030344 116677 000002 150600  MOVB      2(SP),@$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 030352 122766 000015 000002  CMPB      #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
(1) 030360 001003          BNE      1$          ;;BRANCH IF NO
(1) 030362 105037 030402      CLRB      $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
(1) 030366 000406          BR        $TYPEX      ;;EXIT
(1) 030370 122766 000012 000002  1$:      CMPB      #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
(1) 030376 001402          BEQ      $TYPEX      ;;BRANCH IF YES
(1) 030400 105227          INCB      (PC)+        ;;COUNT THE CHARACTER
(1) 030402 000000      $CHARCNT: WORD 0      ;;CHARACTER COUNT STORAGE
(1) 030404 000207      $TYPEX: RTS        PC
(1)
2562
(1)          .SBTTL  APT COMMUNICATIONS ROUTINE
(2)          ;*****
(1) 030406 112737 000001 030652  $ATY1:  MOVB      #1,$FFLG      ;;TO REPORT FATAL ERROR
(1) 030414 112737 000001 030650  $ATY3:  MOVB      #1,$MFLG      ;;TO TYPE A MESSAGE
(1) 030422 000403          BR        $ATYC
(1) 030424 112737 000001 030652  $ATY4:  MOVB      #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
(1) 030432          $ATYC:
(3) 030432 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 030434 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(1) 030436 105737 030650      TSTB      $MFLG      ;;SHOULD TYPE A MESSAGE?
(1) 030442 001450          BEQ      5$          ;;IF NOT: BR
(1) 030444 122737 000001 001226  CMPB      #APTENV,$ENV      ;;OPERATING UNDER APT?
(1) 030452 001031          BNE      3$          ;;IF NOT: BR
(1) 030454 132737 000100 001227  BITB      #APTSPOOL,$ENVM      ;;SHOULD SPOOL MESSAGES?
(1) 030462 001425          BEQ      3$          ;;IF NOT: BR
(1) 030464 017600 000004          MOV      @4(SP),R0      ;;GET MESSAGE ADDR.
(1) 030470 062766 000002 000004  ADD      #2,4(SP)      ;;BUMP RETURN ADDR.
(1) 030476 005737 001206      1$:      TST      $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
(1) 030502 001375          BNE      1$          ;;IF NOT: WAIT
(1) 030504 010037 001222      MOV      R0,$MSGAD      ;;PUT ADDR IN MAILBOX
(1) 030510 105720          2$:      TSTB      (R0)+        ;;FIND END OF MESSAGE
(1) 030512 001376          BNE      2$
(1) 030514 163700 001222      SUB      $MSGAD,R0      ;;SUB START OF MESSAGE
(1) 030520 006200          ASR      R0          ;;GET MESSAGE LNTH IN WORDS
(1) 030522 010037 001224      MOV      R0,$MSGGLT      ;;PUT LENGTH IN MAILBOX
(1) 030526 012737 000004 001206  MOV      #4,$MSGTYPE      ;;TELL APT TO TAKE MSG.
(1) 030534 000413          BR        5$
(1) 030536 017637 000004 030562  3$:      MOV      @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
(1) 030544 062766 000002 000004  ADD      #2,4(SP)      ;;BUMP RETURN ADDRESS
(3) 030552 013746 177776          MOV      177776,-(SP)      ;;PUSH 177776 ON STACK
(1) 030556 004737 030124          JSR      PC,$TYPE      ;;CALL TYPE MACRO
```

```
(1) 030562 000000 4$: .WORD 0
(1) 030564 105737 030652 5$:
(1) 030570 001416 10$: TSTB $FFLG :: SHOULD REPORT FATAL ERROR?
(1) 030572 005737 001226 BEQ 12$ :: IF NOT: BR
(1) 030576 001413 TST $ENV :: RUNNING UNDER APT?
(1) 030600 005737 001206 BEQ 12$ :: IF NOT: BR
(1) 030604 001375 11$: TST $MSGTYPE :: FINISHED LAST MESSAGE?
(1) 030606 017637 000004 001210 BNE 11$ :: IF NOT: WAIT
(1) 030614 062766 000002 000004 MOV @4(SP), $FATAL :: GET ERROR #
(1) 030622 065237 001206 INC #2, 4(SP) :: BUMP RETURN ADDR.
(1) 030626 105037 030652 12$: CLRB $FFLG :: TELL APT TO TAKE ERROR
(1) 030632 105037 030651 CLRB $LFLG :: CLEAR FATAL FLAG
(1) 030636 105037 030650 CLRB $MFLG :: CLEAR LOG FLAG
(3) 030642 012601 MOV (SP)+, R1 :: CLEAR MESSAGE FLAG
(3) 030644 012600 MOV (SP)+, R0 :: POP STACK INTO R1
(1) 030646 000207 RTS PC :: POP STACK INTO R0
(1) 030650 000 $MFLG: .BYTE 0 :: RETURN
(1) 030651 000 $LFLG: .BYTE 0 :: MESSG. FLAG
(1) 030652 000 $FFLG: .BYTE 0 :: LOG FLAG
(1) 030654 .EVEN :: FATAL FLAG
(1) 000200 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTSPool=100
(1) 000040 APTCSUP=040
2563 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)
(1) ::*****
(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) 030654 011646 000004 000002 $RDOCT: MOV (SP), -(SP) :: PROVIDE SPACE FOR THE
(1) 030656 016666 MOV 4(SP), 2(SP) :: INPUT NUMBER
(3) 030664 010046 MOV R0, -(SP) :: PUSH R0 ON STACK
(3) 030666 010146 MOV R1, -(SP) :: PUSH R1 ON STACK
(3) 030670 010246 MOV R2, -(SP) :: PUSH R2 ON STACK
(1) 030672 104411 1$: RDLIN :: READ AN ASCII LINE
(1) 030674 012600 MOV (SP)+, R0 :: GET ADDRESS OF 1ST CHARACTER
(1) 030676 005001 CLR R1 :: CLEAR DATA WORD
(1) 030700 005002 CLR R2
(1) 030702 112046 2$: MOVB (R0)+, -(SP) :: PICKUP THIS CHARACTER
(1) 030704 001412 BEQ 3$ :: IF ZERO GET OUT
(1) 030706 006301 ASL R1 :: *2
(1) 030710 006102 ROL R2
(1) 030712 006301 ASL R1 :: *4
(1) 030714 006102 ROL R2
(1) 030716 006301 ASL R1 :: *8
(1) 030720 006102 ROL R2
(1) 030722 042716 177770 BIC #^C7, (SP) :: STRIP THE ASCII JUNK
(1) 030726 062601 ADD (SP)+, R1 :: ADD IN THIS DIGIT
(1) 030730 000764 BR 2$ :: LOOP
(1) 030732 005726 3$: TST (SP)+ :: CLEAN TERMINATOR FROM STACK
```



```
(1) 031134 104401 001203 14$: TYPE $CRLF ;;ECHO <CR> AND <LF>
(1) 031140 123727 001135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
(1) 031146 001003 BNE 15$ ;;BRANCH IF NOT
(1) 031150 012777 000100 147766 MOV #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
(1) 031156 000002 15$: RTI ;;RETURN
(1) 031160 004737 030336 16$: JSR PC,$TYPEC ;;ECHO CHAR
(1) 031164 021627 000060 CMP (SP),#60 ;;CHAR < 0?
(1) 031170 002420 BLT 18$ ;;BRANCH IF YES
(1) 031172 021627 000067 CMP (SP),#67 ;;CHAR > 7?
(1) 031176 003015 BGT 18$ ;;BRANCH IF YES
(1) 031200 042726 000060 BIC #60,(SP)+ ;;STRIP-OFF ASCII
(1) 031204 005766 000002 TST 2(SP) ;;IS THIS THE FIRST CHAR
(1) 031210 001403 BEQ 17$ ;;BRANCH IF YES
(1) 031212 006316 ASL (SP) ;;NO, SHIFT PRESENT
(1) 031214 006316 ASL (SP) ;; CHAR OVER TO MAKE
(1) 031216 006316 ASL (SP) ;; ROOM FOR NEW ONE.
(1) 031220 005266 000002 17$: INC 2(SP) ;;KEEP COUNT OF CHAR
(1) 031224 056616 177776 BIS -2(SP),(SP) ;;SET IN NEW CHAR
(1) 031230 000707 BR 7$ ;;GET THE NEXT ONE
(1) 031232 104401 001202 18$: TYPE $QUES ;;TYPE ?<CR><LF>
(1) 031236 000720 BR 20$ ;;SIMULATE CONTROL-U
(1) .DSABL LSB
(1)
(1)
(2)
(1) ;;*****
(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) 031240 011646 $RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
(1) 031242 016666 000004 000002 MOV 4(SP),2(SP) ;;SAVE THE PS
(1) 031250 105777 147670 1$: TSTB @STKS ;;WAIT FOR
(1) 031254 100375 BPL 1$ ;;A CHARACTER
(1) 031256 117766 147664 000004 MOVB @STKB,4(SP) ;;READ THE TTY
(1) 031264 042766 177600 000004 BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
(1) 031272 026627 000004 000023 CMP 4(SP),#23 ;;IS IT A CONTROL-S?
(1) 031300 001013 BNE 3$ ;;BRANCH IF NO
(1) 031302 105777 147636 2$: TSTB @STKS ;;WAIT FOR A CHARACTER
(1) 031306 100375 BPL 2$ ;;LOOP UNTIL ITS THERE
(1) 031310 117746 147632 MOVB @STKB,-(SP) ;;GET CHARACTER
(1) 031314 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 031320 022627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
(1) 031324 001366 BNE 2$ ;;IF NOT DISCARD IT
(1) 031326 000750 BR 1$ ;;YES, RESUME
(1) 031330 026627 000004 000140 3$: CMP 4(SP),#140 ;;IS IT UPPER CASE?
(1) 031336 002407 BLT 4$ ;;BRANCH IF YES
(1) 031340 026627 000004 000175 CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
(1) 031346 003003 BGT 4$ ;;BRANCH IF YES
(1) 031350 042766 000040 000004 BIC #40,4(SP) ;;MAKE IT UPPER CASE
(1) 031356 000002 4$: RTI ;;GO BACK TO USER
(2)
(1) ;;*****
(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) 031360 010346 $RDLIN: MOV R3,-(SP)      :: SAVE R3
(1) 031362 012703 031466 1$: MOV #STTYIN,R3  :: GET ADDRESS
(1) 031366 022703 031476 2$: CMP #STTYIN+8.,R3  :: BUFFER FULL?
(1) 031372 101405 BLOS 4$          :: BR IF YES
(1) 031374 104410 RDCHR          :: GO READ ONE CHARACTER FROM THE TTY
(1) 031376 112613 MOVB (SP)+,(R3)  :: GET CHARACTER
(1) 031400 122713 000177 10$: CMPB #177,(R3)  :: IS IT A RUBOUT
(1) 031404 001003 BNE 3$          :: SKIP IF NOT
(1) 031406 104401 001202 4$: TYPE ,SQUES  :: TYPE A '?'
(1) 031412 000763 BR 1$          :: CLEAR THE BUFFER AND LOOP
(1) 031414 111337 031464 3$: MOVB (R3),9$  :: ECHO THE CHARACTER
(1) 031420 104401 031464 TYPE ,9$
(1) 031424 122723 000015 CMPB #15,(R3)+  :: CHECK FOR RETURN
(1) 031430 001356 BNE 2$          :: LOOP IF NOT RETURN
(1) 031432 105063 177777 CLRB -1(R3)     :: CLEAR RETURN (THE 15)
(1) 031436 104401 001204 TYPE ,LF        :: TYPE A LINE FEED
(1) 031442 012603 MOV (SP)+,R3    :: RESTORE R3
(1) 031444 011646 MOV (SP),-(SP)  :: ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 031446 016666 000004 000002 MOV 4(SP),2(SP)  :: FIRST ASCII CHARACTER ON IT
(1) 031454 012766 031466 000004 MOV #STTYIN,4(SP)
(1) 031462 000002 RTI          :: RETURN
(1) 031464 000 9$: .BYTE 0      :: STORAGE FOR ASCII CHAR. TO TYPE
(1) 031465 000 .BYTE 0        :: TERMINATOR
(1) 031466 000010 $TTYIN: .BLKB 8.  :: RESERVE 8 BYTES FOR TTY INPUT
(1) 031476 052536 005015 000 $CNTLU: .ASCIZ /^U/<15><12>  :: CONTROL 'U'
(1) 031503 136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12>  :: CONTROL 'G'
(1) 031510 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 031516 020075 000
(1) 031521 040 047040 053505 $MNEW: .ASCIZ / NEW = /
(1) 031526 036440 000040
2565 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)
(1) :: *****
(1) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1) *OCTAL (ASCII) NUMBER AND TYPE IT.
(1) *$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) 031532 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
(1) 031536 116637 000001 031755  MOVB     1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
(1) 031544 112637 031757          MOVB     (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
(1) 031550 062716 000002          ADD      #2,(SP)          ;;ADJUST RETURN ADDRESS
(1) 031554 000406          BR      $TYPON
(1) 031556 112737 000001 031755  $TYPOC: MOVB     #1, $OFILL      ;;SET THE ZERO FILL SWITCH
(1) 031564 112737 000006 031757  MOVB     #6, $OMODE+1      ;;SET FOP SIX(6) DIGITS
(1) 031572 112737 000005 031754  $TYPON: MOVB     #5, $OCNT      ;;SET THE ITERATION COUNT
(1) 031600 010346          MOV      R3,-(SP)          ;;SAVE R3
(1) 031602 010446          MOV      R4,-(SP)          ;;SAVE R4
(1) 031604 010546          MOV      R5,-(SP)          ;;SAVE R5
(1) 031606 113704 031757          MOVB     $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 031612 005404          NEG      R4
(1) 031614 062704 000006          ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 031620 110437 031756          MOVB     R4, $OMODE      ;;SAVE IT FOR USE
(1) 031624 113704 031755          MOVB     $OFILL,R4      ;;GET THE ZERO FILL SWITCH
(1) 031630 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
(1) 031634 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
(1) 031636 006105          1$: ROL      R5          ;;ROTATE MSB INTO 'C'
(1) 031640 000404          BR      3$
(1) 031642 006105          2$: ROL      R5          ;;FORM THIS DIGIT
(1) 031644 006105          ROL      R5
(1) 031646 006105          ROL      R5
(1) 031650 010503          MOV      R5,R3
(1) 031652 006103          3$: ROL      R3          ;;GET LSB OF THIS DIGIT
(1) 031654 105337 031756          DECB     $OMODE      ;;TYPE THIS DIGIT?
(1) 031660 100016          BPL      7$          ;;BR IF NO
(1) 031662 042703 177770          BIC      #177770,R3      ;;GET RID OF JUNK
(1) 031666 001002          BNE      4$          ;;TEST FOR 0
(1) 031670 005704          TST      R4          ;;SUPPRESS THIS 0?
(1) 031672 001403          BEQ      5$          ;;BR IF YES
(1) 031674 005204          4$: INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
(1) 031676 052703 000060          BIS      #'0,R3          ;;MAKE THIS DIGIT ASCII
(1) 031702 052703 000040          5$: BIS      #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1) 031706 110337 031752          MOVB     R3,8$          ;;SAVE FOR TYPING
(1) 031712 104401 031752          TYPE     8$          ;;GO TYPE THIS DIGIT
(1) 031716 105337 031754          7$: DECB     $OCNT      ;;COUNT BY 1
(1) 031722 003347          BGT      2$          ;;BR IF MORE TO DO
(1) 031724 002402          BLT      6$          ;;BR IF DONE
(1) 031726 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 031730 000744          BR      2$          ;;GO DO THE LAST DIGIT
(1) 031732 012605          6$: MOV      (SP)+,R5      ;;RESTORE R5
(1) 031734 012604          MOV      (SP)+,R4      ;;RESTORE R4
(1) 031736 012603          MOV      (SP)+,R3      ;;RESTORE R3
(1) 031740 016666 000002 000004  MOV      2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
(1) 031746 012616          MOV      (SP)+,(SP)
(1) 031750 000002          RTI          ;;RETURN
(1) 031752 000          8$: .BYTE 0          ;;STORAGE FOR ASCII DIGIT
(1) 031753 000          .BYTE 0          ;;TERMINATOR FOR TYPE ROUTINE
(1) 031754 000          $OCNT: .BYTE 0          ;;OCTAL DIGIT COUNTER
(1) 031755 000          $OFILL: .BYTE 0          ;;ZERO FILL SWITCH
(1) 031756 000000          $OMODE: .WORD 0          ;;NUMBER OF DIGITS TO TYPE
2566          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
```







```

(1) 032374 020040 047125 054105 11$: .ASCIZ ' UNEXPECTED TRAP TO '
(1) 032402 042520 052103 042105
(1) 032410 052040 040522 020120
(1) 032416 047524 000040
(1)
(1) 032422 12$: .EVEN
(1) 032422 004737 032534 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
(1) 032426 104401 001203 TYPE , $CRLF
(1) 032432 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 032432 122737 000001 001226 BNE 2$ ;;NO,SKIP APT ERROR REPORT
(1) 032440 001007 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 032442 113737 001114 032454 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
(1) 032450 004737 030424
(1) 032454 000 21$: .BYTE 0
(1) 032455 000 .BYTE 0
(1) 032456 000777 22$: BR 22$ ;;APT ERROR LOOP
(1) 032460 005777 146454 2$: TST @SWR ;;HALT ON ERROR
(1) 032464 100002 BPL 3$ ;;SKIP IF CONTINUE
(1) 032466 000000 HALT ;;HALT ON ERROR!
(1) 032470 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 032472 032777 001000 146440 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(1) 032500 001402 BEQ 4$ ;;BR IF NO
(1) 032502 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 032506 005737 001174 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(1) 032512 001402 BEQ 5$ ;;BR IF NONE
(1) 032514 013716 001174 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 032520 5$:
(1) 032520 022737 021640 000042 CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 032526 001001 BNE 6$ ;;BRANCH IF NO
(1) 032530 000000 HALT ;;YES
(1) 032532 6$:
(1) 032532 000002 RTI ;;RETURN
2568 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)
(1) ;;*****
(1) ;;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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) $ERRTYP:
(1) 032534 TYPE , $CRLF ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 032534 104401 001203 MOV R0,-(SP) ;;SAVE R0
(1) 032540 010046 CLR R0 ;;PICKUP THE ITEM INDEX
(1) 032542 005000 BISB @#$ITEMB,R0
(1) 032544 153700 001114 BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
(1) 032550 001004 MOV $ERRPC,-(SP) ;;TYPE THE PC OF THE ERROR
(2) 032552 013746 001116 ;;SAVE $ERRPC FOR TYPEOUT
(2) ;;ERROR ADDRESS
(2) 032556 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 032560 000426 BR 6$ ;;GET OUT
(1) 032562 005300 1$: DEC R0 ;;ADJUST THE INDEX SO THAT IT WILL
(1) 032564 006300 ASL R0 ;; WORK FOR THE ERROR TABLE
(1) 032566 006300 ASL R0
(1) 032570 006300 ASL R0
(1) 032572 062700 001332 ADD # $ERRTB,R0 ;;FORM TABLE POINTER
(1) 032576 012037 032606 MOv (R0)+,2$ ;;PICKUP "ERROR MESSAGE" POINTER

```

```
(1) 032602 001404 BEQ 3$ ::SKIP TYPEOUT IF NO POINTER
(1) 032604 104401 TYPE ::TYPE THE 'ERROR MESSAGE'
(1) 032606 000000 2$: .WORD 0 ::'ERROR MESSAGE' POINTER GOES HERE
(1) 032610 104401 001203 TYPE , $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 032614 012037 032624 3$: MOV (R0)+,4$ ::PICKUP 'DATA HEADER' POINTER
(1) 032620 001404 BEQ 5$ ::SKIP TYPEOUT IF 0
(1) 032622 104401 TYPE ::TYPE THE 'DATA HEADER'
(1) 032624 000000 4$: .WORD 0 ::'DATA HEADER' POINTER GOES HERE
(1) 032626 104401 001203 TYPE , $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 032632 011000 5$: MOV (R0),R0 ::PICKUP 'DATA TABLE' POINTER
(1) 032634 001004 BNE 7$ ::GO TYPE THE DATA
(1) 032636 012600 6$: MOV (SP)+,R0 ::RESTORE R0
(1) 032640 104401 001203 TYPE , $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 032644 000207 RTS PC ::RETURN
(1) 032646 7$:
(2) 032646 013046 MOV @ (R0)+, -(SP) ::SAVE @ (R0)+ FOR TYPEOUT
(2) 032650 104402 TYPOC ::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 032652 005710 TST (R0) ::IS THERE ANOTHER NUMBER?
(1) 032654 001770 BEQ 6$ ::BR IF NO
(1) 032656 104401 032664 TYPE , 8$ ::TYPE TWO(2) SPACES
(1) 032662 000771 BR 7$ ::LOOP
(1) 032664 020040 000 8$: .ASCIZ / / ::TWO(2) SPACES
(1) 032670 .EVEN
2569 .SBTTL SCOPE HANDLER ROUTINE
(1)
(2)
(1)
(1) *****
(1) *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) *AND LOAD THE TEST NUMBER($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) $SCOPE:
(1) 032670 104407 CKSWR ::TEST FOR CHANGE IN SOFT-SWR
(1) ::GO TO ERROR ROUTINE IF RETURN PC LESS THAN 1002
(1) ::OTHERWISE CONTINUE
(1) 032672 021627 001002 CMP (SP),#1002 ::UNEXPECTED TRAP OR INTERRUPT
(1) 032676 101002 BHI 1$ ::ARE TRAPPED HERE VIA IOT
(1) 032700 000137 032204 JMP $ERROR ::GO PROCESS UNEXPECTED TRAP
(1) 032704 032777 040000 146226 1$: BIT #BIT14,@SWR ::LOOP ON PRESENT TEST?
(1) 032712 001114 BNE $OVER ::YES IF SW14=1
(1) :*****START OF CODE FOR THE XOR TESTER*****
(1) 032714 000416 $XTSTR: BR 6$ ::IF RUNNING ON THE 'XOR' TESTER CHANGE
(1) ::THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1) 032716 013746 000004 MOV @ERRVEC, -(SP) ::SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 032722 012737 032742 000004 MOV #5$, @ERRVEC ::SET FOR TIMEOUT
(1) 032730 005737 177060 TST @177060 ::TIME OUT ON XOR?
(1) 032734 012637 000004 MOV (SP)+, @ERRVEC ::RESTORE THE ERROR VECTOR
(1) 032740 000463 BR $SVLAD ::GO TO THE NEXT TEST
(1) 032742 022626 5$: CMP (SP)+, (SP)+ ::CLEAR THE STACK AFTER A TIME OUT
(1) 032744 012637 000004 MOV (SP)+, @ERRVEC ::RESTORE THE ERROR VECTOR
```

```
(1) 032750 000423 BR 7$ ;; LOOP ON THE PRESENT TEST
(1) 032752 6S:;*****END OF CODE FOR THE XOR TESTER*****
(1) 032752 032777 000400 146160 BIT #BIT08,@SWR ;; LOOP ON SPEC. TEST?
(1) 032760 001404 BEQ 2$ ;; BR IF NO
(1) 032762 127737 146152 001102 CMPB @SWR,$STSTM ;; ON THE RIGHT TEST? SWR<7:0>
(1) 032770 001465 BEQ $OVER ;; BR IF YES
(1) 032772 105737 001103 2$: TSTB $ERFLG ;; HAS AN ERROR OCCURRED?
(1) 032776 001421 BEQ 3$ ;; BR IF NO
(1) 033000 123737 001115 001103 CMPB $ERMAX,$ERFLG ;; MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 033006 101015 BHI 3$ ;; BR IF NO
(1) 033010 032777 001000 146122 BIT #BIT09,@SWR ;; LOOP ON ERROR?
(1) 033016 001404 BEQ 4$ ;; BR IF NO
(1) 033020 013737 001110 001106 7$: MOV $LPERR,$LPADR ;; SET LOOP ADDRESS TO LAST SCOPE
(1) 033026 000446 BR $OVER
(1) 033030 105037 001103 4$: CLRB $ERFLG ;; ZERO THE ERROR FLAG
(1) 033034 005037 001172 CLR $TIMES ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 033040 000415 BR 1$ ;; ESCAPE TO THE NEXT TEST
(1) 033042 032777 004000 146070 3$: BIT #BIT11,@SWR ;; INHIBIT ITERATIONS?
(1) 033050 001011 BNE 1$ ;; BR IF YES
(1) 033052 005737 001214 TST $PASS ;; IF FIRST PASS OF PROGRAM
(1) 033056 001406 BEQ 1$ ;; INHIBIT ITERATIONS
(1) 033060 005237 001104 INC $ICNT ;; INCREMENT ITERATION COUNT
(1) 033064 023737 001172 001104 CMP $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
(1) 033072 002024 BGE $OVER ;; BR IF MORE ITERATION REQUIRED
(1) 033074 012737 000001 001104 1$: MOV #1,$ICNT ;; REINITIALIZE THE ITERATION COUNTER
(1) 033102 013737 033160 001172 MOV $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
(1) 033110 105237 001102 $SVLAD: INCB $STSTM ;; COUNT TEST NUMBERS
(1) 033114 113737 001102 001212 MOVB $STSTM,$TESTN ;; SET TEST NUMBER IN APT MAILBOX
(1) 033122 011637 001106 MOV (SP),$LPADR ;; SAVE SCOPE LOOP ADDRESS
(1) 033126 011637 001110 MOV (SP),$LPERR ;; SAVE ERROR LOOP ADDRESS
(1) 033132 005037 001174 CLR $ESCAPE ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 033136 112737 000001 001115 MOVB #1,$ERMAX ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 033144 013777 001102 145770 $OVER: MOV $STSTM,@DISPLAY ;; DISPLAY TEST NUMBER
(1) 033152 013716 001106 MOV $LPADR,(SP) ;; FUDGE RETURN ADDRESS
(1) 033156 000002 RTI ;; FIXES PS
(1) 033160 003720 $MXCNT: 2000. ;; MAX. NUMBER OF ITERATIONS
2570 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)
(1)
(1) 033162 012737 033326 000024 $PWRDN: MOV #SILLUP,@PWRVEC ;; SET FOR FAST UP
(1) 033170 012737 000340 000026 MOV #340,@PWRVEC+2 ;; PRIO:7
(3) 033176 010046 MOV R0,-(SP) ;; PUSH R0 ON STACK
(3) 033200 010146 MOV R1,-(SP) ;; PUSH R1 ON STACK
(3) 033202 010246 MOV R2,-(SP) ;; PUSH R2 ON STACK
(3) 033204 010346 MOV R3,-(SP) ;; PUSH R3 ON STACK
(3) 033206 010446 MOV R4,-(SP) ;; PUSH R4 ON STACK
(3) 033210 010546 MOV R5,-(SP) ;; PUSH R5 ON STACK
(3) 033212 017746 145722 MOV @SWR,-(SP) ;; PUSH @SWR ON STACK
(1) 033216 010637 033332 MOV SP,$SAVR6 ;; SAVE SP
(1) 033222 012737 033234 000024 MOV #SPWRUP,@PWRVEC ;; SET UP VECTOR
(1) 033230 000000 HALT
(1) 033232 000776 BR -2 ;; HANG UP
(1)
(2)
(1)
(1) *****
:POWER UP ROUTINE
*****
```



```

(3)      :      ROUTINE
(3)      :      -----
(3) 033424 033412 $TRPAD: .WORD $TRAP2
(3) 033426 030124 $TYPE  ::CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
(3) 033430 031556 $TYPOC ::CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 033432 031532 $TYPOS ::CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 033434 031572 $TYPON ::CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 033436 031760 $TYPDS ::CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(1)
(3) 033440 031026 $GTSWR ::CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
(1)
(3) 033442 030756 $CKSWR ::CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
(3) 033444 031240 $RDCHR ::CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3) 033446 031360 $RDLIN ::CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3) 033450 030654 $RDOCT ::CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
2574 .SBTTL RANDOM NUMBER GENERATOR ROUTINE
(1)
(2)
(1) ::*****
(1) ::*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
(1) ::*WITH A RANGE OF 0 TO 2(+33)-1.
(1) ::*CALL:
(1) ::* JSR PC,$RAND ::CALL THE ROUTINE
(1) ::* RETURN ::RETURN HERE THE RANDOM
(1) ::* ::NUMBER WILL BE IN
(1) ::* ::$HINUM,$LONUM
(1)
(1) 033452 $RAND:
(3) 033452 010046 MOV RO,-(SP) ::PUSH R0 ON STACK
(3) 033454 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
(3) 033456 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
(1) 033460 013700 033552 MOV $LONUM,R0 ::SET R0 WITH LOW
(1) 033464 013701 033550 MOV $HINUM,R1 ::SET R1 WITH HIGH
(1) 033470 012702 177771 MOV #-7,R2 ::SET SHIFT COUNT
(1) 033474 006300 1$: ASL R0 ::SHIFT R0 LEFT AND
(1) 033476 006101 ROL R1 ::ROTATE CARRY INTO R1 AND
(1) 033500 005202 INC R2 ::CHECK FOR DONE
(1) 033502 001374 BNE 1$ ::CONTINUE SHIFT LOOP
(1) 033504 063700 033552 ADD $LONUM,R0 ::ADD NUMBER TO MAKE X 129
(1) 033510 005501 ADC R1 ::PROPOGATE CARRY
(1) 033512 063701 033550 ADD $HINUM,R1 ::ADD NUMBER TO MAKE X 129
(1) 033516 062700 001057 ADD #1057,R0 ::ADD LOW CONSTANT
(1) 033522 005501 ADC R1 ::PROPOGATE CARRY
(1) 033524 062701 047401 ADD #47401,R1 ::ADD HIGH CONSTANT
(1) 033530 010037 033552 MOV RO,$LONUM ::SAVE R0
(1) 033534 010137 033550 MOV R1,$HINUM ::SAVE R1
(3) 033540 012602 MOV (SP)+,R2 ::POP STACK INTO R2
(3) 033542 012601 MOV (SP)+,R1 ::POP STACK INTO R1
(3) 033544 012600 MOV (SP)+,R0 ::POP STACK INTO R0
(1) 033546 000207 RTS PC ::RETURN
(1) 033550 176543 $HINUM: .WORD 176543
(1) 033552 123456 $LONUM: .WORD 123456

```

```
2576 .SBTTL ASCII MESSAGES
2577
2578 033554 047520 052123 051055 EM1: .ASCIZ /POST-RESET CONDITIONS NOT MET IN CSR./
      033562 051505 052105 041440
      033570 047117 044504 044524
      033576 047117 020123 047516
      033604 020124 042515 020124
      033612 047111 041440 051123
      033620 000056
2579 033622 042522 042101 053457 EM2: .ASCIZ %READ/WRITE BIT FAILURE IN CSR.%
      033630 044522 042524 041040
      033636 052111 043040 044501
      033644 052514 042522 044440
      033652 020116 051503 027122
      033660 000
2580 033661 102 052131 020105 EM3: .ASCIZ /BYTE ACCESS FAILURE IN CSR./
      033666 041501 042503 051523
      033674 043040 044501 052514
      033702 042522 044440 020116
      033710 051503 027122 000
2581 033715 105 051122 051117 EM4: .ASCIZ /ERROR IN CONNECT PROCESS./
      033722 044440 020116 047503
      033730 047116 041505 020124
      033736 051120 041517 051505
      033744 027123 000
2582 033747 111 052116 051105 EM5: .ASCIZ /INTERUPT DETECTED WITH INTERUPT ENABLE CLEARED./
      033754 050125 020124 042504
      033762 042524 052103 042105
      033770 053440 052111 020110
      033776 047111 042524 052522
      034004 052120 042440 040516
      034012 046102 020105 046103
      034020 040505 042522 027104
      034026 000
2583 034027 103 052517 042114 EM6: .ASCIZ /COULD NOT CONNECT./
      034034 047040 052117 041440
      034042 047117 042516 052103
      034050 000056
2584 034052 051105 047522 020122 EM7: .ASCIZ /ERROR IN RELEASE PROCESS./
      034060 047111 051040 046105
      034066 040505 042523 050040
      034074 047522 042503 051523
      034102 000056
2585 034104 047516 044440 052116 EM10: .ASCIZ /NO INTERUPT DETECTED./
      034112 051105 050125 020124
      034120 042504 042524 052103
      034126 042105 000056
2586 034132 047111 042524 052522 EM11: .ASCIZ /INTERUPT AT WRONG PRIORITY./
      034140 052120 040440 020124
      034146 051127 047117 020107
      034154 051120 047511 044522
      034162 054524 000056
2587 034166 047503 047116 041505 EM12: .ASCIZ /CONNECT CONDITION FAILURE./
      034174 020124 047503 042116
      034202 052111 047511 020116
      034210 040506 046111 051125
```



2588	034216	027105	000			
	034221	105	051122	051117	EM13:	.ASCIZ /ERROR IN CONNECT FAILURE PROCESS./
	034226	044440	020116	047503		
	034234	047116	041505	020124		
	034242	040506	046111	051125		
	034250	020105	051120	041517		
2589	034256	051505	027123	000		
	034263	105	051122	051117	EM14:	.ASCIZ /ERROR IN RELEASE TEST WITHOUT BUSS MASTERSHIP./
	034270	044440	020116	042522		
	034276	042514	051501	020105		
	034304	042524	052123	053440		
	034312	052111	047510	052125		
	034320	041040	051525	020123		
	034326	040515	052123	051105		
2590	034334	044123	050111	000056		
	034342	042522	042514	051501	EM15:	.ASCIZ /RELEASE CONDITIONS NOT MET./
	034350	020105	047503	042116		
	034356	052111	047511	051516		
	034364	047040	052117	046440		
2591	034372	052105	000056			
	034376	046524	020117	044502	EM16:	.ASCIZ /TMO BIT NOT SET./
	034404	020124	047516	020124		
	034412	042523	027124	000		
2592	034417	103	052517	042114	EM17:	.ASCIZ %COULDN'T READ/WRITE EXTERNAL DEVICE.%
	034424	023516	020124	042522		
	034432	042101	053457	044522		
	034440	042524	042440	052130		
	034446	051105	040516	020114		
	034454	042504	044526	042503		
	034462	000056				
2593	034464	042522	042523	020124	EM20:	.ASCIZ /RESET HOLD (HLD) DIDN'T DISABLE RESET./
	034472	047510	042114	024040		
	034500	046110	024504	042040		
	034506	042111	023516	020124		
	034514	044504	040523	046102		
	034522	020105	042522	042523		
	034530	027124	000			
2594	034533	122	051505	052105	EM21:	.ASCIZ /RESET (RST) DIDN'T WORK./
	034540	024040	051522	024524		
	034546	042040	042111	023516		
	034554	020124	047527	045522		
	034562	000056				
2595	034564	044124	020105	050117	EM22:	.ASCIZ /THE OPERATION OF THE RESET-IN-NEUTRAL HAS CHANGED./
	034572	051105	052101	047511		
	034600	020116	043117	052040		
	034606	042510	051040	051505		
	034614	052105	044455	026516		
	034622	042516	052125	040522		
	034630	020114	040510	020123		
	034636	044103	047101	042507		
	034644	027104	000			
2596	034647	040	000		SPACE:	.ASCIZ / /
2597	034651	200	036460	044124	SELMS:	.ASCII <200>/0=THIS MESSAGE/
	034656	051511	046440	051505		
	034664	040523	042507			
2598	034670	030600	046475	047117		.ASCII <200>/1=MONOPORT TEST/

	034676	050117	051117	020124					
	034704	042524	052123						
2599	034710	031200	046475	046125		.ASCII	<200>/2=	MULTI	PORT TEST/
	034716	044524	047520	052122					
	034724	052040	051505	124					
2600	034731	200	036463	040515		.ASCII	<200>/3=	MANUAL	INTERVENTION TEST/
	034736	052516	046101	044440					
	034744	052116	051105	042526					
	034752	052116	047511	020116					
	034760	042524	052123						
2601	034764	032200	052075	051505		.ASCIIZ	<200>/4=	TEST	NPR LINE/
	034772	020124	050116	020122					
	035000	044514	042516	000					
2602	035005	105	051122	051117	DH1:	.ASCII	/ERROR	TEST#	SHOULD WAS FROM/<200>
	035012	052011	051505	021524					
	035020	051411	047510	046125					
	035026	004504	040527	004523					
	035034	051106	046517	200					
2603	035041	120	004503	041011		.ASCIIZ	/PC		BE/
	035046	000105							
2604	035050	051105	047522	004522	DH2:	.ASCII	/ERROR	TEST#	WROTE READ DEVICE ADDRESS/<200>
	035056	042524	052123	004443					
	035064	051127	052117	004505					
	035072	042522	042101	042011					
	035100	053105	041511	020105					
	035106	042101	051104	051505					
	035114	100123							
2605	035116	041520	000			.ASCIIZ	/PC/		
2606	035121	105	051122	051117	DH3:	.ASCII	/ERROR	TEST#	BITS FAILING RESET DEVICE ADDRESS/<200>
	035126	052011	051505	021524					
	035134	041011	052111	020123					
	035142	040506	046111	047111					
	035150	020107	042522	042523					
	035156	004524	042504	044526					
	035164	042503	040440	042104					
	035172	042522	051523	200					
2607	035177	120	000103			.ASCIIZ	/PC/		
2608	035202	051105	047522	004522	DH4:	.ASCII	/ERROR	TEST#	RESET READ DEVICE ADDRESS/<200>
	035210	042524	052123	004443					
	035216	042522	042523	004524					
	035224	042522	042101	042011					
	035232	053105	041511	020105					
	035240	042101	051104	051505					
	035246	100123							
2609	035250	041520	020040	020040		.ASCIIZ	/PC		BITS/
	035256	020040	020040	020040					
	035264	020040	041040	052111					
	035272	000123							
2610	035274	051105	047522	100122	DH5:	.ASCII	/ERROR/<200>		
2611	035302	050040	020103	000		.ASCIIZ	/PC/		
2612	035307	105	051122	051117	DH6:	.ASCII	/ERROR		BR/<200>
	035314	020040	020040	020040					
	035322	020040	020040	051102					
	035330	200							
2613	035331	040	041520	020040		.ASCIIZ	/PC		LEVEL/
	035336	020040	020040	020040					

	035344	020040	046040	053105		
	035352	046105	000			
2614		035356				.EVEN
2615	035356	001116	001632	001124	DT1:	\$ERRPC,TSTNUM,GOOD,BAD,BADA,0
	035364	001126	001122	000000		
2616	035372	001116	001632	001640	DT2:	\$ERRPC,TSTNUM,DEVWR,BAD,DEVAD,0
	035400	001126	001636	000000		
2617	035406	001116	001632	001126	DT3:	\$ERRPC,TSTNUM,BAD,DEVAD,0
	035414	001636	000000			
2618	035420	001116	000000		DT4:	\$ERRPC,0
2619	035424	001116	027406	000000	DT5:	\$ERRPC,BRCNT,0
2620		000001				.END

ABASE = 177420	388#	400													
ACDW1 = 000000	400														
ACDW2 = 000000	400														
ACPUOP = 000000	400														
ADDW0 = 000000	400														
ADDW1 = 000000	400														
ADDW10 = 000000	400														
ADDW11 = 000000	400														
ADDW12 = 000000	400														
ADDW13 = 000000	400														
ADDW14 = 000000	400														
ADDW15 = 000000	400														
ADDW2 = 000000	400														
ADDW3 = 000000	400														
ADDW4 = 000000	400														
ADDW5 = 000000	400														
ADDW6 = 000000	400														
ADDW7 = 000000	400														
ADDW8 = 000000	400														
ADDW9 = 000000	400														
ADEVCT = 000000	400														
ADEVN = 000000	400														
AENV = 000000	400														
AENVN = 000000	400														
AFATAL = 000000	400														
AMADR1 = 000000	400														
AMADR2 = 000000	400														
AMADR3 = 000000	400														
AMADR4 = 000000	400														
AMAMS1 = 000000	400														
AMAMS2 = 000000	400														
AMAMS3 = 000000	400														
AMAMS4 = 000000	400														
AMSGAD = 000000	400														
AMSGLG = 000000	400														
AMSGTY = 000000	400														
AMTY1 = 000000	400														
AMTY2 = 000000	400														
AMTY3 = 000000	400														
AMTY4 = 000000	400														
APASS = 000000	400														
APRIOR = 000000	400														
APTCSU = 000040	2561	2562#													
APTENV = 000001	2561	2562#	2567												
APTSIZ = 000200	579	2562#													
APTSP0 = 000100	2561	2562#													
ASWREG = 000000	400														
ATESTN = 000000	400														
AUNIT = 000000	400														
AUSWR = 000000	400														
AVECT1 = 160350	389#	400													
AVECT2 = 000000	400														
BAD = 001126	391#	670*	672	686*	687*	688*	689	696*	697*	698	704*	705	719*		
	720	726*	727	747*	748*	749	758*	759	767*	781*	782	797*	798		
	806*	807	814*	833*	834*	835	845*	846*	847	855*	856	864*	879*		
	880	895*	896	904*	905	912*	924*	933*	938	941*	944	961*	962		

















SDDW0	001272	400#							
SDDW1	001274	400#							
SDDW10	001316	400#							
SDDW11	001320	400#							
SDDW12	001322	400#							
SDDW13	001324	400#							
SDDW14	001326	400#							
SDDW15	001330	400#							
SDDW2	001276	400#							
SDDW3	001300	400#							
SDDW4	001302	400#							
SDDW5	001304	400#							
SDDW6	001306	400#							
SDDW7	001310	400#							
SDDW8	001312	400#							
SDDW9	001314	400#							
SDEVCT	001216	400#							
SDEVM	001264	400#							
SDOAGN	021650	1940#							
SDTBL	032164	2566#							
SENDAD	021640	397	602	1940#	2567				
SENDCT	021606	579	1940#						
SENDMG	021657	1940#							
SENULL	021654	1940#							
SENV	001226	400#	602	2561	2562	2567			
SENVN	001227	400#	579	2561	2562				
SEOP	021552	1262	1323	1344	1717	1719	1940#		
SEOPCT	021600	579*	1940#						
SERFLG	001103	400#	2567*	2569*					
SERMAX	001115	400#	579*	2569*					
SERROR	032204	579	2567#	2569					
SERRPC	001116	400#	2567*	2568	2615	2616	2617	2618	2619
SERRTB	001332	400#	2568						
SERRTY	032534	2567	2568#						
SERTTL	001112	400#	2567*						
SESCAP	001174	400#	579*	2567	2569*				
SETABL	001226	400#							
SETEND	001332	399	400#						
SFATAL	001210	400#	2562*						
SFFLG	030652	2562#*							
SFILLC	001156	400#	2561						
SFILLS	001155	400#	2561						
SGADR	001120	400#							
SGDDAT	001124	390	400#						
SGET42	021630	1940#							
SGTSWR	031026	2564#	2573						
\$HIBTS	001000	399#							
\$HINUM	033550	2574#*							
\$HIOCT	030754	2563#*							
\$ICNT	001104	400#	2569*						
\$ILLUP	033326	2570#							
\$INTAG	001135	400#	2564						
\$ITEMB	001114	400#	2567*	2568					
\$LF	001204	400#	2561	2564	2567				
\$LFLG	030651	2562#*							
\$LONUM	033552	2574#*							







.KT11	379#	394
.SETUP	379#	577
.SWRHI	381#	
.\$ACT1	382#	397
.\$APT8	382#	400#
.\$APTH	382#	399
.\$APTY	382#	2562
.\$CMTA	380#	400
.\$EOP	381#	1940
.\$ERRO	381#	2567
.\$ERRT	381#	2568
.\$POWE	380#	2570
.\$RAND	379#	2574
.\$RDOC	379#	2563
.\$READ	382#	2564
.\$SCOP	381#	2569
.\$STRAP	379#	2573
.\$TYPD	381#	2566
.\$TYPE	382#	2561
.\$TYPO	380#	2565
.\$4OCA	380#	395

. ABS. 035432 000

ERRORS DETECTED: 0

CRDTAB,CRDTAB.LST/CRF=CRDTAB.P11  
RUN-TIME: 122 66 6 SECONDS  
RUN-TIME RATIO: 459/195=2.3  
CORE USED: 26K (51 PAGES)