MINC-11

MNCDI DIAGNOSTIC

CVMNBA0

AH-B089A-MC

FICHE 1 OF 1

DEC 1978

digital

MADE IN USA

IDENTIFICATION

PRODUCT CODE:	AC-B088A-MC
PRODUCT NAME:	CVMNBA0 MNCDI (DIGITAL IN) DIAGNOSTIC TEST
DATE:	AUGUST 1978
MAINTAINER:	DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1.0 ABSTRACT

THE MNCDI DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED
TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE.
ADDITIONAL ROUTINES ARE PROVIDED TO VERIFY ADDITIONAL SECTIONS.
TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL
VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11/03 COMPUTER OR LSI-11 PROCESSOR
2. DLV11 WITH I/O TERMINAL (LA36, VT100, ETC.)
3. MNCDI (DIGITAL IN) OPTION

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE <XXDP>

1. ENSURE THAT THE DIAGNOSTIC LOAD MEDIA IS INSTALLED IN DRIVE 0.
2. BOOT THE MEDIA BY TYPING "173000G" IF IN THE ODT MICRO-CODE
   STATE OR CYCLING THE POWER "ON-OFF" SWITCH.
3. UPON SUCCESSFUL BOOTING OF THE LOAD MEDIA, THE XXDP MONITOR
   WILL IDENTIFY ITSELF AND INFORM THE OPERATOR OF THE OPERATING
   OPTIONS THAT MAYBE SELECTED.
4. THE OPERATOR SHOULD TYPE "R VMNB??" FOLLOWED BY A "RETURN"
   THE XXDP MONITOR WILL LOAD THE PROGRAM INTO MEMORY AND START
   THE PROGRAM AT LOCATION 200.

4.0 STARTING PROCEDURE

1. THE PROGRAM WILL RESPOND BY TYPING THE PROGRAM TITLE.
2. THE PROGRAM WILL NOW ASK FOR AN INITIAL SWITCH REGISTER
   VALUE TO BE STORED IN THE SOFTWARE SWITCH REGISTER.
3. THE PROGRAM WILL NOW DISPLAY THE MENU OF TEST OR LOOPS AVAILABLE.
   THE OPERATOR SELECTS THE TESTS BY TYPING THE SELECTED CHARACTER
   FOLLOWED BY DEPRESSING THE "RETURN" KEY.

4.1 PROGRAM START

200	STARTING ADDRESS OF THE PROGRAM
204	RESTART ADDRESS OF THE PROGRAM
210	STARTING ADDRESS FOR THE OPTION TESTER.

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW12=1	010000	INHIBIT SIZING THE NUMBER OF MNCDI'S
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <7-0>

5.2 CONTROL

1. THE TEST OR LOOP MAYBE STOPPED BY TYPING THE ''CONTROL & C'' KEYS. THIS OPERATION WILL STOP THE PROGRAM AND ENABLE THE OPERATOR TO SELECT DIFFERENT PROGRAM COMMAND'S.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CONTROL & G' KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.
3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING 'P' (CONTINUE).
4. IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL 'CONTROL & G OR C' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE BY THE PROGRAM.

6.0 ERROR REPORTING

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF.

6.2 ERROR DATA

*UNIT	UNIT NUMBER
*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*BUSADR	MNCDI BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

*ALWAYS REPORTED

7.0 MISCELLANEOUS

SEQ 0005

7.1 MNCDI BUS ADDRESS MODIFICATION

MODIFY LOCATION '$BASE' (LOC. 1244) IF BASE BUS ADDRESS IS NOT 171160.
MODIFY LOCATION '$VECT1' (LOC. 1240) IF INTERRUPT VECTOR IS NOT 130.
MODIFY LOCATION '$CDW1' (LOC. 1250) IF THE OUTPUT BASE ADDRESS
IS NOT 171260 FOR THE WRAP-AROUND TEST.

*NOTE: USE THE 'B'' PROGRAM COMMAND TO MODIFY THIS LOCATIONS
AFTER PROGRAM LOAD.

7.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REQUIRES 8K OR MORE).
THIS DIAGNOSTIC DOES SUPPORT ''APT'' BUT HAS NOT BEEN RUN UNDER IT.

7.3 POWER FAIL

A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT
WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH
NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).

7.4 MULTIPLE MNCDI INTERFACE TESTING

THIS PROGRAM DOES ''AUTO-SIZE'' THE NUMBER OF MNCDI'S CONNECTED.
THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 MNCDI INTERFACES
WITH CONTIGUOUS BUS ADDRESSES. THE ''AUTO-SIZE'' CAN BE INHIBITED
BY THE OPERATOR SETTING BIT 15 OF LOCATION '$ENV (LOC. 1214) OR
SETTING SWITCH REGISTER BIT 12 TO A ONE. USE THE 'B'' PROGRAM
COMMAND TO LOAD THE BASE AND VECTOR ADDRESSES.

7.5 RESTRICTIONS

ALL USER CONNECTIONS MUST BE REMOVED.

8.0 EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS
TO ABOUT 20 SECONDS WITH ITERATIONS ENABLED WITH ONE MNCDI CONNECTED.
AN END PASS MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS.
END OF PASS WILL ALSO REPORT TOTAL ERROR COUNT AND ANY UNIT'S THAT HAD ERRORED.

9.0 PROGRAM TEST DESCRIPTIONS

9.1 L = LOGIC TESTS

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED
TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE MNCDI INPUT CONTROL.
A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS
AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN
EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING.
THE PROGRAM WILL AUTO-SIZE UP TO 8 MNCDI'S TO BE TESTED.

9.2 W = WRAP-AROUND LOGIC AND DATA TEST

THE PURPOSE IS TO VERIFY PROPER OPERATION OF THE DIGITAL INPUT
SIGNALS FROM THE CABLE CONNECTOR. A MNCDO SUPPLIES DATA AND
CONTROL SIGNALS TO THE MNCDI. THE ROUTINE REQUIRES THE USE OF A
MNCDI, MNCDI TEST MODULE, MNCDO, MNCDO TEST MODULE AND A
SHORT BERG CABLE.

9.3 T = TYPEOUT LOOP FOR INPUT TEST MODULE SWITCHES

THE LOOP ENABLES VERIFICATION OF THE CABLE CONNECTOR INPUT WITHOUT
THE USE OF A MNCDO. THE PROGRAM WILL WAIT FOR THE OPERATOR TO
DEPRESS THE MNCDI TEST MODULE PUSH-BUTTON. WHEN THE OPERATOR DEPRESSES
THE BUTTON, A SIGNAL FLAG (STROBE) IS SENSED AND THE PROGRAM READS
A MNCDI REGISTER. THE CONTENTS OF THE MNCDI TEST MODULE SWITCH REGISTER
CAN BE READ BY READING THE MNCDI INPUT DATA REGISTER. THE VALUE
OF THE INPUT IS REPORTED AS AN OCTAL NUMBER AND A 16 BIT BINARY.
THE OPERATOR MAY CHANGE THE TEST MODULE DATA SWITCHES AND DEPRESS THE
PUSH-BUTTON AGAIN.

9.4 L = LOGIC TEST WITH TESTER SUPPORT (SA 210)

THE IN-HOUSE TESTER CONSISTS OF A INPUT/OUTPUT DEVICE TO SENSE THE
OUTPUT CONTROL SIGNAL AND TO STIMULATE THE INPUT DATA AND CONTROL
SIGNALS. AN EXPANDED LOGIC TEST IS EXECUTED INCLUDING A WRAP-AROUND
DATA AND CONTROL TEST.

10.0 LISTING

```
 15                          .TITLE  CVMNB-A MNCDI  DIAGNOSTIC
(1)                          ;*COPYRIGHT (C) 1978
(1)                          ;*DIGITAL EQUIPMENT CORP.
(1)                          ;*MAYNARD, MASS. 01754
(1)                          ;*
(1)                          ;*PROGRAM BY SHOOP
(1)                          ;*
(1)                          ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)                          ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)                          ;*
 16
 17          171160                  ABASE=171160    ;DEFAULT DIGITAL INPUT ADDRESSES
 18          000120                  AVECT1=120      ;DEFAULT DIGITAL INPUT VECTOR
 19          171260                  ACDW1=171260    ;DEFAULT DIGITAL OUTPUT ADDRESSES <WRAP-AROUND>
 20                          .SBTTL  BASIC DEFINITIONS
(1)
(1)                          ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)          001100          STACK=  1100
(1)                          .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
(1)                          .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)                          ;*MISCELLANEOUS DEFINITIONS
(1)          000011          HT=     11              ;;CODE FOR HORIZONTAL TAB
(1)          000012          LF=     12              ;;CODE FOR LINE FEED
(1)          000015          CR=     15              ;;CODE FOR CARRIAGE RETURN
(1)          000200          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)          177776          PS=     177776          ;;PROCESSOR STATUS WORD
(1)                          .EQUIV  PS,PSW
(1)          177774          STKLMT= 177774          ;;STACK LIMIT REGISTER
(1)          177772          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)          177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
(1)          177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
(1)
(1)                          ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)          000000          R0=     %0              ;;GENERAL REGISTER
(1)          000001          R1=     %1              ;;GENERAL REGISTER
(1)          000002          R2=     %2              ;;GENERAL REGISTER
(1)          000003          R3=     %3              ;;GENERAL REGISTER
(1)          000004          R4=     %4              ;;GENERAL REGISTER
(1)          000005          R5=     %5              ;;GENERAL REGISTER
(1)          000006          R6=     %6              ;;GENERAL REGISTER
(1)          000007          R7=     %7              ;;GENERAL REGISTER
(1)          000006          SP=     %6              ;;STACK POINTER
(1)          000007          PC=     %7              ;;PROGRAM COUNTER
(1)
(1)                          ;*PRIORITY LEVEL DEFINITIONS
(1)          000000          PR0=    0               ;;PRIORITY LEVEL 0
(1)          000040          PR1=    40              ;;PRIORITY LEVEL 1
(1)          000100          PR2=    100             ;;PRIORITY LEVEL 2
(1)          000140          PR3=    140             ;;PRIORITY LEVEL 3
(1)          000200          PR4=    200             ;;PRIORITY LEVEL 4
(1)          000240          PR5=    240             ;;PRIORITY LEVEL 5
(1)          000300          PR6=    300             ;;PRIORITY LEVEL 6
(1)          000340          PR7=    340             ;;PRIORITY LEVEL 7
```

```
(1)
(1)                             ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
(1)          100000             SW15=   100000
(1)          040000             SW14=   40000
(1)          020000             SW13=   20000
(1)          010000             SW12=   10000
(1)          004000             SW11=   4000
(1)          002000             SW10=   2000
(1)          001000             SW09=   1000
(1)          000400             SW08=   400
(1)          000200             SW07=   200
(1)          000100             SW06=   100
(1)          000040             SW05=   40
(1)          000020             SW04=   20
(1)          000010             SW03=   10
(1)          000004             SW02=   4
(1)          000002             SW01=   2
(1)          000001             SW00=   1
(1)                             .EQUIV  SW09,SW9
(1)                             .EQUIV  SW08,SW8
(1)                             .EQUIV  SW07,SW7
(1)                             .EQUIV  SW06,SW6
(1)                             .EQUIV  SW05,SW5
(1)                             .EQUIV  SW04,SW4
(1)                             .EQUIV  SW03,SW3
(1)                             .EQUIV  SW02,SW2
(1)                             .EQUIV  SW01,SW1
(1)                             .EQUIV  SW00,SW0
(1)
(1)                             ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)          100000             BIT15=  100000
(1)          040000             BIT14=  40000
(1)          020000             BIT13=  20000
(1)          010000             BIT12=  10000
(1)          004000             BIT11=  4000
(1)          002000             BIT10=  2000
(1)          001000             BIT09=  1000
(1)          000400             BIT08=  400
(1)          000200             BIT07=  200
(1)          000100             BIT06=  100
(1)          000040             BIT05=  40
(1)          000020             BIT04=  20
(1)          000010             BIT03=  10
(1)          000004             BIT02=  4
(1)          000002             BIT01=  2
(1)          000001             BIT00=  1
(1)                             .EQUIV  BIT09,BIT9
(1)                             .EQUIV  BIT08,BIT8
(1)                             .EQUIV  BIT07,BIT7
(1)                             .EQUIV  BIT06,BIT6
(1)                             .EQUIV  BIT05,BIT5
(1)                             .EQUIV  BIT04,BIT4
(1)                             .EQUIV  BIT03,BIT3
(1)                             .EQUIV  BIT02,BIT2
```

```
(1)                             .EQUIV  BIT01,BIT1
(1)                             .EQUIV  BIT00,BIT0
(1)
(1)                             ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
(1)             000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)             000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)             000014          TBITVEC=14              ;;''T'' BIT
(1)             000014          TRTVEC= 14              ;;TRACE TRAP
(1)             000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)             000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)             000024          PWRVEC= 24              ;;POWER FAIL
(1)             000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)             000034          TRAPVEC=34              ;;''TRAP'' TRAP
(1)             000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)             000064          TPVEC=  64              ;;TTY PRINTER VECTOR
(1)             000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
```

```
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            INHIBIT SIZING THE # OF MNCDI'S
                                ;*        11            INHIBIT ITERATIONS
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<7:0>

                                .SBTTL  TRAP CATCHER

                000000          .=0
                                ;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ''.+2''
                                ;*AND ''JSR PC,R0'' SEQUENCE TO CATCH ILLEGAL INTERRUPTS.
                                ;*AND INTERRUPTS TO THE WRONG VECTOR.
                                ;*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
                                ;*VECTORS.
                000004          .=4
000004  014570  000200          .WORD   IOTRD,200       ;HANDLE BUSS ERROR.
                000174          .=174
000174  000000                  DISPREG:  .WORD 0               ;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:    .WORD 0               ;SOFTWARE SWITCH REGISTER

                000200          .=200
000200  000137  001542          JMP     BEGIN
000204  000137  001534          JMP     RESTRT          ;RESTART ADDRESS
000210  000137  001554          JMP     TESTER          ;TESTER STARTING ADDRESS


                000100          .=100

000100  000104  000200  000002  104,200,RTI             ;LSI-11 'B EVENT'' PROTECTION
```

```
                                .SBTTL  ACT11 HOOKS
                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
        000106                          $SVPC=.                 ;SAVE PC
        000046                          .=46
000046  010142                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
        000052                          .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
        000106                          .=$SVPC                 ;; RESTORE PC
        001000                          .=1000
                                .SBTTL  APT PARAMETER BLOCK
                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
        001000                          .$X=.   ;;SAVE CURRENT LOCATION
        000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
        000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
        001000                          .=.$X   ;;RESET LOCATION COUNTER
                                ;;******************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001170                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000030                  $TSTM:  .WORD   30      ;;RUN TIM OF LONGEST TEST
001006  000030                  $PASTM: .WORD   30      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000030                  $UNITM: .WORD   30      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000032                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
 61                                     .SBTTL  COMMON TAGS
(1)
(2)                                    ;;**********************************************************************
(1)                                    ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(1)                                    ;*USED IN THE PROGRAM.
(1)
(1)             001100                         .=1100
(1)  001100                            $CMTAG:                         ;;START OF COMMON TAGS
(1)  001100  000000                            .WORD   0
(1)  001102     000                    $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
(1)  001103     000                    $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
(1)  001104  000000                    $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
(1)  001106  000000                    $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
(1)  001110  000000                    $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
(1)  001112  000000                    $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
(1)  001114     000                    $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
(1)  001115     001                    $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
(1)  001116  000000                    $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
(1)  001120  000000                    $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
(1)  001122  000000                    $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
(1)  001124  000000                    $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
(1)  001126  000000                    $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
(1)  001130  000000                            .WORD   0               ;;RESERVED--NOT TO BE USED
(1)  001132  000000                            .WORD   0
(1)  001134     000                    $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
(1)  001135     000                    $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
(1)  001136  000000                            .WORD   0
(1)  001140  177570                    SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
(1)  001142  177570                    DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
(1)  001144  177560                    $TKS:   177560                  ;;TTY KBD STATUS
(1)  001146  177562                    $TKB:   177562                  ;;TTY KBD BUFFER
(1)  001150  177564                    $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
(1)  001152  177566                    $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
(1)  001154     000                    $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
(1)  001155     002                    $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
(1)  001156     012                    $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
(1)  001157     000                    $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(1)  001160  000000                    $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
(1)  001162  000000                    $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
(1)  001164     077                    $QUES:  .ASCII  /?/             ;;QUESTION MARK
(1)  001165     015                    $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
(1)  001166  000012                    $LF:    .ASCIZ  <12>            ;;LINE FEED
(2)                                    ;;**********************************************************************
(2)                                    .SBTTL  APT MAILBOX-ETABLE
(2)
(3)                                    ;;**********************************************************************
(2)                                    .EVEN
(2)  001170                            $MAIL:                          ;;APT MAILBOX
(2)  001170  000000                    $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
(2)  001172  000000                    $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
(2)  001174  000000                    $TESTN: .WORD   ATESTN          ;;TEST NUMBER
(2)  001176  000000                    $PASS:  .WORD   APASS           ;;PASS COUNT
(2)  001200  000000                    $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
(2)  001202  000000                    $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
```

```
(2)  001204  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(2)  001206  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001210                        $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001210     000                $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001211     000                $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001212  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001214  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001216  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)  001220     000                $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001221     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)  001222  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
(2)  001224     000                $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001225     000                $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001226  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001230     000                $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001231     000                $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001232  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001234     000                $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001235     000                $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001236  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001240  000120                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001242  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001244  171160                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001246  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001250  171260                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001252  000000                $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
(2)  001254                        $ETEND:
(2)                                .MEXIT
```

```
(1)                                     .SBTTL  ERROR POINTER TABLE
(1)
(1)                                     ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1)                                     ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1)                                     ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1)                                     ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1)                                     ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1)                                     ;*      EM              ;;POINTS TO THE ERROR MESSAGE
(1)                                     ;*      DH              ;;POINTS TO THE DATA HEADER
(1)                                     ;*      DT              ;;POINTS TO THE DATA
(1)                                     ;*      DF              ;;POINTS TO THE DATA FORMAT
(1)
(1)
(1)  001254                             $ERRTB:
 62
 63                                     ;ITEM   1
 64  001254  015652  017150  020512             EM1,DH1,DT1,DF0         ;MNCDO BUS ERROR
     001262  020674
 65                                     ;ITEM   2
 66  001264  016513  017406  020614             EM12,DH12,DT12,DF0      ;MNCDO-MNCDI WRAP-AROUND STATUS ERROR
     001272  020674
 67                                     ;ITEM   3
 68  001274  016604  017451  020632             EM13,DH13,DT13,DF0      ;MNCDO-MNCDI WRAP-AROUND DATA ERROR
     001302  020674
 69                                     ;ITEM   4
 70  001304  016673  017275  020554             EM14,DH7,DT7,DF0        ;MNCDI INCORRECT I.D. VALUE
     001312  020674
 71                                     ;ITEM   5
 72  001314  016150  017201  020524             EM5,DH5,DT5,DF0         ;MNCDI BUS ERROR
     001322  020674
 73                                     ;ITEM   6
 74  001324  016255  017241  020540             EM6,DH6,DT6,DF0         ;MNCDI STIMULUS REGISTER ERROR
     001332  020674
 75                                     ;ITEM   7
 76  001334  016330  017275  020554             EM7,DH7,DT7,DF0         ;MNCDI STATUS REGISTER ERROR
     001342  020674
 77                                     ;ITEM   10
 78  001344  016401  017331  020570             EM10,DH10,DT10,DF0      ;MNCDI DATA REGISTER ERROR
     001352  020674
 79                                     ;ITEM   11
 80  001354  016450  017364  020604             EM11,DH11,DT11,DF0      ;MNCDI INTERRUPT ERROR
     001362  020674
 81                                     ;ITEM   12
 82  001364  016740  017514  020650             EM15,DH15,DT15,DF0      ;MNCDI ILLEGAL OR TRAP INTERRUPT
     001372  020674
 83                                     ;ITEM   13
 84  001374  017011  017547  020662             EM16,DH16,DT16,DF0      ;EXISTING MNCDI FAILED TO RESPOND
     001402  020674
```

```
                        OCSR:   0
                        OCSR1:  0                       ;HIGH BYTE ADDRESS

                        DOR:    0
                        DOR1:   0                       ;HIGH BYTE ADDRESS

                        ICSR:   0
                        ICSR1:  0                       ;HIGH BYTE ADDRESS

                        DIR:    0
                        DIR1:   0
                        SBR:    0
                        SBR1:   0                       ;HIGH BYTE ADDRESS

                        DIDINV: 0                       ;INPUT INTERREPT VECTOR # 1
                        DIDINS: 0

                        DIEINV: 0                       ;INPUT INTERRUPT VECTOR # 2
                        DIEINS: 0

                        DWARF:  0       ;0= NO EDGE CONNECTOR, =1 IN-HOUSE TESTER, =2 WRAP-AROUND
                        TEMP:   0
                        TEMP1:  0
                        TEMP2:  0                       ;RESTART INDICATOR
                        TSTR0:  167770                  ;IN-HOUSE TESTER ADDRESS
                        TSTR2:  167772
                        TSTR4:  167774
                        TSTR6:  167776
                        MASKNM: 0                       ;DEVICE MASK
                        VADDRO: 4                       ;MULTIPLE OUTPUT UNITS ADDRESSESS DIFFERENCE
                        VADDR:  10                      ;MULTIPLE INPUT UNITS, ADDRESS DIFFERENCE
                        VECLST: AVECT1                  ;VECTOR FOR UNIT #0
                                AVECT1+10               ;               #1
                                AVECT1+20               ;               #2
                                AVECT1+30               ;               #3
                                AVECT1+350              ;               #4
                                AVECT1+340              ;               #5
                                AVECT1+330              ;               #6
                                AVECT1+320              ;               #7
                        VECOFF: 0                       ;VECTOR OFFSET FROM DEFAULT
                                10                      ;               #1
                                20                      ;               #2
                                30                      ;               #3
                                350                     ;               #4
                                340                     ;               #5
                                330                     ;               #6
                                320                     ;               #7
                        EVER:   0
                        BADUNT: 0                       ;BAD UNIT INDICATOR
                        UNITBD: 0

                                BITDAT=BIT12            ;MAINT INPUT INHIBIT
                                BITEXT=BIT11            ;INPUT MAINT STROBE
```

```
  212  001534  005237  001446          RESTRT: INC     TEMP2           ;INDICATE RESTART
  213  001540  000412                          BR      RBEG0
  214  001542  005037  001440          BEGIN:  CLR     DWARF           ;CLEAR DWARF MODE INDICATOR
  215  001546  005037  001446                  CLR     TEMP2
  216  001552  000405                          BR      RBEG0
  217  001554  012737  000001  001440  TESTER: MOV     #1,DWARF        ;INDICATE MNCDI IN-HOUSE TESTER MODE
  218  001562  005037  001446                  CLR     TEMP2
  219  001566                          RBEG0:
  (1)                                 .SBTTL  INITIALIZE THE COMMON TAGS
  (1)                                 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
  (1)  001566  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
  (1)  001572  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
  (1)  001574  022706  001140                  CMP     #SWR,R6 ;;DONE?
  (1)  001600  001374                          BNE     .-6             ;;LOOP BACK IF NO
  (1)  001602  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
  (1)                                 ;;INITIALIZE A FEW VECTORS
  (1)  001606  012737  012006  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
  (1)  001614  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
  (1)  001622  012737  012326  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
  (1)  001630  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
  (1)  001636  012737  014504  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
  (1)  001644  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
  (1)  001652  012737  013624  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
  (1)  001660  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
  (1)  001666  013737  010110  010102          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
  (1)  001674  005037  001160                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
  (1)  001700  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
  (1)  001704  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
  (1)  001712  012737  001712  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
  (1)  001720  012737  001720  001110          MOV     #.,$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)  001726  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
  (2)  001732  012737  001766  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
  (2)  001740  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
  (2)  001746  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
  (2)  001754  022777  177777  177156          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
  (2)  001762  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
  (2)                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
  (2)  001764  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
  (2)  001766  012716  001774          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
  (2)  001772  000002                          RTI
  (2)  001774  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
  (2)  002002  012737  000174  001142          MOV     #DISPREG,DISPLAY
  (2)  002010  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
  (1)
  (2)  002014  005037  001176                  CLR     $PASS           ;;CLEAR PASS COUNT
  (2)  002020  132737  000200  001211          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
  (2)  002026  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
  (2)  002030  012737  001212  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
  (2)  002036                          67$:
```

```
                                        ;ROUTINE TO OVERLAY THE '$TYPE'' ROUTINE
002036  012737  005046  014146                MOV     #5046,$TYPE     ;CLR -(SP)
002044  012737  012746  014150                MOV     #12746,$TYPE+2  ;MOV #$TYPE+12,-(SP)
002052  012737  014160  014152                MOV     #$TYPE+12,$TYPE+4
002060  012737  000002  014154                MOV     #RTI,$TYPE+6    ;RTI
002066  004737  010524                        JSR     PC,$TKINT       ;ENABLE TKB INTR.
                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002072  005227  177777                        INC     #-1             ;;FIRST TIME?
002076  001053                                BNE     68$             ;;BRANCH IF NO
002100  022737  010142  000042                CMP     #$ENDAD,@#42    ;;ACT-11?
002106  001447                                BEQ     68$             ;;BRANCH IF YES
002110  104401  002156                        TYPE    ,69$            ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002114  005737  000042                        TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
002120  001012                                BNE     70$             ;;BRANCH IF YES
002122  123727  001210  000001                CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
002130  001406                                BEQ     70$             ;;BRANCH IF YES
002132  023727  001140  000176                CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
002140  001005                                BNE     71$             ;;BRANCH IF NO
002142  104407                                GTSWR                   ;;GET SOFT-SWR SETTINGS
002144  000403                                BR      71$
002146  112737  000001  001134  70$:          MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
002154                                71$:
002154  000424                                BR      68$             ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <CRLF>#CVMNB-A  MNCDI (DIGITAL IN) DIAGNOSTIC#<CRLF>
002226                                  68$:
002226  105737  001134                        TSTB    $AUTOB          ;TEST IF ACT/XXDP/ACT AUTO MODE
002232  001402                                BEQ     3$              ;BR IF NOT
002234  000137  002520                        JMP     LOGIC           ;RUN LOGIC TEST
002240  005737  001446                3$:     TST     TEMP2           ;TEST IF RESTART
002244  001015                                BNE     MTEST1          ;BR IF YES
002246  104401  020044                        TYPE,   SETUP0          ;INFORM OPERATOR ABOUT FRONT SWITCH
002252  022737  000001  001440                CMP     #1,DWARF        ;TEST IF TESTER
002260  001005                                BNE     MTEST           ;BR IF NOT AND TELL OPERATOR THE HEADER
002262  104401  020145                        TYPE,   SETUP1          ;INFORM OPERATOR ABOUT TESTER
002266  013737  001450  001250                MOV     TSTR0,$CDW1     ;LOAD TESTER DEFAULT FOR WRAPAROUND
```

```
                                                .SBTTL  KEYBOARD COMMAND DECODER
002274  104401  015046          MTEST:  TYPE,   PRIME0          ;TELL THE OPER. THE TESTS AVAIL
002300  000005                  MTEST1: RESET
002302  052777  000100  176634          BIS     #BIT6,@$TKS     ;ENABLE TKB INTR.
002310  005037  001176                  CLR     $PASS           ;PRIME THE PASS COUNT
002314  005037  001112                  CLR     $ERTTL          ;PRIME THE TOTAL # OF ERRORS
002320  005037  001526                  CLR     EVER            ;INIT. THE UNIT TYPEOUT
002324  004737  003010                  JSR     PC,FIXADR       ;ENSURE BASE AND VECTOR ADDRESS IS LOADED
002330  104401  015562                  TYPE,   DOT             ;INDICATE THE REST POINT
002334  104412                          RDLIN                   ;GET OPER. INPUT
002336  013637  002512                  MOV     @(SP)+,RUNIT    ;SAVE THE FIRST CHAR.
002342  142737  000040  002512          BICB    #40,RUNIT       ;ENSURE UPPER CASE
002350  122737  000102  002512          CMPB    #'B,RUNIT       ;TEST IF 'B'
002356  001002                          BNE     1$              ;BR IF NOT
002360  000137  003252                  JMP     BASEXC          ;CHANGE INPUT BASE ADDRESS
002364  122737  000107  002512  1$:     CMPB    #'G,RUNIT       ;TEST IF 'G'
002372  001002                          BNE     2$              ;BR IF NOT
002374  104407                          GTSWR                   ;GET SWITCH VALUE
002376  000740                          BR      MTEST1          ;AND RETYPE THE DOT
002400  122737  000110  002512  2$:     CMPB    #'H,RUNIT       ;TEST IF 'H'
002406  001732                          BEQ     MTEST           ;BR IF YES
002410  122737  000114  002512          CMPB    #'L,RUNIT       ;TEST IF 'L'
002416  001005                          BNE     3$              ;BR IF NOT
002420  042737  000002  001440          BIC     #2,DWARF        ;REMOVE DWARF CONNECTED INDICATOR
002426  000137  002520                  JMP     LOGIC           ;RUN LOGIC TEST NON-WRAPAROUND
002432  122737  000117  002512  3$:     CMPB    #'O,RUNIT       ;TEST IF 'O'
002440  001002                          BNE     4$              ;BR IF NOT
002442  000137  003210                  JMP     BASEXD          ;GET OUTPUT BASE ADDRESS
002446  122737  000124  002512  4$:     CMPB    #'T,RUNIT       ;TEST IF 'T'
002454  001002                          BNE     5$              ;BR IF NOT
002456  000137  010262                  JMP     DIDATA          ;RUN INPUT SWITCH TYPEOUT
002462  122737  000127  002512  5$:     CMPB    #'W,RUNIT       ;TEST IF 'W'
002470  001005                          BNE     77$             ;BR IF NOT
002472  012737  000002  001440          MOV     #2,DWARF        ;INDICATE WRAPAROUND MODE
002500  000137  002514                  JMP     LOGIC0          ;RUN LOGIC TEST
002504  104401  001164          77$:    TYPE,   $QUES
002510  000673                          BR      MTEST1          ;TRY AGAIN
002512  000000                  RUNIT:  0
```

```
                                                .SBTTL  DETERMINE THE NUMBER OF MNCDI'S ON THE SYSTEM
002514  104401  020220                  LOGIC0: TYPE,   SETUP2          ;TELL OPER. THE CABLE MUST BE THERE
002520  013737  001244  001126          LOGIC:  MOV     $BASE,$BDDAT    ;GET BASE ADDRESS
002526  005037  001460                          CLR     MASKNM          ;CLR DEVICE MASK
002532  005037  001202                          CLR     $UNIT           ;CLR UNIT NUMBER
002536  012737  002612  000004                  MOV     #2$,ERRVEC      ;LOAD RETURN ADDRESS
002544  005777  176356                  1$:     TST     @$BDDAT         ;TEST IF ADDRESS EXISTS
002550  063737  001464  001126                  ADD     VADDR,$BDDAT    ;UPDATE BUS ADDRESS
002556  005237  001202                          INC     $UNIT           ;UPDATE UNIT COUNT
002562  005737  001210                          TST     $ENV            ;TEST IF 'DO NOT SIZE''
002566  100423                                  BMI     3$              ;BR IF NO SIZEING
002570  032777  010000  176342                  BIT     #SW12,@SWR      ;TEST IF INHIBIT SIZING IS SET
002576  001017                                  BNE     3$              ;BR IF SET
002600  022737  000010  001202                  CMP     #8.,$UNIT       ;TEST IF MAX NUMBER
002606  001356                                  BNE     1$              ;BR IF NOT
002610  000412                                  BR      3$              ;BR IF MAX
002612  022626                          2$:     CMP     (SP)+,(SP)+     ;RESTORE STACK
002614  005737  001202                          TST     $UNIT           ;TEST IF ANY EXIST
002620  001006                                  BNE     3$              ;BR IF ANY ARE THERE
002622  005737  000042                          TST     @#42            ;TEST IF XXDP CHAIN MODE
002626  001003                                  BNE     3$              ;BR IF YES
002630  104001                                  ERROR   1               ;BASE ADDRESS CAUSED A BUS TRAP
002632  000137  010054                          JMP     $EOP
002636  012737  014570  000004  3$:     MOV     #IOTRD,ERRVEC
002644  012737  000200  000006                  MOV     #200,ERRVEC+2
002652  005737  001526                          TST     EVER            ;TEST IF # HAS BEEN REPORTED
002656  100427                                  BMI     4$              ;IF YES BRANCH
002660  023727  001440  000001                  CMP     DWARF,#1        ;TEST IF IN TESTER MODE
002666  001414                                  BEQ     6$              ;BR IF TESTER
002670  104401  017075                          TYPE    ,FOUND1         ;TELL OPERATOR # OF MNCDI'S FOUND
002674  013746  001202                          MOV     $UNIT,-(SP)     ;PUT # TO BE TYPED ON STACK
002700  104405                                  TYPDS
002702  104401  017117                          TYPE    ,FOUND2         ;FINISH MESSAGE
002706  005737  001202                          TST     $UNIT           ;ANY UNITS
002712  001002                                  BNE     6$              ;BR IF SOME
002714  000137  010054                          JMP     $EOP            ;REPORT EOP
002720  013737  001202  001526  6$:     MOV     $UNIT,EVER      ;SAVE THE # OF MNCDI'S FOR LATER
002726  052737  100000  001526          BIS     #BIT15,EVER     ;SET 'REPORTED # FLAG''
002734  000410                                  BR      5$
002736  123737  001526  001202  4$:     CMPB    EVER,$UNIT      ;TEST IF ANY HAVE GONE AWAY
002744  001404                                  BEQ     5$              ;BR IF ALL ARE STILL THERE
002746  113737  001526  001442                  MOVB    EVER,TEMP       ;SAVE FOR ERROR REPORT
002754  104013                                  ERROR   13              ;EXISTING DEVICE FAILED TO RESPOND
002756  005037  001202                  5$:     CLR     $UNIT           ;RESET UNIT POINTER
002762  004737  003010                          JSR     PC,FIXADR       ;FIX BUS ADDRESSES
002766  012737  000001  001460                  MOV     #BIT0,MASKNM    ;LOAD DEVICE MASK
002774  005037  001530                          CLR     BADUNT          ;RESET BAD UNIT INDICATOR
003000  005046                                  CLR     -(SP)           ;LOWER PRIORITY LEVEL 0
003002  012746  003424                          MOV     #TST1,-(SP)
003006  000002                                  RTI
```

```
                                                ;SUBROUTINE TO FIX DEVICE ADDRESS AND BUS VECTORS
003010  012700  001404          FIXADR: MOV     #OCSR,R0        ;LOAD ADDRESS POINTER
003014  013701  001250                  MOV     $CDW1,R1        ;LOAD INITIAL BUS ADDRESS
003020  010120                  1$:     MOV     R1,(R0)+        ;LOAD DEVICE ADDRESS
003022  005201                          INC     R1              ;UPDATE BUS ADDRESS VALUE
003024  020027  001414                  CMP     R0,#ICSR        ;TEST IF DONE WITH BUS ADDRESSES
003030  001373                          BNE     1$              ;BR IF NOT
003032  013701  001244                  MOV     $BASE,R1        ;LOAD INITIAL INPUT BUS ADDRESS
003036  010120                  2$:     MOV     R1,(R0)+        ;LOAD THE ADDRESS
003040  005201                          INC     R1              ;UPDATE THE ADDRESS
003042  020027  001430                  CMP     R0,#DIDINV      ;TEST IF AT END
003046  001373                          BNE     2$              ;BRANCH IF NOT
003050  013701  001240                  MOV     $VECT1,R1       ;LOAD INITIAL INPUT VECTOR
003054  010120                  4$:     MOV     R1,(R0)+        ;LOAD THE VECTOR
003056  005721                          TST     (R1)+           ;BUMP THE ADDRESS
003060  020027  001440                  CMP     R0,#DIEINS+2    ;TEST IF DONE
003064  001373                          BNE     4$              ;BRANCH IF NOT
003066  012700  001466                  MOV     #VECLST,R0      ;GET ACTUAL VECTOR AREA
003072  012701  001506                  MOV     #VECOFF,R1      ;GET VECTOR OFFSET POINTER
003076  013710  001240                  MOV     $VECT1,(R0)     ;GET BASE
003102  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003104  013710  001240          (1)     MOV     $VECT1,(R0)     ;GET BASE
003110  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003112  013710  001240          (1)     MOV     $VECT1,(R0)     ;GET BASE
003116  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003120  013710  001240          (1)     MOV     $VECT1,(R0)     ;GET BASE
003124  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003126  013710  001240          (1)     MOV     $VECT1,(R0)     ;GET BASE
003132  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003134  013710  001240          (1)     MOV     $VECT1,(R0)     ;GET BASE
003140  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003142  013710  001240          (1)     MOV     $VECT1,(R0)     ;GET BASE
003146  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003150  013710  001240          (1)     MOV     $VECT1,(R0)     ;GET BASE
003154  062120                  (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET

                                                ;ALSO TO LOAD INTELLIGENT TRAP CATCHER
003156  012700  000250                  MOV     #250,R0         ;LOAD FIRST ADDRESS OF TRAP CATCHER
003162  012701  000252                  MOV     #252,R1         ;LOAD .+2
003166  012702  004700                  MOV     #4700,R2        ;LOAD ILLEGAL INST.
003172  010120                  5$:     MOV     R1,(R0)+        ;LOAD .+2
003174  010220                          MOV     R2,(R0)+        ;LOAD ILLEGAL INST.
003176  022121                          CMP     (R1)+,(R1)+     ;BUMP R1 TWICE
003200  020027  001000                  CMP     R0,#1000        ;TEST IF DONE
003204  001372                          BNE     5$              ;BRANCH IF NOT
003206  000207                          RTS     PC
```

```
                                        ;SUBROUTINE TO ASK THE OPERATOR FOR OUT ADRS
003210  104401                  BASEXD: TYPE
003212  020320                          ADROUT                  ;ASK FOR OUTPUT ADDR.
003214  013746  001250                  MOV     $CDW1,-(SP)     ;GET DEFAULT VALUE
003220  104402                          TYPOC                   ;TELL OPER.
003222  104401  020475                  TYPE,   ENDOUT          ;ADD END
003226  104413                          RDOCT                   ;AND WAIT FOR INPUT
003230  005726                          TST     (SP)+           ;WAS IT A <CR> FOR DEFAULT
003232  001403                          BEQ     1$              ;YES- BRANCH
003234  016637  177776  001250          MOV     -2(SP),$CDW1    ;NO-LOAD NEW ADDR.
003242  004737  003010          1$:     JSR     PC,FIXADR       ;LOAD NEW ADDRESSES
003246  000137  002300                  JMP     MTEST1          ;RETURN

                                        ;SUBROUTINE TO ASK FOR INPUT ADRS AND VEC
003252  104401                  BASEXC: TYPE
003254  020364                          ADRIN                   ;ASK FOR INPUT ADDR.
003256  013746  001244                  MOV     $BASE,-(SP)     ;GET DEFAULT
003262  104402                          TYPOC                   ;TELL OPER. DEFAULT
003264  104401  020475                  TYPE,   ENDOUT          ;ADD END
003270  104413                          RDOCT                   ;AND WAIT FOR INPUT
003272  005726                          TST     (SP)+           ;WAS IT A <CR> FOR DEFAULT
003274  001403                          BEQ     1$              ;YES-BRANCH
003276  016637  177776  001244          MOV     -2(SP),$BASE    ;NO-LOAD NEW ADDR
003304  104401                  1$:     TYPE
003306  020427                          VECIN                   ;ASK FOR INPUT VECTOR
003310  013746  001240                  MOV     $VECT1,-(SP)    ;GET DEFAULT
003314  104402                          TYPOC                   ;TELL OPER THE DEFAULT
003316  104401  020475                  TYPE,   ENDOUT          ;ADD END
003322  104413                          RDOCT                   ;AND WAIT FOR INPUT
003324  005726                          TST     (SP)+           ;WAS IT A <CR> FOR DEFAULT
003326  001403                          BEQ     2$              ;YES BRANCH
003330  016637  177776  001240          MOV     -2(SP),$VECT1   ;LOAD NEW VECTOR
003336  004737  003010          2$:     JSR     PC,FIXADR       ;FIX ADDRESSES AND VECTORS
003342  000137  002300                  JMP     MTEST1          ;AND RETYPE THE DOT

                                .SBTTL  SUBROUTINE TO HANDLE CONTROL C/G

003346  105777  175572          CTRLCG: TSTB    @$TKS           ;INPUT FLAG ?
003352  100022                          BPL     2$              ;BR IF NOT
003354  017737  175566  003422          MOV     @$TKB,CTRCHA    ;READ CHAR.
003362  042737  177640  003422          BIC     #177640,CTRCHA  ;MASK OFF BITS
003370  022737  000003  003422          CMP     #3,CTRCHA       ;TEST IF CONTROL C
003376  001003                          BNE     1$              ;BR IF NOT
003400  005726                          TST     (SP)+           ;CLEAN STACK
003402  000137  002300                  JMP     MTEST1          ;AND RETYPE THE DOT
003406  022737  000007  003422  1$:     CMP     #7,CTRCHA       ;TEST IF CTRL G
003414  001001                          BNE     2$
003416  104407                          GTSWR                   ;GET SWITCHES
003420  000207                  2$:     RTS     PC              ;EXIT
003422  000000                  CTRCHA: 0               ;CHAR. THE OPER TYPED
```

```
                                ;;*****************************************************************
                                ;*TEST 1         VERIFY CORRECT I.D. CODE FOR MNCDI (IN-HOUSE TESTER)
                                ;;*****************************************************************
003424  000004                  TST1:   SCOPE
003426  022737  000001  001440          CMP     #1,DWARF        ;TEST IF ''IN-HOUSE TESTER'' MODE
003434  001020                          BNE     TST2            ;;BR IF NOT
003436  005077  176010                  CLR     @TSTR2          ;ENSURE TESTER MODE
003442  017737  176006  001126          MOV     @TSTR4,$BDDAT   ;READ I.D. VALUE
003450  042737  177417  001126          BIC     #177417,$BDDAT  ;MASK TO OTHER BITS
003456  012737  000140  001124          MOV     #140,$GDDAT     ;LOAD EXPECTED I.D. VALUE
003464  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;COMPARE
003472  001401                          BEQ     TST2            ;;BR IF SAME
003474  104004                          ERROR   4               ;INCORRECT I.D. VALUE FOR MNCDI
                                ;;*****************************************************************
                                ;*TEST 2         VERIFY A MNCDI BUS ADDRESS RESPONSE
                                ;;*****************************************************************
003476  000004                  TST2:   SCOPE
003500  012737  003524  000004          MOV     #1$,ERRVEC              ;LOAD BUS TRAP VECTOR
003506  005777  175702                  TST     @ICSR                   ;TEST INPUT STATUS
003512  005777  175702                  TST     @DIR                    ;TEST INPUT DATA REGISTER
003516  005777  175702                  TST     @SBR                    ;TEST STIM. BUFFER REGISTER
003522  000411                          BR      2$              ;;BR IF NO TIMEOUT
003524  104005                  1$:     ERROR   5               ;BUS TIMEOUT WHEN REFERENCING THE MNCDI
003526  012737  014570  000004          MOV     #IOTRD,ERRVEC   ;RESTORE TRAP
003534  012737  000200  000006          MOV     #200,ERRVEC+2   ;VECTOR
003542  000137  007700                  JMP     REMAIN          ;CHECK FOR MORE UNITS
003546  012737  014570  000004  2$:     MOV     #IOTRD,ERRVEC   ;RESTORE TRAP VECTOR
003554  012737  000200  000006          MOV     #200,ERRVEC+2
                                ;;*****************************************************************
                                ;*TEST 3         FLOAT A 1 ACROSS THE MNCDI STIMULUS BIT REGISTER
                                ;;*****************************************************************
003562  000004                  TST3:   SCOPE
003564  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED BIT
003572  012737  003600  001106          MOV     #1$,$LPADR              ;LOAD LOOP ADDRESS
003600  013777  001124  175616  1$:     MOV     $GDDAT,@SBR             ;LOAD MNCDI STIMULUS BIT REGISTER
003606  017737  175612  001126          MOV     @SBR,$BDDAT             ;READ MNCDI STIMULUS BIT REGISTER
003614  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003622  001401                          BEQ     2$              ;;BR IF EXPECTED
003624  104006                          ERROR   6               ; MNCDI STIMULUS BIT REGISTER FAILED TO HOLD A FLOATING
003626  006337  001124          2$:     ASL     $GDDAT                  ;CHANGE THE DATA
003632  001362                          BNE     1$                      ;BR IF MORE DATA
```

```
                                ;;*******************************************************************
                                ;*TEST 4         FLOAT A 0 ACROSS THE MNCDI STIMULUS BIT REGISTER
                                ;;*******************************************************************
003634  000004                  TST4:   SCOPE
003636  012737  000001  001442          MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
003644  012737  003652  001106          MOV     #1$,$LPADR              ;LOAD LOOP ADDRESS
003652  013737  001442  001124  1$:     MOV     TEMP,$GDDAT             ;LOAD EXPECTED
003660  005137  001124                  COM     $GDDAT                  ;COMPLEMENT
003664  013777  001124  175532          MOV     $GDDAT,@SBR             ;LOAD MNCDI STIMULUS BIT REGISTER
003672  017737  175526  001126          MOV     @SBR,$BDDAT             ;READ MNCDI STIMULUS BIT REGISTER
003700  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003706  001401                          BEQ     2$              ;;BR IF EXPECTED
003710  104006                          ERROR   6               ;MNCDI STIMULUS BIT REGISTER FAILED TO HOLD A FLOATING 0
003712  006337  001442          2$:     ASL     TEMP                    ;CHANGE THE DATA
003716  001355                          BNE     1$                      ;BR IF MORE DATA
                                ;;*******************************************************************
                                ;*TEST 5         ENSURE THAT 'RESET' CLEARS THE MNCDI STIMULUS BIT REGISTER
                                ;;*******************************************************************
003720  000004                  TST5:   SCOPE
003722  012737  000040  001160          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
003730  012777  177777  175466          MOV     #-1,@SBR                ;LOAD BITS TO BE RESET
003736  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
003742  000005                          RESET                           ;CLEAR THE DEVICE
003744  052777  000100  175172          BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
003752  017737  175446  001126          MOV     @SBR,$BDDAT             ;READ MNCDI STIMULUS BIT REGISTER
003760  001401                          BEQ     TST6            ;;BR IF CLEARED
003762  104006                          ERROR   6               ;MNCDI STIMULUS BIT REGISTER FAILED TO CLEAR WITH 'RESET
                                ;;*******************************************************************
                                ;*TEST 6         VERIFY BYTE OPERATION ON THE MNCDI STIMULUS BIT REGISTER
                                ;;*******************************************************************
003764  000004                  TST6:   SCOPE
003766  012737  003774  001106          MOV     #1$,$LPADR              ;LOAD RETURN ADDRESS
003774  012777  177777  175422  1$:     MOV     #-1,@SBR                ;LOAD MNCDI STIMULUS BIT REGISTER
004002  012737  000377  001124          MOV     #377,$GDDAT             ;LOAD EXPECTED
004010  105077  175412                  CLRB    @SBR1                   ;CLEAR HIGH BYTE
004014  017737  175404  001126          MOV     @SBR,$BDDAT             ;READ MNCDI STIMULUS BIT REGISTER
004022  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004030  001404                          BEQ     2$              ;;BR IF SAME
004032  104006                          ERROR   6               ;CLEARING HIGH BYTE CHANGED LOW BYTE
004034  012737  004042  001106          MOV     #2$,$LPADR      ;LOAD LOOP RETURN
004042  012777  177777  175354  2$:     MOV     #-1,@SBR                ;LOAD MNCDI STIMULUS BIT REGISTER
004050  012737  177400  001124          MOV     #177400,$GDDAT          ;LOAD EXPECTED
004056  105077  175342                  CLRB    @SBR                    ;CLEAR LOW BYTE
004062  017737  175336  001126          MOV     @SBR,$BDDAT             ;READ MNCDI STIMULUS BIT REGISTER
004070  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004076  001401                          BEQ     3$              ;;BR IF SAME
004100  104006                          ERROR   6               ;CLEARING LOW BYTE CHANGED HIGH BYTE
004102                          3$:
```

```
                        ;;*****************************************************************
                        ;*TEST 7         TEST THAT BIT1 OF MNCDI STATUS REGISTER IS READ-WRITE
                        ;;*****************************************************************
004102  000004          TST7:   SCOPE
004104  012737  000002  001124          MOV     #BIT1,$GDDAT            ;LOAD EXPECTED
004112  013777  001124  175274          MOV     $GDDAT,@ICSR            ;LOAD BIT1 INTO MNCDI STATUS REGISTER
004120  017737  175270  001126          MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
004126  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
004134  001401                          BEQ     1$              ;;BR IF SET
004136  104007                          ERROR   7               ;BIT1 OF MNCDI STATUS REGISTER FAILED TO SET
                        
004140  043777  001124  175246  1$:     BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
004146  017737  175242  001126          MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
004154  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST THE BIT
004162  001001                          BNE     TST10           ;;BR IF CLEARED
004164  104007                          ERROR   7               ;BIT1 OF MNCDI STATUS REGISTER FAILED TO CLEAR
                        ;;*****************************************************************
                        ;*TEST 10        TEST THAT BIT2 OF MNCDI STATUS REGISTER IS READ-WRITE
                        ;;*****************************************************************
004166  000004          TST10:  SCOPE
004170  012737  000004  001124          MOV     #BIT2,$GDDAT            ;LOAD EXPECTED
004176  013777  001124  175210          MOV     $GDDAT,@ICSR            ;LOAD BIT2 INTO MNCDI STATUS REGISTER
004204  017737  175204  001126          MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
004212  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
004220  001401                          BEQ     1$              ;;BR IF SET
004222  104007                          ERROR   7               ;BIT2 OF MNCDI STATUS REGISTER FAILED TO SET
                        
004224  043777  001124  175162  1$:     BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
004232  017737  175156  001126          MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
004240  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST THE BIT
004246  001001                          BNE     TST11           ;;BR IF CLEARED
004250  104007                          ERROR   7               ;BIT2 OF MNCDI STATUS REGISTER FAILED TO CLEAR
                        ;;*****************************************************************
                        ;*TEST 11        TEST THAT BIT3 OF MNCDI STATUS REGISTER IS READ-WRITE
                        ;;*****************************************************************
004252  000004          TST11:  SCOPE
004254  012737  000010  001124          MOV     #BIT3,$GDDAT            ;LOAD EXPECTED
004262  013777  001124  175124          MOV     $GDDAT,@ICSR            ;LOAD BIT3 INTO MNCDI STATUS REGISTER
004270  017737  175120  001126          MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
004276  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
004304  001401                          BEQ     1$              ;;BR IF SET
004306  104007                          ERROR   7               ;BIT3 OF MNCDI STATUS REGISTER FAILED TO SET
                        
004310  043777  001124  175076  1$:     BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
004316  017737  175072  001126          MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
004324  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST THE BIT
004332  001001                          BNE     TST12           ;;BR IF CLEARED
004334  104007                          ERROR   7               ;BIT3 OF MNCDI STATUS REGISTER FAILED TO CLEAR
```

```
452                                     ;;*******************************************************************
(4)                                     ;*TEST 12       TEST THAT BIT4 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004336  000004                     TST12:  SCOPE
(1)  004340  012737  000020  001124             MOV     #BIT4,$GDDAT            ;LOAD EXPECTED
(1)  004346  013777  001124  175040             MOV     $GDDAT,@ICSR            ;LOAD BIT4 INTO MNCDI STATUS REGISTER
(1)  004354  017737  175034  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004362  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004370  001401                             BEQ     1$              ;;BR IF SET
(1)  004372  104007                             ERROR   7               ;BIT4 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004374  043777  001124  175012  1$:      BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004402  017737  175006  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004410  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004416  001001                             BNE     TST13           ;;BR IF CLEARED
(1)  004420  104007                             ERROR   7               ;BIT4 OF MNCDI STATUS REGISTER FAILED TO CLEAR
453                                     ;;*******************************************************************
(4)                                     ;*TEST 13       TEST THAT BIT5 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004422  000004                     TST13:  SCOPE
(1)  004424  012737  000040  001124             MOV     #BIT5,$GDDAT            ;LOAD EXPECTED
(1)  004432  013777  001124  174754             MOV     $GDDAT,@ICSR            ;LOAD BIT5 INTO MNCDI STATUS REGISTER
(1)  004440  017737  174750  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004446  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004454  001401                             BEQ     1$              ;;BR IF SET
(1)  004456  104007                             ERROR   7               ;BIT5 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004460  043777  001124  174726  1$:      BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004466  017737  174722  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004474  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004502  001001                             BNE     TST14           ;;BR IF CLEARED
(1)  004504  104007                             ERROR   7               ;BIT5 OF MNCDI STATUS REGISTER FAILED TO CLEAR
454                                     ;;*******************************************************************
(4)                                     ;*TEST 14       TEST THAT BIT6 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004506  000004                     TST14:  SCOPE
(1)  004510  012737  000100  001124             MOV     #BIT6,$GDDAT            ;LOAD EXPECTED
(1)  004516  013777  001124  174670             MOV     $GDDAT,@ICSR            ;LOAD BIT6 INTO MNCDI STATUS REGISTER
(1)  004524  017737  174664  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004532  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004540  001401                             BEQ     1$              ;;BR IF SET
(1)  004542  104007                             ERROR   7               ;BIT6 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004544  043777  001124  174642  1$:      BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004552  017737  174636  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004560  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004566  001001                             BNE     TST15           ;;BR IF CLEARED
(1)  004570  104007                             ERROR   7               ;BIT6 OF MNCDI STATUS REGISTER FAILED TO CLEAR
```

```
456                                     ;;*******************************************************************
(4)                                     ;*TEST 15       TEST THAT BIT8 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004572  000004                     TST15:  SCOPE
(1)  004574  012737  000400  001124             MOV     #BIT8,$GDDAT            ;LOAD EXPECTED
(1)  004602  013777  001124  174604             MOV     $GDDAT,@ICSR            ;LOAD BIT8 INTO MNCDI STATUS REGISTER
(1)  004610  017737  174600  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004616  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004624  001401                             BEQ     1$              ;;BR IF SET
(1)  004626  104007                             ERROR   7               ;BIT8 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004630  043777  001124  174556  1$:       BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004636  017737  174552  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004644  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004652  001001                             BNE     TST16           ;;BR IF CLEARED
(1)  004654  104007                             ERROR   7               ;BIT8 OF MNCDI STATUS REGISTER FAILED TO CLEAR
457                                     ;;*******************************************************************
(4)                                     ;*TEST 16       TEST THAT BIT9 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004656  000004                     TST16:  SCOPE
(1)  004660  012737  001000  001124             MOV     #BIT9,$GDDAT            ;LOAD EXPECTED
(1)  004666  013777  001124  174520             MOV     $GDDAT,@ICSR            ;LOAD BIT9 INTO MNCDI STATUS REGISTER
(1)  004674  017737  174514  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004702  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004710  001401                             BEQ     1$              ;;BR IF SET
(1)  004712  104007                             ERROR   7               ;BIT9 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004714  043777  001124  174472  1$:       BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004722  017737  174466  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004730  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004736  001001                             BNE     TST17           ;;BR IF CLEARED
(1)  004740  104007                             ERROR   7               ;BIT9 OF MNCDI STATUS REGISTER FAILED TO CLEAR
458                                     ;;*******************************************************************
(4)                                     ;*TEST 17       TEST THAT BIT12 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004742  000004                     TST17:  SCOPE
(1)  004744  012737  010000  001124             MOV     #BIT12,$GDDAT           ;LOAD EXPECTED
(1)  004752  013777  001124  174434             MOV     $GDDAT,@ICSR            ;LOAD BIT12 INTO MNCDI STATUS REGISTER
(1)  004760  017737  174430  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004766  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004774  001401                             BEQ     1$              ;;BR IF SET
(1)  004776  104007                             ERROR   7               ;BIT12 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  005000  043777  001124  174406  1$:       BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  005006  017737  174402  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  005014  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  005022  001001                             BNE     TST20           ;;BR IF CLEARED
(1)  005024  104007                             ERROR   7               ;BIT12 OF MNCDI STATUS REGISTER FAILED TO CLEAR
```

```
 460                                ;;*****************************************************************
 (4)                                ;*TEST 20       TEST THAT BIT14 OF MNCDI STATUS REGISTER IS READ-WRITE
 (4)                                ;;*****************************************************************
 (3)  005026  000004                TST20:  SCOPE
 (1)  005030  012737  040000  001124        MOV     #BIT14,$GDDAT           ;LOAD EXPECTED
 (1)  005036  013777  001124  174350        MOV     $GDDAT,@ICSR            ;LOAD BIT14 INTO MNCDI STATUS REGISTER
 (1)  005044  017737  174344  001126        MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
 (1)  005052  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
 (2)  005060  001401                        BEQ     1$              ;;BR IF SET
 (1)  005062  104007                        ERROR   7               ;BIT14 OF MNCDI STATUS REGISTER FAILED TO SET
 (1)                                
 (1)  005064  043777  001124  174322  1$:   BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
 (1)  005072  017737  174316  001126        MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
 (1)  005100  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;TEST THE BIT
 (3)  005106  001001                        BNE     TST21           ;;BR IF CLEARED
 (1)  005110  104007                        ERROR   7               ;BIT14 OF MNCDI STATUS REGISTER FAILED TO CLEAR
 461                                ;;*****************************************************************
 (4)                                ;*TEST 21       ENSURE THAT "RESET" CLEARS THE MNCDI STATUS REGISTER
 (4)                                ;;*****************************************************************
 (3)  005112  000004                TST21:  SCOPE
 (2)  005114  012737  000040  001160        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
 (1)  005122  012777  040426  174264        MOV     #40426,@ICSR            ;LOAD BITS TO BE RESET
 (1)  005130  005037  001124                CLR     $GDDAT                  ;CLEAR EXPECTED
 (1)  005134  000005                        RESET                           ;CLEAR THE DEVICE
 (1)  005136  052777  000100  174000        BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
 (1)  005144  017737  174244  001126        MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
 (3)  005152  001401                        BEQ     TST22           ;;BR IF CLEARED
 (1)  005154  104007                        ERROR   7               ;MNCDI STATUS REGISTER FAILED TO CLEAR WITH "RESET"
 462                                ;;*****************************************************************
 (3)                                ;*TEST 22       VERIFY HIGH BYTE OPERATION ON THE INPUT STATUS REGISTER
 (3)                                ;;*****************************************************************
 (2)  005156  000004                TST22:  SCOPE
 463  005160  012777  040426  174226        MOV     #40426,@ICSR            ;LOAD INPUT REG. BIT
 464  005166  105077  174224                CLRB    @ICSR1                  ;CLEAR HIGH BYTE
 465  005172  012737  000026  001124        MOV     #BIT4!BIT2!BIT1,$GDDAT  ;LOAD EXPECTED
 466  005200  017737  174210  001126        MOV     @ICSR,$BDDAT            ;READ INPUT STATUS REG.
 467  005206  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
 468  005214  001401                        BEQ     TST23           ;;BR IF SAME
 469  005216  104007                        ERROR   7               ;CLERING HIGH BYTE CHANGED LOW BYTE
 470
 471                                ;;*****************************************************************
 (3)                                ;*TEST 23       VERIFY LOW BYTE OPERATION ON THE INPUT STATUS REGISTER
 (3)                                ;;*****************************************************************
 (2)  005220  000004                TST23:  SCOPE
 472  005222  012777  040426  174164        MOV     #40426,@ICSR            ;LOAD INPUT REG.
 473  005230  105077  174160                CLRB    @ICSR                   ;CLEAR LOW BYTE
 474  005234  012737  040400  001124        MOV     #40400,$GDDAT           ;LOAD EXPECTED
 475  005242  017737  174146  001126        MOV     @ICSR,$BDDAT            ;READ INPUT STATUS REG.
 476  005250  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
 477  005256  001401                        BEQ     TST24           ;;BR IF SAME
 478  005260  104007                        ERROR   7               ;CLEARING LOW BYTE CHANGED HIGH BYTE
 479
```

```
                                ;;*****************************************************************
                                ;*TEST 24        VERIFY THAT MAINT. STROBE SETS ''INPUT DATA READY''
                                ;;*****************************************************************
005262  000004                  TST24:  SCOPE
005264  005077  174124                  CLR     @ICSR                   ;ENSURE CLEAR FLAG
005270  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED DATA
005276  012777  004200  174110          MOV     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
005304  017737  174104  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS REGISTER
005312  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
005320  001401                        BEQ     TST25           ;;BR IF SAME
005322  104007                        ERROR   7               ;MAINT. STROBE FAILED TO SET ''INPUT DATA READY''

                                ;;*****************************************************************
                                ;*TEST 25        VERIFY THAT ''INPUT DATA READY'' CAN BE WRITTEN TO A ZERO
                                ;;*****************************************************************
005324  000004                  TST25:  SCOPE
005326  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED DATA
005332  012777  004000  174054          MOV     #BITEXT,@ICSR           ;GENERATE MAINT. STROBE
005340  005077  174050                  CLR     @ICSR                   ;CLEAR DATA READY FLAG
005344  017737  174044  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS REGISTER
005352  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005360  001401                        BEQ     TST26           ;;BR IF SAME
005362  104007                        ERROR   7               ;''INPUT DATA READY'' FAILED TO BE WRITTEN TO A ZERO

                                ;;*****************************************************************
                                ;*TEST 26        VERIFY THAT ''INPUT DATA READY'' CAN BE CLEARED BY A 'RESET''
                                ;;*****************************************************************
005364  000004                  TST26:  SCOPE
005366  012737  000040  001160          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
005374  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED DATA
005400  012777  004000  174006          MOV     #BITEXT,@ICSR           ;GENERATE MAINT. STROBE
005406  000005                        RESET
005410  052777  000100  173526          BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
005416  017737  173772  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS REGISTER
005424  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005432  001401                        BEQ     TST27           ;;BR IF CLEARED
005434  104007                        ERROR   7               ;''INPUT DATA READY'' FAILED TO BE CLEARED BY 'RESET''

                                ;;*****************************************************************
                                ;*TEST 27        ''INPUT DATA READY'' WILL NOT SET IF IN STIMILUS MODE AND NO SBR MATCH
                                ;;*****************************************************************
005436  000004                  TST27:  SCOPE
005440  005077  173760                  CLR     @SBR                    ;CLEAR SBR REGISTER
005444  012777  177777  173746          MOV     #-1,@DIR                ;CLEAR INPUT REGISTER
005452  012777  000004  173734          MOV     #BIT2,@ICSR             ;SET STILILUS MODE
005460  052777  004000  173726          BIS     #BITEXT,@ICSR           ;GENERATE MAINT. STROBE
005466  012737  000004  001124          MOV     #BIT2,$GDDAT            ;LOAD EXPECTED
005474  017737  173714  001126          MOV     @ICSR,$BDDAT            ;READ STATUS
005502  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005510  001401                        BEQ     TST30           ;;BR IF CLEARED
005512  104007                        ERROR   7               ;INPUT STROBE SET INPUT READY WHEN IN STIMILUS MODE
```

```
                                ;;*****************************************************************
                                ;*TEST 30        VERIFY THAT 'OVERRUN ERROR' SETS
                                ;;*****************************************************************
005514  000004                  TST30:  SCOPE
005516  012737  100202  001124          MOV     #BIT15!BIT7!BIT1,$GDDAT ;LOAD EXPECTED
005524  012777  000002  173662          MOV     #BIT1,@ICSR             ;SET STROBE MODE
005532  052777  004200  173654          BIS     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
005540  052777  104200  173646          BIS     #BIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE AGAIN
005546  017737  173642  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS REGISTER
005554  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005562  001401                          BEQ     TST31           ;;BR IF SAME
005564  104007                          ERROR   7                       ;'OVER RUN' FAILED TO SET

                                ;;*****************************************************************
                                ;*TEST 31        VERIFY THAT 'OVERRUN ERROR' CAN BE WRITTEN TO A ZERO
                                ;;*****************************************************************
005566  000004                  TST31:  SCOPE
005570  012737  000202  001124          MOV     #BIT7!BIT1,$GDDAT       ;LOAD EXPECTED VALUE
005576  012777  000002  173610          MOV     #BIT1,@ICSR             ;SET STROBE MODE
005604  052777  004200  173602          BIS     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
005612  052777  104200  173574          BIS     #BIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE AGAN
005620  105077  173572                  CLRB    @ICSR1                  ;CLEAR HIGH BYTE OF THE INPUT STATUS REGISTER
005624  017737  173564  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS REGISTER
005632  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005640  001401                          BEQ     TST32           ;;BR IF SAME
005642  104007                          ERROR   7                       ;'OVERRUN ERROR' FAILED TO BE WRITTEN TO A ZERO

                                ;;*****************************************************************
                                ;*TEST 32        VERIFY THAT 'RESET' CLEARS 'OVERRUN ERROR'
                                ;;*****************************************************************
005644  000004                  TST32:  SCOPE
005646  012737  000040  001160          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
005654  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
005660  012777  000002  173526          MOV     #BIT1,@ICSR             ;SET STROBE MODE
005666  052777  004000  173520          BIS     #BITEXT,@ICSR           ;GENERATE MAINT. STROBE
005674  052777  004000  173512          BIS     #BITEXT,@ICSR           ;GENERATE MAINT. STROBE AGAIN
005702  000005                          RESET
005704  052777  000100  173232          BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
005712  017737  173476  001126          MOV     @ICSR,$BDDAT            ;READ INPUT REGISTER
005720  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005726  001401                          BEQ     TST33           ;;BR IF SAME
005730  104007                          ERROR   7                       ;RESET FAILED TO CLEAR 'OVERRUN ERROR'
```

```
                                        ;;*****************************************************************
                                        ;*TEST 33       VERIFY INVERT DATA FUNCTION
                                        ;;*****************************************************************
005732  000004                  TST33:  SCOPE
005734  005037  001124                  CLR     $GDDAT                  ;LOADEXPECTED
005740  012777  010000  173446          MOV     #BITDAT,@ICSR           ;SET INPUT INHIBIT
005746  017737  173446  001126          MOV     @DIR,$BDDAT             ;READ INPUT
005754  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005762  001401                          BEQ     1$              ;;BR IF SAME
005764  104007                          ERROR   7               ;INPUT INHIBIT FAILED TO INHIBIT INPUT
005766  012777  010060  173420  1$:     MOV     #BITDAT!BIT5!BIT4,@ICSR ;SET INVERT DATA AND INPUT INHIBIT
005774  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD EXPECTED
006002  017737  173412  001126          MOV     @DIR,$BDDAT             ;READ INPUT
006010  001001                          BNE     2$                      ;BR IF NON-ZERO
006012  104007                          ERROR   7                       ;INVERT DATA FUNCTION FAILED
006014  023737  001124  001126  2$:     CMP     $GDDAT,$BDDAT           ;COMPARE DATA
006022  001401                          BEQ     3$              ;;BR IF SAME
006024  104007                          ERROR   7               ;INVERT DATA - DATA PATH ERROR

006026  012777  010000  173360  3$:     MOV     #BITDAT,@ICSR           ;SET INPUT INHIBIT
006034  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
006040  017737  173354  001126          MOV     @DIR,$BDDAT             ;READ INPUT
006046  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
006054  001401                          BEQ     TST34           ;;BR IF SAME
006056  104007                          ERROR   7               ;INVERT DATA FUNCTION OR INPUT INHIBIT FAILED

                                        ;;*****************************************************************
                                        ;*TEST 34       VERIFY EACH BIT OF THE MNCDI INPUT DATA REGISTER CAN BE CLEARED
                                        ;;*****************************************************************
006060  000004                  TST34:  SCOPE
006062  012737  006076  001110          MOV     #1$,$LPERR              ;LOAD LOOP ADDRESS ON ERROR
006070  012737  000001  001442          MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
006076  012777  010060  173310  1$:     MOV     #BITDAT!BIT4!BIT5,@ICSR ;LOAD INHIBIT INPUT AND INVERT DATA
006104  013737  001442  001124          MOV     TEMP,$GDDAT             ;LOAD EXPECTED
006112  005137  001124                  COM     $GDDAT                  ;MAKE OPPOSITE
006116  017700  173276                  MOV     @DIR,R0                 ;READ INPUT
006122  013777  001442  173270          MOV     TEMP,@DIR               ;CLEAR THE INPUT BIT
006130  042777  000040  173256          BIC     #BIT5,@ICSR             ;REM INVERT DATA BITOVE
006136  052777  000002  173250          BIS     #BIT1,@ICSR             ;ENABLE EXT. STROBE TO PREVENT  DATA INPUT BEING
006144  017737  173250  001126          MOV     @DIR,$BDDAT             ;READ INPUT REG.
006152  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
006160  001401                          BEQ     2$              ;;BR IF SAME
006162  104010                          ERROR   10              ;INPUT REGISTER BIT FAILED TO CLEAR
006164  006337  001442          2$:     ASL     TEMP                    ;SHIFT THE DATA
006170  001342                          BNE     1$                      ;TRY MORE BITS
```

```
 593                                    ;;*****************************************************************
 (3)                                    ;*TEST 35       VERIFY THAT 'RESET' CLEARS MNCDI INPUT DATA REGISTER
 (3)                                    ;;*****************************************************************
 (2)  006172  000004                    TST35:  SCOPE
 (1)  006174  012737  000040  001160            MOV     #40,$TIMES      ;;DO 40 ITERATIONS
 594  006202  012777  004060  173204            MOV     #BITEXT!BIT5!BIT4,@ICSR ;INVERT DATA AND INHIBIT INPUT
 595  006210  017737  173204  001442            MOV     @DIR,TEMP               ;READ REGISTER
 596  006216  005037  001124                    CLR     $GDDAT                  ;LOAD EXPECTED
 597  006222  000005                            RESET                           ;CLEAR THE INPUT REG.
 598  006224  052777  000100  172712            BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
 599  006232  052777  000004  173154            BIS     #BIT2,@ICSR             ;INHIBIT REG. FROM BEING CLOCKED
 600  006240  017737  173154  001126            MOV     @DIR,$BDDAT             ;READ REGISTER
 601  006246  001401                            BEQ     TST36           ;;BR IF CLEARED
 602  006250  104010                            ERROR   10                      ;RESET FAILED TO CLEAR INPUT REGISTER
 603
 604                                    ;;*****************************************************************
 (3)                                    ;*TEST 36       VERIFY THAT A 2ND STROBE PULSE WILL NOT CHANGE THE DIR DATA
 (3)                                    ;;*****************************************************************
 (2)  006252  000004                    TST36:  SCOPE
 605  006254  012777  010062  173132            MOV     #BIT12!BIT5!BIT4!BIT1,@ICSR     ;DISABLE INPUTS, ENABLE INVERT DATA, EXT
 606  006262  052777  004200  173124            BIS     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
 607  006270  042777  000040  173116            BIC     #BIT5,@ICSR             ;REMOVE INVERT DATA
 608  006276  052777  004200  173110            BIS     #BITEXT!BIT7,@ICSR      ;SET MAINT. STROBE AGAIN
 609  006304  012737  177777  001124            MOV     #-1,$GDDAT              ;LOAD EXPECTED DATA
 610  006312  017737  173102  001126            MOV     @DIR,$BDDAT             ;READ REGISTER
 611  006320  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
 612  006326  001401                            BEQ     TST37           ;;BR IF SAME
 613  006330  104010                            ERROR   10                      ;DATA READY FAILED TO INHIBIT 2ND
 614                                                                            ;STROBE FROM CHAINING THE DIR
```

```
 616                                    ;;*********************************************************************
 (3)                                    ;*TEST 37       INTERRUPT TEST -- VERIFY MNCDI INTERRUPTS VIA DATA READY VECTOR
 (3)                                    ;;*********************************************************************
 (2)  006332  000004                    TST37:  SCOPE
 617  006334  012737  006342  001106             MOV     #64$,$LPADR
 618  006342  012777  006422  173060  64$:      MOV     #1$,@DIDINV             ;LOAD RETURN VECTOR
 619  006350  012777  000200  173054             MOV     #200,@DIDINS            ;LOAD RETURN LEVEL
 620  006356  005046                            CLR     -(SP)
 621  006360  012746  006366                    MOV     #10$,-(SP)
 622  006364  000002                            RTI
 623  006366  012777  000102  173020  10$:      MOV     #BIT6!BIT1,@ICSR        ;SET STROBE MODE
 624  006374  052777  004200  173012             BIS     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
 625  006402  000240                            NOP
 626  006404  000240                            NOP
 627  006406  000240                            NOP
 628  006410  000240                            NOP
 629  006412  005077  172776                    CLR     @ICSR                   ;CLEAR STATUS
 630  006416  104011                            ERROR   11                      ;MNCDI INPUT DATA READY FAILED TO INTERRUPT
 631  006420  000401                            BR      2$              ;;RESET VECTOR
 632
 633  006422  022626                    1$:     CMP     (SP)+,(SP)+             ;CLEAN STACK
 634  006424  005077  172764            2$:     CLR     @ICSR                   ;CLEAR DEVICE
 635  006430  013777  001432  172772             MOV     DIDINS,@DIDINV
 636  006436  012777  004700  172766             MOV     #4700,@DIDINS
 637
 638
 639                                    ;;*********************************************************************
 (3)                                    ;*TEST 40       INTERRUPT TEST -- VERIFY MNCDI INTERRUPTS VIA OVERRUN ERROR
 (3)                                    ;;*********************************************************************
 (2)  006444  000004                    TST40:  SCOPE
 640  006446  012777  006534  172760             MOV     #1$,@DIEINV             ;LOAD RETURN VECTOR
 641  006454  012777  000200  172754             MOV     #200,@DIEINS            ;LOAD RETURN STATUS
 642  006462  005046                            CLR     -(SP)
 643  006464  012746  006472                    MOV     #10$,-(SP)
 644  006470  000002                            RTI
 645  006472  012777  040002  172714  10$:      MOV     #BIT14!BIT1,@ICSR       ;ENABLE INTR. AND STROBE MODE
 646  006500  052777  104200  172706             BIS     #BIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE
 647  006506  052777  104200  172700             BIS     #BIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE AGAIN TO SET OVE
 648  006514  000240                            NOP
 649  006516  000240                            NOP
 650  006520  000240                            NOP
 651  006522  000240                            NOP
 652  006524  005077  172664                    CLR     @ICSR                   ;DISABLE INTR.
 653  006530  104011                            ERROR   11                      ;MNCDI FAILED TO INTERRUPT ON ''OVERRUN ERROR''
 654  006532  000401                            BR      2$              ;;RESET VECTOR
 655
 656  006534  022626                    1$:     CMP     (SP)+,(SP)+             ;CLEAN THE STACK
 657  006536  005077  172652            2$:     CLR     @ICSR                   ;CLEAR DEVICE
 658  006542  013777  001436  172664             MOV     DIEINS,@DIEINV
 659  006550  012777  004700  172660             MOV     #4700,@DIEINS
 660  006556  005046                            CLR     -(SP)
 661  006560  012746  006566                    MOV     #11$,-(SP)
 662  006564  000002                            RTI
 663  006566                    11$:
```

```
                                        ;;*****************************************************************
                                        ;*TEST 41        VERFIY MODE 00  -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
                                        ;;*****************************************************************
(2) 006566 000004                       TST41:  SCOPE
678 006570 005737 001440                        TST     DWARF                   ;TEST IF WRAP-AROUND OR TESTER MODE
679 006574 001002                               BNE     1$                      ;BR IF YES
680 006576 000137 007700                        JMP     REMAIN                  ;NO REPORT END OF PASS
681 006602 012777 177777 172610 1$:             MOV     #-1,@DIR                ;CLEAR INPUT REG.
682 006610 005077 172610                        CLR     @SBR                    ;CLEAR STIM. REG.
683 006614 005077 172574                        CLR     @ICSR                   ;CLEAR INPUT DATA READY FLAG
684 006620 005077 172560                        CLR     @OCSR                   ;CLEAR OUTPUT STATUS
685 006624 012777 025252 172556                 MOV     #25252,@DOR             ;WRITE TO THE OUTPUT DATA REG.
686 006632 012737 000200 001124                 MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
687 006640 017737 172550 001126                 MOV     @ICSR,$BDDAT            ;READ STATUS
688 006646 023737 001124 001126                 CMP     $GDDAT,$BDDAT           ;COMPARE
689 006654 001401                               BEQ     TST42           ;;BR IF SET
690 006656 104002                               ERROR   2                       ;MODE 00  -- EXT. STROBE FAILED TO SET INPUT DAT
691
692                                     ;;*****************************************************************
                                        ;*TEST 42        VERIFY MODE 01  -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
                                        ;;*****************************************************************
(2) 006660 000004                       TST42:  SCOPE
693 006662 012777 177777 172530                 MOV     #-1,@DIR                ;CLEAR INPUT REG.
694 006670 005077 172530                        CLR     @SBR                    ;CLEAR STIM. REG.
695 006674 012777 000002 172512                 MOV     #BIT1,@ICSR             ;CLEAR INPUT DATA READY FLAG
696 006702 005077 172476                        CLR     @OCSR                   ;CLEAR OUTPUT STATUS
697 006706 012777 052525 172474                 MOV     #52525,@DOR             ;WRITE TO THE OUTPUT DATA REG.
698 006714 012737 000202 001124                 MOV     #BIT7!BIT1,$GDDAT       ;LOAD EXPECTED
699 006722 017737 172466 001126                 MOV     @ICSR,$BDDAT            ;READ STATUS
700 006730 023737 001124 001126                 CMP     $GDDAT,$BDDAT           ;COMPARE
701 006736 001401                               BEQ     TST43           ;;BR IF SET
702 006740 104002                               ERROR   2                       ;MODE 01  -- EXT. STROBE FAILED TO SET INPUT DAT
703
704                                     ;;*****************************************************************
                                        ;*TEST 43        VERIFY MODE 10  -- INPUT STROBE WILL NOT SET INPUT DATA READY FLAG
                                        ;;*****************************************************************
(2) 006742 000004                       TST43:  SCOPE
705 006744 012777 177777 172446                 MOV     #-1,@DIR                ;CLEAR INPUT REG.
706 006752 005077 172446                        CLR     @SBR                    ;CLEAR STIM. REG.
707 006756 012777 000004 172430                 MOV     #BIT2,@ICSR             ;CLEAR INPUT DATA READY FLAG
708 006764 005077 172414                        CLR     @OCSR                   ;CLEAR OUTPUT STATUS
709 006770 012777 070707 172412                 MOV     #70707,@DOR             ;WRITE TO THE OUTPUT DATA REG.
710 006776 012737 000004 001124                 MOV     #BIT2,$GDDAT            ;LOAD EXPECTED
711 007004 017737 172404 001126                 MOV     @ICSR,$BDDAT            ;READ STATUS
712 007012 023737 001124 001126                 CMP     $GDDAT,$BDDAT           ;COMPARE
713 007020 001401                               BEQ     TST44           ;;BR IF CLEARED
714 007022 104002                               ERROR   2                       ;MODE 10  -- EXT. STROBE SET INPUT DATA READY FL
```

```
                                ;;*****************************************************************
                                ;*TEST 44        VERIFY INPUT REPLY SETS OUTPUT DONE FLAG
                                ;;*****************************************************************
007024  000004                  TST44:  SCOPE
007026  005077  172356                  CLR     @DOR                    ;CLEAR OUTPUT DATA
007032  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
007040  022737  000001  001440          CMP     #1,DWARF                ;CHECK IF IN TESTER MODE
007046  001003                          BNE     1$                      ;BR IF NOT
007050  012737  100000  001124          MOV     #BIT15,$GDDAT           ;LOAD TESTER WRAPAROUND STATUS
007056  005077  172322          1$:     CLR     @OCSR                   ;CLEAR OUTPUT DONE FLAG
007062  012777  004200  172324          MOV     #BITEXT!BIT7,@ICSR      ;SET INPUT READY FLAG
007070  005077  172320                  CLR     @ICSR                   ;CLEAR INPUT READY FLAG<GEN. INPUT REPLY>
007074  017737  172304  001126          MOV     @OCSR,$BDDAT            ;READ OUTPUT STATUS
007102  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007110  001401                          BEQ     TST45           ;;BR IF SET
007112  104002                          ERROR   2                       ;INPUT REPLY FAILED TO SET OUTPUT DONE FLAG

                                ;;*****************************************************************
                                ;*TEST 45        VERIFY THE MNCDO - WRAPAROUND - MNCDI DATA PATH
                                ;;*****************************************************************
007114  000004                  TST45:  SCOPE
007116  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
007124  012777  177777  172266          MOV     #-1,@DIR                ;CLEAR INPUT REG.
007132  005077  172246                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
007136  005077  172252                  CLR     @ICSR                   ;CLEAR INPUT STATUS
007142  012737  000001  001442          MOV     #BIT0,TEMP              ;LOAD EXPECTED

007150  013737  001442  001124  1$:     MOV     TEMP,$GDDAT             ;LOAD TYPEOUT EXPECTED
007156  013777  001124  172224          MOV     $GDDAT,@DOR             ;LOAD OUTPUT DATA REG.
007164  017737  172230  001126          MOV     @DIR,$BDDAT             ;READ INPUT DATA REGISTER
007172  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007200  001401                          BEQ     2$              ;;BR IF SAME
007202  104003                          ERROR   3                       ;INPUT DATA PATH ERROR

007204  006337  001442          2$:     ASL     TEMP                    ;TRY NEXT BIT
007210  001357                          BNE     1$                      ;BR IF MORE BITS
```

```
747                                     ;;*****************************************************************
(3)                                     ;*TEST 46        VERIFY THE MNCDO - WRAPAROUND - MNCDI INVERTED DATA PATH
(3)                                     ;;*****************************************************************
(2)  007212  000004                     TST46:  SCOPE
(1)  007214  012737  000100  001160             MOV     #100,$TIMES     ;;DO 100 ITERATIONS
748  007222  012777  177777  172170             MOV     #-1,@DIR                ;CLEAR INPUT REG.
749  007230  005077  172150                     CLR     @OCSR                   ;CLEAR OUTPUT STATUS
750  007234  012777  000060  172152             MOV     #BIT5!BIT4,@ICSR        ;LOAD INPUT STATUS <INVERT DATA>
751  007242  012737  000001  001442             MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
752
753  007250  013777  001442  172132  1$:        MOV     TEMP,@DOR               ;LOAD OUTPUT DATA REG.
754  007256  017737  172136  001126             MOV     @DIR,$BDDAT             ;READ INPUT DATA REGISTER
755  007264  013737  001442  001124             MOV     TEMP,$GDDAT             ;GET THE BIT
756  007272  005137  001124                     COM     $GDDAT                  ;INVERT EXPECTED INPUT DATA
757  007276  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
758  007304  001401                             BEQ     2$              ;;BR IF SAME
759  007306  104003                             ERROR   3                       ;INVERTED INPUT DATA PATH ERROR
760
761  007310  006337  001442          2$:       ASL     TEMP                    ;TRY NEXT BIT
762  007314  001355                             BNE     1$                      ;BR IF MORE BITS
763
```

```
765                                     ;;*****************************************************************
(3)                                     ;*TEST 47       VERIFY IN 10 MODE THAT SBR AND INPUT  BITS SET INPUT READY
(3)                                     ;;*****************************************************************
(2)  007316  000004                     TST47:  SCOPE
(1)  007320  012737  000100  001160             MOV     #100,$TIMES     ;;DO 100 ITERATIONS
766                                     ;        INPUT DATA READY FLAG
767                                     ;                LOAD A FLOATING 1 ACROSS THE SBR
768                                     ;                VERIFY THAT ONLY THE CORRECT BIT SET DATA READY
769
770  007326  005077  172056                         CLR     @DOR
771  007332  012737  000001  001442         MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
772
773  007340  005077  172044         1$:     CLR     @DOR                    ;CLEAR OUTPUT BITS
774  007344  005077  172054                 CLR     @SBR                    ;CLEAR SBR REG.
775  007350  012777  000004  172036         MOV     #BIT2,@ICSR             ;CLEAR INPUT READY AND SET MODE 10
776  007356  012777  177777  172034         MOV     #-1,@DIR                ;CLEAR INPUT REG.
777
778  007364  013777  001442  172032         MOV     TEMP,@SBR               ;LOAD SBR REG.
779  007372  042777  000200  172014         BIC     #BIT7,@ICSR             ;CLEAR INPUT READY BIT
780  007400  013777  001442  172002         MOV     TEMP,@DOR               ;LOAD OUTPUT REG.
781
782  007406  012737  100204  001124         MOV     #BIT15!BIT7!BIT2,$GDDAT ;LOAD EXPECTED STATUS
783  007414  017737  171774  001126         MOV     @ICSR,$BDDAT            ;READ STATUS
784  007422  023737  001124  001126         CMP     $GDDAT,$BDDAT           ;COMPARE
785  007430  001401                 BEQ     2$              ;;BR IF SET
786  007432  104002                 ERROR   2               ;INPUT DATA READY FLAG FAILED
787                                                         ;TO SET IN MODE 10 <STIMILUS MODE>
788
789                                     ;NOW LOAD ALL BITS EXCEPT THE FLOATING BIT AND ENSURE INPUT DATA READY DOES NOT SET
790  007434  013737  001442  001444 2$:      MOV     TEMP,TEMP1              ;COPY EXPECTED
791  007442  005137  001444                 COM     TEMP1                   ;USE REVERSE PATTERN
792  007446  005077  171736                 CLR     @DOR                    ;CLEAR OUTPUT REG.
793  007452  012777  177777  171740         MOV     #-1,@DIR                ;CLEAR INPUT REG.
794  007460  042777  100200  171726         BIC     #BIT15!BIT7,@ICSR       ;CLEAR INPUT DATA READY
795  007466  012737  000004  001124         MOV     #BIT2,$GDDAT            ;LOAD EXPECTED
796
797  007474  013777  001444  171706         MOV     TEMP1,@DOR              ;LOAD ALL OTHER DATA BITS
798  007502  017737  171706  001126         MOV     @ICSR,$BDDAT            ;READ INPUT STATUS
799  007510  023737  001124  001126         CMP     $GDDAT,$BDDAT           ;COMPARE
800  007516  001401                 BEQ     3$              ;;BR IF CLEARED
801  007520  104002                 ERROR   2               ;INPUT DATA READY FLAG SET IN ERROR
802                                                         ;UNEXPECTED SBR BIT SET INPUT DATA READY
803  007522  006337  001442         3$:     ASL     TEMP                    ;TRY NEXT BIT
804  007526  001304                 BNE     1$              ;BR IF MORE BITS
805
806                                     ;;*****************************************************************
(3)                                     ;*TEST 50       TEST THE TRANSITION ENABLE AND TRANSITION DETECTION
(3)                                     ;;*****************************************************************
(2)  007530  000004                     TST50:  SCOPE
807  007532  005077  171652                 CLR     @DOR
808  007536  012777  177777  171654         MOV     #-1,@DIR                ;CLEAR INPUT REGISTER
809  007544  012737  010000  001442         MOV     #BIT12,TEMP             ;LOAD INITIAL TRANSITION BIT
810  007552  012777  000404  171634         MOV     #BIT8!BIT2,@ICSR        ;SET STIM. CLEAR READY, ENABLE TRANS.
811  007560  012737  100604  001124         MOV     #BIT15!BIT8!BIT7!BIT2,$GDDAT    ;LOAD EXPECTED STATUS
```

```
007566  042777  100200  171620  1$:     BIC     #BIT15!BIT7,@ICSR               ;CLEAR READY
007574  013777  001442  171622          MOV     TEMP,@SBR                       ;LOAD STIMULUS REG.
007602  013777  001442  171600          MOV     TEMP,@DOR                       ;LOAD INPUT REG. <VIA OUTPUT REG.>
007610  017737  171600  001126          MOV     @ICSR,$BDDAT                    ;READ INPUT STATUS
007616  023737  001124  001126          CMP     $GDDAT,$BDDAT                   ;COMPARE
007624  001401                          BEQ     2$                      ;;BR IF SAME
007626  104002                          ERROR   2                               ;TRANSITION ENABLE OR TRANSITION TO A ONE FAILED
                                ;NOW REMOVE THE TRANSITION DATA BIT (THIS SHOULD CAUSE THE INPUT READY FLAG TO SET AGAIN
007630  012777  177777  171562  2$:     MOV     #-1,@DIR                        ;CLEAR INPUT REG
007636  042777  100200  171550          BIC     #BIT15!BIT7,@ICSR               ;CLEAR INPUT READY FLAG
007644  043777  001442  171536          BIC     TEMP,@DOR                       ;REMOVE THE INPUT DATA
                                                                        ;THIS SHOULD CAUSE THE TRANSITION TO A ZERO
007652  017737  171536  001126          MOV     @ICSR,$BDDAT                    ;READ INPUT STATUS
007660  023737  001124  001126          CMP     $GDDAT,$BDDAT                   ;COMPARE
007666  001401                          BEQ     3$                      ;;BR IF SET
007670  104002                          ERROR   2                               ;TRANSITION TO A ZERO FAILED
007672  006337  001442          3$:     ASL     TEMP                            ;TRY ANOTHER BIT ?
007676  001333                          BNE     1$                              ;BR IF YES
007700                          REMAIN:
                                ;;*****************************************************************
                                ;*TEST 51       DETERMINE IF MORE MNCDI'S REMAIN TO BE TESTED
                                ;;*****************************************************************
007700  000004                  TST51:  SCOPE
007702  012737  000001  001160          MOV     #1,$TIMES               ;;DO 1 ITERATION
007710  005237  001202                  INC     $UNIT                           ;UPDATE UNIT NUMBER
007714  123737  001202  001526          CMPB    $UNIT,EVER                      ;TEST IF MORE
007722  001454                          BEQ     3$                      ;;BR IF NOT
007724  012701  001404                  MOV     #OCSR,R1                        ;LOAD POINTER TO OUTPUT STATUS ADDRESS
007730  063721  001462          1$:     ADD     VADDRO,(R1)+                    ;UPDATE OUTPUT BUS ADDRESS
007734  020127  001414                  CMP     R1,#DOR1+2                      ;TEST IF DONE
007740  001373                          BNE     1$                              ;BRANCH IF NOT
007742  063721  001464          2$:     ADD     VADDR,(R1)+                     ;UPDATE INPUT ADDRESS
007746  020127  001430                  CMP     R1,#SBR1+2                      ;TEST IF DONE
007752  001373                          BNE     2$                              ;BRANCH IF NOT
007754  006337  001460                  ASL     MASKNM                          ;UPDATE ERROR FLAG BIT
007760  004737  012272                  JSR     PC,WHICHU                       ;DETERMINE UNIT #
007764  013700  001532                  MOV     UNITBD,R0                       ;GET UNIT #
007770  006300                          ASL     R0                              ;MAKE WORD
007772  016037  001466  001430          MOV     VECLST(R0),DIDINV               ;GET VALUE
010000  013737  001430  001432          MOV     DIDINV,DIDINS                   ;MAKE OTHER VALUES
010006  062737  000002  001432          ADD     #2,DIDINS
010014  013737  001430  001434          MOV     DIDINV,DIEINV
010022  062737  000004  001434          ADD     #4,DIEINV
010030  013737  001430  001436          MOV     DIDINV,DIEINS
010036  062737  000006  001436          ADD     #6,DIEINS
010044  005037  001102                  CLR     $TSTNM                          ;RESET TEST NUMBER
010050  000137  003424                  JMP     TST1                            ;TEST NEXT UNIT
010054                          3$:
```

```
857                                     .SBTTL  END OF PASS ROUTINE
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;*INCREMENT THE PASS NUMBER ($PASS)
(1)                                     ;*TYPE 'END PASS #XXXXX'' (WHERE XXXXX IS A DECIMAL NUMBER)
(1)                                     ;*IF THERES A MONITOR GO TO IT
(1)                                     ;*IF THERE ISN'T JUMP TO EXTMSG
(1)
(1)  010054                             $EOP:
(1)  010054  000004                             SCOPE
(1)  010056  005037  001102                     CLR     $TSTNM          ;;ZERO THE TEST NUMBER
(1)  010062  005037  001160                     CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
(1)  010066  005237  001176                     INC     $PASS           ;;INCREMENT THE PASS NUMBER
(1)  010072  042737  100000  001176             BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
(1)  010100  005327                             DEC     (PC)+           ;;LOOP?
(1)  010102  000001                     $EOPCT: .WORD   1
(1)  010104  003022                             BGT     $DOAGN          ;;YES
(1)  010106  012737                             MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
(1)  010110  000001                     $ENDCT: .WORD   1
(1)  010112  010102                             $EOPCT
(1)  010114  104401  010161                     TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
(2)  010120  013746  001176                     MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
(2)  010124  104405                             TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1)  010126  104401  010156                     TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
(1)  010132  013700  000042             $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
(1)  010136  001405                             BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
(1)  010140  000005                             RESET                   ;;CLEAR THE WORLD
(1)  010142  004710                     $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
(1)  010144  000240                             NOP                     ;;SAVE ROOM
(1)  010146  000240                             NOP                     ;;FOR
(1)  010150  000240                             NOP                     ;;ACT11
(1)  010152                             $DOAGN:
(1)  010152  000137                             JMP     @(PC)+          ;;RETURN
(1)  010154  010176                     $RTNAD: .WORD   EXTMSG
(1)  010156     377     377     000     $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
(1)  010161     015  042412  042116     $ENDMG: .ASCIZ  <15><12>/END PASS #/
(1)  010166  050040  051501  020123
(1)  010174  000043
858
859  010176  052777  000100  170740  EXTMSG: BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
860  010204  005737  001112                  TST     $ERTTL                  ;TEST IF ANY ERRORS
861  010210  001416                          BEQ     1$                      ;BR IF NONE
862  010212  104401  017600                  TYPE    ,ERRTOT                 ;TYPE TOTAL ERRORS MESSAGE
863  010216  013746  001112                  MOV     $ERTTL,-(SP)            ;PUSH TOTAL ERRORS ON STACK
864  010222  104405                          TYPDS                           ;TYPE IT
865  010224  022737  000001  001460          CMP     #1,MASKNM               ;TEST IF MULTIPLE
866  010232  001405                          BEQ     1$                      ;BR IF NOT
867  010234  104401  017627                  TYPE    ,MESGD                  ;TYPE BAD UNITS
868  010240  013746  001530                  MOV     BADUNT,-(SP)            ;PUSH BAD UNITS ON STACK FOR TYPE OUT
869  010244  104406                          TYPBN                           ;TYPE IT
870  010246  104401  010156          1$:     TYPE,   $ENULL                  ;ENSURE ALL TEXT GOT TYPED
871  010252  004737  003346                  JSR     PC,CTRLCG               ;TEST FOR CTRL C/G
872  010256  000137  002520                  JMP     LOGIC
```

```
                                                .SBTTL  MNCDI TEST MODULE SWITCH TYPEOUT LOOP
010262  012706  001100            DIDATA: MOV     #STACK,SP       ;LOAD THE STACK POINTER
010266  004737  003010                    JSR     PC,FIXADR               ;FIX DEVICE ADDRESSES
010272  104401  010350                    TYPE,   WAITO
010276  005077  171112            1$:     CLR     @ICSR                   ;CLEAR INPUT STATUS
010302  012777  177777  171110            MOV     #-1,@DIR                ;CLEAR INPUT DATA REGISTER
010310  004737  003346            2$:     JSR     PC,CTRLCG               ;TEST IF CTRL C/G
010314  105777  171074                    TSTB    @ICSR                   ;WAIT FOR INPUT READY
010320  100373                            BPL     2$
010322  017746  171072                    MOV     @DIR,-(SP)              ;GET INPUT DATA
010326  104402                            TYPOC                           ;TYPE THE OCTAL VALUE
010330  104401  020502                    TYPE,   ADASH                   ;TYPE A DASH
010334  017746  171060                    MOV     @DIR,-(SP)              ;GET INPUT DATA AGAIN
010340  104406                            TYPBN                           ;TYPE THE BINARY VALUE
010342  104401  020506                    TYPE,   TCRLF                   ;TYPE A CR-LF
010346  000753                            BR      1$

010350     015     012            WAITO:  .BYTE   15,12
010352  040527  052111  047111            .ASCII  \WAITING FOR OPERATOR TO ACTIVIATE MNCDI (D/I)\
010360  020107  047506  020122
010366  050117  051105  052101
010374  051117  052040  020117
010402  041501  044524  044526
010410  052101  020105  047115
010416  042103  020111  042050
010424  044457     051
010427     040  042524  052123            .ASCII  \ TEST MODULE BUTTON\
010434  046440  042117  046125
010442  020105  052502  052124
010450  047117
010452     015     012     000            .BYTE   15,12,0
            010456                      .EVEN
```

```
898                                     .SBTTL  TTY INPUT ROUTINE
(1)
(2)                                     ;;**********************************************************************
(1)                                     .ENABL  LSB
(1)  010456  000000                     $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
(1)  010460  000000                     $TKQIN: .WORD   0               ;;INPUT POINTER
(1)  010462  000000                     $TKQOUT: .WORD  0               ;;OUTPUT POINTER
(1)  010464  000040                     $TKQSRT: .BLKB  32.             ;;TTY KEYBOARD QUEUE
(1)          010524                     $TKQEND=.
(1)
(1)                                     ;*TK INITIALIZE ROUTINE
(1)                                     ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
(1)                                     ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
(1)                                     ;
(1)                                     ;*CALL:
(1)                                     ;*      JSR     PC,$TKINT
(1)                                     ;*      RETURN
(1)                                     ;
(1)  010524  005037  010456             $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
(1)  010530  012737  010464  010460             MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
(1)  010536  013737  010460  010462             MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1)  010544  012737  010574  000060             MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
(1)  010552  012737  000200  000062             MOV     #200,@#TKVEC+2  ;;'BR'' LEVEL 4
(1)  010560  005777  170362                     TST     @$TKB           ;;CLEAR DONE FLAG
(1)  010564  012777  000100  170352             MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
(1)  010572  000207                             RTS     PC              ;;RETURN TO CALLER
(1)
(1)                                     ;*TK SERVICE ROUTINE
(1)                                     ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
(1)                                     ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
(1)                                     ;*IT IN THE QUEUE.
(1)                                     ;*IF THE CHARACTER IS A ''CONTROL-C'' (^C) $TKINT IS CALLED AND
(1)                                     ;*UPON RETURN EXIT IS MADE TO THE ''CONTROL-C'' RESTART ADDRESS (MTEST1)
(1)                                     ;
(1)  010574  117746  170346             $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
(1)  010600  042716  177600                     BIC     #^C177,(SP)     ;;STRIP THE JUNK
(1)  010604  021627  000003                     CMP     (SP),#3         ;;IS IT A CONTROL C?
(1)  010610  001007                             BNE     1$              ;;BRANCH IF NO
(1)  010612  104401  011744                     TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
(1)  010616  004737  010524                     JSR     PC,$TKINT       ;;INIT THE KEYBOARD
(1)  010622  005726                             TST     (SP)+           ;;CLEAN UP STACK
(1)  010624  000137  002300                     JMP     MTEST1          ;;CONTROL C RESTART
(1)  010630  021627  000007             1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
(1)  010634  001004                             BNE     2$              ;;BRANCH IF NO
(1)  010636  022737  000176  001140             CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
(1)  010644  001500                             BEQ     6$              ;;GO TO SWR CHANGE
(1)
(1)  010646                             2$:
(1)  010646  022737  000040  010456             CMP     #32.,$TKCNT     ;;IS THE QUEUE FULL?
(1)  010654  001004                             BNE     3$              ;;BRANCH IF NO
(1)  010656  104401  011740                     TYPE    ,$BELL          ;;RING THE TTY BELL
(1)  010662  005726                             TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
(1)  010664  000451                             BR      5$              ;;EXIT
(1)  010666  021627  000023             3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
```

```
(1)  010672  001021                          BNE     32$             ;;BRANCH IF NO
(1)  010674  005077  170244                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  010700  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
(1)  010702  105777  170236          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
(1)  010706  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
(1)  010710  117746  170232                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
(1)  010714  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  010720  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  010724  001366                          BNE     31$             ;;BRANCH IF NO
(1)  010726  012777  000100  170210          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
(1)  010734  000002                          RTI                     ;;RETURN
(1)  010736  005237  010456          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
(1)  010742  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
(1)  010746  002405                          BLT     4$              ;;BRANCH IF YES
(1)  010750  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
(1)  010754  003002                          BGT     4$              ;;BRANCH IF YES
(1)  010756  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
(1)  010762  112677  177472          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
(1)  010766  005237  010460                  INC     $TKQIN          ;;UPDATE THE POINTER
(1)  010772  023727  010460  010524          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
(1)  011000  001003                          BNE     5$              ;;BRANCH IF NO
(1)  011002  012737  010464  010460          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
(1)  011010  000002                  5$:     RTI                     ;;RETURN
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)                                  ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
(1)                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
(1)                                  ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
(1)  011012  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
(1)  011020  001124                          BNE     15$             ;;EXIT IF NOT
(1)  011022  105777  170116                  TSTB    @$TKS           ;;IS A CHAR WAITING?
(1)  011026  100121                          BPL     15$             ;;IF NOT, EXIT
(1)  011030  117746  170112                  MOVB    @$TKB,-(SP)     ;;YES
(1)  011034  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
(1)  011040  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
(1)  011044  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
(1)                                                                  ;;AND EXIT
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
(1)                                  ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
(1)                                  ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
(1)  011046  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  011054  001674                          BEQ     2$              ;;BRANCH IF YES
(1)  011056  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
(1)  011060  004737  010524                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
(1)  011064  005077  170054                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  011070  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
(1)
(1)  011076  104401  011756                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  011102  104401  011763          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  011106  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  011112  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
(1)  011114  104401  011774              TYPE   ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  011120  005046              19$:    CLR    -(SP)           ;;CLEAR COUNTER
(1)  011122  005046                      CLR    -(SP)           ;;THE NEW SWR
(1)  011124  105777  170014      7$:     TSTB   @$TKS           ;;CHAR THERE?
(1)  011130  100375                      BPL    7$              ;;IF NOT TRY AGAIN
(1)
(1)  011132  117746  170010              MOVB   @$TKB,-(SP)     ;;PICK UP CHAR
(1)  011136  042716  177600              BIC    #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)  011142  021627  000003              CMP    (SP),#3         ;;IS IT A CONTROL-C?
(1)  011146  001015                      BNE    9$              ;;BRANCH IF NOT
(1)  011150  104401  011744              TYPE   ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
(1)  011154  062706  000006              ADD    #6,SP           ;;CLEAN UP STACK
(1)  011160  123727  001135  000001      CMPB   $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
(1)  011166  001003                      BNE    8$              ;;BRANCH IF NO
(1)  011170  012777  000100  167746      MOV    #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
(1)  011176  000137  002300      8$:     JMP    MTEST1          ;;CONTROL-C RESTART
(1)
(1)
(1)  011202  021627  000025      9$:     CMP    (SP),#25        ;;IS IT A CONTROL-U?
(1)  011206  001005                      BNE    10$             ;;BRANCH IF NOT
(1)  011210  104401  011751              TYPE   ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  011214  062706  000006      20$:    ADD    #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  011220  000737                      BR     19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  011222  021627  000015      10$:    CMP    (SP),#15        ;;IS IT A <CR>?
(1)  011226  001022                      BNE    16$             ;;BRANCH IF NO
(1)  011230  005766  000004              TST    4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  011234  001403                      BEQ    11$             ;;BRANCH IF YES
(1)  011236  016677  000002  167674      MOV    2(SP),@SWR      ;;SAVE NEW SWR
(1)  011244  062706  000006      11$:    ADD    #6,SP           ;;CLEAR UP STACK
(1)  011250  104401  001165      14$:    TYPE   ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  011254  123727  001135  000001      CMPB   $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  011262  001003                      BNE    15$             ;;BRANCH IF NOT
(1)  011264  012777  000100  167652      MOV    #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  011272  000002              15$:    RTI                    ;;RETURN
(1)  011274  004737  014360      16$:    JSR    PC,$TYPEC       ;;ECHO CHAR
(1)  011300  021627  000060              CMP    (SP),#60        ;;CHAR < 0?
(1)  011304  002420                      BLT    18$             ;;BRANCH IF YES
(1)  011306  021627  000067              CMP    (SP),#67        ;;CHAR > 7?
(1)  011312  003015                      BGT    18$             ;;BRANCH IF YES
(1)  011314  042726  000060              BIC    #60,(SP)+       ;;STRIP-OFF ASCII
(1)  011320  005766  000002              TST    2(SP)           ;;IS THIS THE FIRST CHAR
(1)  011324  001403                      BEQ    17$             ;;BRANCH IF YES
(1)  011326  006316                      ASL    (SP)            ;;NO, SHIFT PRESENT
(1)  011330  006316                      ASL    (SP)            ;;   CHAR OVER TO MAKE
(1)  011332  006316                      ASL    (SP)            ;;   ROOM FOR NEW ONE.
(1)  011334  005266  000002      17$:    INC    2(SP)           ;;KEEP COUNT OF CHAR
(1)  011340  056616  177776              BIS    -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  011344  000667                      BR     7$              ;;GET THE NEXT ONE
(1)  011346  104401  001164      18$:    TYPE   ,$QUES          ;;TYPE ?<CR><LF>
(1)  011352  000720                      BR     20$             ;;SIMULATE CONTROL-U
(1)                              .DSABL  LSB
```

```
(1)
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
(1)                                     ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                     ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                     ;
(1)
(1)  011354  011646                    $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
(1)  011356  016666  000004  000002            MOV     4(SP),2(SP)     ;;THE PS
(1)  011364  005066  000004                    CLR     4(SP)           ;;GET READY FOR A CHARACTER
(2)  011370  005046                            CLR     -(SP)           ;;PUT NEW PS ON STACK
(2)  011372  012746  011400                    MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
(2)  011376  000002                            RTI                     ;;POP NEW PC AND PS
(2)  011400                             64$:
(1)  011400  005737  010456            1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
(1)  011404  001775                            BEQ     1$
(1)  011406  005337  010456                    DEC     $TKCNT          ;;DECREMENT THE COUNTER
(1)  011412  117766  177044  000004            MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
(1)  011420  005237  010462                    INC     $TKQOUT         ;;UPDATE THE POINTER
(1)  011424  023727  010462  010524            CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
(1)  011432  001003                            BNE     2$              ;;BRANCH IF NO
(1)  011434  012737  010464  010462            MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
(1)  011442  000002                    2$:     RTI                     ;;RETURN
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                     ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  011444  010346                    $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  011446  005046                            CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  011450  012703  011700            1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  011454  022703  011740            2$:     CMP     #$TTYIN+32.,R3  ;;BUFFER FULL?
(1)  011460  101456                            BLOS    4$              ;;BR IF YES
(1)  011462  104411                            RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  011464  112613                            MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  011466  122713  000177            10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  011472  001022                            BNE     5$              ;;BR IF NO
(1)  011474  005716                            TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  011476  001007                            BNE     6$              ;;BR IF NO
(1)  011500  112737  000134  011676            MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  011506  104401  011676                    TYPE    ,9$
(1)  011512  012716  177777                    MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  011516  005303                    6$:     DEC     R3              ;;BACKUP BY ONE
(1)  011520  020327  011700                    CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  011524  103434                            BLO     4$              ;;BR IF YES
(1)  011526  111337  011676                    MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  011532  104401  011676                    TYPE    ,9$             ;;GO TYPE
(1)  011536  000746                            BR      2$              ;;GO READ ANOTHER CHAR.
(1)  011540  005716                    5$:     TST     (SP)            ;;RUBOUT KEY SET?
```

```
(1)  011542  001406                          BEQ     7$              ;;BR IF NO
(1)  011544  112737  000134  011676          MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  011552  104401  011676                  TYPE    ,9$
(1)  011556  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  011560  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  011564  001003                          BNE     8$              ;;BR IF NO
(1)  011566  104401  011751                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U''
(1)  011572  000726                          BR      1$              ;;GO START OVER
(1)  011574  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A ''^R''?
(1)  011600  001011                          BNE     3$              ;;BRANCH IF NO
(1)  011602  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
(1)  011604  104401  001165                  TYPE    ,$CRLF          ;;TYPE A ''CR'' & ''LF''
(1)  011610  104401  011700                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  011614  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  011616  104401  001164          4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  011622  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  011624  111337  011676          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  011630  104401  011676                  TYPE    ,9$
(1)  011634  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  011640  001305                          BNE     2$              ;;LOOP IF NOT RETURN
(1)  011642  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  011646  104401  001166                  TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  011652  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  011654  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  011656  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  011660  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  011666  012766  011700  000004          MOV     #$TTYIN,4(SP)
(1)  011674  000002                          RTI                     ;;RETURN
(1)  011676     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  011677     000                  .BYTE   0                       ;;TERMINATOR
(1)  011700  000040          $TTYIN: .BLKB   32.                     ;;RESERVE 32. BYTES FOR TTY INPUT
(1)  011740  177607  000377  $BELL:  .ASCIZ  <207><377><377>         ;;CODE FOR BELL
(1)  011744  041536  005015  000     $CNTLC: .ASCIZ  /^C/<15><12>     ;;CONTROL ''C''
(1)  011751     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>     ;;CONTROL ''U''
(1)  011756  043536  005015  000     $CNTLG: .ASCIZ  /^G/<15><12>     ;;CONTROL ''G''
(1)  011763     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  011770  036440  000040
(1)  011774  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
(1)  012002  020075     000
(1)          012006                  .EVEN
899                                  .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                  ;*SW14=1        LOOP ON TEST
(1)                                  ;*SW11=1        INHIBIT ITERATIONS
(1)                                  ;*SW09=1        LOOP ON ERROR
(1)                                  ;*SW08=1        LOOP ON TEST IN SWR<7:0>
(1)                                  ;*CALL
(1)                                  ;*      SCOPE           ;;SCOPE=IOT
(1)
```

```
(1)  012006                                $SCOPE:
(1)  012006  104410                                CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
(2)  012010  004737  003346                        JSR     PC,CTRLCG
(1)  012014  032777  040000  167116  1$:          BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
(1)  012022  001114                                BNE     $OVER           ;;YES IF SW14=1
(1)                                ;#####START OF CODE FOR THE XOR TESTER#####
(1)  012024  000416                $XTSTR: BR      6$                      ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
(1)                                                                        ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
(1)  012026  013746  000004                        MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  012032  012737  012052  000004                MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
(1)  012040  005737  177060                        TST     @#177060        ;;TIME OUT ON XOR?
(1)  012044  012637  000004                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  012050  000463                                BR      $SVLAD          ;;GO TO THE NEXT TEST
(1)  012052  022626                5$:     CMP     (SP)+,(SP)+             ;;CLEAR THE STACK AFTER A TIME OUT
(1)  012054  012637  000004                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  012060  000423                                BR      7$              ;;LOOP ON THE PRESENT TEST
(1)  012062                        6$:;#####END OF CODE FOR THE XOR TESTER#####
(1)  012062  032777  000400  167050                BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
(1)  012070  001404                                BEQ     2$              ;;BR IF NO
(1)  012072  127737  167042  001102                CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
(1)  012100  001465                                BEQ     $OVER           ;;BR IF YES
(1)  012102  105737  001103        2$:     TSTB    $ERFLG                  ;;HAS AN ERROR OCCURRED?
(1)  012106  001421                                BEQ     3$              ;;BR IF NO
(1)  012110  123737  001115  001103                CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  012116  101015                                BHI     3$              ;;BR IF NO
(1)  012120  032777  001000  167012                BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
(1)  012126  001404                                BEQ     4$              ;;BR IF NO
(1)  012130  013737  001110  001106  7$:          MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  012136  000446                                BR      $OVER
(1)  012140  105037  001103        4$:     CLRB    $ERFLG                  ;;ZERO THE ERROR FLAG
(1)  012144  005037  001160                        CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  012150  000415                                BR      1$              ;;ESCAPE TO THE NEXT TEST
(1)  012152  032777  004000  166760  3$:          BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
(1)  012160  001011                                BNE     1$              ;;BR IF YES
(1)  012162  005737  001176                        TST     $PASS           ;;IF FIRST PASS OF PROGRAM
(1)  012166  001406                                BEQ     1$              ;;      INHIBIT ITERATIONS
(1)  012170  005237  001104                        INC     $ICNT           ;;INCREMENT ITERATION COUNT
(1)  012174  023737  001160  001104                CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)  012202  002024                                BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
(1)  012204  012737  000001  001104  1$:          MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
(1)  012212  013737  012270  001160                MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
(1)  012220  105237  001102        $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
(1)  012224  113737  001102  001174                MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1)  012232  011637  001106                        MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1)  012236  011637  001110                        MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  012242  005037  001162                        CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  012246  112737  000001  001115                MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  012254  013777  001102  166660  $OVER:       MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  012262  013716  001106                        MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(1)  012266  000002                                RTI                     ;;FIXES PS
(1)  012270  003720                $MXCNT: 2000.                           ;;MAX. NUMBER OF ITERATIONS
```

```
  901
  907                                   ;SUBROUTINE TO DETERMINE UNIT #
  908  012272  012737  000000  001532  WHICHU: MOV     #0,UNITBD               ;PRIME THE VALUE
  909  012300  013737  001460  012322          MOV     MASKNM,11$              ;LOAD VALUE
  910  012306  006237  012322          10$:    ASR     11$                     ;SHIFT
  911  012312  001404                          BEQ     12$                     ;BR WHEN DONE
  912  012314  005237  001532                  INC     UNITBD                  ;UPDATE BIT
  913  012320  000772                          BR      10$
  914  012322  000000                  11$:    0
  915  012324  000207                  12$:    RTS     PC                      ;EXIT
  916                                   .SBTTL  ERROR HANDLER ROUTINE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
  (1)                                   ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
  (1)                                   ;*AND GO TO $ERRTYP ON ERROR
  (1)                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
  (1)                                   ;*SW15=1        HALT ON ERROR
  (1)                                   ;*SW13=1        INHIBIT ERROR TYPEOUTS
  (1)                                   ;*SW09=1        LOOP ON ERROR
  (1)                                   ;*CALL
  (1)                                   ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
  (1)
  (1)  012326                           $ERROR:
  (1)  012326  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
  (3)  012330  004737  003346                   JSR     PC,CTRLCG               ;TEST FOR CTRL C/G
  (3)  012334  053737  001460  001530           BIS     MASKNM,BADUNT           ;INCIDATE BAD BIT
  (3)  012342  004737  012272                   JSR     PC,WHICHU               ;DETERMINE UNIT #
  (1)  012346  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
  (1)  012352  001775                           BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
  (1)  012354  013777  001102  166560           MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
  (1)  012362  005237  001112                   INC     $ERTTL          ;;INC THE ERROR COUNT
  (1)  012366  011637  001116                   MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
  (1)  012372  162737  000002  001116           SUB     #2,$ERRPC
  (1)  012400  117737  166512  001114           MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
  (1)  012406  032777  020000  166524           BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
  (1)  012414  001004                           BNE     20$             ;;SKIP TYPEOUTS
  (1)  012416  004737  012530                   JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
  (1)  012422  104401  001165                   TYPE    ,$CRLF
  (1)  012426                           20$:
  (1)  012426  122737  000001  001210           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)  012434  001007                           BNE     2$              ;;NO,SKIP APT ERROR REPORT
  (1)  012436  113737  001114  012450           MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
  (1)  012444  004737  013374                   JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
  (1)  012450     000                   21$:    .BYTE   0
  (1)  012451     000                           .BYTE   0
  (1)  012452  000777                   22$:    BR      22$             ;;APT ERROR LOOP
  (1)  012454  005777  166460           2$:     TST     @SWR            ;;HALT ON ERROR
  (1)  012460  100002                           BPL     3$              ;;SKIP IF CONTINUE
  (1)  012462  000000                           HALT                    ;;HALT ON ERROR!
  (1)  012464  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
  (1)  012466  032777  001000  166444   3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
  (1)  012474  001402                           BEQ     4$              ;;BR IF NO
  (1)  012476  013716  001110                   MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
```

```
(1)  012502  005737  001162          4$:     TST     $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
(1)  012506  001402                          BEQ     5$               ;;BR IF NONE
(1)  012510  013716  001162                  MOV     $ESCAPE,(SP)     ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  012514                          5$:
(1)  012514  022737  010142  000042          CMP     #$ENDAD,@#42     ;;ACT-11 AUTO-ACCEPT?
(1)  012522  001001                          BNE     6$               ;;BRANCH IF NO
(1)  012524  000000                          HALT                     ;;YES
(1)  012526                          6$:
(1)  012526  000002                          RTI                      ;;RETURN
 917                                 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE USES THE ''ITEM CONTROL BYTE'' ($ITEMB) TO DETERMINE WHICH
(1)                                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE'' ($ERRTB),
(1)                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  012530                          $ERRTYP:
(1)  012530  104401  001165                  TYPE    ,$CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
(1)  012534  010046                          MOV     R0,-(SP)         ;;SAVE R0
(1)  012536  005000                          CLR     R0               ;;PICKUP THE ITEM INDEX
(1)  012540  153700  001114                  BISB    @#$ITEMB,R0
(1)  012544  001004                          BNE     1$               ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                                   ;;TYPE THE PC OF THE ERROR
(2)  012546  013746  001116                  MOV     $ERRPC,-(SP)     ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                                   ;;ERROR ADDRESS
(2)  012552  104402                          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  012554  000445                          BR      10$              ;;GET OUT
(1)  012556  005300                  1$:     DEC     R0               ;;ADJUST THE INDEX SO THAT IT WILL
(1)  012560  006300                          ASL     R0               ;;      WORK FOR THE ERROR TABLE
(1)  012562  006300                          ASL     R0
(1)  012564  006300                          ASL     R0
(1)  012566  062700  001254                  ADD     #$ERRTB,R0       ;;FORM TABLE POINTER
(1)  012572  012037  012602                  MOV     (R0)+,2$         ;;PICKUP 'ERROR MESSAGE'' POINTER
(1)  012576  001404                          BEQ     3$               ;;SKIP TYPEOUT IF NO POINTER
(1)  012600  104401                          TYPE                     ;;TYPE THE 'ERROR MESSAGE''
(1)  012602  000000                  2$:     .WORD   0                ;;''ERROR MESSAGE'' POINTER GOES HERE
(1)  012604  104401  001165                  TYPE    ,$CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
(1)  012610  012037  012620          3$:     MOV     (R0)+,4$         ;;PICKUP 'DATA HEADER'' POINTER
(1)  012614  001404                          BEQ     5$               ;;SKIP TYPEOUT IF 0
(1)  012616  104401                          TYPE                     ;;TYPE THE 'DATA HEADER''
(1)  012620  000000                  4$:     .WORD   0                ;;''DATA HEADER'' POINTER GOES HERE
(1)  012622  104401  001165                  TYPE    ,$CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
(1)  012626  010146                  5$:     MOV     R1,-(SP)         ;;SAVE R1
(1)  012630  012001                          MOV     (R0)+,R1         ;;PICKUP 'DATA TABLE'' POINTER
(1)  012632  001415                          BEQ     9$               ;;BR IF NO DATA TO BE TYPED
(1)  012634  012000                          MOV     (R0)+,R0         ;;PICKUP 'DATA FORMAT'' POINTER
(1)  012636  105720                  6$:     TSTB    (R0)+            ;;''OCTAL'' OR 'DECIMAL''
(1)  012640  001003                          BNE     7$               ;;BR IF DECIMAL
(2)  012642  013146                          MOV     @(R1)+,-(SP)     ;;SAVE @(R1)+ FOR TYPEOUT
(2)  012644  104402                          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  012646  000402                          BR      8$
(2)  012650                          7$:
(2)  012650  013146                          MOV     @(R1)+,-(SP)     ;;SAVE @(R1)+ FOR TYPEOUT
(2)  012652  104405                          TYPDS                    ;;GO TYPE--DECIMAL ASCII WITH SIGN
```

```
(1)  012654  005711                 8$:    TST     (R1)          ;;IS THERE ANOTHER NUMBER?
(1)  012656  001403                        BEQ     9$            ;;BR IF NO
(1)  012660  104401  012700               TYPE    ,11$          ;;TYPE TWO(2) SPACES
(1)  012664  000764                        BR      6$            ;;LOOP
(1)
(1)  012666  012601                 9$:    MOV     (SP)+,R1      ;;RESTORE R1
(1)  012670  012600                 10$:   MOV     (SP)+,R0      ;;RESTORE R0
(1)  012672  104401  001165               TYPE    ,$CRLF        ;;''CARRIAGE RETURN'' & 'LINE FEED''
(1)  012676  000207                        RTS     PC            ;;RETURN
(1)  012700  020040     000        11$:   .ASCIZ  /  /          ;;TWO(2) SPACES
(1)          012704                        .EVEN
```

```
  919                                   .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
  (1)                                   ;*OCTAL (ASCII) NUMBER AND TYPE IT.
  (1)                                   ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
  (1)                                   ;*CALL:
  (1)                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                   ;*      TYPOS                   ;;CALL FOR TYPEOUT
  (1)                                   ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
  (1)                                   ;*      .BYTE   M               ;;M=1 OR 0
  (1)                                   ;*                                      ;;1=TYPE LEADING ZEROS
  (1)                                   ;*                                      ;;0=SUPPRESS LEADING ZEROS
  (1)                                   ;*
  (1)                                   ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
  (1)                                   ;*$TYPOS OR $TYPOC
  (1)                                   ;*CALL:
  (1)                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                   ;*      TYPON                   ;;CALL FOR TYPEOUT
  (1)                                   ;*
  (1)                                   ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
  (1)                                   ;*CALL:
  (1)                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                   ;*      TYPOC                   ;;CALL FOR TYPEOUT
  (1)
  (1)  012704  017646  000000           $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
  (1)  012710  116637  000001  013127           MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
  (1)  012716  112637  013131                   MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
  (1)  012722  062716  000002                   ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
  (1)  012726  000406                           BR      $TYPON
  (1)  012730  112737  000001  013127  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
  (1)  012736  112737  000006  013131          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
  (1)  012744  112737  000005  013126  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
  (1)  012752  010346                           MOV     R3,-(SP)        ;;SAVE R3
  (1)  012754  010446                           MOV     R4,-(SP)        ;;SAVE R4
  (1)  012756  010546                           MOV     R5,-(SP)        ;;SAVE R5
  (1)  012760  113704  013131                   MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
  (1)  012764  005404                           NEG     R4
  (1)  012766  062704  000006                   ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
  (1)  012772  110437  013130                   MOVB    R4,$OMODE       ;;SAVE IT FOR USE
  (1)  012776  113704  013127                   MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
  (1)  013002  016605  000012                   MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
  (1)  013006  005003                           CLR     R3              ;;CLEAR THE OUTPUT WORD
  (1)  013010  006105                   1$:     ROL     R5              ;;ROTATE MSB INTO ''C''
  (1)  013012  000404                           BR      3$              ;;GO DO MSB
  (1)  013014  006105                   2$:     ROL     R5              ;;FORM THIS DIGIT
  (1)  013016  006105                           ROL     R5
  (1)  013020  006105                           ROL     R5
  (1)  013022  010503                           MOV     R5,R3
  (1)  013024  006103                   3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
  (1)  013026  105337  013130                   DECB    $OMODE          ;;TYPE THIS DIGIT?
  (1)  013032  100016                           BPL     7$              ;;BR IF NO
  (1)  013034  042703  177770                   BIC     #177770,R3      ;;GET RID OF JUNK
  (1)  013040  001002                           BNE     4$              ;;TEST FOR 0
```

```
(1)  013042  005704                        TST     R4              ;;SUPPRESS THIS 0?
(1)  013044  001403                        BEQ     5$              ;;BR IF YES
(1)  013046  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  013050  052703  000060               BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  013054  052703  000040        5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  013060  110337  013124               MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  013064  104401  013124               TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  013070  105337  013126        7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  013074  003347                        BGT     2$              ;;BR IF MORE TO DO
(1)  013076  002402                        BLT     6$              ;;BR IF DONE
(1)  013100  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  013102  000744                        BR      2$              ;;GO DO THE LAST DIGIT
(1)  013104  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  013106  012604                        MOV     (SP)+,R4        ;;RESTORE R4
(1)  013110  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  013112  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  013120  012616                        MOV     (SP)+,(SP)
(1)  013122  000002                        RTI                     ;;RETURN
(1)  013124     000                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  013125     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  013126     000                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  013127     000                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  013130  000000                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 920                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)                                 ;;**********************************************************************
(1)                                 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)                                 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)                                 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)                                 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)                                 ;*REPLACED WITH SPACES.
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
(1)                                 ;*      TYPDS                   ;;GO TO THE ROUTINE
(1)
(2)  013132                         $TYPDS:
(3)  013132  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  013134  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  013136  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  013140  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  013142  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  013144  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  013150  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  013154  100004                        BPL     1$              ;;BR IF INPUT IS POS.
(1)  013156  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  013160  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  013166  005000                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  013170  012703  013346                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  013174  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  013200  005002                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  013202  016001  013336                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  013206  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
(1)  013210  002402                        BLT     4$              ;;BR IF DONE
(1)  013212  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
```

```
(1)  013214  000774                          BR      3$
(1)  013216  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  013220  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  013222  001002                          BNE     5$              ;;FALL THROUGH IF 0
(1)  013224  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  013226  100407                          BMI     7$              ;;BR IF YES
(1)  013230  106316                  5$:     ASLB    (SP)            ;;MSD?
(1)  013232  103003                          BCC     6$              ;;BR IF NO
(1)  013234  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  013242  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  013246  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  013252  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  013254  005720                          TST     (R0)+           ;;JUST INCREMENTING
(1)  013256  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
(1)  013262  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
(1)  013264  003002                          BGT     8$              ;;GO TO EXIT
(1)  013266  010502                          MOV     R5,R2           ;;GET THE LSD
(1)  013270  000764                          BR      6$              ;;GO CHANGE TO ASCII
(1)  013272  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  013274  100003                          BPL     9$              ;;BR IF NO
(1)  013276  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  013304  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
(3)  013306  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  013310  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  013312  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  013314  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  013316  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  013320  104401  013346                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  013324  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  013332  012616                          MOV     (SP)+,(SP)
(1)  013334  000002                          RTI                     ;;RETURN TO USER
(1)  013336  023420                  $DTBL:  10000.
(1)  013340  001750                          1000.
(1)  013342  000144                          100.
(1)  013344  000012                          10.
(1)  013346  000004                  $DBLK:  .BLKW   4
921                                  .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)  013356  112737  000001  013622  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  013364  112737  000001  013620  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  013372  000403                          BR      $ATYC
(1)  013374  112737  000001  013622  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(2)  013402                          $ATYC:
(3)  013402  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  013404  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  013406  105737  013620                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  013412  001450                          BEQ     5$              ;;IF NOT:  BR
(1)  013414  122737  000001  001210          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  013422  001031                          BNE     3$              ;;IF NOT:  BR
(1)  013424  132737  000100  001211          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  013432  001425                          BEQ     3$              ;;IF NOT:  BR
(1)  013434  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  013440  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
```

```
(1)  013446  005737  001170          1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  013452  001375                        BNE     1$              ;;IF NOT:  WAIT
(1)  013454  010037  001204                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  013460  105720                  2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  013462  001376                        BNE     2$
(1)  013464  163700  001204                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  013470  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  013472  010037  001206                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  013476  012737  000004  001170        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  013504  000413                        BR      5$
(1)  013506  017637  000004  013532  3$:   MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  013514  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  013522  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  013526  004737  014146                JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  013532  000000                  4$:   .WORD   0
(1)  013534                          5$:
(1)  013534  105737  013622          10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  013540  001416                        BEQ     12$             ;;IF NOT:  BR
(1)  013542  005737  001210                TST     $ENV            ;;RUNNING UNDER APT?
(1)  013546  001413                        BEQ     12$             ;;IF NOT:  BR
(1)  013550  005737  001170          11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  013554  001375                        BNE     11$             ;;IF NOT:  WAIT
(1)  013556  017637  000004  001172        MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  013564  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  013572  005237  001170                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  013576  105037  013622          12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  013602  105037  013621                CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  013606  105037  013620                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  013612  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  013614  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  013616  000207                        RTS     PC              ;;RETURN
(1)  013620     000           $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  013621     000           $LFLG:  .BYTE   0               ;;LOG FLAG
(1)  013622     000           $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          013624                  .EVEN
(1)          000200          APTSIZE=200
(1)          000001          APTENV=001
(1)          000100          APTSPOOL=100
(1)          000040          APTCSUP=040
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;**********************************************************************
                                        ;POWER DOWN ROUTINE
013624  012737  013770  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
013632  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
013640  010046                        MOV     R0,-(SP)         ;;PUSH R0 ON STACK
013642  010146                        MOV     R1,-(SP)         ;;PUSH R1 ON STACK
013644  010246                        MOV     R2,-(SP)         ;;PUSH R2 ON STACK
013646  010346                        MOV     R3,-(SP)         ;;PUSH R3 ON STACK
013650  010446                        MOV     R4,-(SP)         ;;PUSH R4 ON STACK
013652  010546                        MOV     R5,-(SP)         ;;PUSH R5 ON STACK
013654  017746  165260                MOV     @SWR,-(SP)       ;;PUSH @SWR ON STACK
013660  010637  013774                MOV     SP,$SAVR6        ;;SAVE SP
013664  012737  013676  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
013672  000000                        HALT
013674  000776                        BR      .-2              ;;HANG UP
                                        ;;**********************************************************************
                                        ;POWER UP ROUTINE
013676  012737  013770  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
013704  013706  013774                MOV     $SAVR6,SP        ;;GET SP
013710  005037  013774                CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
013714  005237  013774          1$:   INC     $SAVR6           ;;WAIT FOR THE INC
013720  001375                        BNE     1$               ;;OF  WORD
013722  012677  165212                MOV     (SP)+,@SWR       ;;POP STACK INTO @SWR
013726  012605                        MOV     (SP)+,R5         ;;POP STACK INTO R5
013730  012604                        MOV     (SP)+,R4         ;;POP STACK INTO R4
013732  012603                        MOV     (SP)+,R3         ;;POP STACK INTO R3
013734  012602                        MOV     (SP)+,R2         ;;POP STACK INTO R2
013736  012601                        MOV     (SP)+,R1         ;;POP STACK INTO R1
013740  012600                        MOV     (SP)+,R0         ;;POP STACK INTO R0
013742  012737  013624  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
013750  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
013756  104401                        TYPE                     ;;REPORT THE POWER FAILURE
013760  013776                $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
013762  012716                        MOV     (PC)+,(SP)       ;;RESTART AT BEGIN
013764  001542                $PWRAD: .WORD   BEGIN            ;;RESTART ADDRESS
013766  000002                        RTI
013770  000000                $ILLUP: HALT                     ;;THE POWER UP SEQUENCE WAS STARTED
013772  000776                        BR      .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
013774  000000                $SAVR6: 0                        ;;PUT THE SP HERE
013776  005015  042522  052123  PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
014004  051101  044524  043516
014012  040440  052106  051105
014020  040440  050040  053517
014026  051105  043040  044501
014034  052514  042522  005015
014042     000
                014044                .EVEN
```

```
  928                                   .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
  (1)                                   ;*CHANGE IT TO BINARY.
  (1)                                   ;*CALL:
  (1)                                   ;*      RDOCT                           ;;READ AN OCTAL NUMBER
  (1)                                   ;*      RETURN HERE                     ;;LOW ORDER BITS ARE ON TOP OF THE STACK
  (1)                                   ;*                                      ;;HIGH ORDER BITS ARE IN $HIOCT
  (1)
  (1)  014044  011646                   $RDOCT: MOV     (SP),-(SP)              ;;PROVIDE SPACE FOR THE
  (1)  014046  016666  000004  000002           MOV     4(SP),2(SP)             ;;INPUT NUMBER
  (3)  014054  010046                           MOV     R0,-(SP)                ;;PUSH R0 ON STACK
  (3)  014056  010146                           MOV     R1,-(SP)                ;;PUSH R1 ON STACK
  (3)  014060  010246                           MOV     R2,-(SP)                ;;PUSH R2 ON STACK
  (1)  014062  104412                   1$:     RDLIN                           ;;READ AN ASCIZ LINE
  (1)  014064  012600                           MOV     (SP)+,R0                ;;GET ADDRESS OF 1ST CHARACTER
  (1)  014066  005001                           CLR     R1                      ;;CLEAR DATA WORD
  (1)  014070  005002                           CLR     R2
  (1)  014072  112046                   2$:     MOVB    (R0)+,-(SP)             ;;PICKUP THIS CHARACTER
  (1)  014074  001412                           BEQ     3$                      ;;IF ZERO GET OUT
  (1)  014076  006301                           ASL     R1                      ;;*2
  (1)  014100  006102                           ROL     R2
  (1)  014102  006301                           ASL     R1                      ;;*4
  (1)  014104  006102                           ROL     R2
  (1)  014106  006301                           ASL     R1                      ;;*8
  (1)  014110  006102                           ROL     R2
  (1)  014112  042716  177770                   BIC     #^C7,(SP)               ;;STRIP THE ASCII JUNK
  (1)  014116  062601                           ADD     (SP)+,R1                ;;ADD IN THIS DIGIT
  (1)  014120  000764                           BR      2$                      ;;LOOP
  (1)  014122  005726                   3$:     TST     (SP)+                   ;;CLEAN TERMINATOR FROM STACK
  (1)  014124  010166  000012                   MOV     R1,12(SP)               ;;SAVE THE RESULT
  (1)  014130  010237  014144                   MOV     R2,$HIOCT
  (3)  014134  012602                           MOV     (SP)+,R2                ;;POP STACK INTO R2
  (3)  014136  012601                           MOV     (SP)+,R1                ;;POP STACK INTO R1
  (3)  014140  012600                           MOV     (SP)+,R0                ;;POP STACK INTO R0
  (1)  014142  000002                           RTI                             ;;RETURN
  (1)  014144  000000                   $HIOCT: .WORD   0                       ;;HIGH ORDER BITS GO HERE
```

```
                                        ;CAUTION        THE FIRST 4 LOC. ARE OVERLAYED TO LOWER INTERRPUT LEVEL
                                        ;       THE OVERLAY OCCURS AFTER THE ''SETUP'' CODE
                                        .SBTTL  TYPE ROUTINE
(1)
(2)                                     ;;*********************************************************************
(1)                                     ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)                                     ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)                                     ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)                                     ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)                                     ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)                                     ;*
(1)                                     ;*CALL:
(1)                                     ;*1) USING A TRAP INSTRUCTION
(1)                                     ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)                                     ;*OR
(1)                                     ;*      TYPE
(1)                                     ;*      MESADR
(1)                                     ;*
(1)
(1)  014146  105737  001157        $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
(1)  014152  100002                        BPL     1$              ;;BR IF YES
(1)  014154  000000                        HALT                    ;;HALT HERE IF NO TERMINAL
(1)  014156  000430                        BR      3$              ;;LEAVE
(1)  014160  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
(1)  014162  017600  000002                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
(1)  014166  122737  000001  001210        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  014174  001011                        BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
(1)  014176  132737  000100  001211        BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1)  014204  001405                        BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
(1)  014206  010037  014216                MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
(1)  014212  004737  013364                JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
(1)  014216  000000                61$:    .WORD   0               ;;MESSAGE ADDRESS
(1)  014220  132737  000040  001211  62$:   BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
(1)  014226  001003                        BNE     60$             ;;YES,SKIP TYPE OUT
(1)  014230  112046                2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1)  014232  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
(1)  014234  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
(1)  014236  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
(1)  014240  062716  000002        3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
(1)  014244  000002                        RTI                     ;;RETURN
(1)  014246  122716  000011        4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
(1)  014252  001430                        BEQ     8$
(1)  014254  122716  000200                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
(1)  014260  001006                        BNE     5$
(1)  014262  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
(1)  014264  104401                        TYPE                    ;;TYPE A CR AND LF
(1)  014266  001165                        $CRLF
(1)  014270  105037  014424                CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
(1)  014274  000755                        BR      2$              ;;GET NEXT CHARACTER
(1)  014276  004737  014360        5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
(1)  014302  123726  001156        6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
(1)  014306  001350                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
(1)  014310  013746  001154                MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
(1)                                                                 ;;AND THE NULL CHAR.
```

```
(1)  014314  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
(1)  014320  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1)  014322  004737  014360                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  014326  105337  014424                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  014332  000770                          BR      7$              ;;LOOP
(1)
(1)                                  ;HORIZONTAL TAB PROCESSOR
(1)
(1)  014334  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  014340  004737  014360          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  014344  132737  000007  014424          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  014352  001372                          BNE     9$              ;;TAB STOP
(1)  014354  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
(1)  014356  000724                          BR      2$              ;;GET NEXT CHARACTER
(1)  014360  105777  164564          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  014364  100375                          BPL     $TYPEC
(1)  014366  116677  000002  164556          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  014374  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  014402  001003                          BNE     1$              ;;BRANCH IF NO
(1)  014404  105037  014424                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  014410  000406                          BR      $TYPEX          ;;EXIT
(1)  014412  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  014420  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
(1)  014422  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  014424  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  014426  000207                  $TYPEX: RTS     PC
(1)
933                                  .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
(1)
(2)                                  ;;******************************************************************
(1)                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
(1)                                  ;*BINARY-ASCII NUMBER AND TYPE IT.
(1)                                  ;*CALL:
(1)                                  ;*      MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
(1)                                  ;*      TYPBN                   ;;TYPE IT
(1)
(1)  014430  010146                  $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
(1)  014432  016601  000006                  MOV     6(SP),R1        ;;GET THE INPUT NUMBER
(1)  014436  000261                          SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1)  014440  112737  000060  014502  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
(1)  014446  006101                          ROL     R1              ;;GET THIS BIT
(1)  014450  001406                          BEQ     2$              ;;DONE?
(1)  014452  105537  014502                  ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1)  014456  104401  014502                  TYPE    ,$BIN           ;;GO TYPE THIS BIT
(1)  014462  000241                          CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
(1)  014464  000765                          BR      1$              ;;GO DO THE NEXT BIT
(1)  014466  012601                  2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
(1)  014470  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  014476  012616                          MOV     (SP)+,(SP)
(1)  014500  000002                          RTI                     ;;RETURN TO USER
(1)  014502     000     000          $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
```

```
                                        .SBTTL  TRAP DECODER

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

014504  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
014506  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
014512  005740                          TST     -(R0)           ;;BACKUP BY 2
014514  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
014516  006300                          ASL     R0              ;;POSITION FOR INDEXING
014520  016000  014540                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
014524  000200                          RTS     R0              ;;GO TO ROUTINE


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

014526  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
014530  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
014536  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
014540  014526                  $TRPAD: .WORD   $TRAP2
014542  014146                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
014544  012730                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
014546  012704                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
014550  012744                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
014552  013132                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
014554  014430                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

014556  011102                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING

014560  011012                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
014562  011354                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
014564  011444                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
014566  014044                          $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
```

```
                                        ;*
                                        ;*THIS ROUTINE WILL PROTECT THE PROGRAM
                                        ;*FROM INTERRUPTS (BAD ONES).
                                        ;*
                                        ;*THE TRAP CATCHER IS SET UP FOR
                                        ;*      .WORD   .+2
                                        ;*      JSR PC,R0
                                        ;*
                                        ;*ILLEGAL INERRUPTS OR INTERRUPTS TO THE WRONG VECTOR
                                        ;*GOTO THE VECTOR AND PICK UP THE ".+2" AS AN ADDRESS

                                        ;*AND "4700" AS NEW STATUS.
                                        ;*THE .+2 AS A PC WILL CAUSE EXECUTION OF THE *JSR PC,R0* (AN ILLEGAL INSTR.).
                                        ;*AND TRAP TO LOCATION "4". IN LOCATION 4 WE HAVE A
                                        ;*POINTER HERE. IF THIS CONDITION CAUSES A TRAP TO LOC. 4.
                                        ;*WE WILL REPORT IT IN THE SAME MANNER THAT WE WOULD
                                        ;*REPORT ANY OTHER ERROR.
                                        ;*IF A BUSS ERROR TRAP DID OCCUR AND CAUSE A TRAP TO 4.
                                        ;*WE WILL HALT.

014570  011637  015042          IOTRD:  MOV     (6),TRTO        ;GET WHERE WE CAME TO.
014574  162737  000004  015042          SUB     #4,TRTO         ;FORM READ ADDR.

014602  023727  015042  001000          CMP     TRTO,#1000      ;DID TRAP FROM LESS THAN ADDR. 1000?
014610  003402                          BLE     2$              ;NO-CONTINUE.

014612  000000                  1$:     HALT                    ;A BUSS ERROR TIME OUT TRAP BROUGHT US HERE.
                                        ;ADDRESS CONTAINED IN TRTO.

014614  000776                          BR      1$              ;DONT'T ALLOW CONTINUE.

014616  016637  000004  015044  2$:     MOV     4(6),TRFRO      ;GET TRAPPED FROM ADDR.
014624  062706  000004                  ADD     #4,SP           ;ADJUST THE STACK POINTER

014630  122737  000037  001102          CMPB    #37,$TSTNM      ;LESS THAN INTERRUPT TESTS?
014636  003402                          BLE     3$              ;NO MUST BE WRONG VECTOR.
                                ;$$$$$$$$$$$$$$$$$$$$$$$$>>>ERROR<<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
014640  104012                          ERROR   12              ;ERROR! ILLEGAL INTERRUPT OR
                                        ;INTERRUPT TO WRONG VECTOR.
                                        ;IF TEST NO. IS LESS THAN 37,ITS
                                        ;LIKELY(BUT NO EXCLUSIVELY)TO BE A
                                        ;DEVICE OTHER THAN THE DEVICE UNDER TEST,
                                        ;IF THE INTERRUPT OCCURED
                                        ;DURING AN INTERRUPT TEST, I'D
                                        ;SUSPECT A PROBLEM WITH THE DEVICE UNDER TEST.
                                        ;IF THE ADDRESS THE INTERRUPT
                                        ;VECTORED TO IS WITHIN THE RANGE OF
                                        ;VECTORS ASSIGNED TO THE DEVICE,
                                        ;THEN I'D SUSPECT THE DEVICE
                                        ;INTERRUPTED ILLEGALLY.
                                        ;IF THE ADDRESS THE INTERRUPT
                                        ;VECTORED TO IS OUTSIDE OF THE
                                        ;RANGE ASSIGNED TO THE DEVICE
                                        ;I'D SUSPECT THAT THE
```

```
                                                        ;DEVICE PUT THE WRONG INTERRUPT
                                                        ;VECTOR ON THE BUS DURING THE INTERRUPT
                                                        ;PROCESS.
                                                        ;       NOTE:
                                                        ;FOR THIS ERROR - DON'T USE
                                                        ; 'LOOP ON ERROR' OPTION.
                                                        ;ALSO EXPECT THAT THE INTERRUPT TEST TO
                                                        ;WILL REPORT THAT THE DEVICE DIDN'T
                                                        ;INTERRUPT.
                                                        ;FOLLOW THE RECOMMENDED PROCEDURE
                                                        ; IN THE DOCUMENT (ON THIS DIAGNOSTIC)
                                                        ;FOR LOOPING ON TEST.

                                ;$$$$$$$$$$$$$$$$$$$$$$>>>ERROR<<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
014642  000002                          RTI
014644  022626                  3$:     CMP     (SP)+,(SP)+     ;CLEAN OFF STACK
014646  005737  001176                  TST     $PASS           ;IS THIS THE FIRST PASS?
014652  001025                          BNE     4$              ;;NO, DON'T REPORT
014654  104401  017644                  TYPE,   VTMSG           ;TELL OPER.
014660  013746  001202                  MOV     $UNIT,-(SP)     ;GET UNIT #
014664  104405                          TYPDS
014666  000240                          NOP
014670  000240                          NOP
014672  104401  017677                  TYPE    ,VTMSG3         ;TYPE 'EXPECTED INTR. AT ''
014676  013746  001430                  MOV     DIDINV,-(SP)    ;;SAVE DIDINV FOR TYPEOUT
  (1)   014702  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
  (1)   014704     003                  .BYTE   3               ;;TYPE 3 DIGIT(S)
  (1)   014705     001                  .BYTE   1               ;;TYPE LEADING ZEROS
014706  104401  017727                  TYPE    ,VTMSG1         ;TYPE '' RECEIVED INTR. AT ''
014712  013746  015042                  MOV     TRTO,-(SP)      ;;SAVE TRTO FOR TYPEOUT
  (1)   014716  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
  (1)   014720     003                  .BYTE   3               ;;TYPE 3 DIGIT(S)
  (1)   014721     001                  .BYTE   1               ;;TYPE LEADING ZEROS
014722  104401  017757                  TYPE    ,VTMSG2         ;TYPE 'RESTARTING TEST''
014726  013777  001432  164474  4$:     MOV     DIDINS,@DIDINV  ;RESTORE INCORRECT VECTOR
014734  013777  001436  164472          MOV     DIEINS,@DIEINV
014742  012777  004700  164466          MOV     #4700,@DIEINS
014750  012777  004700  164454          MOV     #4700,@DIDINS   ;LOCATION WITH .+2 + 4700
014756  013737  015042  001430          MOV     TRTO,DIDINV     ;SET UP ACTUAL VECTOR LOCATION
014764  042737  000007  001430          BIC     #7,DIDINV       ;CLR LOW BITS
014772  013737  001430  001432          MOV     DIDINV,DIDINS   ;SET UP NEW VECTOR VALUES
015000  013737  001430  001434          MOV     DIDINV,DIEINV
015006  013737  001430  001436          MOV     DIDINV,DIEINS
015014  062737  000002  001432          ADD     #2,DIDINS       ;
015022  062737  000004  001434          ADD     #4,DIEINV
015030  062737  000006  001436          ADD     #6,DIEINS
015036  000177  164044                  JMP     @$LPADR         ;START TEST OVER AGAIN.
015042  000000                  TRTO:   .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTERRUPTED TO.
015044  000000                  TRFRO:  .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTR. FROM.
```

```
 1037                                          .SBTTL  ASCII MESSAGES
 1038  015046     015     012                 PRIME0: .BYTE   15,12
 1039  015050  020114  020075  047514                 .ASCII  \L = LOGIC TEST WITH OR WITHOUT TEST MODULE CONNECTED\
       015056  044507  020103  042524
       015064  052123  053440  052111
       015072  020110  051117  053440
       015100  052111  047510  052125
       015106  052040  051505  020124
       015114  047515  052504  042514
       015122  041440  047117  042516
       015130  052103  042105
 1040  015134     015     012                         .BYTE   15,12
 1041  015136  020127  020075  051127                 .ASCII  \W = WRAPAROUND LOGIC TEST WITH MNCDO CONNECTED TO MNCDI\
       015144  050101  051101  052517
       015152  042116  046040  043517
       015160  041511  052040  051505
       015166  020124  044527  044124
       015174  046440  041516  047504
       015202  041440  047117  042516
       015210  052103  042105  052040
       015216  020117  047115  042103
       015224     111
 1042  015225     015     012                         .BYTE   15,12
 1043  015227     124  036440  052040                 .ASCII  \T = TYPEOUT LOOP FOR MNCDI TEST MODULE SWITCHES\
       015234  050131  047505  052125
       015242  046040  047517  020120
       015250  047506  020122  047115
       015256  042103  020111  042524
       015264  052123  046440  042117
       015272  046125  020105  053523
       015300  052111  044103  051505
 1044  015306     015     012                         .BYTE   15,12
 1045  015310  020107  020075  042507                 .ASCII  \G = GET NEW SWITCH REGISTER VALUE\
       015316  020124  042516  020127
       015324  053523  052111  044103
       015332  051040  043505  051511
       015340  042524  020122  040526
       015346  052514     105
 1046  015351     015     012                         .BYTE   15,12
 1047  015353     102  036440  024040                 .ASCII  \B = (DIGITAL IN) BASE OR VECTOR ADDRESS CHANGES\
       015360  044504  044507  040524
       015366  020114  047111  020051
       015374  040502  042523  047440
       015402  020122  042526  052103
       015410  051117  040440  042104
       015416  042522  051523  041440
       015424  040510  043516  051505
 1048  015432     015     012                         .BYTE   15,12
 1049  015434  020117  020075  042050                 .ASCII  \O = (DIGITAL OUT) BASE ADDRESS CHANGE\
       015442  043511  052111  046101
       015450  047440  052125  020051
       015456  040502  042523  040440
       015464  042104  042522  051523
       015472  041440  040510  043516
```

```
                015500   105
1050            015501   015   012                    .BYTE   15,12
1051            015503   110   036440  044040         .ASCIZ  \H = HELP THE OPERATOR AND RETYPE THIS LIST    \
                015510  046105  020120  044124
                015516  020105  050117  051105
                015524  052101  051117  040440
                015532  042116  051040  052105
                015540  050131  020105  044124
                015546  051511  046040  051511
                015554  020124  020040  000040
1052            015562   015   012            DOT:    .BYTE   15,12
1053            015564  054524  042520  052040         .ASCIZ  /TYPE THE ''TEST CHARACTER'' THEN DEPRESS 'RETURN KEY''  /
                015572  042510  021040  042524
                015600  052123  041440  040510
                015606  040522  052103  051105
                015614  020042  044124  047105
                015622  042040  050105  042522
                015630  051523  021040  042522
                015636  052524  047122  045440
                015644  054505  020042  000040
1054
1055            015652  047115  042103  020117  EM1:    .ASCIZ  \MNCDO (DIGITAL OUT)    DOES NOT EXIST  <BUS ERROR>  CHECK ADDRESS SWITC
                015660  042050  043511  052111
                015666  046101  047440  052125
                015674  004451  047504  051505
                015702  047040  052117  042440
                015710  044530  052123  020040
                015716  041074  051525  042440
                015724  051122  051117  020076
                015732  041440  042510  045503
                015740  040440  042104  042522
                015746  051523  051440  044527
                015754  041524  042510  000123
1056            015762  047115  042103  020117  EM2:    .ASCIZ  \MNCDO (DIGITAL OUT)    DATA REGISTER ERROR\
                015770  042050  043511  052111
                015776  046101  047440  052125
                016004  004451  040504  040524
                016012  051040  043505  051511
                016020  042524  020122  051105
                016026  047522  000122
1057            016032  047115  042103  020117  EM3:    .ASCIZ  \MNCDO (DIGITAL OUT)    STATUS REGISTER ERROR\
                016040  042050  043511  052111
                016046  046101  047440  052125
                016054  004451  052123  052101
                016062  051525  051040  043505
                016070  051511  042524  020122
                016076  051105  047522  000122
1058            016104  047115  042103  020117  EM4:    .ASCIZ  \MNCDO (DIGITAL OUT)    INTERRUPT ERROR\
                016112  042050  043511  052111
                016120  046101  047440  052125
                016126  004451  047111  042524
                016134  051122  050125  020124
                016142  051105  047522  000122
1059            016150  047115  042103  020111  EM5:    .ASCIZ  \MNCDI (DIGITAL IN)     DOES NOT EXIST  <BUS ERROR> CHECK ADDRESS SWITCH
```

```
                016156  042050  043511  052111
                016164  046101  044440  024516
                016172  042011  042517  020123
                016200  047516  020124  054105
                016206  051511  004524  041074
                016214  051525  042440  051122
                016222  051117  020076  044103
                016230  041505  020113  042101
                016236  051104  051505  020123
                016244  053523  052111  044103
                016252  051505     000
1060            016255     115  041516  044504  EM6:    .ASCIZ  \MNCDI (DIGITAL IN)     STIMULUS REGISTER ERROR\
                016262  024040  044504  044507
                016270  040524  020114  047111
                016276  004451  052123  046511
                016304  046125  051525  051040
                016312  043505  051511  042524
                016320  020122  051105  047522
                016326  000122
1061            016330  047115  042103  020111  EM7:    .ASCIZ  \MNCDI (DIGITAL IN)     STATUS REGISTER ERROR\
                016336  042050  043511  052111
                016344  046101  044440  024516
                016352  051411  040524  052524
                016360  020123  042522  044507
                016366  052123  051105  042440
                016374  051122  051117     000
1062            016401     115  041516  044504  EM10:   .ASCIZ  \MNCDI (DIGITAL IN)     DATA REGISTER ERROR\
                016406  024040  044504  044507
                016414  040524  020114  047111
                016422  004451  040504  040524
                016430  051040  043505  051511
                016436  042524  020122  051105
                016444  047522  000122
1063            016450  047115  042103  020111  EM11:   .ASCIZ  \MNCDI (DIGITAL IN)     INTERRUPT ERROR\
                016456  042050  043511  052111
                016464  046101  044440  024516
                016472  044411  052116  051105
                016500  052522  052120  042440
                016506  051122  051117     000
1064            016513     115  041516  044504  EM12:   .ASCIZ  /MNCDI MNCDO    DIGITAL INPUT-OUTPUT WRAPAROUND STATUS ERROR/
                016520  046440  041516  047504
                016526  042011  043511  052111
                016534  046101  044440  050116
                016542  052125  047455  052125
                016550  052520  020124  051127
                016556  050101  051101  052517
                016564  042116  051440  040524
                016572  052524  020123  051105
                016600  047522  000122
1065            016604  047115  042103  020111  EM13:   .ASCIZ  /MNCDI MNCDO    DIGITAL INPUT-OUTPUT WRAPAROUND DATA ERROR/
                016612  047115  042103  004517
                016620  044504  044507  040524
                016626  020114  047111  052520
                016634  026524  052517  050124
```

```
                016642  052125  053440  040522
                016650  040520  047522  047125
                016656  020104  040504  040524
                016664  042440  051122  051117
                016672     000
1066            016673     115  041516  044504  EM14:   .ASCIZ  /MNCDI  INCORRECT I.D. VALUE FOR MNCDI/
                016700  044411  041516  051117
                016706  042522  052103  044440
                016714  042056  020056  040526
                016722  052514  020105  047506
                016730  020122  047115  042103
                016736  000111
1067            016740  047115  042103  020111  EM15:   .ASCIZ  /MNCDI (DIGITAL IN)     ILLEGAL INTR. OR TRAP/
                016746  042050  043511  052111
                016754  046101  044440  024516
                016762  044411  046114  043505
                016770  046101  044440  052116
                016776  027122  047440  020122
                017004  051124  050101     000
1068            017011     115  041516  044504  EM16:   .ASCIZ  /MNCDI (DIGITAL IN)     EXISTING MNCDI FAILED TO RESPOND/
                017016  024040  044504  044507
                017024  040524  020114  047111
                017032  004451  054105  051511
                017040  044524  043516  046440
                017046  041516  044504  043040
                017054  044501  042514  020104
                017062  047524  051040  051505
                017070  047520  042116     000
1069            017075     200  051120  043517  FOUND1: .ASCIZ  <200>/PROGRAM DETECTED/
                017102  040522  020115  042504
                017110  042524  052103  042105
                017116     000
1070            017117     040  047115  042103  FOUND2: .ASCIZ  \ MNCDI (DIGITAL IN)'S   \
                017124  020111  042050  043511
                017132  052111  046101  044440
                017140  024516  051447  020040
                017146  000040
1071            017150  047125  052111  042411  DH1:    .ASCIZ  /UNIT   ERRPC   OUTCSR  OUTDAT/
                017156  051122  041520  047411
                017164  052125  051503  004522
                017172  052517  042124  052101
                017200     000
1072            017201     125  044516  004524  DH5:    .ASCIZ  /UNIT   ERRPC   IN CSR  IN DIR  IN SBR/
                017206  051105  050122  004503
                017214  047111  041440  051123
                017222  044411  020116  044504
                017230  004522  047111  051440
                017236  051102     000
1073            017241     125  044516  004524  DH6:    .ASCIZ  /UNIT   ERRPC   IN SBR  GOOD     BAD/
                017246  051105  050122  004503
                017254  047111  051440  051102
                017262  043411  047517  004504
                017270  041040  042101     000
1074            017275     125  044516  004524  DH7:    .ASCIZ  /UNIT   ERRPC   IN CSR  GOOD     BAD/
```

```
                017302  051105  050122  004503
                017310  047111  041440  051123
                017316  043411  047517  004504
                017324  041040  042101     000
1075            017331     125  044516  004524  DH10:   .ASCIZ  /UNIT   ERRPC   IN DIR  GOOD    BAD/
                017336  051105  050122  004503
                017344  047111  042040  051111
                017352  043411  047517  004504
                017360  040502  000104
1076            017364  047125  052111  042411  DH11:   .ASCIZ  /UNIT   ERRPC   IN CSR/
                017372  051122  041520  044411
                017400  020116  051503  000122
1077            017406  047125  052111  042411  DH12:   .ASCIZ  /UNIT   ERRPC   OUTCSR  IN CSR  GOOD     BAD/
                017414  051122  041520  047411
                017422  052125  051503  004522
                017430  047111  041440  051123
                017436  043411  047517  004504
                017444  041040  042101     000
1078            017451     125  044516  004524  DH13:   .ASCIZ  /UNIT   ERROC   OUTDOR  IN DIR  GOOD     BAD/
                017456  051105  047522  004503
                017464  052517  042124  051117
                017472  044411  020116  044504
                017500  004522  047507  042117
                017506  020011  040502  000104
1079            017514  047125  052111  042411  DH15:   .ASCIZ  /UNIT   ERRPC     TO    FROM ADRS./
                017522  051122  041520  020011
                017530  052040  004517  051106
                017536  046517  040440  051104
                017544  027123     000
1080            017547     125  044516  004524  DH16:   .ASCIZ  /UNIT   ERRPC    WERE   ARE/
                017554  051105  050122  020103
                017562  020040  053440  051105
                017570  020105  020040  051101
                017576  000105
1081            017600  035440  047524  040524  ERRTOT: .ASCIZ  / ;TOTAL ERROR COUNT = /
                017606  020114  051105  047522
                017614  020122  047503  047125
                017622  020124  020075     000
1082            017627     040  041073  042101  MESGD:  .ASCIZ  / ;BAD UNITS /
                017634  052440  044516  051524
                017642  000040
1083            017644  046600  041516  044504  VTMSG:  .ASCIZ  <200>/MNCDI (DIGITAL IN) UNIT #/
                017652  024040  044504  044507
                017660  040524  020114  047111
                017666  020051  047125  052111
                017674  021440     000
1084            017677     200  054105  042520  VTMSG3: .ASCIZ  <200>/EXPECTED INTERRUPT AT /
                017704  052103  042105  044440
                017712  052116  051105  052522
                017720  052120  040440  020124
                017726     000
1085            017727     040  042522  042503  VTMSG1: .ASCIZ  / RECEIVED INTERRUPT AT /
                017734  053111  042105  044440
                017742  052116  051105  052522
```

```
                017750  052120  040440  020124
                017756     000
1086            017757     200  046120  040505  VTMSG2: .ASCII  <200>/PLEASE CHECK VECTOR SWITCHES/<200>
                017764  042523  041440  042510
                017772  045503  053040  041505
                020000  047524  020122  053523
                020006  052111  044103  051505
                020014     200
1087            020015     011  042522  052123          .ASCIZ  /       RESTARTING LOGIC TEST/
                020022  051101  044524  043516
                020030  046040  043517  041511
                020036  052040  051505  000124
1088            020044     015     012                  SETUP0: .BYTE   15,12
1089            020046  047105  052523  042522                  .ASCII  \ENSURE MNCDI (DIGITAL IN) DATA SWITCH IS IN THE ''-'' POSITION\
                020054  046440  041516  044504
                020062  024040  044504  044507
                020070  040524  020114  047111
                020076  020051  040504  040524
                020104  051440  044527  041524
                020112  020110  051511  044440
                020120  020116  044124  020105
                020126  026442  020042  047520
                020134  044523  044524  047117
1090            020142     015     012     000          .BYTE   15,12,0
1091            020145     103  047117  042516  SETUP1: .ASCII  \CONNECT MNCDI (DIGITAL IN) TO THE TESTER\
                020152  052103  046440  041516
                020160  044504  024040  044504
                020166  044507  040524  020114
                020174  047111  020051  047524
                020202  052040  042510  052040
                020210  051505  042524     122
1092            020215     015     012     000          .BYTE   15,12,0
1093            020220     015     012                  SETUP2: .BYTE   15,12
1094            020222  047115  042103  020111                  .ASCII  \MNCDI (DIGITAL IN) MUST BE CONNECTED TO MNCDO (DIGITAL OUT)\
                020230  042050  043511  052111
                020236  046101  044440  024516
                020244  046440  051525  020124
                020252  042502  041440  047117
                020260  042516  052103  042105
                020266  052040  020117  047115
                020274  042103  020117  042050
                020302  043511  052111  046101
                020310  047440  052125     051
1095            020315     015     012     000          .BYTE   15,12,0
1096            020320  005015  047115  042103  ADROUT: .ASCIZ  <15><12>/MNCDO (DIGITAL OUT) BUS ADDRESS </
                020326  020117  042050  043511
                020334  052111  046101  047440
                020342  052125  020051  052502
                020350  020123  042101  051104
                020356  051505  020123  000074
1097            020364  005015  047115  042103  ADRIN:  .ASCIZ  <15><12>/MNCDI (DIGITAL IN) BUS ADDRESS </
                020372  020111  042050  043511
                020400  052111  046101  044440
                020406  024516  041040  051525
```

```
        020414  040440  042104  042522
        020422  051523  036040     000
1098    020427     015  046412  041516  VECIN:  .ASCIZ  <15><12>/MNCDI (DIGITAL IN) VECTOR ADDRESS </
        020434  044504  024040  044504
        020442  044507  040524  020114
        020450  047111  020051  042526
        020456  052103  051117  040440
        020464  042104  042522  051523
        020472  036040     000
1099    020475     076  037440  000040  ENDOUT: .ASCIZ  /> ? /
1100    020502  026440  000040          ADASH:  .ASCIZ  / - /
1101    020506     015     012     000  TCRLF:  .BYTE   15,12,0
1102            020512                       .EVEN
1103    020512  001532  001116  001404  DT1:    UNITBD,$ERRPC,OCSR,DOR,0
        020520  001410  000000
1104    020524  001532  001116  001414  DT5:    UNITBD,$ERRPC,ICSR,DIR,SBR,0
        020532  001420  001424  000000
1105    020540  001532  001116  001424  DT6:    UNITBD,$ERRPC,SBR,$GDDAT,$BDDAT,0
        020546  001124  001126  000000
1106    020554  001532  001116  001414  DT7:    UNITBD,$ERRPC,ICSR,$GDDAT,$BDDAT,0
        020562  001124  001126  000000
1107    020570  001532  001116  001420  DT10:   UNITBD,$ERRPC,DIR,$GDDAT,$BDDAT,0
        020576  001124  001126  000000
1108    020604  001532  001116  001414  DT11:   UNITBD,$ERRPC,ICSR,0
        020612  000000
1109    020614  001532  001116  001404  DT12:   UNITBD,$ERRPC,OCSR,ICSR,$GDDAT,$BDDAT,0
        020622  001414  001124  001126
        020630  000000
1110    020632  001532  001116  001410  DT13:   UNITBD,$ERRPC,DOR,DIR,$GDDAT,$BDDAT,0
        020640  001420  001124  001126
        020646  000000
1111    020650  001532  001116  015042  DT15:   UNITBD,$ERRPC,TRTO,TRFRO,0
        020656  015044  000000
1112    020662  001532  001116  001442  DT16:   UNITBD,$ERRPC,TEMP,$UNIT,0
        020670  001202  000000
1113    020674     000     000     000  DF0:    .BYTE   0,0,0,0,0,0,0,0,0,0,0
        020677     000     000     000
        020702     000     000     000
        020705     000     000
1114
1115            000001                       .END
```

```
ABASE = 171160              17#     61
ACDW1 = 171260              19#     61
ACDW2 = 000000              61
ACPUOP= 000000              61
ADASH   020502             885    1100#
ADDW0 = 000000              61
ADDW1 = 000000              61
ADDW10= 000000              61
ADDW11= 000000              61
ADDW12= 000000              61
ADDW13= 000000              61
ADDW14= 000000              61
ADDW15= 000000              61
ADDW2 = 000000              61
ADDW3 = 000000              61
ADDW4 = 000000              61
ADDW5 = 000000              61
ADDW6 = 000000              61
ADDW7 = 000000              61
ADDW8 = 000000              61
ADDW9 = 000000              61
ADEVCT= 000000              61
ADEVM = 000000              61
ADRIN   020364             379    1097#
ADROUT  020320             366    1096#
AENV  = 000000              61
AENVM = 000000              61
AFATAL= 000000              61
AMADR1= 000000              61
AMADR2= 000000              61
AMADR3= 000000              61
AMADR4= 000000              61
AMAMS1= 000000              61
AMAMS2= 000000              61
AMAMS3= 000000              61
AMAMS4= 000000              61
AMSGAD= 000000              61
AMSGLG= 000000              61
AMSGTY= 000000              61
AMTYP1= 000000              61
AMTYP2= 000000              61
AMTYP3= 000000              61
AMTYP4= 000000              61
APASS = 000000              61
APRIOR= 000000              61
APTCSU= 000040             921#    932
APTENV= 000001             916     921#    932
APTSIZ= 000200             219     921#
APTSPO= 000100             921#    932
ASWREG= 000000              61
ATESTN= 000000              61
AUNIT = 000000              61
AUSWR = 000000              61
AVECT1= 000120              18#     61      188     189     190     191     192     193     194     195
```

```
AVECT2= 000000   61
BADUNT  001530   205#  324*  868  916*
BASEXC  003252   253   378#
BASEXD  003210   266   365#
BEGIN   001542    48   214#  924
BITDAT= 010000   208#  555   560  569  579
BITEXT= 004000   209#  484   492  501  513  523  524  533  534  544  545  594  606
                 608   624   646  647  723
BIT0  = 000001    20#  323   442  444  578  734  751  771
BIT00 = 000001    20#
BIT01 = 000002    20#
BIT02 = 000004    20#
BIT03 = 000010    20#
BIT04 = 000020    20#
BIT05 = 000040    20#
BIT06 = 000100    20#
BIT07 = 000200    20#
BIT08 = 000400    20#  899
BIT09 = 001000    20#  899  916
BIT1  = 000002    20#  448  465  521  522  531  532  543  585  605  623  645  695
                  698
BIT10 = 002000    20#
BIT11 = 004000    20#  209  899
BIT12 = 010000    20#  208  458  605  809
BIT13 = 020000    20#  916
BIT14 = 040000    20#  460  645  899
BIT15 = 100000    20#  315  521  524  534  646  647  721  782  794  811  812  821
BIT2  = 000004    20#  449  465  512  514  599  707  710  775  782  795  810  811
BIT3  = 000010    20#  450
BIT4  = 000020    20#  452  465  560  579  594  605  750
BIT5  = 000040    20#  453  560  579  584  594  605  607  750
BIT6  = 000100    20#  242  445  454  461  503  547  598  623  859
BIT7  = 000200    20#  483  484  521  523  524  531  533  534  606  608  624  646
                  647  686  698  718  723  779  782  794  811  812  821
BIT8  = 000400    20#  456  810  811
BIT9  = 001000    20#  457
BPTVEC= 000014    20#
CKSWR = 104410   899   916  936#
CR    = 000015    20#  932
CRLF  = 000200    20#  227  932
CTRCHA  003422   403*  404*  405  409  413#
CTRLCG  003346   401#  871   880  899  916
DDISP = 177570    20#   61   219
DF0     020674    64    66    68   70   72   74   76   78   80   82   84  1113#
DH1     017150    64  1071#
DH10    017331    78  1075#
DH11    017364    80  1076#
DH12    017406    66  1077#
DH13    017451    68  1078#
DH15    017514    82  1079#
DH16    017547    84  1080#
DH5     017201    72  1072#
DH6     017241    74  1073#
DH7     017275    70    76  1074#
```

```
DIDATA  010262          269   875#
DIDINS  001432          172#  619*  635   636*  847*  848*  1020  1023*  1026*  1029*
DIDINV  001430          171#  339   618*  635*  846*  847   849   851    1016   1020*  1024*  1025*  1026
                       1027  1028
DIEINS  001436          175#  344   641*  658   659*  851*  852*  1021   1022*  1028*  1031*
DIEINV  001434          174#  640*  658*  849*  850*  1021* 1027* 1030*
DIR     001420          166#  433   511*  556   562   571   582   583*   586    595    600    610    681*
                        693*  705*  731*  738   748*  754   776*  793*   808*   820*   879*   883    886
                       1104  1107  1110
DIR1    001422          167#
DISPLA  001142           61#  219*  899*  916*
DISPRE  000174           44#  219
DOR     001410          160#  685*  697*  709*  717*  737*  753*  770*   773*   780*   792*   797*   807*
                        814*  822*  1103  1110
DOR1    001412          161#  837
DOT     015562          247  1052#
DSWR  = 177570           20#   61   219
DT1     020512           64  1103#
DT10    020570           78  1107#
DT11    020604           80  1108#
DT12    020614           66  1109#
DT13    020632           68  1110#
DT15    020650           82  1111#
DT16    020662           84  1112#
DT5     020524           72  1104#
DT6     020540           74  1105#
DT7     020554           70   76   1106#
DWARF   001440          177#  214*  217*  234   262*  272*  305   421    678    719
EMTVEC= 000030           20#  219*
EM1     015652           64  1055#
EM10    016401           78  1062#
EM11    016450           80  1063#
EM12    016513           66  1064#
EM13    016604           68  1065#
EM14    016673           70  1066#
EM15    016740           82  1067#
EM16    017011           84  1068#
EM2     015762         1056#
EM3     016032         1057#
EM4     016104         1058#
EM5     016150           72  1059#
EM6     016255           74  1060#
EM7     016330           76  1061#
ENDOUT  020475          369   382   391   1099#
ERRTOT  017600          862  1081#
ERRVEC= 000004           20#  219*  283*  301*  302*  431*  437*  438*   440*   441*   899*
EVER    001526          204#  245*  303   314*  315*  317   319   833
EXTMSG  010176          857   859#
FIXADR  003010          246   322   330#  374   396   876
FOUND1  017075          307  1069#
FOUND2  017117          310  1070#
GNS   = ****** U        227   936
GTSWR = 104407          227   256   411   936#
HT    = 000011           20#  932
```

```
ICSR    001414  163# 334  432  448* 449* 450* 452* 453* 454* 456* 457* 458* 460*
                461* 463* 466  472* 473* 475  482* 484* 485  492* 493* 494  501*
                504  512* 513* 515  522* 523* 524* 525  532* 533* 534* 536  543*
                544* 545* 548  555* 560* 569* 579* 584* 585* 594* 599* 605* 606*
                607* 608* 623* 624* 629* 634* 645* 646* 647* 652* 657* 683* 687
                695* 699  707* 711  723* 724* 733* 750* 775* 779* 783  794* 798
                810* 812* 815  821* 824  878* 881  1104  1106  1108  1109
ICSR1   001416  164# 464* 535*
IOTRD   014570   42  301  437  440  958#
IOTVEC= 000020   20# 219*
LF    = 000012   20# 932
LOGIC   002520  230  263  280# 872
LOGIC0  002514  273  279#
MASKNM  001460  185# 281* 323* 842* 865  909  916
MESGD   017627  867  1082#
MTEST   002274  235  240# 259
MTEST1  002300  232  241# 257  275  375  397  408  898
OCSR    001404  157# 330  684* 696* 708* 722* 725  732* 749* 835  1103 1109
OCSR1   001406  158#
PC    =%000007   20# 226* 246* 322* 362* 374* 396* 412* 843* 857* 871* 876* 880*
                898* 899* 915* 916* 917* 921* 924  932*
PIRQ  = 177772   20#
PIRQVE= 000240   20#
PRIME0  015046  240  1038#
PR0   = 000000   20#
PR1   = 000040   20#
PR2   = 000100   20#
PR3   = 000140   20#
PR4   = 000200   20#
PR5   = 000240   20#
PR6   = 000300   20#
PR7   = 000340   20#
PS    = 177776   20#
PSW   = 177776   20#
PWRMSG  013776  924  925#
PWRVEC= 000024   20# 219* 924*
RBEG0   001566  213  216  219#
RDCHR = 104411  898  936#
RDLIN = 104412  248  928  936#
RDOCT = 104413  370  383  392  936#
REMAIN  007700  439  680  830#
RESTRT  001534   49  212#
RESVEC= 000010   20#
RUNIT   002512  249* 250* 251  254  258  260  264  267  270  276#
R0    =%000000   20# 330* 332* 334  337* 339  342* 344  346* 351* 354* 357* 358*
                360  582* 844* 845* 846  857* 917* 920* 921* 924* 928* 932* 936*
R1    =%000001   20# 331* 332  333* 336* 337  338* 341* 342  343  347* 351  355*
                357  359  835* 836* 837  839* 840  917* 920* 921* 924* 928* 933*
R2    =%000002   20# 356* 358  920* 924* 928*
R3    =%000003   20# 898* 919* 920* 924*
R4    =%000004   20# 919* 924*
R5    =%000005   20# 919* 920* 924*
R6    =%000006   20# 219*
R7    =%000007   20#
```

```
SBR      001424          168#   434    442*   444*   445*   446*   510*   682*   694*   706*   774*   778*   813*
                        1104   1105
SBR1     001426          169#   446*   840
SETUP0   020044          233   1088#
SETUP1   020145          236   1091#
SETUP2   020220          279   1093#
SP     =%000006           20#   219*   249    294    308*   325*   326*   367*   371    373    380*   384    386
                         389*   393    395    407    620*   621*   633    642*   643*   656    660*   661*   857*
                         863*   868*   875*   883*   886*   898*   899*   916*   917*   919*   920*   921*   924*
                         928*   932*   933*   936*   970*  1007   1011*  1016*  1018*
STACK  = 001100           20#   219    875
STKLMT= 177774            20#
SWR      001140           61#   219*   227    289    898*   899    916    924*
SWREG    000176           45#   219    227    898
SW0    = 000001           20#
SW00   = 000001           20#
SW01   = 000002           20#
SW02   = 000004           20#
SW03   = 000010           20#
SW04   = 000020           20#
SW05   = 000040           20#
SW06   = 000100           20#
SW07   = 000200           20#
SW08   = 000400           20#
SW09   = 001000           20#
SW1    = 000002           20#
SW10   = 002000           20#
SW11   = 004000           20#
SW12   = 010000           20#   289
SW13   = 020000           20#
SW14   = 040000           20#
SW15   = 100000           20#
SW2    = 000004           20#
SW3    = 000010           20#
SW4    = 000020           20#
SW5    = 000040           20#
SW6    = 000100           20#
SW7    = 000200           20#
SW8    = 000400           20#
SW9    = 001000           20#
TBITVE= 000014            20#
TCRLF    020506          888   1101#
TEMP     001442          178#   319*   444*   578*   580    583    590*   595*   734*   736    743*   751*   753
                         755    761*   771*   778    780    790    803*   809*   813    814    822    828*  1112
TEMP1    001444          179#   790*   791*   797
TEMP2    001446          180#   212*   215*   218*   231
TESTER   001554           50    217#
TKVEC  = 000060           20#   898*
TPVEC  = 000064           20#
TRAPVE= 000034            20#   219*
TRFRO    015044          969*  1034#  1111
TRTO     015042          958*   959*   961   1018   1024   1033#  1111
TRTVEC= 000014            20#
TSTRO    001450          181#   237
```

```
TSTR2   001452          182#    423*
TSTR4   001454          183#    424
TSTR6   001456          184#
TST1    003424          326     420#    854
TST10   004166          448     449#
TST11   004252          449     450#
TST12   004336          450     452#
TST13   004422          452     453#
TST14   004506          453     454#
TST15   004572          454     456#
TST16   004656          456     457#
TST17   004742          457     458#
TST2    003476          422     428     430#
TST20   005026          458     460#
TST21   005112          460     461#
TST22   005156          461     462#
TST23   005220          468     471#
TST24   005262          477     481#
TST25   005324          487     490#
TST26   005364          496     499#
TST27   005436          506     509#
TST3    003562          442#
TST30   005514          517     520#
TST31   005566          527     530#
TST32   005644          538     541#
TST33   005732          550     553#
TST34   006060          573     576#
TST35   006172          593#
TST36   006252          601     604#
TST37   006332          612     616#
TST4    003634          444#
TST40   006444          639#
TST41   006566          677#
TST42   006660          689     692#
TST43   006742          701     704#
TST44   007024          713     716#
TST45   007114          727     730#
TST46   007212          747#
TST47   007316          765#
TST5    003720          445#
TST50   007530          806#
TST51   007700          831#
TST6    003764          445     446#
TST7    004102          448#
TYPBN = 104406          869     887     936#
TYPDS = 104405          309     857     864     917     936#    1012
TYPE  = 104401          227     233     236     240     247     274     279     307     310     365     369     378     382
                        387     391     857     862     867     870     877     885     888     898     916     917     919
                        920     924     932     933     936#    1010    1015    1017    1019
TYPOC = 104402          368     381     390     884     898     917     936#
TYPON = 104404          936#
TYPOS = 104403          936#    1016    1018
UNITBD  001532          206#    844     908*    912*    1103    1104    1105    1106    1107    1108    1109    1110    1111
                        1112
```

```
VADDR   001464          187#    285     839
VADDRO  001462          186#    836
VECIN   020427          388    1098#
VECLST  001466          188#    346     846
VECOFF  001506          196#    347
VTMSG   017644         1010    1083#
VTMSG1  017727         1017    1085#
VTMSG2  017757         1019    1086#
VTMSG3  017677         1015    1084#
WAITO   010350          877     891#
WHICHU  012272          843     908#    916
$APTHD  001000           60#
$ASTAT= ****** U        921
$ATYC   013402          921#
$ATY1   013356          921#
$ATY3   013364          921#    932
$ATY4   013374          916     921#
$AUTOB  001134           61#    227*    228     898
$BASE   001244           61#    280     336     380     386*
$BDADR  001122           61#
$BDDAT  001126           61#    280*    284     285*    424*    425*    427     442*    444*    445*    446*    448*    449*
                        450*    452*    453*    454*    456*    457*    458*    460*    461*    466*    467     475*    476
                        485*    486     494*    495     504*    505     515*    516     525*    526     536*    537     548*
                        549     556*    557     562*    565     571*    572     586*    587     600*    610*    611     687*
                        688     699*    700     711*    712     725*    726     738*    739     754*    757     783*    784
                        798*    799     815*    816     824*    825    1105    1106    1107    1109    1110
$BELL   011740          898#
$BIN    014502          933#*
$CDW1   001250           61#    237*    331     367     373*
$CDW2   001252           61#
$CHARC  014424          932#*
$CKSWR  011012          898#    936
$CMTAG  001100           61#    219
$CM3  = 000000           61#
$CNTLC  011744          898#
$CNTLG  011756          898#
$CNTLU  011751          898#
$CPUOP  001216           61#
$CRLF   001165           61#    898     916     917     932
$DBLK   013346          920#
$DEVCT  001200           61#
$DEVM   001246           61#
$DOAGN  010152          857#
$DTBL   013336          920#
$ENDAD  010142           58     227     857#    916
$ENDCT  010110          219     857#
$ENDMG  010161          857#
$ENULL  010156          857#    870
$ENV    001210           61#    227     287     916     921     932
$ENVM   001211           61#    219     921     932
$EOP    010054          300     313     857#
$EOPCT  010102          219*    857#
$ERFLG  001103           61#    899*    916*
$ERMAX  001115           61#    219*    899*
```

```
$ERROR  012326          219     916#
$ERRPC  001116           61#    916*    917     1103    1104    1105    1106    1107    1108    1109    1110    1111    1112
$ERRTB  001254           61#    917
$ERRTY  012530          916     917#
$ERTTL  001112           61#    244*    860     863     916*
$ESCAP  001162           61#    219*    899*    916
$ETABL  001210           61#
$ETEND  001254           60      61#
$FATAL  001172           61#    921*
$FFLG   013622          921#*
$FILLC  001156           61#    932
$FILLS  001155           61#    932
$GDADR  001120           61#
$GDDAT  001124           61#    426*    427     442*    444*    445*    446*    448*    449*    450*    452*    453*    454*
                        456*    457*    458*    460*    461*    465*    467     474*    476     483*    486     491*    495
                        500*    505     514*    516     521*    526     531*    537     542*    549     554*    557     561*
                        565     570*    572     580*    581*    587     596*    609*    611     686*    688     698*    700
                        710*    712     718*    721*    726     736*    737     739     755*    756*    757     782*    784
                        795*    799     811*    816     825     1105    1106    1107    1109    1110
$GET42  010132          857#
$GTSWR  011102          898#    936
$HD   = 000000           15
$HIBTS  001000           60#
$HIOCT  014144          928#*
$ICNT   001104           61#    899*
$ILLUP  013770          924#
$INTAG  001135           61#    898*
$ITEMB  001114           61#    916*    917
$LF     001166           61#    898     916     932
$LFLG   013621          921#*
$LPADR  001106           61#    219*    442*    444*    446*    617*    899*    1032
$LPERR  001110           61#    219*    577*    899*    916
$MADR1  001222           61#
$MADR2  001226           61#
$MADR3  001232           61#
$MADR4  001236           61#
$MAIL   001170           60      61#     219     227     899     916     932
$MAMS1  001220           61#
$MAMS2  001224           61#
$MAMS3  001230           61#
$MAMS4  001234           61#
$MBADR  001002           60#
$MFLG   013620          921#*
$MNEW   011774          898#
$MSGAD  001204           61#    921*
$MSGLG  001206           61#    921*
$MSGTY  001170           61#    921*
$MSWR   011763          898#
$MTYP1  001221           61#
$MTYP2  001225           61#
$MTYP3  001231           61#
$MTYP4  001235           61#
$MXCNT  012270          899#
$NULL   001154           61#    932
```

```
$NWTST= 000001          420#  430#  442#  444#  445#  446#  448#  449#  450#  452#  453#  454#  456#
                        457#  458#  460#  461#  462#  471#  481#  490#  499#  509#  520#  530#  541#
                        553#  576#  593#  604#  616#  639#  677#  692#  704#  716#  730#  747#  765#
                        806#  831#
$OCNT   013126          919#*
$OMODE  013130          919#*
$OVER   012254          899#
$PASS   001176           61#  219*  243*  857*  899  1008
$PASTM  001006           60#
$PWRAD  013764          924#
$PWRDN  013624          219  924#
$PWRMG  013760          924#
$PWRUP  013676          924#
$QUES   001164           61#  274  898  916  932
$RDCHR  011354          898#  936
$RDDEC= ****** U        936
$RDLIN  011444          898#  936
$RDOCT  014044          928#  936
$RDSZ = 000040          898#
$RTNAD  010154          857#
$R2A  = ****** U        936
$SAVRE= ****** U        936
$SAVR6  013774          924#*
$SCOPE  012006          219  899#
$SETUP= 000137          211#  219  227  857  898  899  916
$STUP = 177777          211#
$SVLAD  012220          899#
$SVPC = 000106           58#
$SWR  = 165400            8#   15   22   61  219  420  430  442  444  445  446  448  449
                        450  452  453  454  456  457  458  460  461  462  471  481  490
                        499  509  520  530  541  553  576  593  604  616  639  677  692
                        704  716  730  747  765  806  831  857  899  916  924
$SWREG  001212           61#  219
$SWRMK= 000000           22   899
$TESTN  001174           61#  899*
$TIMES  001160           61#  219*  445*  461*  499*  541*  593*  730*  747*  765*  831*  857*  899*
$TKB    001146           61#  403  898
$TKCNT  010456          898#*
$TKINT  010524          226  898#
$TKQEN= 010524          898#
$TKQIN  010460          898#*
$TKQOU  010462          898#*
$TKQSR  010464          898#
$TKS    001144           61#  242*  401  445*  461*  503*  547*  598*  859*  898*
$TKSRV  010574          898#
$TN   = 000052            9#   15  420#  422  428  430#  442#  444#  445#  446#  448#  449#  450#
                        452#  453#  454#  456#  457#  458#  460#  461#  462#  468  471#  477  481#
                        487  490#  496  499#  506  509#  517  520#  527  530#  538  541#  550
                        553#  573  576#  593#  601  604#  612  616#  639#  677#  689  692#  701
                        704#  713  716#  727  730#  747#  765#  806#  831#
$TPB    001152           61#  932*
$TPFLG  001157           61#  932
$TPS    001150           61#  932
$TRAP   014504          219  936#
```

```
$TRAP2  014526            936#
$TRP   = 000014           936#
$TRPAD  014540            936#
$TSTM   001004             60#
$TSTNM  001102             61#   853*  857*  899*  916   972
$TTYIN  011700            898#
$TYPBN  014430            933#   936
$TYPDS  013132            920#   936
$TYPE   014146            222*   223*  224*  225*  921   932#   936
$TYPEC  014360            898    932#
$TYPEX  014426            932#
$TYPOC  012730            919#   936
$TYPON  012744            919#   936
$TYPOS  012704            919#   936
$UNIT   001202             61#   282*  286*  291   295   308   311   314   317   321*  832*  833   1011
                         1112
$UNITM  001010             60#
$USWR   001214             61#
$VECT1  001240             61#   341   351   389   395*
$VECT2  001242             61#
$XTSTR  012024            899#
$$GET4= 000000            857#
$OFILL  013127            919#*
$40CAT= ****** U          899    916
.      = 020707            26#    37    38#   41#   43#   47#   53#   58#   59#   60#   61#   219   857
                          895#   898#  899   916   917#  920#  921#  924   926#  932   1102#
.$ASTA= ****** U          921
.$X    = 001000            60#
```

```
COMMEN    20#
ENDCOM    20#
ERROR     20#   299   320   429   436   442   444   445   446   448   449   450   452   453   454
          456   457   458   460   461   469   478   488   497   507   518   528   539   551   559
          564   567   574   589   602   613   630   653   690   702   714   728   741   759   786
          801   818   827   975
ESCAPE    20#
FIXUNT    902#  916
FLOAT0    118#  444
FLOAT1    106#  442
GETPRI    20#
GETSWR    12#    20#   227#
HIBYTE    86#   446
MULT      20#
NEWTST    20#   420   430   442   444   445   446   448   449   450   452   453   454   456   457
          458   460   461   462   471   481   490   499   509   520   530   541   553   576   593
          604   616   639   677   692   704   716   730   747   765   806   831
POP       20#   920   921   924   928
PUSH      20#   920   921   924   928
READRW    142#  448   449   450   452   453   454   456   457   458   460
REPORT    20#
RESET0    132#  445   461
SCOPE     20#   420   430   442   444   445   446   448   449   450   452   453   454   456   457
          458   460   461   462   471   481   490   499   509   520   530   541   553   576   593
          604   616   639   677   692   704   716   730   747   765   806   831   857
SETPRI    20#   898
SETTRA    936#
SETUP     20#   219
SKIP      20#   422   428   435   442   444   445   446   448   449   450   452   453   454   456
          457   458   460   461   468   477   487   496   506   517   527   538   550   558   566
          573   588   601   612   631   654   689   701   713   727   740   758   785   800   817
          826   834   1009
SLASH     20#
SPACE     20#
STARS     20#    58    60    61   420   430   442   444   445   446   448   449   450   452   453
          454   456   457   458   460   461   462   471   481   490   499   509   520   530   541
          553   576   593   604   616   639   677   692   704   716   730   747   765   806   831
          857   898   899   916   917   919   920   921   924   928   932   933   936
SWRSU     20#   219#
TRMTRP    936#
TYPBIN    20#
TYPDEC    20#   857   917
TYPNAM    20#   227
TYPNUM    20#
TYPOCS    20#  1016  1018
TYPOCT    20#   898   917
TYPTXT    20#
$$CMRE    61#
$$CMTM    61#
$$ESCA    20#
$$NEWT    20#   420   430   442   444   445   446   448   449   450   452   453   454   456   457
          458   460   461   462   471   481   490   499   509   520   530   541   553   576   593
          604   616   639   677   692   704   716   730   747   765   806   831
$$SET     936#
```

```
$$SETM     219#
$$SKIP      20#    422     428     445     448     449     450     452     453     454     456     457     458     460     451
           468     477     487     496     506     517     527     538     550     573     601     612     689     701     713
           727
.EQUAT      10#     20
.HEADE      10#     15
.SETUP      11#    211
.SWRHI      12#     22
.SWRLO      22#
.$ACT1      13#     58
.$APTB      13#     61#
.$APTH      13#     60
.$APTY      13#    921
.$CATC      10#
.$CMTA      10#     61
.$EOP       10#    857
.$ERRO      10#    916
.$ERRT      12#    917
.$PARM      11#
.$POWE      11#    924
.$RDOC      12#    928
.$READ      11#    898
.$SAVE      11#
.$SCOP      11#    899
.$SPAC      11#
.$SWDO      11#
.$TRAP      11#    936
.$TYPB      12#    933
.$TYPD      12#    920
.$TYPE      10#     11#    932
.$TYPO      10#    919
```

ERRORS DETECTED: 0

```
 *CVMNBA,CVMNBA/CRF=CVMNBA
 RUN-TIME: 26 13 2 SECONDS
 CORE USED:  26K
```