MNCDI

MNCDI DIAG
CVMNBB0

AH-B089B-MC
COPYRIGHT 78-80
FICHE 1 OF 1

JAN 1980
digital
MADE IN USA

IDENTIFICATION

PRODUCT CODE: AC-B088B-MC

PRODUCT NAME: CVMNBBO MNCDI (DIGITAL IN) DIAGNOSTIC TEST

DATE: JULY 1979

MAINTAINER: DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1.0 ABSTRACT

THE "B" VERSION CORRECTS A PROGRAM BUG IN THE TESTER SECTION.
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 120.
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

SEQ 0006

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-B MNCDI   DIAGNOSTIC
(1)                              ;*COPYRIGHT (C) 1979
(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  014702  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  010234                          $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)
```

```
                                .SBTTL  COMMON TAGS

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

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

                        ;;**********************************************************************
                        .EVEN
001170                  $MAIL:                          ;;APT MAILBOX
001170  000000          $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001172  000000          $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001174  000000          $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001176  000000          $PASS:  .WORD   APASS           ;;PASS COUNT
001200  000000          $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001202  000000          $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001204  000000          $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
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  015764  017262  020624             EM1,DH1,DT1,DF0         ;MNCDO BUS ERROR
     001262  021006
 65                                     ;ITEM   2
 66  001264  016625  017520  020726             EM12,DH12,DT12,DF0      ;MNCDO-MNCDI WRAP-AROUND STATUS ERROR
     001272  021006
 67                                     ;ITEM   3
 68  001274  016716  017563  020744             EM13,DH13,DT13,DF0      ;MNCDO-MNCDI WRAP-AROUND DATA ERROR
     001302  021006
 69                                     ;ITEM   4
 70  001304  017005  017407  020666             EM14,DH7,DT7,DF0        ;MNCDI INCORRECT I.D. VALUE
     001312  021006
 71                                     ;ITEM   5
 72  001314  016262  017313  020636             EM5,DH5,DT5,DF0         ;MNCDI BUS ERROR
     001322  021006
 73                                     ;ITEM   6
 74  001324  016367  017353  020652             EM6,DH6,DT6,DF0         ;MNCDI STIMULUS REGISTER ERROR
     001332  021006
 75                                     ;ITEM   7
 76  001334  016442  017407  020666             EM7,DH7,DT7,DF0         ;MNCDI STATUS REGISTER ERROR
     001342  021006
 77                                     ;ITEM   10
 78  001344  016513  017443  020702             EM10,DH10,DT10,DF0      ;MNCDI DATA REGISTER ERROR
     001352  021006
 79                                     ;ITEM   11
 80  001354  016562  017476  020716             EM11,DH11,DT11,DF0      ;MNCDI INTERRUPT ERROR
     001362  021006
 81                                     ;ITEM   12
 82  001364  017052  017626  020762             EM15,DH15,DT15,DF0      ;MNCDI ILLEGAL OR TRAP INTERRUPT
     001372  021006
 83                                     ;ITEM   13
 84  001374  017123  017661  020774             EM16,DH16,DT16,DF0      ;EXISTING MNCDI FAILED TO RESPOND
     001402  021006
```

```
157  001404  000000                OCSR:   0
158  001406  000000                OCSR1:  0                       ;HIGH BYTE ADDRESS
159
160  001410  000000                DOR:    0
161  001412  000000                DOR1:   0                       ;HIGH BYTE ADDRESS
162
163  001414  000000                ICSR:   0
164  001416  000000                ICSR1:  0                       ;HIGH BYTE ADDRESS
165
166  001420  000000                DIR:    0
167  001422  000000                DIR1:   0
168  001424  000000                SBR:    0
169  001426  000000                SBR1:   0                       ;HIGH BYTE ADDRESS
170
171  001430  000000                DIDINV: 0                       ;INPUT INTERREPT VECTOR # 1
172  001432  000000                DIDINS: 0
173
174  001434  000000                DIEINV: 0                       ;INPUT INTERRUPT VECTOR # 2
175  001436  000000                DIEINS: 0
176
177  001440  000000                DWARF:  0       ;0= NO EDGE CONNECTOR, =1 IN-HOUSE TESTER, =2 WRAP-AROUND
178  001442  000000                TEMP:   0
179  001444  000000                TEMP1:  0
180  001446  000000                TEMP2:  0                       ;RESTART INDICATOR
181  001450  167770                TSTR0:  167770                  ;IN-HOUSE TESTER ADDRESS
182  001452  167772                TSTR2:  167772
183  001454  167774                TSTR4:  167774
184  001456  167776                TSTR6:  167776
185  001460  000000                MASKNM: 0                       ;DEVICE MASK
186  001462  000004                VADDRO: 4                       ;MULTIPLE OUTPUT UNITS ADDRESSESS DIFFERENCE
187  001464  000010                VADDR:  10                      ;MULTIPLE INPUT UNITS, ADDRESS DIFFERENCE
188  001466  000120                VECLST: AVECT1                  ;VECTOR FOR UNIT #0
189  001470  000130                        AVECT1+10               ;                  #1
190  001472  000140                        AVECT1+20               ;                  #2
191  001474  000150                        AVECT1+30               ;                  #3
192  001476  000470                        AVECT1+350              ;                  #4
193  001500  000460                        AVECT1+340              ;                  #5
194  001502  000450                        AVECT1+330              ;                  #6
195  001504  000440                        AVECT1+320              ;                  #7
196  001506  000000                VECOFF: 0                       ;VECTOR OFFSET FROM DEFAULT
197  001510  000010                        10                      ;                  #1
198  001512  000020                        20                      ;                  #2
199  001514  000030                        30                      ;                  #3
200  001516  000350                        350                     ;                  #4
201  001520  000340                        340                     ;                  #5
202  001522  000330                        330                     ;                  #6
203  001524  000320                        320                     ;                  #7
204  001526  000000                EVER:   0
205  001530  000000                BADUNT: 0                       ;BAD UNIT INDICATOR
206  001532  000000                UNITBD: 0
207
208          010000                        BITDAT=BIT12            ;MAINT INPUT INHIBIT
209          004000                        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  012120  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
  (1)  001614  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
  (1)  001622  012737  012440  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
  (1)  001630  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
  (1)  001636  012737  014616  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
  (1)  001644  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
  (1)  001652  012737  013736  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
  (1)  001660  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
  (1)  001666  013737  010202  010174          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 '$TYPE'' ROUTINE
002036  012737  005046  014260          MOV     #5046,$TYPE     ;CLR -(SP)
002044  012737  012746  014262          MOV     #12746,$TYPE+2  ;MOV #$TYPE+12,-(SP)
002052  012737  014272  014264          MOV     #$TYPE+12,$TYPE+4
002060  012737  000002  014266          MOV     #RTI,$TYPE+6    ;RTI
002066  004737  010636                  JSR     PC,$TKINT       ;ENABLE TKB INTR.
                                        .SBTTL  TYPE PROGRAM NAME
(1)                                     ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1)  002072  005227  177777             INC     #-1             ;;FIRST TIME?
(1)  002076  001053                     BNE     68$             ;;BRANCH IF NO
(1)  002100  022737  010234  000042     CMP     #$ENDAD,@#42    ;;ACT-11?
(1)  002106  001447                     BEQ     68$             ;;BRANCH IF YES
(1)  002110  104401  002156             TYPE    ,69$            ;;TYPE ASCIZ STRING
(2)                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
(2)  002114  005737  000042             TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
(2)  002120  001012                     BNE     70$             ;;BRANCH IF YES
(2)  002122  123727  001210  000001     CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
(2)  002130  001406                     BEQ     70$             ;;BRANCH IF YES
(2)  002132  023727  001140  000176     CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
(2)  002140  001005                     BNE     71$             ;;BRANCH IF NO
(2)  002142  104407                     GTSWR                   ;;GET SOFT-SWR SETTINGS
(2)  002144  000403                     BR      71$
(2)  002146  112737  000001  001134  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
(2)  002154                          71$:
(1)  002154  000424                     BR      68$             ;;GET OVER THE ASCIZ
(1)                                  ;;69$:  .ASCIZ  <CRLF>#CVMNB-B  MNCDI (DIGITAL IN) DIAGNOSTIC#<CRLF>
(1)  002226                          68$:
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$: TST     TEMP2           ;TEST IF RESTART
002244  001015                          BNE     MTEST1          ;BR IF YES
002246  104401  020156                  TYPE,   SETUP0          ;INFORM OPERATOR ABOUT FRONT SWITCH
002252  022737  000001  001440          CMP     #1,DWARF        ;TEST IF TESTER
002260  001005                          BNE     MTEST           ;BR IF NOT AND TELL OPERATOR THE HEADER
002262  104401  020257                  TYPE,   SETUP1          ;INFORM OPERATOR ABOUT TESTER
002266  013737  001450  001250          MOV     TSTR0,$CDW1     ;LOAD TESTER DEFAULT FOR WRAPAROUND
```

```
                                        .SBTTL  KEYBOARD COMMAND DECODER
002274  104401  015160          MTEST:  TYPE,   PRIME0          ;TELL THE OPER. THE TESTS AVAIL
002300  000240                  MTEST1: NOP
002302  052777  000100  176634          BIS     #BIT6,@$TKS     ;ENABLE TKB INTR.
002310  005037  001176                  CLR     $PASS           ;PRIME THE PASS COUNT
002314  005037  001112                  CLR     $ERTTL          ;PRIME THE TOTAL # OF ERRORS
002320  005037  001526                  CLR     EVER            ;INIT. THE UNIT TYPEOUT
002324  004737  003010                  JSR     PC,FIXADR       ;ENSURE BASE AND VECTOR ADDRESS IS LOADED
002330  104401  015674                  TYPE,   DOT             ;INDICATE THE REST POINT
002334  104412                          RDLIN                   ;GET OPER. INPUT
002336  013637  002512                  MOV     @(SP)+,RUNIT    ;SAVE THE FIRST CHAR.
002342  142737  000040  002512          BICB    #40,RUNIT       ;ENSURE UPPER CASE
002350  122737  000102  002512          CMPB    #'B,RUNIT       ;TEST IF 'B'
002356  001002                          BNE     1$              ;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  010354                  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  020332          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  010146                  JMP     $EOP
002636  012737  014702  000004  3$:     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  017207                  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  017231                  TYPE    ,FOUND2         ;FINISH MESSAGE
002706  005737  001202                  TST     $UNIT           ;ANY UNITS
002712  001002                          BNE     6$              ;BR IF SOME
002714  000137  010146                  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                MOV     $VECT1,(R0)     ;GET BASE
003110  062120                (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003112  013710  001240                MOV     $VECT1,(R0)     ;GET BASE
003116  062120                (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003120  013710  001240                MOV     $VECT1,(R0)     ;GET BASE
003124  062120                (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003126  013710  001240                MOV     $VECT1,(R0)     ;GET BASE
003132  062120                (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003134  013710  001240                MOV     $VECT1,(R0)     ;GET BASE
003140  062120                (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003142  013710  001240                MOV     $VECT1,(R0)     ;GET BASE
003146  062120                (1)     ADD     (R1)+,(R0)+     ;ADD OFFSET
003150  013710  001240                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  020432                      ADROUT                  ;ASK FOR OUTPUT ADDR.
003214  013746  001250              MOV     $CDW1,-(SP)     ;GET DEFAULT VALUE
003220  104402                      TYPOC                   ;TELL OPER.
003222  104401  020607              TYPE,   ENDOUT          ;ADD END
003226  104413                      RDOCT                   ;AND WAIT FOR INPUT
003230  005726                      TST     (SP)+           ;WAS IT A <CR> FOR DEFAULT
003232  001403                      BEQ     1$              ;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  020476                      ADRIN                   ;ASK FOR INPUT ADDR.
003256  013746  001244              MOV     $BASE,-(SP)     ;GET DEFAULT
003262  104402                      TYPOC                   ;TELL OPER. DEFAULT
003264  104401  020607              TYPE,   ENDOUT          ;ADD END
003270  104413                      RDOCT                   ;AND WAIT FOR INPUT
003272  005726                      TST     (SP)+           ;WAS IT A <CR> FOR DEFAULT
003274  001403                      BEQ     1$              ;YES-BRANCH
003276  016637  177776  001244      MOV     -2(SP),$BASE    ;NO-LOAD NEW ADDR
003304  104401              1$:     TYPE
003306  020541                      VECIN                   ;ASK FOR INPUT VECTOR
003310  013746  001240              MOV     $VECT1,-(SP)    ;GET DEFAULT
003314  104402                      TYPOC                   ;TELL OPER THE DEFAULT
003316  104401  020607              TYPE,   ENDOUT          ;ADD END
003322  104413                      RDOCT                   ;AND WAIT FOR INPUT
003324  005726                      TST     (SP)+           ;WAS IT A <CR> FOR DEFAULT
003326  001403                      BEQ     2$              ;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
```

```
420                                     ;;*****************************************************************
(3)                                     ;*TEST 1        VERIFY CORRECT I.D. CODE FOR MNCDI (IN-HOUSE TESTER)
(3)                                     ;;*****************************************************************
(2)  003424  000004                     TST1:   SCOPE
421  003426  022737  000001  001440             CMP     #1,DWARF        ;TEST IF ''IN-HOUSE TESTER'' MODE
422  003434  001020                             BNE     TST2            ;;BR IF NOT
423  003436  005077  176010                     CLR     @TSTR2          ;ENSURE TESTER MODE
424  003442  017737  176006  001126             MOV     @TSTR4,$BDDAT   ;READ I.D. VALUE
425  003450  042737  177417  001126             BIC     #177417,$BDDAT  ;MASK TO OTHER BITS
426  003456  012737  000140  001124             MOV     #140,$GDDAT     ;LOAD EXPECTED I.D. VALUE
427  003464  023737  001124  001126             CMP     $GDDAT,$BDDAT   ;COMPARE
428  003472  001401                             BEQ     TST2            ;;BR IF SAME
429  003474  104004                             ERROR   4               ;INCORRECT I.D. VALUE FOR MNCDI
430                                     ;;*****************************************************************
(3)                                     ;*TEST 2        VERIFY A MNCDI BUS ADDRESS RESPONSE
(3)                                     ;;*****************************************************************
(2)  003476  000004                     TST2:   SCOPE
431  003500  012737  003524  000004             MOV     #1$,ERRVEC              ;LOAD BUS TRAP VECTOR
432  003506  005777  175702                     TST     @ICSR                   ;TEST INPUT STATUS
433  003512  005777  175702                     TST     @DIR                    ;TEST INPUT DATA REGISTER
434  003516  005777  175702                     TST     @SBR                    ;TEST STIM. BUFFER REGISTER
435  003522  000411                             BR      2$              ;;BR IF NO TIMEOUT
436  003524  104005                     1$:     ERROR   5               ;BUS TIMEOUT WHEN REFERENCING THE MNCDI
437  003526  012737  014702  000004             MOV     #IOTRD,ERRVEC   ;RESTORE TRAP
438  003534  012737  000200  000006             MOV     #200,ERRVEC+2   ;VECTOR
439  003542  000137  007772                     JMP     REMAIN          ;CHECK FOR MORE UNITS
440  003546  012737  014702  000004     2$:     MOV     #IOTRD,ERRVEC   ;RESTORE TRAP VECTOR
441  003554  012737  000200  000006             MOV     #200,ERRVEC+2
442                                     ;;*****************************************************************
(4)                                     ;*TEST 3        FLOAT A 1 ACROSS THE MNCDI STIMULUS BIT REGISTER
(4)                                     ;;*****************************************************************
(3)  003562  000004                     TST3:   SCOPE
(1)  003564  012737  000001  001124             MOV     #BIT0,$GDDAT                    ;LOAD EXPECTED BIT
(1)  003572  012737  003600  001106             MOV     #1$,$LPADR                      ;LOAD LOOP ADDRESS
(1)  003600  013777  001124  175616     1$:     MOV     $GDDAT,@SBR                     ;LOAD MNCDI STIMULUS BIT REGISTER
(1)  003606  017737  175612  001126             MOV     @SBR,$BDDAT                     ;READ MNCDI STIMULUS BIT REGISTER
(1)  003614  023737  001124  001126             CMP     $GDDAT,$BDDAT                   ;COMPARE
(2)  003622  001401                             BEQ     2$              ;;BR IF EXPECTED
(1)  003624  104006                             ERROR   6               ; MNCDI STIMULUS BIT REGISTER FAILED TO HOLD A FLOATING
(1)  003626  006337  001124             2$:     ASL     $GDDAT                  ;CHANGE THE DATA
(1)  003632  001362                             BNE     1$                      ;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$:
```

```
448                                     ;;*******************************************************************
(4)                                     ;*TEST 7        TEST THAT BIT1 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004102  000004                     TST7:   SCOPE
(1)  004104  012737  000002  001124             MOV     #BIT1,$GDDAT            ;LOAD EXPECTED
(1)  004112  013777  001124  175274             MOV     $GDDAT,@ICSR            ;LOAD BIT1 INTO MNCDI STATUS REGISTER
(1)  004120  017737  175270  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004126  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004134  001401                             BEQ     1$              ;;BR IF SET
(1)  004136  104007                             ERROR   7               ;BIT1 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004140  043777  001124  175246  1$:       BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004146  017737  175242  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004154  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004162  001001                             BNE     TST10           ;;BR IF CLEARED
(1)  004164  104007                             ERROR   7               ;BIT1 OF MNCDI STATUS REGISTER FAILED TO CLEAR
449                                     ;;*******************************************************************
(4)                                     ;*TEST 10       TEST THAT BIT2 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004166  000004                     TST10:  SCOPE
(1)  004170  012737  000004  001124             MOV     #BIT2,$GDDAT            ;LOAD EXPECTED
(1)  004176  013777  001124  175210             MOV     $GDDAT,@ICSR            ;LOAD BIT2 INTO MNCDI STATUS REGISTER
(1)  004204  017737  175204  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004212  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004220  001401                             BEQ     1$              ;;BR IF SET
(1)  004222  104007                             ERROR   7               ;BIT2 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004224  043777  001124  175162  1$:       BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004232  017737  175156  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004240  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004246  001001                             BNE     TST11           ;;BR IF CLEARED
(1)  004250  104007                             ERROR   7               ;BIT2 OF MNCDI STATUS REGISTER FAILED TO CLEAR
450                                     ;;*******************************************************************
(4)                                     ;*TEST 11       TEST THAT BIT3 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)                                     ;;*******************************************************************
(3)  004252  000004                     TST11:  SCOPE
(1)  004254  012737  000010  001124             MOV     #BIT3,$GDDAT            ;LOAD EXPECTED
(1)  004262  013777  001124  175124             MOV     $GDDAT,@ICSR            ;LOAD BIT3 INTO MNCDI STATUS REGISTER
(1)  004270  017737  175120  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER
(1)  004276  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004304  001401                             BEQ     1$              ;;BR IF SET
(1)  004306  104007                             ERROR   7               ;BIT3 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1)  004310  043777  001124  175076  1$:       BIC     $GDDAT,@ICSR            ;CLEAR THAT BIT
(1)  004316  017737  175072  001126             MOV     @ICSR,$BDDAT            ;READ MNCDI STATUS REGISTER AGAIN
(1)  004324  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004332  001001                             BNE     TST12           ;;BR IF CLEARED
(1)  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
```

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

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

```
                        ;;*****************************************************************
                        ;*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
                                ;;*****************************************************************
(2) 005732 000004               TST33:  SCOPE
554 005734 005037 001124                CLR     $GDDAT                  ;LOADEXPECTED
555 005740 012777 010000 173446         MOV     #BITDAT,@ICSR           ;SET INPUT INHIBIT
556 005746 017737 173446 001126         MOV     @DIR,$BDDAT             ;READ INPUT
557 005754 023737 001124 001126         CMP     $GDDAT,$BDDAT           ;COMPARE
558 005762 001401                       BEQ     1$              ;;BR IF SAME
559 005764 104007                       ERROR   7                       ;INPUT INHIBIT FAILED TO INHIBIT INPUT
560 005766 012777 010060 173420 1$:     MOV     #BITDAT!BIT5!BIT4,@ICSR ;SET INVERT DATA AND INPUT INHIBIT
561 005774 012737 177777 001124         MOV     #-1,$GDDAT              ;LOAD EXPECTED
562 006002 017737 173412 001126         MOV     @DIR,$BDDAT             ;READ INPUT
563 006010 001001                       BNE     2$                      ;BR IF NON-ZERO
564 006012 104007                       ERROR   7                       ;INVERT DATA FUNCTION FAILED
565 006014 023737 001124 001126 2$:     CMP     $GDDAT,$BDDAT           ;COMPARE DATA
566 006022 001401                       BEQ     3$              ;;BR IF SAME
567 006024 104007                       ERROR   7                       ;INVERT DATA - DATA PATH ERROR
568
569 006026 012777 010000 173360 3$:     MOV     #BITDAT,@ICSR           ;SET INPUT INHIBIT
570 006034 005037 001124                CLR     $GDDAT                  ;CLEAR EXPECTED
571 006040 017737 173354 001126         MOV     @DIR,$BDDAT             ;READ INPUT
572 006046 023737 001124 001126         CMP     $GDDAT,$BDDAT           ;COMPARE
573 006054 001401                       BEQ     TST34           ;;BR IF SAME
574 006056 104007                       ERROR   7                       ;INVERT DATA FUNCTION OR INPUT INHIBIT FAILED
575
576                             ;;*****************************************************************
                                ;*TEST 34       VERIFY EACH BIT OF THE MNCDI INPUT DATA REGISTER CAN BE CLEARED
                                ;;*****************************************************************
(2) 006060 000004               TST34:  SCOPE
577 006062 012737 006076 001110         MOV     #1$,$LPERR              ;LOAD LOOP ADDRESS ON ERROR
578 006070 012737 000001 001442         MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
579 006076 012777 010060 173310 1$:     MOV     #BITDAT!BIT4!BIT5,@ICSR ;LOAD INHIBIT INPUT AND INVERT DATA
580 006104 013737 001442 001124         MOV     TEMP,$GDDAT             ;LOAD EXPECTED
581 006112 005137 001124                COM     $GDDAT                  ;MAKE OPPOSITE
582 006116 017700 173276                MOV     @DIR,R0                 ;READ INPUT
583 006122 013777 001442 173270         MOV     TEMP,@DIR               ;CLEAR THE INPUT BIT
584 006130 042777 000040 173256         BIC     #BIT5,@ICSR             ;REM INVERT DATA BITOVE
585 006136 052777 000002 173250         BIS     #BIT1,@ICSR             ;ENABLE EXT. STROBE TO PREVENT  DATA INPUT BEING
586 006144 017737 173250 001126         MOV     @DIR,$BDDAT             ;READ INPUT REG.
587 006152 023737 001124 001126         CMP     $GDDAT,$BDDAT           ;COMPARE
588 006160 001401                       BEQ     2$              ;;BR IF SAME
589 006162 104010                       ERROR   10                      ;INPUT REGISTER BIT FAILED TO CLEAR
590 006164 006337 001442        2$:     ASL     TEMP                    ;SHIFT THE DATA
591 006170 001342                       BNE     1$                      ;TRY MORE BITS
```

```
                        ;;*****************************************************************
                        ;*TEST 35       VERIFY THAT 'RESET' CLEARS MNCDI INPUT DATA REGISTER
                        ;;*****************************************************************
006172  000004          TST35:  SCOPE
006174  012737  000040  001160          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
006202  012777  004060  173204          MOV     #BITEXT!BIT5!BIT4,@ICSR ;INVERT DATA AND INHIBIT INPUT
006210  017737  173204  001442          MOV     @DIR,TEMP               ;READ REGISTER
006216  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED
006222  000005                          RESET                           ;CLEAR THE INPUT REG.
006224  052777  000100  172712          BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
006232  052777  000004  173154          BIS     #BIT2,@ICSR             ;INHIBIT REG. FROM BEING CLOCKED
006240  017737  173154  001126          MOV     @DIR,$BDDAT             ;READ REGISTER
006246  001401                          BEQ     TST36           ;;BR IF CLEARED
006250  104010                          ERROR   10                      ;RESET FAILED TO CLEAR INPUT REGISTER

                        ;;*****************************************************************
                        ;*TEST 36       VERIFY THAT A 2ND STROBE PULSE WILL NOT CHANGE THE DIR DATA
                        ;;*****************************************************************
006252  000004          TST36:  SCOPE
006254  012777  010062  173132          MOV     #BIT12!BIT5!BIT4!BIT1,@ICSR     ;DISABLE INPUTS, ENABLE INVERT DATA, EXT
006262  052777  004200  173124          BIS     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
006270  042777  000040  173116          BIC     #BIT5,@ICSR             ;REMOVE INVERT DATA
006276  052777  004200  173110          BIS     #BITEXT!BIT7,@ICSR      ;SET MAINT. STROBE AGAIN
006304  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD EXPECTED DATA
006312  017737  173102  001126          MOV     @DIR,$BDDAT             ;READ REGISTER
006320  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
006326  001401                          BEQ     TST37           ;;BR IF SAME
006330  104010                          ERROR   10                      ;DATA READY FAILED TO INHIBIT 2ND
                                                                        ;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
                                ;;*****************************************************************
006566  000004                  TST41:  SCOPE
006570  005737  001440                  TST     DWARF                   ;TEST IF WRAP-AROUND OR TESTER MODE
006574  001002                          BNE     1$                      ;BR IF YES
006576  000137  007772                  JMP     REMAIN                  ;NO REPORT END OF PASS
006602  012777  177777  172610  1$:     MOV     #-1,@DIR                ;CLEAR INPUT REG.
006610  005077  172610                  CLR     @SBR                    ;CLEAR STIM. REG.
006614  005077  172574                  CLR     @ICSR                   ;CLEAR INPUT DATA READY FLAG
006620  005077  172560                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
006624  012777  025252  172556          MOV     #25252,@DOR             ;WRITE TO THE OUTPUT DATA REG.
006632  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
006640  004737  010550                  JSR     PC,DELAY0               ;DELAY A SHORT TIME
006644  017737  172544  001126          MOV     @ICSR,$BDDAT            ;READ STATUS
006652  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
006660  001401                          BEQ     TST42           ;;BR IF SET
006662  104002                          ERROR   2                       ;MODE 00  -- EXT. STROBE FAILED TO SET INPUT DAT
                                ;;*****************************************************************
                                ;*TEST 42        VERIFY MODE 01  -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
                                ;;*****************************************************************
006664  000004                  TST42:  SCOPE
006666  012777  177777  172524          MOV     #-1,@DIR                ;CLEAR INPUT REG.
006674  005077  172524                  CLR     @SBR                    ;CLEAR STIM. REG.
006700  012777  000002  172506          MOV     #BIT1,@ICSR             ;CLEAR INPUT DATA READY FLAG
006706  005077  172472                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
006712  012777  052525  172470          MOV     #52525,@DOR             ;WRITE TO THE OUTPUT DATA REG.
006720  012737  000202  001124          MOV     #BIT7!BIT1,$GDDAT       ;LOAD EXPECTED
006726  004737  010550                  JSR     PC,DELAY0               ;DELAY A SHORT TIME
006732  017737  172456  001126          MOV     @ICSR,$BDDAT            ;READ STATUS
006740  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
006746  001401                          BEQ     TST43           ;;BR IF SET
006750  104002                          ERROR   2                       ;MODE 01  -- EXT. STROBE FAILED TO SET INPUT DAT

                                ;;*****************************************************************
                                ;*TEST 43        VERIFY MODE 10  -- INPUT STROBE WILL NOT SET INPUT DATA READY FLAG
                                ;;*****************************************************************
006752  000004                  TST43:  SCOPE
006754  012777  177777  172436          MOV     #-1,@DIR                ;CLEAR INPUT REG.
006762  005077  172436                  CLR     @SBR                    ;CLEAR STIM. REG.
006766  012777  000004  172420          MOV     #BIT2,@ICSR             ;CLEAR INPUT DATA READY FLAG
006774  005077  172404                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
007000  012777  070707  172402          MOV     #70707,@DOR             ;WRITE TO THE OUTPUT DATA REG.
007006  012737  000004  001124          MOV     #BIT2,$GDDAT            ;LOAD EXPECTED
007014  004737  010550                  JSR     PC,DELAY0               ;DELAY A SHORT TIME
007020  017737  172370  001126          MOV     @ICSR,$BDDAT            ;READ STATUS
007026  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007034  001401                          BEQ     TST44           ;;BR IF CLEARED
007036  104002                          ERROR   2                       ;MODE 10  -- EXT. STROBE SET INPUT DATA READY FL
```

```
                                ;;*****************************************************************
                                ;*TEST 44       VERIFY INPUT REPLY SETS OUTPUT DONE FLAG
                                ;;*****************************************************************
007040  000004                  TST44:  SCOPE
007042  005077  172342                  CLR     @DOR                    ;CLEAR OUTPUT DATA
007046  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
007054  022737  000001  001440          CMP     #1,DWARF                ;CHECK IF IN TESTER MODE
007062  001003                          BNE     1$                      ;BR IF NOT
007064  012737  100000  001124          MOV     #BIT15,$GDDAT           ;LOAD TESTER WRAPAROUND STATUS
007072  005077  172306          1$:     CLR     @OCSR                   ;CLEAR OUTPUT DONE FLAG
007076  012777  004200  172310          MOV     #BITEXT!BIT7,@ICSR      ;SET INPUT READY FLAG
007104  005077  172304                  CLR     @ICSR                   ;CLEAR INPUT READY FLAG<GEN. INPUT REPLY>
007110  004737  010550                  JSR     PC,DELAY0               ;DELAY A SHORT TIME
007114  017737  172264  001126          MOV     @OCSR,$BDDAT            ;READ OUTPUT STATUS
007122  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007130  001401                          BEQ     TST45           ;;BR IF SET
007132  104002                          ERROR   2                       ;INPUT REPLY FAILED TO SET OUTPUT DONE FLAG

                                ;;*****************************************************************
                                ;*TEST 45       VERIFY THE MNCDO - WRAPAROUND - MNCDI DATA PATH
                                ;;*****************************************************************
007134  000004                  TST45:  SCOPE
007136  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
007144  012777  177777  172246          MOV     #-1,@DIR                ;CLEAR INPUT REG.
007152  005077  172226                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
007156  012777  000002  172230          MOV     #BIT1,@ICSR             ;INITILIZE THE INPUT STATUS REGISTER
007164  012737  000001  001442          MOV     #BIT0,TEMP              ;LOAD EXPECTED

007172  013737  001442  001124  1$:     MOV     TEMP,$GDDAT             ;LOAD TYPEOUT EXPECTED
007200  012777  000002  172206          MOV     #BIT1,@ICSR             ;INITILIZE THE INPUT STATUS REG.
007206  013777  001124  172174          MOV     $GDDAT,@DOR             ;LOAD OUTPUT DATA REG.
007214  105777  172174          2$:     TSTB    @ICSR                   ;WAIT FOR STROBE PULSE
007220  100375                          BPL     2$                      ; TO OCCUR
007222  017737  172172  001126          MOV     @DIR,$BDDAT             ;READ INPUT DATA REGISTER
007230  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007236  001401                          BEQ     3$              ;;BR IF SAME
007240  104003                          ERROR   3                       ;INPUT DATA PATH ERROR

007242  006337  001442          3$:     ASL     TEMP                    ;TRY NEXT BIT
007246  001351                          BNE     1$                      ;BR IF MORE BITS
```

```
754                                     ;;*****************************************************************
(3)                                     ;*TEST 46       VERIFY THE MNCDO - WRAPAROUND - MNCDI INVERTED DATA PATH
(3)                                     ;;*****************************************************************
(2) 007250  000004                      TST46:  SCOPE
(1) 007252  012737  000100  001160              MOV     #100,$TIMES     ;;DO 100 ITERATIONS
755 007260  012777  177777  172132              MOV     #-1,@DIR                ;CLEAR INPUT REG.
756 007266  005077  172112                      CLR     @OCSR                   ;CLEAR OUTPUT STATUS
757 007272  012777  000062  172114              MOV     #BIT5!BIT4!BIT1,@ICSR   ;LOAD INPUT STATUS <INVERT DATA><READ ENABLE>
758 007300  012737  000001  001442              MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
759
760 007306  013777  001442  172074  1$:         MOV     TEMP,@DOR               ;LOAD OUTPUT DATA REG.
761 007314  013737  001442  001124              MOV     TEMP,$GDDAT             ;GET THE BIT
762 007322  005137  001124                      COM     $GDDAT                  ;INVERT EXPECTED INPUT DATA
763 007326  105777  172062          2$:         TSTB    @ICSR                   ;WAIT FOR INPUT READY
764 007332  100375                              BPL     2$                      ; TO OCCUR
765 007334  017737  172060  001126              MOV     @DIR,$BDDAT             ;READ INPUT DATA REG.
766 007342  023737  001124  001126              CMP     $GDDAT,$BDDAT           ;COMPARE
767 007350  001401                              BEQ     3$              ;;BR IF SAME
768 007352  104003                              ERROR   3                       ;INVERTED INPUT DATA PATH ERROR
769
770 007354  042777  000200  172032  3$:         BIC     #BIT7,@ICSR             ;CLEAR INPUT READY
771 007362  006337  001442                      ASL     TEMP                    ;TRY NEXT BIT
772 007366  001347                              BNE     1$                      ;BR IF MORE BITS
773
```

```
                                        ;;*********************************************************************
                                        ;*TEST 47       VERIFY IN 10 MODE THAT SBR AND INPUT  BITS SET INPUT READY
                                        ;;*********************************************************************
007370  000004                  TST47:  SCOPE
007372  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
                                ;       INPUT DATA READY FLAG
                                ;               LOAD A FLOATING 1 ACROSS THE SBR
                                ;               VERIFY THAT ONLY THE CORRECT BIT SET DATA READY

007400  005077  172004          1$:     CLR     @DOR
007404  012737  000001  001442          MOV     #BIT0,TEMP              ;LOAD INITIAL BIT

007412  005077  171772                  CLR     @DOR                    ;CLEAR OUTPUT BITS
007416  005077  172002                  CLR     @SBR                    ;CLEAR SBR REG.
007422  012777  000004  171764          MOV     #BIT2,@ICSR             ;CLEAR INPUT READY AND SET MODE 10
007430  012777  177777  171762          MOV     #-1,@DIR                ;CLEAR INPUT REG.

007436  013777  001442  171760          MOV     TEMP,@SBR               ;LOAD SBR REG.
007444  042777  000200  171742          BIC     #BIT7,@ICSR             ;CLEAR INPUT READY BIT
007452  013777  001442  171730          MOV     TEMP,@DOR               ;LOAD OUTPUT REG.

007460  012737  100204  001124          MOV     #BIT15!BIT7!BIT2,$GDDAT ;LOAD EXPECTED STATUS
007466  004737  010550                  JSR     PC,DELAY0               ;DELAY A SHORT TIME
007472  017737  171716  001126          MOV     @ICSR,$BDDAT            ;READ STATUS
007500  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007506  001401                          BEQ     2$              ;;BR IF SET
007510  104002                          ERROR   2                       ;INPUT DATA READY FLAG FAILED
                                                                ;TO SET IN MODE 10 <STIMILUS MODE>

                                ;NOW LOAD ALL BITS EXCEPT THE FLOATING BIT AND ENSURE INPUT DATA READY DOES NOT SET
007512  013737  001442  001444  2$:     MOV     TEMP,TEMP1              ;COPY EXPECTED
007520  005137  001444                  COM     TEMP1                   ;USE REVERSE PATTERN
007524  005077  171660                  CLR     @DOR                    ;CLEAR OUTPUT REG.
007530  012777  177777  171662          MOV     #-1,@DIR                ;CLEAR INPUT REG.
007536  042777  100200  171650          BIC     #BIT15!BIT7,@ICSR       ;CLEAR INPUT DATA READY
007544  012737  000004  001124          MOV     #BIT2,$GDDAT            ;LOAD EXPECTED

007552  013777  001444  171630          MOV     TEMP1,@DOR              ;LOAD ALL OTHER DATA BITS
007560  004737  010550                  JSR     PC,DELAY0               ;DELAY FOR A SHORT TIME
007564  017737  171624  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS
007572  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007600  001401                          BEQ     3$              ;;BR IF CLEARED
007602  104002                          ERROR   2                       ;INPUT DATA READY FLAG SET IN ERROR
                                                                ;UNEXPECTED SBR BIT SET INPUT DATA READY
007604  006337  001442          3$:     ASL     TEMP                    ;TRY NEXT BIT
007610  001300                          BNE     1$                      ;BR IF MORE BITS

                                        ;;*********************************************************************
                                        ;*TEST 50       TEST THE TRANSITION ENABLE AND TRANSITION DETECTION
                                        ;;*********************************************************************
007612  000004                  TST50:  SCOPE
007614  005077  171570                  CLR     @DOR
007620  012777  177777  171572          MOV     #-1,@DIR                ;CLEAR INPUT REGISTER
007626  012737  010000  001442          MOV     #BIT12,TEMP             ;LOAD INITIAL TRANSITION BIT
007634  012777  000404  171552          MOV     #BIT8!BIT2,@ICSR        ;SET STIM. CLEAR READY, ENABLE TRANS.
007642  012737  100604  001124          MOV     #BIT15!BIT8!BIT7!BIT2,$GDDAT    ;LOAD EXPECTED STATUS
```

```
  824  007650  042777  100200  171536  1$:     BIC     #BIT15!BIT7,@ICSR       ;CLEAR READY
  825  007656  013777  001442  171540          MOV     TEMP,@SBR               ;LOAD STIMULUS REG.
  826  007664  013777  001442  171516          MOV     TEMP,@DOR               ;LOAD INPUT REG. <VIA OUTPUT REG.>
  827  007672  004737  010550                  JSR     PC,DELAY0               ;DELAY FOR A SHORT TIME
  828  007676  017737  171512  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS
  829  007704  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
  830  007712  001401                          BEQ     2$              ;;BR IF SAME
  831  007714  104002                          ERROR   2               ;TRANSITION ENABLE OR TRANSITION TO A ONE FAILED
  832                          ;NOW REMOVE THE TRANSITION DATA BIT (THIS SHOULD CAUSE THE INPUT READY FLAG TO SET AGAIN
  833  007716  012777  177777  171474  2$:     MOV     #-1,@DIR                ;CLEAR INPUT REG
  834  007724  042777  100200  171462          BIC     #BIT15!BIT7,@ICSR       ;CLEAR INPUT READY FLAG
  835  007732  043777  001442  171450          BIC     TEMP,@DOR               ;REMOVE THE INPUT DATA
  836                                                  ;THIS SHOULD CAUSE THE TRANSITION TO A ZERO
  837  007740  004737  010550                  JSR     PC,DELAY0               ;DO A SHORT DELAY
  838  007744  017737  171444  001126          MOV     @ICSR,$BDDAT            ;READ INPUT STATUS
  839  007752  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
  840  007760  001401                          BEQ     3$              ;;BR IF SET
  841  007762  104002                          ERROR   2               ;TRANSITION TO A ZERO FAILED
  842  007764  006337  001442          3$:     ASL     TEMP                    ;TRY ANOTHER BIT ?
  843  007770  001327                          BNE     1$                      ;BR IF YES
  844  007772                          REMAIN:
  845                              ;;*****************************************************************
  (3)                              ;*TEST 51       DETERMINE IF MORE MNCDI'S REMAIN TO BE TESTED
  (3)                              ;;*****************************************************************
  (2)  007772  000004              TST51:  SCOPE
  (1)  007774  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
  846  010002  005237  001202                  INC     $UNIT                   ;UPDATE UNIT NUMBER
  847  010006  123737  001202  001526          CMPB    $UNIT,EVER              ;TEST IF MORE
  848  010014  001454                          BEQ     3$              ;;BR IF NOT
  849  010016  012701  001404                  MOV     #OCSR,R1                ;LOAD POINTER TO OUTPUT STATUS ADDRESS
  850  010022  063721  001462          1$:     ADD     VADDRO,(R1)+            ;UPDATE OUTPUT BUS ADDRESS
  851  010026  020127  001414                  CMP     R1,#DOR1+2              ;TEST IF DONE
  852  010032  001373                          BNE     1$                      ;BRANCH IF NOT
  853  010034  063721  001464          2$:     ADD     VADDR,(R1)+             ;UPDATE INPUT ADDRESS
  854  010040  020127  001430                  CMP     R1,#SBR1+2              ;TEST IF DONE
  855  010044  001373                          BNE     2$                      ;BRANCH IF NOT
  856  010046  006337  001460                  ASL     MASKNM                  ;UPDATE ERROR FLAG BIT
  857  010052  004737  012404                  JSR     PC,WHICHU               ;DETERMINE UNIT #
  858  010056  013700  001532                  MOV     UNITBD,R0               ;GET UNIT #
  859  010062  006300                          ASL     R0                      ;MAKE WORD
  860  010064  016037  001466  001430          MOV     VECLST(R0),DIDINV       ;GET VALUE
  861  010072  013737  001430  001432          MOV     DIDINV,DIDINS           ;MAKE OTHER VALUES
  862  010100  062737  000002  001432          ADD     #2,DIDINS
  863  010106  013737  001430  001434          MOV     DIDINV,DIEINV
  864  010114  062737  000004  001434          ADD     #4,DIEINV
  865  010122  013737  001430  001436          MOV     DIDINV,DIEINS
  866  010130  062737  000006  001436          ADD     #6,DIEINS
  867  010136  005037  001102                  CLR     $TSTNM                  ;RESET TEST NUMBER
  868  010142  000137  003424                  JMP     TST1                    ;TEST NEXT UNIT
  869  010146                          3$:
```

```
                        .SBTTL  END OF PASS ROUTINE

                        ;;**********************************************************************
                        ;*INCREMENT THE PASS NUMBER ($PASS)
                        ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
                        ;*IF THERES A MONITOR GO TO IT
                        ;*IF THERE ISN'T JUMP TO EXTMSG

010146                  $EOP:
010146  000004                 SCOPE
010150  005037  001102          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
010154  005037  001160          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
010160  005237  001176          INC     $PASS           ;;INCREMENT THE PASS NUMBER
010164  042737  100000  001176  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
010172  005327                  DEC     (PC)+           ;;LOOP?
010174  000001          $EOPCT: .WORD   1
010176  003022                  BGT     $DOAGN          ;;YES
010200  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
010202  000001          $ENDCT: .WORD   1
010204  010174                  $EOPCT
010206  104401  010253          TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
010212  013746  001176          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
010216  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
010220  104401  010250          TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
010224  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
010230  001405                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
010232  000005                  RESET                   ;;CLEAR THE WORLD
010234  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
010236  000240                  NOP                     ;;SAVE ROOM
010240  000240                  NOP                     ;;FOR
010242  000240                  NOP                     ;;ACT11
010244                  $DOAGN:
010244  000137                  JMP     @(PC)+          ;;RETURN
010246  010270          $RTNAD: .WORD   EXTMSG
010250     377     377  000     $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
010253     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
010260  050040  051501  020123
010266  000043

010270  052777  000100  170646  EXTMSG: BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
010276  005737  001112          TST     $ERTTL          ;TEST IF ANY ERRORS
010302  001416                  BEQ     1$              ;BR IF NONE
010304  104401  017712          TYPE    ,ERRTOT         ;TYPE TOTAL ERRORS MESSAGE
010310  013746  001112          MOV     $ERTTL,-(SP)    ;PUSH TOTAL ERRORS ON STACK
010314  104405                  TYPDS                   ;TYPE IT
010316  022737  000001  001460  CMP     #1,MASKNM       ;TEST IF MULTIPLE
010324  001405                  BEQ     1$              ;BR IF NOT
010326  104401  017741          TYPE    ,MESGD          ;TYPE BAD UNITS
010332  013746  001530          MOV     BADUNT,-(SP)    ;PUSH BAD UNITS ON STACK FOR TYPE OUT
010336  104406                  TYPBN                   ;TYPE IT
010340  104401  010250  1$:     TYPE,   $ENULL          ;ENSURE ALL TEXT GOT TYPED
010344  004737  003346          JSR     PC,CTRLCG       ;TEST FOR CTRL C/G
010350  000137  002520          JMP     LOGIC
```

```
                                        .SBTTL  MNCDI TEST MODULE SWITCH TYPEOUT LOOP
010354 012706 001100            DIDATA: MOV     #STACK,SP       ;LOAD THE STACK POINTER
010360 004737 003010                    JSR     PC,FIXADR               ;FIX DEVICE ADDRESSES
010364 104401 010442                    TYPE,   WAITO
010370 005077 171020            1$:     CLR     @ICSR                   ;CLEAR INPUT STATUS
010374 012777 177777 171016             MOV     #-1,@DIR                ;CLEAR INPUT DATA REGISTER
010402 004737 003346            2$:     JSR     PC,CTRLCG               ;TEST IF CTRL C/G
010406 105777 171002                    TSTB    @ICSR                   ;WAIT FOR INPUT READY
010412 100373                           BPL     2$
010414 017746 171000                    MOV     @DIR,-(SP)              ;GET INPUT DATA
010420 104402                           TYPOC                           ;TYPE THE OCTAL VALUE
010422 104401 020614                    TYPE,   ADASH                   ;TYPE A DASH
010426 017746 170766                    MOV     @DIR,-(SP)              ;GET INPUT DATA AGAIN
010432 104406                           TYPBN                           ;TYPE THE BINARY VALUE
010434 104401 020620                    TYPE,   TCRLF                   ;TYPE A CR-LF
010440 000753                           BR      1$

010442     015     012          WAITO:  .BYTE   15,12
010444 040527 052111 047111             .ASCII  \WAITING FOR OPERATOR TO ACTIVIATE MNCDI (D/I)\
010452 020107 047506 020122
010460 050117 051105 052101
010466 051117 052040 020117
010474 041501 044524 044526
010502 052101 020105 047115
010510 042103 020111 042050
010516 044457    051
010521     040 042524 052123            .ASCII  \ TEST MODULE BUTTON\
010526 046440 042117 046125
010534 020105 052502 052124
010542 047117
010544     015     012     000          .BYTE   15,12,0
        010550                          .EVEN
                                ;SUBROUTINE TO DELAY A SHORT AMOUNT OF TIME
010550 012737 000012 010566     DELAY0: MOV     #10.,10$                ;LOAD DELAY COUNTER
010556 005337 010566            1$:     DEC     10$                     ;DELAY
010562 100375                           BPL     1$                      ; A WHILE
010564 000207                           RTS     PC                      ;EXIT
010566 000000                   10$:    0
```

```
918                                     .SBTTL  TTY INPUT ROUTINE
(1)
(2)                                     ;;**********************************************************************
(1)                                     .ENABL  LSB
(1)  010570  000000                     $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
(1)  010572  000000                     $TKQIN: .WORD   0               ;;INPUT POINTER
(1)  010574  000000                     $TKQOUT: .WORD  0               ;;OUTPUT POINTER
(1)  010576  000040                     $TKQSRT: .BLKB  32.             ;;TTY KEYBOARD QUEUE
(1)          010636                     $TKQEND=.
(1)
(1)                                     ;*TK INITIALIZE ROUTINE
(1)                                     ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
(1)                                     ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
(1)                                     ;
(1)                                     ;*CALL:
(1)                                     ;*      JSR     PC,$TKINT
(1)                                     ;*      RETURN
(1)                                     ;
(1)  010636  005037  010570             $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
(1)  010642  012737  010576  010572             MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
(1)  010650  013737  010572  010574             MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1)  010656  012737  010706  000060             MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
(1)  010664  012737  000200  000062             MOV     #200,@#TKVEC+2  ;;'BR'' LEVEL 4
(1)  010672  005777  170250                     TST     @$TKB           ;;CLEAR DONE FLAG
(1)  010676  012777  000100  170240             MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
(1)  010704  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)  010706  117746  170234             $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
(1)  010712  042716  177600                     BIC     #^C177,(SP)     ;;STRIP THE JUNK
(1)  010716  021627  000003                     CMP     (SP),#3         ;;IS IT A CONTROL C?
(1)  010722  001007                             BNE     1$              ;;BRANCH IF NO
(1)  010724  104401  012056                     TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
(1)  010730  004737  010636                     JSR     PC,$TKINT       ;;INIT THE KEYBOARD
(1)  010734  005726                             TST     (SP)+           ;;CLEAN UP STACK
(1)  010736  000137  002300                     JMP     MTEST1          ;;CONTROL C RESTART
(1)  010742  021627  000007             1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
(1)  010746  001004                             BNE     2$              ;;BRANCH IF NO
(1)  010750  022737  000176  001140             CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
(1)  010756  001500                             BEQ     6$              ;;GO TO SWR CHANGE
(1)
(1)  010760                             2$:
(1)  010760  022737  000040  01057C             CMP     #32.,$TKCNT     ;;IS THE QUEUE FULL?
(1)  010766  001004                             BNE     3$              ;;BRANCH IF NO
(1)  010770  104401  012052                     TYPE    ,$BELL          ;;RING THE TTY BELL
(1)  010774  005726                             TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
(1)  010776  000451                             BR      5$              ;;EXIT
(1)  011000  021627  000023             3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
(1)  011004  001021                             BNE     32$             ;;BRANCH IF NO
(1)  011006  005077  170132                     CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
```

```
(1)  011012  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
(1)  011014  105777  170124          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
(1)  011020  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
(1)  011022  117746  170120                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
(1)  011026  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  011032  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  011036  001366                          BNE     31$             ;;BRANCH IF NO
(1)  011040  012777  000100  170076          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
(1)  011046  000002                          RTI                     ;;RETURN
(1)  011050  005237  010570          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
(1)  011054  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
(1)  011060  002405                          BLT     4$              ;;BRANCH IF YES
(1)  011062  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
(1)  011066  003002                          BGT     4$              ;;BRANCH IF YES
(1)  011070  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
(1)  011074  112677  177472          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
(1)  011100  005237  010572                  INC     $TKQIN          ;;UPDATE THE POINTER
(1)  011104  023727  010572  010636          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
(1)  011112  001003                          BNE     5$              ;;BRANCH IF NO
(1)  011114  012737  010576  010572          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
(1)  011122  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)  011124  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
(1)  011132  001124                          BNE     15$             ;;EXIT IF NOT
(1)  011134  105777  170004                  TSTB    @$TKS           ;;IS A CHAR WAITING?
(1)  011140  100121                          BPL     15$             ;;IF NOT, EXIT
(1)  011142  117746  170000                  MOVB    @$TKB,-(SP)     ;;YES
(1)  011146  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
(1)  011152  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
(1)  011156  001300                          BNE     2$              ;;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)  011160  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  011166  001674                          BEQ     2$              ;;BRANCH IF YES
(1)  011170  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
(1)  011172  004737  010636                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
(1)  011176  005077  167742                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  011202  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
(1)
(1)  011210  104401  012070                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  011214  104401  012075  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  011220  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(1)  011224  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  011226  104401  012106                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  011232  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
(1)  011234  005046                          CLR     -(SP)           ;;THE NEW SWR
(1)  011236  105777  167702          7$.     TSTB    @$TKS           ;;CHAR THERE?
```

```
(1)  011242  100375                        BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  011244  117746  167676                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  011250  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)  011254  021627  000003                CMP     (SP),#3         ;;IS IT A CONTROL-C?
(1)  011260  001015                        BNE     9$              ;;BRANCH IF NOT
(1)  011262  104401  012056                TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
(1)  011266  062706  000006                ADD     #6,SP           ;;CLEAN UP STACK
(1)  011272  123727  001135  000001        CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
(1)  011300  001003                        BNE     8$              ;;BRANCH IF NO
(1)  011302  012777  000100  167634        MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
(1)  011310  000137  002300          8$:   JMP     MTEST1          ;;CONTROL-C RESTART
(1)
(1)
(1)  011314  021627  000025          9$:   CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  011320  001005                        BNE     10$             ;;BRANCH IF NOT
(1)  011322  104401  012063                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  011326  062706  000006          20$:  ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  011332  000737                        BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  011334  021627  000015          10$:  CMP     (SP),#15        ;;IS IT A <CR>?
(1)  011340  001022                        BNE     16$             ;;BRANCH IF NO
(1)  011342  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  011346  001403                        BEQ     11$             ;;BRANCH IF YES
(1)  011350  016677  000002  167562        MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  011356  062706  000006          11$:  ADD     #6,SP           ;;CLEAR UP STACK
(1)  011362  104401  001165          14$:  TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  011366  123727  001135  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  011374  001003                        BNE     15$             ;;BRANCH IF NOT
(1)  011376  012777  000100  167540        MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  011404  000002          15$:  RTI                             ;;RETURN
(1)  011406  004737  014472          16$:  JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  011412  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
(1)  011416  002420                        BLT     18$             ;;BRANCH IF YES
(1)  011420  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
(1)  011424  003015                        BGT     18$             ;;BRANCH IF YES
(1)  011426  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  011432  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  011436  001403                        BEQ     17$             ;;BRANCH IF YES
(1)  011440  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  011442  006316                        ASL     (SP)            ;;    CHAR OVER TO MAKE
(1)  011444  006316                        ASL     (SP)            ;;    ROOM FOR NEW ONE.
(1)  011446  005266  000002          17$:  INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  011452  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  011456  000667                        BR      7$              ;;GET THE NEXT ONE
(1)  011460  104401  001164          18$:  TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  011464  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)  011466  011646                     $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
(1)  011470  016666  000004  000002             MOV     4(SP),2(SP)     ;;THE PS
(1)  011476  005066  000004                     CLR     4(SP)           ;;GET READY FOR A CHARACTER
(2)  011502  005046                             CLR     -(SP)           ;;PUT NEW PS ON STACK
(2)  011504  012746  011512                     MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
(2)  011510  000002                             RTI                     ;;POP NEW PC AND PS
(2)  011512                             64$:
(1)  011512  005737  010570             1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
(1)  011516  001775                             BEQ     1$
(1)  011520  005337  010570                     DEC     $TKCNT          ;;DECREMENT THE COUNTER
(1)  011524  117766  177044  000004             MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
(1)  011532  005237  010574                     INC     $TKQOUT         ;;UPDATE THE POINTER
(1)  011536  023727  010574  010636             CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
(1)  011544  001003                             BNE     2$              ;;BRANCH IF NO
(1)  011546  012737  010576  010574             MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
(1)  011554  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)  011556  010346                     $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  011560  005046                             CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  011562  012703  012012             1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  011566  022703  012052             2$:     CMP     #$TTYIN+32.,R3  ;;BUFFER FULL?
(1)  011572  101456                             BLOS    4$              ;;BR IF YES
(1)  011574  104411                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  011576  112613                             MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  011600  122713  000177             10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  011604  001022                             BNE     5$              ;;BR IF NO
(1)  011606  005716                             TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  011610  001007                             BNE     6$              ;;BR IF NO
(1)  011612  112737  000134  012010             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  011620  104401  012010                     TYPE    ,9$
(1)  011624  012716  177777                     MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  011630  005303                     6$:     DEC     R3              ;;BACKUP BY ONE
(1)  011632  020327  012012                     CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  011636  103434                             BLO     4$              ;;BR IF YES
(1)  011640  111337  012010                     MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  011644  104401  012010                     TYPE    ,9$             ;;GO TYPE
(1)  011650  000746                             BR      2$              ;;GO READ ANOTHER CHAR.
(1)  011652  005716                     5$:     TST     (SP)            ;;RUBOUT KEY SET?
(1)  011654  001406                             BEQ     7$              ;;BR IF NO
(1)  011656  112737  000134  012010             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  011664  104401  012010                     TYPE    ,9$
(1)  011670  005016                             CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  011672  122713  000025             7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  011676  001003                             BNE     8$              ;;BR IF NO
(1)  011700  104401  012063                     TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U''
(1)  011704  000726                             BR      1$              ;;GO START OVER
```

```
(1)  011706  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A ''^R''?
(1)  011712  001011                          BNE     3$              ;;BRANCH IF NO
(1)  011714  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
(1)  011716  104401  001165                  TYPE    ,$CRLF          ;;TYPE A ''CR'' & ''LF''
(1)  011722  104401  012012                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  011726  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  011730  104401  001164          4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  011734  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  011736  111337  012010          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  011742  104401  012010                  TYPE    ,9$
(1)  011746  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  011752  001305                          BNE     2$              ;;LOOP IF NOT RETURN
(1)  011754  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  011760  104401  001166                  TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  011764  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  011766  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  011770  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  011772  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  012000  012766  012012  000004          MOV     #$TTYIN,4(SP)
(1)  012006  000002                          RTI                     ;;RETURN
(1)  012010     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  012011     000                  .BYTE   0                       ;;TERMINATOR
(1)  012012  000040          $TTYIN: .BLKB   32.                     ;;RESERVE 32. BYTES FOR TTY INPUT
(1)  012052  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
(1)  012056  041536  005015     000  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL ''C''
(1)  012063     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL ''U''
(1)  012070  043536  005015     000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL ''G''
(1)  012075     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  012102  036440  000040
(1)  012106  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
(1)  012114  020075     000
(1)          012120                  .EVEN
919                                  .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)                                  ;;*****************************************************************
(1)                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)                                  ;*AND LOAD THE TEST NUMBER($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)  012120                          $SCOPE:
(1)  012120  104410                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
(2)  012122  004737  003346                  JSR     PC,CTRLCG
(1)  012126  032777  040000  167004  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
(1)  012134  001114                          BNE     $OVER           ;;YES IF SW14=1
(1)                                  ;#####START OF CODE FOR THE XOR TESTER#####
(1)  012136  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
(1)                                                                  ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
(1)  012140  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  012144  012737  012164  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
```

```
(1)  012152  005737  177060            TST     @#177060          ;;TIME OUT ON XOR?
(1)  012156  012637  000004            MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
(1)  012162  000463                    BR      $SVLAD            ;;GO TO THE NEXT TEST
(1)  012164  022626                5$:  CMP     (SP)+,(SP)+       ;;CLEAR THE STACK AFTER A TIME OUT
(1)  012166  012637  000004            MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
(1)  012172  000423                    BR      7$                ;;LOOP ON THE PRESENT TEST
(1)  012174                        6$:;#####END OF CODE FOR THE XOR TESTER#####
(1)  012174  032777  000400  166736    BIT     #BIT08,@SWR       ;;LOOP ON SPEC. TEST?
(1)  012202  001404                    BEQ     2$                ;;BR IF NO
(1)  012204  127737  166730  001102    CMPB    @SWR,$TSTNM       ;;ON THE RIGHT TEST?    SWR<7:0>
(1)  012212  001465                    BEQ     $OVER             ;;BR IF YES
(1)  012214  105737  001103        2$: TSTB    $ERFLG            ;;HAS AN ERROR OCCURRED?
(1)  012220  001421                    BEQ     3$                ;;BR IF NO
(1)  012222  123737  001115  001103    CMPB    $ERMAX,$ERFLG     ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  012230  101015                    BHI     3$                ;;BR IF NO
(1)  012232  032777  001000  166700    BIT     #BIT09,@SWR       ;;LOOP ON ERROR?
(1)  012240  001404                    BEQ     4$                ;;BR IF NO
(1)  012242  013737  001110  001106 7$: MOV     $LPERR,$LPADR     ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  012250  000446                    BR      $OVER
(1)  012252  105037  001103        4$: CLRB    $ERFLG            ;;ZERO THE ERROR FLAG
(1)  012256  005037  001160            CLR     $TIMES            ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  012262  000415                    BR      1$                ;;ESCAPE TO THE NEXT TEST
(1)  012264  032777  004000  166646 3$: BIT     #BIT11,@SWR       ;;INHIBIT ITERATIONS?
(1)  012272  001011                    BNE     1$                ;;BR IF YES
(1)  012274  005737  001176            TST     $PASS             ;;IF FIRST PASS OF PROGRAM
(1)  012300  001406                    BEQ     1$                ;;      INHIBIT ITERATIONS
(1)  012302  005237  001104            INC     $ICNT             ;;INCREMENT ITERATION COUNT
(1)  012306  023737  001160  001104    CMP     $TIMES,$ICNT      ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)  012314  002024                    BGE     $OVER             ;;BR IF MORE ITERATION REQUIRED
(1)  012316  012737  000001  001104 1$: MOV     #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
(1)  012324  013737  012402  001160    MOV     $MXCNT,$TIMES     ;;SET NUMBER OF ITERATIONS TO DO
(1)  012332  105237  001102        $SVLAD: INCB $TSTNM            ;;COUNT TEST NUMBERS
(1)  012336  113737  001102  001174    MOVB    $TSTNM,$TESTN     ;;SET TEST NUMBER IN APT MAILBOX
(1)  012344  011637  001106            MOV     (SP),$LPADR       ;;SAVE SCOPE LOOP ADDRESS
(1)  012350  011637  001110            MOV     (SP),$LPERR       ;;SAVE ERROR LOOP ADDRESS
(1)  012354  005037  001162            CLR     $ESCAPE           ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  012360  112737  000001  001115    MOVB    #1,$ERMAX         ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  012366  013777  001102  166546 $OVER: MOV  $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER
(1)  012374  013716  001106            MOV     $LPADR,(SP)       ;;FUDGE RETURN ADDRESS
(1)  012400  000002                    RTI                       ;;FIXES PS
(1)  012402  003720                $MXCNT: 2000.                 ;;MAX. NUMBER OF ITERATIONS
```

```
                                        ;SUBROUTINE TO DETERMINE UNIT #
012404  012737  000000  001532  WHICHU: MOV     #0,UNITBD               ;PRIME THE VALUE
012412  013737  001460  012434          MOV     MASKNM,11$              ;LOAD VALUE
012420  006237  012434          10$:    ASR     11$                     ;SHIFT
012424  001404                          BEQ     12$                     ;BR WHEN DONE
012426  005237  001532                  INC     UNITBD                  ;UPDATE BIT
012432  000772                          BR      10$
012434  000000                  11$:    0
012436  000207                  12$:    RTS     PC                      ;EXIT
                                .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)  012440                     $ERROR:
(1)  012440  104410                     CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(3)  012442  004737  003346              JSR     PC,CTRLCG               ;TEST FOR CTRL C/G
(3)  012446  053737  001460  001530      BIS     MASKNM,BADUNT           ;INCIDATE BAD BIT
(3)  012454  004737  012404              JSR     PC,WHICHU               ;DETERMINE UNIT #
(1)  012460  105237  001103          7$: INCB    $ERFLG          ;;SET THE ERROR FLAG
(1)  012464  001775                     BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
(1)  012466  013777  001102  166446      MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  012474  005237  001112              INC     $ERTTL          ;;INC THE ERROR COUNT
(1)  012500  011637  001116              MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  012504  162737  000002  001116      SUB     #2,$ERRPC
(1)  012512  117737  166400  001114      MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  012520  032777  020000  166412      BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
(1)  012526  001004                      BNE     20$             ;;SKIP TYPEOUTS
(1)  012530  004737  012642              JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(1)  012534  104401  001165              TYPE    ,$CRLF
(1)  012540                         20$:
(1)  012540  122737  000001  001210      CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  012546  001007                      BNE     2$              ;;NO,SKIP APT ERROR REPORT
(1)  012550  113737  001114  012562      MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1)  012556  004737  013506              JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
(1)  012562     000                 21$: .BYTE   0
(1)  012563     000                      .BYTE   0
(1)  012564  000777                 22$: BR      22$             ;;APT ERROR LOOP
(1)  012566  005777  166346          2$: TST     @SWR            ;;HALT ON ERROR
(1)  012572  100002                      BPL     3$              ;;SKIP IF CONTINUE
(1)  012574  000000                      HALT                    ;;HALT ON ERROR!
(1)  012576  104410                      CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(1)  012600  032777  001000  166332  3$: BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
(1)  012606  001402                      BEQ     4$              ;;BR IF NO
(1)  012610  013716  001110              MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
(1)  012614  005737  001162          4$: TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
(1)  012620  001402                      BEQ     5$              ;;BR IF NONE
```

```
(1)  012622  013716  001162                    MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  012626                              5$:
(1)  012626  022737  010234  000042            CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
(1)  012634  001001                            BNE     6$              ;;BRANCH IF NO
(1)  012636  000000                            HALT                    ;;YES
(1)  012640                              6$:
(1)  012640  000002                            RTI                     ;;RETURN
 937                                    .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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)  012642                             $ERRTYP:
(1)  012642  104401  001165                    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  012646  010046                            MOV     R0,-(SP)        ;;SAVE R0
(1)  012650  005000                            CLR     R0              ;;PICKUP THE ITEM INDEX
(1)  012652  153700  001114                    BISB    @#$ITEMB,R0
(1)  012656  001004                            BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                                    ;;TYPE THE PC OF THE ERROR
(2)  012660  013746  001116                    MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                                    ;;ERROR ADDRESS
(2)  012664  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  012666  000445                            BR      10$             ;;GET OUT
(1)  012670  005300                      1$:   DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1)  012672  006300                            ASL     R0              ;;       WORK FOR THE ERROR TABLE
(1)  012674  006300                            ASL     R0
(1)  012676  006300                            ASL     R0
(1)  012700  062700  001254                    ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(1)  012704  012037  012714                    MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
(1)  012710  001404                            BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1)  012712  104401                            TYPE                    ;;TYPE THE "ERROR MESSAGE"
(1)  012714  000000                      2$:   .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
(1)  012716  104401  001165                    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  012722  012037  012732              3$:   MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
(1)  012726  001404                            BEQ     5$              ;;SKIP TYPEOUT IF 0
(1)  012730  104401                            TYPE                    ;;TYPE THE "DATA HEADER"
(1)  012732  000000                      4$:   .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
(1)  012734  104401  001165                    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  012740  010146                      5$:   MOV     R1,-(SP)        ;;SAVE R1
(1)  012742  012001                            MOV     (R0)+,R1        ;;PICKUP "DATA TABLE" POINTER
(1)  012744  001415                            BEQ     9$              ;;BR IF NO DATA TO BE TYPED
(1)  012746  012000                            MOV     (R0)+,R0        ;;PICKUP "DATA FORMAT" POINTER
(1)  012750  105720                      6$:   TSTB    (R0)+           ;;"OCTAL" OR "DECIMAL"
(1)  012752  001003                            BNE     7$              ;;BR IF DECIMAL
(2)  012754  013146                            MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(2)  012756  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  012760  000402                            BR      8$
(1)  012762                              7$:
(2)  012762  013146                            MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(2)  012764  104405                            TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1)  012766  005711                      8$:   TST     (R1)            ;;IS THERE ANOTHER NUMBER?
(1)  012770  001403                            BEQ     9$              ;;BR IF NO
(1)  012772  104401  013012                    TYPE    ,11$            ;;TYPE TWO(2) SPACES
(1)  012776  000764                            BR      6$              ;;LOOP
```

```
(1)
(1)  013000  012601                  9$:     MOV     (SP)+,R1        ;;RESTORE R1
(1)  013002  012600                  10$:    MOV     (SP)+,R0        ;;RESTORE R0
(1)  013004  104401  001165                  TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
(1)  013010  000207                          RTS     PC              ;;RETURN
(1)  013012  020040      000         11$:    .ASCIZ  / /             ;;TWO(2) SPACES
(1)          013016                          .EVEN
938                                  .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)  013016  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  013022  116637  000001  013241         MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  013030  112637  013243                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  013034  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  013040  000406                          BR      $TYPON
(1)  013042  112737  000001  013241  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  013050  112737  000006  013243         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  013056  112737  000005  013240  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  013064  010346                          MOV     R3,-(SP)        ;;SAVE R3
(1)  013066  010446                          MOV     R4,-(SP)        ;;SAVE R4
(1)  013070  010546                          MOV     R5,-(SP)        ;;SAVE R5
(1)  013072  113704  013243                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  013076  005404                          NEG     R4
(1)  013100  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  013104  110437  013242                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  013110  113704  013241                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  013114  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  013120  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  013122  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO ''C''
(1)  013124  000404                          BR      3$              ;;GO DO MSB
(1)  013126  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  013130  006105                          ROL     R5
(1)  013132  006105                          ROL     R5
(1)  013134  010503                          MOV     R5,R3
```

```
(1)  013136  006103                    3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  013140  105337  013242                    DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  013144  100016                            BPL     7$              ;;BR IF NO
(1)  013146  042703  177770                    BIC     #177770,R3      ;;GET RID OF JUNK
(1)  013152  001002                            BNE     4$              ;;TEST FOR 0
(1)  013154  005704                            TST     R4              ;;SUPPRESS THIS 0?
(1)  013156  001403                            BEQ     5$              ;;BR IF YES
(1)  013160  005204                    4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  013162  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  013166  052703  000040            5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  013172  110337  013236                    MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  013176  104401  013236                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  013202  105337  013240            7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  013206  003347                            BGT     2$              ;;BR IF MORE TO DO
(1)  013210  002402                            BLT     6$              ;;BR IF DONE
(1)  013212  005204                            INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  013214  000744                            BR      2$              ;;GO DO THE LAST DIGIT
(1)  013216  012605                    6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  013220  012604                            MOV     (SP)+,R4        ;;RESTORE R4
(1)  013222  012603                            MOV     (SP)+,R3        ;;RESTORE R3
(1)  013224  016666  000002  000004            MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  013232  012616                            MOV     (SP)+,(SP)
(1)  013234  000002                            RTI                     ;;RETURN
(1)  013236     000                    8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  013237     000                            .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  013240     000                    $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  013241     000                    $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  013242  000000                    $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 939                                   .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)                                    ;;**********************************************************************
(1)                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)                                    ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)                                    ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)                                    ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)                                    ;*REPLACED WITH SPACES.
(1)                                    ;*CALL:
(1)                                    ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
(1)                                    ;*      TYPDS                   ;;GO TO THE ROUTINE
(1)
(1)  013244                            $TYPDS:
(3)  013244  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  013246  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  013250  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  013252  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  013254  010546                            MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  013256  012746  020200                    MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  013262  016605  000020                    MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  013266  100004                            BPL     1$              ;;BR IF INPUT IS POS.
(1)  013270  005405                            NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  013272  112766  000055  000001            MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  013300  005000                    1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  013302  012703  013460                    MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  013306  112723  000040                    MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  013312  005002                    2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  013314  016001  013450                    MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
```

```
(1)  013320  160105                3$:   SUB     R1,R5             ;;FORM THIS BCD DIGIT
(1)  013322  002402                      BLT     4$                ;;BR IF DONE
(1)  013324  005202                      INC     R2                ;;INCREASE THE BCD DIGIT BY 1
(1)  013326  000774                      BR      3$
(1)  013330  060105                4$:   ADD     R1,R5             ;;ADD BACK THE CONSTANT
(1)  013332  005702                      TST     R2                ;;CHECK IF BCD DIGIT=0
(1)  013334  001002                      BNE     5$                ;;FALL THROUGH IF.0
(1)  013336  105716                      TSTB    (SP)              ;;STILL DOING LEADING 0'S?
(1)  013340  100407                      BMI     7$                ;;BR IF YES
(1)  013342  106316                5$:   ASLB    (SP)              ;;MSD?
(1)  013344  103003                      BCC     6$                ;;BR IF NO
(1)  013346  116663  000001 177777       MOVB    1(SP),-1(R3)      ;;YES--SET THE SIGN
(1)  013354  052702  000060         6$:   BIS     #'0,R2            ;;MAKE THE BCD DIGIT ASCII
(1)  013360  052702  000040         7$:   BIS     #' ,R2            ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  013364  110223                      MOVB    R2,(R3)+          ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  013366  005720                      TST     (R0)+             ;;JUST INCREMENTING
(1)  013370  020027  000010              CMP     R0,#10            ;;CHECK THE TABLE INDEX
(1)  013374  002746                      BLT     2$                ;;GO DO THE NEXT DIGIT
(1)  013376  003002                      BGT     8$                ;;GO TO EXIT
(1)  013400  010502                      MOV     R5,R2             ;;GET THE LSD
(1)  013402  000764                      BR      6$                ;;GO CHANGE TO ASCII
(1)  013404  105726                8$:   TSTB    (SP)+             ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  013406  100003                      BPL     9$                ;;BR IF NO
(1)  013410  116663  177777 177776       MOVB    -1(SP),-2(R3)     ;;YES--SET THE SIGN FOR TYPING
(1)  013416  105013                9$:   CLRB    (R3)              ;;SET THE TERMINATOR
(3)  013420  012605                      MOV     (SP)+,R5          ;;POP STACK INTO R5
(3)  013422  012603                      MOV     (SP)+,R3          ;;POP STACK INTO R3
(3)  013424  012602                      MOV     (SP)+,R2          ;;POP STACK INTO R2
(3)  013426  012601                      MOV     (SP)+,R1          ;;POP STACK INTO R1
(3)  013430  012600                      MOV     (SP)+,R0          ;;POP STACK INTO R0
(1)  013432  104401  013460              TYPE    ,$DBLK            ;;NOW TYPE THE NUMBER
(1)  013436  016666  000002 000004       MOV     2(SP),4(SP)       ;;ADJUST THE STACK
(1)  013444  012616                      MOV     (SP)+,(SP)
(1)  013446  000002                      RTI                       ;;RETURN TO USER
(1)  013450  023420               $DTBL:  10000.
(1)  013452  001750                       1000.
(1)  013454  000144                       100.
(1)  013456  000012                       10.
(1)  013460  000004               $DBLK:  .BLKW   4
940                                .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                ;;**********************************************************************
(1)  013470  112737  000001 013734 $ATY1:  MOVB    #1,$FFLG          ;;TO REPORT FATAL ERROR
(1)  013476  112737  000001 013732 $ATY3:  MOVB    #1,$MFLG          ;;TO TYPE A MESSAGE
(1)  013504  000403                       BR      $ATYC
(1)  013506  112737  000001 013734 $ATY4:  MOVB    #1,$FFLG          ;;TO ONLY REPORT FATAL ERROR
(1)  013514                        $ATYC:
(3)  013514  010046                       MOV     R0,-(SP)          ;;PUSH R0 ON STACK
(3)  013516  010146                       MOV     R1,-(SP)          ;;PUSH R1 ON STACK
(1)  013520  105737  013732               TSTB    $MFLG             ;;SHOULD TYPE A MESSAGE?
(1)  013524  001450                       BEQ     5$                ;;IF NOT:  BR
(1)  013526  122737  000001 001210       CMPB    #APTENV,$ENV      ;;OPERATING UNDER APT?
(1)  013534  001031                       BNE     3$                ;;IF NOT:  BR
(1)  013536  132737  000100 001211       BITB    #APTSPOOL,$ENVM   ;;SHOULD SPOOL MESSAGES?
(1)  013544  001425                       BEQ     3$                ;;IF NOT:  BR
(1)  013546  017600  000004               MOV     @4(SP),R0         ;;GET MESSAGE ADDR.
```

```
(1)  013552  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  013560  005737  001170            1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  013564  001375                           BNE     1$              ;;IF NOT:  WAIT
(1)  013566  010037  001204                   MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  013572  105720                    2$:    TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  013574  001376                           BNE     2$
(1)  013576  163700  001204                   SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  013602  006200                           ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  013604  010037  001206                   MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  013610  012737  000004  001170           MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  013616  000413                           BR      5$
(1)  013620  017637  000004  013644    3$:    MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  013626  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  013634  013746  177776                   MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  013640  004737  014260                   JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  013644  000000                    4$:    .WORD   0
(1)  013646                            5$:
(1)  013646  105737  013734            10$:   TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  013652  001416                           BEQ     12$             ;;IF NOT:  BR
(1)  013654  005737  001210                   TST     $ENV            ;;RUNNING UNDER APT?
(1)  013660  001413                           BEQ     12$             ;;IF NOT:  BR
(1)  013662  005737  001170            11$:   TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  013666  001375                           BNE     11$             ;;IF NOT:  WAIT
(1)  013670  017637  000004  001172           MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  013676  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  013704  005237  001170                   INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  013710  105037  013734            12$:   CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  013714  105037  013733                   CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  013720  105037  013732                   CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  013724  012601                           MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  013726  012600                           MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  013730  000207                           RTS     PC              ;;RETURN
(1)  013732     000                $MFLG:     .BYTE   0               ;;MESSG. FLAG
(1)  013733     000                $LFLG:     .BYTE   0               ;;LOG FLAG
(1)  013734     000                $FFLG:     .BYTE   0               ;;FATAL FLAG
(1)          013736                                .EVEN
(1)          000200                    APTSIZE=200
(1)          000001                    APTENV=001
(1)          000100                    APTSPOOL=100
(1)          000040                    APTCSUP=040
```

```
942
943                                     .SBTTL  POWER DOWN AND UP ROUTINES
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;POWER DOWN ROUTINE
(1)  013736  012737  014102  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1)  013744  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
(3)  013752  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  013754  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  013756  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  013760  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  013762  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
(3)  013764  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(3)  013766  017746  165146                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
(1)  013772  010637  014106                  MOV     SP,$SAVR6       ;;SAVE SP
(1)  013776  012737  014010  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
(1)  014004  000000                          HALT
(1)  014006  000776                          BR      .-2             ;;HANG UP
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;POWER UP ROUTINE
(1)  014010  012737  014102  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1)  014016  013706  014106                  MOV     $SAVR6,SP       ;;GET SP
(1)  014022  005037  014106                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
(1)  014026  005237  014106          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
(1)  014032  001375                          BNE     1$              ;;OF  WORD
(3)  014034  012677  165100                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
(3)  014040  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  014042  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
(3)  014044  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  014046  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  014050  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  014052  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  014054  012737  013736  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1)  014062  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
(1)  014070  104401                          TYPE                    ;;REPORT THE POWER FAILURE
(1)  014072  014110                  $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
(1)  014074  012716                          MOV     (PC)+,(SP)      ;;RESTART AT BEGIN
(1)  014076  001542                  $PWRAD: .WORD   BEGIN           ;;RESTART ADDRESS
(1)  014100  000002                          RTI
(1)  014102  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
(1)  014104  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
(1)  014106  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
944  014110  005015  042522  052123  PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
     014116  051101  044524  043516
     014124  040440  052106  051105
     014132  040440  050040  053517
     014140  051105  043040  044501
     014146  052514  042522  005015
     014154     000
945          014156                      .EVEN
```

```
947
(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)  014156  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1)  014160  016666  000004  000002  MOV     4(SP),2(SP)     ;;INPUT NUMBER
(3)  014166  010046          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  014170  010146          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  014172  010246          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(1)  014174  104412          1$:     RDLIN           ;;READ AN ASCIZ LINE
(1)  014176  012600          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
(1)  014200  005001          CLR     R1              ;;CLEAR DATA WORD
(1)  014202  005002          CLR     R2
(1)  014204  112046          2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
(1)  014206  001412          BEQ     3$              ;;IF ZERO GET OUT
(1)  014210  006301          ASL     R1              ;;*2
(1)  014212  006102          ROL     R2
(1)  014214  006301          ASL     R1              ;;*4
(1)  014216  006102          ROL     R2
(1)  014220  006301          ASL     R1              ;;*8
(1)  014222  006102          ROL     R2
(1)  014224  042716  177770  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
(1)  014230  062601          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
(1)  014232  000764          BR      2$              ;;LOOP
(1)  014234  005726          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
(1)  014236  010166  000012  MOV     R1,12(SP)       ;;SAVE THE RESULT
(1)  014242  010237  014256  MOV     R2,$HIOCT
(3)  014246  012602          MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  014250  012601          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  014252  012600          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  014254  000002          RTI                     ;;RETURN
(1)  014256  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
                        ;
                        ;;*****************************************************************
                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                        ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                        ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                        ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                        ;*
                        ;*CALL:
                        ;*1) USING A TRAP INSTRUCTION
                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                        ;*OR
                        ;*      TYPE
                        ;*      MESADR
                        ;*
                        ;
014260  105737  001157  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
014264  100002                 BPL     1$              ;;BR IF YES
014266  000000                 HALT                    ;;HALT HERE IF NO TERMINAL
014270  000430                 BR      3$              ;;LEAVE
014272  010046          1$:    MOV     R0,-(SP)        ;;SAVE R0
014274  017600  000002         MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
014300  122737  000001  001210 CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
014306  001011                 BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
014310  132737  000100  001211 BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
014316  001405                 BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
014320  010037  014330         MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
014324  004737  013476         JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
014330  000000          61$:   .WORD   0               ;;MESSAGE ADDRESS
014332  132737  000040  001211 62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
014340  001003                 BNE     60$             ;;YES,SKIP TYPE OUT
014342  112046          2$:    MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
014344  001005                 BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
014346  005726                 TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
014350  012600          60$:   MOV     (SP)+,R0        ;;RESTORE R0
014352  062716  000002         3$:    ADD     #2,(SP)         ;;ADJUST RETURN PC
014356  000002                 RTI                     ;;RETURN
014360  122716  000011         4$:    CMPB    #HT,(SP)        ;;BRANCH IF <HT>
014364  001430                 BEQ     8$
014366  122716  000200         CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
014372  001006                 BNE     5$
014374  005726                 TST     (SP)+           ;;POP  <CR><LF> EQUIV
014376  104401                 TYPE                    ;;TYPE A CR AND LF
014400  001165                 $CRLF
014402  105037  014536         CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
014406  000755                 BR      2$              ;;GET NEXT CHARACTER
014410  004737  014472  5$:    JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
014414  123726  001156  6$:    CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
014420  001350                 BNE     2$              ;;IF NO GO GET NEXT CHAR.
014422  013746  001154         MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                       ;;AND THE NULL CHAR.
014426  105366  000001  7$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
014432  002770                 BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

```
(1)  014434  004737  014472                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  014440  105337  014536                    DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  014444  000770                            BR      7$              ;;LOOP
(1)
(1)                                     ;HORIZONTAL TAB PROCESSOR
(1)
(1)  014446  112716  000040            8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  014452  004737  014472            9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  014456  132737  000007  014536            BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  014464  001372                            BNE     9$              ;;TAB STOP
(1)  014466  005726                            TST     (SP)+           ;;POP SPACE OFF STACK
(1)  014470  000724                            BR      2$              ;;GET NEXT CHARACTER
(1)  014472  105777  164452            $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  014476  100375                            BPL     $TYPEC
(1)  014500  116677  000002  164444            MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  014506  122766  000015  000002            CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  014514  001003                            BNE     1$              ;;BRANCH IF NO
(1)  014516  105037  014536                    CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  014522  000406                            BR      $TYPEX          ;;EXIT
(1)  014524  122766  000012  000002  1$:      CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  014532  001402                            BEQ     $TYPEX          ;;BRANCH IF YES
(1)  014534  105227                            INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  014536  000000                    $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  014540  000207                    $TYPEX: RTS     PC
(1)
952                                     .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)  014542  010146                    $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
(1)  014544  016601  000006                    MOV     6(SP),R1        ;;GET THE INPUT NUMBER
(1)  014550  000261                            SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1)  014552  112737  000060  014614    1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
(1)  014560  006101                            ROL     R1              ;;GET THIS BIT
(1)  014562  001406                            BEQ     2$              ;;DONE?
(1)  014564  105537  014614                    ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1)  014570  104401  014614                    TYPE    ,$BIN           ;;GO TYPE THIS BIT
(1)  014574  000241                            CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
(1)  014576  000765                            BR      1$              ;;GO DO THE NEXT BIT
(1)  014600  012601                    2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
(1)  014602  016666  000002  000004            MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  014610  012616                            MOV     (SP)+,(SP)
(1)  014612  000002                            RTI                     ;;RETURN TO USER
(1)  014614     000     000            $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
```

```
954
955                                     .SBTTL  TRAP DECODER
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE ''TRAP'' INSTRUCTION
(1)                                     ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)                                     ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)                                     ;*GO TO THAT ROUTINE.
(1)
(1)  014616  010046                     $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
(1)  014620  016600  000002                     MOV     2(SP),R0        ;;GET TRAP ADDRESS
(1)  014624  005740                             TST     -(R0)           ;;BACKUP BY 2
(1)  014626  111000                             MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
(1)  014630  006300                             ASL     R0              ;;POSITION FOR INDEXING
(1)  014632  016000  014652                     MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
(1)  014636  000200                             RTS     R0              ;;GO TO ROUTINE
(1)
(1)
(1)                                     ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
(1)
(1)  014640  011646                     $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
(1)  014642  016666  000004  000002             MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
(1)  014650  000002                             RTI                     ;;RESTORE THE PSW
(1)
(3)                                     .SBTTL  TRAP TABLE
(3)
(3)                                     ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)                                     ;*BY THE ''TRAP'' INSTRUCTION.
(3)
(3)                                     ;       ROUTINE
(3)                                     ;       -------
(3)  014652  014640                     $TRPAD: .WORD   $TRAP2
(3)  014654  014260                             $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3)  014656  013042                             $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)  014660  013016                             $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)  014662  013056                             $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)  014664  013244                             $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
(3)  014666  014542                             $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
(1)
(3)  014670  011214                             $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
(1)
(3)  014672  011124                             $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
(3)  014674  011466                             $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
(3)  014676  011556                             $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
(3)  014700  014156                             $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.

014702  011637  015154          IOTRD:  MOV     (6),TRTO        ;GET WHERE WE CAME TO.
014706  162737  000004  015154          SUB     #4,TRTO         ;FORM READ ADDR.

014714  023727  015154  001000          CMP     TRTO,#1000      ;DID TRAP FROM LESS THAN ADDR. 1000?
014722  003402                          BLE     2$              ;NO-CONTINUE.

014724  000000                  1$:     HALT                    ;A BUSS ERROR TIME OUT TRAP BROUGHT US HERE.
                                        ;ADDRESS CONTAINED IN TRTO.

014726  000776                          BR      1$              ;DONT'T ALLOW CONTINUE.

014730  016637  000004  015156  2$:     MOV     4(6),TRFRO      ;GET TRAPPED FROM ADDR.
014736  062706  000004                  ADD     #4,SP           ;ADJUST THE STACK POINTER

014742  122737  000037  001102          CMPB    #37,$TSTNM      ;LESS THAN INTERRUPT TESTS?
014750  003402                          BLE     3$              ;NO MUST BE WRONG VECTOR.
                                ;$$$$$$$$$$$$$$$$$$$$$$$$$>>>ERROR<<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
014752  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.

                                ;$$$$$$$$$$$$$$$$$$$$$$$$>>>ERROR<<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
014754  000002                        RTI
014756  022626                3$:     CMP     (SP)+,(SP)+     ;CLEAN OFF STACK
014760  005737  001176                TST     $PASS           ;IS THIS THE FIRST PASS?
014764  001025                        BNE     4$              ;;NO, DON'T REPORT
014766  104401  017756                TYPE,   VTMSG           ;TELL OPER.
014772  013746  001202                MOV     $UNIT,-(SP)     ;GET UNIT #
014776  104405                        TYPDS
015000  000240                        NOP
015002  000240                        NOP
015004  104401  020011                TYPE    ,VTMSG3         ;TYPE 'EXPECTED INTR. AT ''
015010  013746  001430                MOV     DIDINV,-(SP)    ;;SAVE DIDINV FOR TYPEOUT
(1)     015014  104403                TYPOS                   ;;GO TYPE--OCTAL ASCII
(1)     015016     003                .BYTE   3               ;;TYPE 3 DIGIT(S)
(1)     015017     001                .BYTE   1               ;;TYPE LEADING ZEROS
015020  104401  020041                TYPE    ,VTMSG1         ;TYPE '' RECEIVED INTR. AT ''
015024  013746  015154                MOV     TRTO,-(SP)      ;;SAVE TRTO FOR TYPEOUT
(1)     015030  104403                TYPOS                   ;;GO TYPE--OCTAL ASCII
(1)     015032     003                .BYTE   3               ;;TYPE 3 DIGIT(S)
(1)     015033     001                .BYTE   1               ;;TYPE LEADING ZEROS
015034  104401  020071                TYPE    ,VTMSG2         ;TYPE 'RESTARTING TEST''
015040  013777  001432  164362  4$:   MOV     DIDINS,@DIDINV  ;RESTORE INCORRECT VECTOR
015046  013777  001436  164360        MOV     DIEINS,@DIEINV
015054  012777  004700  164354        MOV     #4700,@DIEINS
015062  012777  004700  164342        MOV     #4700,@DIDINS   ;LOCATION WITH .+2 + 4700
015070  013737  015154  001430        MOV     TRTO,DIDINV     ;SET UP ACTUAL VECTOR LOCATION
015076  042737  000007  001430        BIC     #7,DIDINV       ;CLR LOW BITS
015104  013737  001430  001432        MOV     DIDINV,DIDINS   ;SET UP NEW VECTOR VALUES
015112  013737  001430  001434        MOV     DIDINV,DIEINV
015120  013737  001430  001436        MOV     DIDINV,DIEINS
015126  062737  000002  001432        ADD     #2,DIDINS       ;
015134  062737  000004  001434        ADD     #4,DIEINV
015142  062737  000006  001436        ADD     #6,DIEINS
015150  000177  163732                JMP     @$LPADR         ;START TEST OVER AGAIN.
015154  000000                TRTO:   .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTERRUPTED TO.
015156  000000                TRFRO:  .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTR. FROM.
```

```
1056                                        .SBTTL  ASCII MESSAGES
1057  015160     015     012                PRIME0: .BYTE   15,12
1058  015162  020114  020075  047514                .ASCII  \L = LOGIC TEST WITH OR WITHOUT TEST MODULE CONNECTED\
      015170  044507  020103  042524
      015176  052123  053440  052111
      015204  020110  051117  053440
      015212  052111  047510  052125
      015220  052040  051505  020124
      015226  047515  052504  042514
      015234  041440  047117  042516
      015242  052103  042105
1059  015246     015     012                        .BYTE   15,12
1060  015250  020127  020075  051127                .ASCII  \W = WRAPAROUND LOGIC TEST WITH MNCDO CONNECTED TO MNCDI\
      015256  050101  051101  052517
      015264  042116  046040  043517
      015272  041511  052040  051505
      015300  020124  044527  044124
      015306  046440  041516  047504
      015314  041440  047117  042516
      015322  052103  042105  052040
      015330  020117  047115  042103
      015336     111
1061  015337     015     012                        .BYTE   15,12
1062  015341     124  036440  052040                .ASCII  \T = TYPEOUT LOOP FOR MNCDI TEST MODULE SWITCHES\
      015346  050131  047505  052125
      015354  046040  047517  020120
      015362  047506  020122  047115
      015370  042103  020111  042524
      015376  052123  046440  042117
      015404  046125  020105  053523
      015412  052111  044103  051505
1063  015420     015     012                        .BYTE   15,12
1064  015422  020107  020075  042507                .ASCII  \G = GET NEW SWITCH REGISTER VALUE\
      015430  020124  042516  020127
      015436  053523  052111  044103
      015444  051040  043505  051511
      015452  042524  020122  040526
      015460  052514     105
1065  015463     015     012                        .BYTE   15,12
1066  015465     102  036440  024040                .ASCII  \B = (DIGITAL IN) BASE OR VECTOR ADDRESS CHANGES\
      015472  044504  044507  040524
      015500  020114  047111  020051
      015506  040502  042523  047440
      015514  020122  042526  052103
      015522  051117  040440  042104
      015530  042522  051523  041440
      015536  040510  043516  051505
1067  015544     015     012                        .BYTE   15,12
1068  015546  020117  020075  042050                .ASCII  \O = (DIGITAL OUT) BASE ADDRESS CHANGE\
      015554  043511  052111  046101
      015562  047440  052125  020051
      015570  040502  042523  040440
      015576  042104  042522  051523
      015604  041440  040510  043516
      015612     105
1069  015613     015     012                        .BYTE   15,12
```

```
1070  015615  000110  036440  044040          .ASCIZ  \H = HELP THE OPERATOR AND RETYPE THIS LIST    \
      015622  046105  020120  044124
      015630  020105  050117  051105
      015636  052101  051117  040440
      015644  042116  051040  052105
      015652  050131  020105  044124
      015660  051511  046040  051511
      015666  020124  020040  000040
1071  015674  000015  000012          DOT:    .BYTE   15,12
1072  015676  054524  042520  052040          .ASCIZ  /TYPE THE ''TEST CHARACTER'' THEN DEPRESS 'RETURN KEY''  /
      015704  042510  021040  042524
      015712  052123  041440  040510
      015720  040522  052103  051105
      015726  020042  044124  047105
      015734  042040  050105  042522
      015742  051523  021040  042522
      015750  052524  047122  045440
      015756  054505  020042  000040
1073
1074  015764  047115  042103  020117  EM1:    .ASCIZ  \MNCDO (DIGITAL OUT)    DOES NOT EXIST  <BUS ERROR>  CHECK ADDRESS SWITC
      015772  042050  043511  052111
      016000  046101  047440  052125
      016006  004451  047504  051505
      016014  047040  052117  042440
      016022  044530  052123  020040
      016030  041074  051525  042440
      016036  051122  051117  020076
      016044  041440  042510  045503
      016052  040440  042104  042522
      016060  051523  051440  044527
      016066  041524  042510  000123
1075  016074  047115  042103  020117  EM2:    .ASCIZ  \MNCDO (DIGITAL OUT)    DATA REGISTER ERROR\
      016102  042050  043511  052111
      016110  046101  047440  052125
      016116  004451  040504  040524
      016124  051040  043505  051511
      016132  042524  020122  051105
      016140  047522  000122
1076  016144  047115  042103  020117  EM3:    .ASCIZ  \MNCDO (DIGITAL OUT)    STATUS REGISTER ERROR\
      016152  042050  043511  052111
      016160  046101  047440  052125
      016166  004451  052123  052101
      016174  051525  051040  043505
      016202  051511  042524  020122
      016210  051105  047522  000122
1077  016216  047115  042103  020117  EM4:    .ASCIZ  \MNCDO (DIGITAL OUT)    INTERRUPT ERROR\
      016224  042050  043511  052111
      016232  046101  047440  052125
      016240  004451  047111  042524
      016246  051122  050125  020124
      016254  051105  047522  000122
1078  016262  047115  042103  020111  EM5:    .ASCIZ  \MNCDI (DIGITAL IN)     DOES NOT EXIST  <BUS ERROR> CHECK ADDRESS SWITCH
      016270  042050  043511  052111
      016276  046101  044440  024516
      016304  042011  042517  020123
      016312  047516  020124  054105
```

```
                016320  051511  004524  041074
                016326  051525  042440  051122
                016334  051117  020076  044103
                016342  041505  020113  042101
                016350  051104  051505  020123
                016356  053523  052111  044103
                016364  051505     000
1079            016367     115  041516  044504  EM6:    .ASCIZ  \MNCDI (DIGITAL IN)     STIMULUS REGISTER ERROR\
                016374  024040  044504  044507
                016402  040524  020114  047111
                016410  004451  052123  046511
                016416  046125  051525  051040
                016424  043505  051511  042524
                016432  020122  051105  047522
                016440  000122
1080            016442  047115  042103  020111  EM7:    .ASCIZ  \MNCDI (DIGITAL IN)     STATUS REGISTER ERROR\
                016450  042050  043511  052111
                016456  046101  044440  024516
                016464  051411  040524  052524
                016472  020123  042522  044507
                016500  052123  051105  042440
                016506  051122  051117     000
1081            016513     115  041516  044504  EM10:   .ASCIZ  \MNCDI (DIGITAL IN)     DATA REGISTER ERROR\
                016520  024040  044504  044507
                016526  040524  020114  047111
                016534  004451  040504  040524
                016542  051040  043505  051511
                016550  042524  020122  051105
                016556  047522  000122
1082            016562  047115  042103  020111  EM11:   .ASCIZ  \MNCDI (DIGITAL IN)     INTERRUPT ERROR\
                016570  042050  043511  052111
                016576  046101  044440  024516
                016604  044411  052116  051105
                016612  052522  052120  042440
                016620  051122  051117     000
1083            016625     115  041516  044504  EM12:   .ASCIZ  /MNCDI MNCDO    DIGITAL INPUT-OUTPUT WRAPAROUND STATUS ERROR/
                016632  046440  041516  047504
                016640  042011  043511  052111
                016646  046101  044440  050116
                016654  052125  047455  052125
                016662  052520  020124  051127
                016670  050101  051101  052517
                016676  042116  051440  040524
                016704  052524  020123  051105
                016712  047522  000122
1084            016716  047115  042103  020111  EM13:   .ASCIZ  /MNCDI MNCDO    DIGITAL INPUT-OUTPUT WRAPAROUND DATA ERROR/
                016724  047115  042103  004517
                016732  044504  044507  040524
                016740  020114  047111  052520
                016746  026524  052517  050124
                016754  052125  053440  040522
                016762  040520  047522  047125
                016770  020104  040504  040524
                016776  042440  051122  051117
                017004     000
1085            017005     115  041516  044504  EM14:   .ASCIZ  /MNCDI  INCORRECT I.D. VALUE FOR MNCDI/
```

```
                017012  044411  041516  051117
                017020  042522  052103  044440
                017026  042056  020056  040526
                017034  052514  020105  047506
                017042  020122  047115  042103
                017050  000111
1086            017052  047115  042103  020111  EM15:   .ASCIZ  /MNCDI (DIGITAL IN)     ILLEGAL INTR. OR TRAP/
                017060  042050  043511  052111
                017066  046101  044440  024516
                017074  044411  046114  043505
                017102  046101  044440  052116
                017110  027122  047440  020122
                017116  051124  050101     000
1087            017123     115  041516  044504  EM16:   .ASCIZ  /MNCDI (DIGITAL IN)     EXISTING MNCDI FAILED TO RESPOND/
                017130  024040  044504  044507
                017136  040524  020114  047111
                017144  004451  054105  051511
                017152  044524  043516  046440
                017160  041516  044504  043040
                017166  044501  042514  020104
                017174  047524  051040  051505
                017202  047520  042116     000
1088            017207     200  051120  043517  FOUND1: .ASCIZ  <200>/PROGRAM DETECTED/
                017214  040522  020115  042504
                017222  042524  052103  042105
                017230     000
1089            017231     040  047115  042103  FOUND2: .ASCIZ  \ MNCDI (DIGITAL IN)'S   \
                017236  020111  042050  043511
                017244  052111  046101  044440
                017252  024516  051447  020040
                017260  000040
1090            017262  047125  052111  042411  DH1:    .ASCIZ  /UNIT   ERRPC   OUTCSR  OUTDAT/
                017270  051122  041520  047411
                017276  052125  051503  004522
                017304  052517  042124  052101
                017312     000
1091            017313     125  044516  004524  DH5:    .ASCIZ  /UNIT   ERRPC   IN CSR  IN DIR  IN SBR/
                017320  051105  050122  004503
                017326  047111  041440  051123
                017334  044411  020116  044504
                017342  004522  047111  051440
                017350  051102     000
1092            017353     125  044516  004524  DH6:    .ASCIZ  /UNIT   ERRPC   IN SBR  GOOD    BAD/
                017360  051105  050122  004503
                017366  047111  051440  051102
                017374  043411  047517  004504
                017402  041040  042101     000
1093            017407     125  044516  004524  DH7:    .ASCIZ  /UNIT   ERRPC   IN CSR  GOOD    BAD/
                017414  051105  050122  004503
                017422  047111  041440  051123
                017430  043411  047517  004504
                017436  041040  042101     000
1094            017443     125  044516  004524  DH10:   .ASCIZ  /UNIT   ERRPC   IN DIR  GOOD    BAD/
                017450  051105  050122  004503
                017456  047111  042040  051111
                017464  043411  047517  004504
```

```
                017472  040502  000104
1095            017476  047125  052111  042411  DH11:   .ASCIZ  /UNIT   ERRPC   IN CSR/
                017504  051122  041520  044411
                017512  020116  051503  000122
1096            017520  047125  052111  042411  DH12:   .ASCIZ  /UNIT   ERRPC   OUTCSR  IN CSR  GOOD     BAD/
                017526  051122  041520  047411
                017534  052125  051503  004522
                017542  047111  041440  051123
                017550  043411  047517  004504
                017556  041040  042101     000
1097            017563     125  044516  004524  DH13:   .ASCIZ  /UNIT   ERROC   OUTDOR  IN DIR  GOOD     BAD/
                017570  051105  047522  004503
                017576  052517  042124  051117
                017604  044411  020116  044504
                017612  004522  047507  042117
                017620  020011  040502  000104
1098            017626  047125  052111  042411  DH15:   .ASCIZ  /UNIT   ERRPC     TO    FROM ADRS./
                017634  051122  041520  020011
                017642  052040  004517  051106
                017650  046517  040440  051104
                017656  027123     000
1099            017661     125  044516  004524  DH16:   .ASCIZ  /UNIT   ERRPC    WERE   ARE/
                017666  051105  050122  020103
                017674  020040  053440  051105
                017702  020105  020040  051101
                017710  000105
1100            017712  035440  047524  040524  ERRTOT: .ASCIZ  / ;TOTAL ERROR COUNT = /
                017720  020114  051105  047522
                017726  020122  047503  047125
                017734  020124  020075     000
1101            017741     040  041073  042101  MESGD:  .ASCIZ  / ;BAD UNITS /
                017746  052440  044516  051524
                017754  000040
1102            017756  046600  041516  044504  VTMSG:  .ASCIZ  <200>/MNCDI (DIGITAL IN) UNIT #/
                017764  024040  044504  044507
                017772  040524  020114  047111
                020000  020051  047125  052111
                020006  021440     000
1103            020011     200  054105  042520  VTMSG3: .ASCIZ  <200>/EXPECTED INTERRUPT AT /
                020016  052103  042105  044440
                020024  052116  051105  052522
                020032  052120  040440  020124
                020040     000
1104            020041     040  042522  042503  VTMSG1: .ASCIZ  / RECEIVED INTERRUPT AT /
                020046  053111  042105  044440
                020054  052116  051105  052522
                020062  052120  040440  020124
                020070     000
1105            020071     200  046120  040505  VTMSG2: .ASCII  <200>/PLEASE CHECK VECTOR SWITCHES/<200>
                020076  042523  041440  042510
                020104  045503  053040  041505
                020112  047524  020122  053523
                020120  052111  044103  051505
                020126     200
1106            020127     011  042522  052123          .ASCIZ  /       RESTARTING LOGIC TEST/
                020134  051101  044524  043516
```

```
                020142  046040  043517  041511
                020150  052040  051505  000124
1107            020156     015     012          SETUP0: .BYTE   15,12
1108            020160  047105  052523  042522          .ASCII  \ENSURE MNCDI (DIGITAL IN) DATA SWITCH IS IN THE ''-'' POSITION\
                020166  046440  041516  044504
                020174  024040  044504  044507
                020202  040524  020114  047111
                020210  020051  040504  040524
                020216  051440  044527  041524
                020224  020110  051511  044440
                020232  020116  044124  020105
                020240  026442  020042  047520
                020246  044523  044524  047117
1109            020254     015     012     000          .BYTE   15,12,0
1110            020257     103  047117  042516  SETUP1: .ASCII  \CONNECT MNCDI (DIGITAL IN) TO THE TESTER\
                020264  052103  046440  041516
                020272  044504  024040  044504
                020300  044507  040524  020114
                020306  047111  020051  047524
                020314  052040  042510  052040
                020322  051505  042524     122
1111            020327     015     012     000          .BYTE   15,12,0
1112            020332     015     012          SETUP2: .BYTE   15,12
1113            020334  047115  042103  020111          .ASCII  \MNCDI (DIGITAL IN) MUST BE CONNECTED TO MNCDO (DIGITAL OUT)\
                020342  042050  043511  052111
                020350  046101  044440  024516
                020356  046440  051525  020124
                020364  042502  041440  047117
                020372  042516  052103  042105
                020400  052040  020117  047115
                020406  042103  020117  042050
                020414  043511  052111  046101
                020422  047440  052125     051
1114            020427     015     012     000          .BYTE   15,12,0
1115            020432  005015  047115  042103  ADROUT: .ASCIZ  <15><12>/MNCDO (DIGITAL OUT) BUS ADDRESS </
                020440  020117  042050  043511
                020446  052111  046101  047440
                020454  052125  020051  052502
                020462  020123  042101  051104
                020470  051505  020123  000074
1116            020476  005015  047115  042103  ADRIN:  .ASCIZ  <15><12>/MNCDI (DIGITAL IN) BUS ADDRESS </
                020504  020111  042050  043511
                020512  052111  046101  044440
                020520  024516  041040  051525
                020526  040440  042104  042522
                020534  051523  036040     000
1117            020541     015  046412  041516  VECIN:  .ASCIZ  <15><12>/MNCDI (DIGITAL IN) VECTOR ADDRESS </
                020546  044504  024040  044504
                020554  044507  040524  020114
                020562  047111  020051  042526
                020570  052103  051117  040440
                020576  042104  042522  051523
                020604  036040     000
1118            020607     076  037440  000040  ENDOUT: .ASCIZ  /> ? /
1119            020614  026440  000040          ADASH:  .ASCIZ  / - /
1120            020620     015     012     000  TCRLF:  .BYTE   15,12,0
```

```
1121            020624                                  .EVEN
1122  020624    001532    001116    001404    DT1:      UNITBD,$ERRPC,OCSR,DOR,0
      020632    001410    000000
1123  020636    001532    001116    001414    DT5:      UNITBD,$ERRPC,ICSR,DIR,SBR,0
      020644    001420    001424    000000
1124  020652    001532    001116    001424    DT6:      UNITBD,$ERRPC,SBR,$GDDAT,$BDDAT,0
      020660    001124    001126    000000
1125  020666    001532    001116    001414    DT7:      UNITBD,$ERRPC,ICSR,$GDDAT,$BDDAT,0
      020674    001124    001126    000000
1126  020702    001532    001116    001420    DT10:     UNITBD,$ERRPC,DIR,$GDDAT,$BDDAT,0
      020710    001124    001126    000000
1127  020716    001532    001116    001414    DT11:     UNITBD,$ERRPC,ICSR,0
      020724    000000
1128  020726    001532    001116    001404    DT12:     UNITBD,$ERRPC,OCSR,ICSR,$GDDAT,$BDDAT,0
      020734    001414    001124    001126
      020742    000000
1129  020744    001532    001116    001410    DT13:     UNITBD,$ERRPC,DOR,DIR,$GDDAT,$BDDAT,0
      020752    001420    001124    001126
      020760    000000
1130  020762    001532    001116    015154    DT15:     UNITBD,$ERRPC,TRTO,TRFRO,0
      020770    015156    000000
1131  020774    001532    001116    001442    DT16:     UNITBD,$ERRPC,TEMP,$UNIT,0
      021002    001202    000000
1132  021006      000       000       000     DF0:      .BYTE   0,0,0,0,0,0,0,0,0,0,0,0
      021011      000       000       000
      021014      000       000       000
      021017      000       000
1133
1134            000001                                  .END
```

```
ABASE = 171160              17#     61
ACDW1 = 171260              19#     61
ACDW2 = 000000              61
ACPUOP= 000000              61
ADASH   020614             899    1119#
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   020476             379    1116#
ADROUT  020432             366    1115#
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             940#    951
APTENV= 000001             936     940#    951
APTSIZ= 000200             219     940#
APTSPO= 000100             940#    951
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*    882     936*
```

```
BASEXC  003252            253    378#
BASEXD  003210            266    365#
BEGIN   001542             48    214#   943
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    726
BIT0  = 000001             20#   323    442    444    578    738    758    781
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#   919
BIT09 = 001000             20#   919    936
BIT1  = 000002             20#   448    465    521    522    531    532    543    585    605    623    645    696
                          699    737    741    757
BIT10 = 002000             20#
BIT11 = 004000             20#   209    919
BIT12 = 010000             20#   208    458    605    821
BIT13 = 020000             20#   936
BIT14 = 040000             20#   460    645    919
BIT15 = 100000             20#   315    521    524    534    646    647    724    792    805    823    824    834
BIT2  = 000004             20#   449    465    512    514    599    709    712    785    792    806    822    823
BIT3  = 000010             20#   450
BIT4  = 000020             20#   452    465    560    579    594    605    757
BIT5  = 000040             20#   453    560    579    584    594    605    607    757
BIT6  = 000100             20#   242    445    454    461    503    547    598    623    873
BIT7  = 000200             20#   483    484    521    523    524    531    533    534    606    608    624    646
                          647    686    699    721    726    770    789    792    805    823    824    834
BIT8  = 000400             20#   456    822    823
BIT9  = 001000             20#   457
BPTVEC= 000014             20#
CKSWR = 104410            919    936    955#
CR    = 000015             20#   951
CRLF  = 000200             20#   227    951
CTRCHA  003422            403*   404*   405    409    413#
CTRLCG  003346            401#   885    894    919    936
DDISP = 177570             20#    61    219
DELAY0  010550            687    700    713    728    793    809    827    837    911#
DF0     021006             64     66     68     70     72     74     76     78     80     82     84   1132#
DH1     017262             64   1090#
DH10    017443             78   1094#
DH11    017476             80   1095#
DH12    017520             66   1096#
DH13    017563             68   1097#
DH15    017626             82   1098#
DH16    017661             84   1099#
DH5     017313             72   1091#
DH6     017353             74   1092#
DH7     017407             70     76   1093#
DIDATA  010354            269    889#
DIDINS  001432            172#   619*   635    636*   861*   862*  1039   1042*  1045*  1048*
DIDINV  001430            171#   339    618*   635*   860*   861    863    865   1035   1039*  1043*  1044*  1045
```

```
                        1046    1047
DIEINS  001436           175#    344     641*    658     659*    865*    866*   1040    1041*   1047*   1050*
DIEINV  001434           174#    640*    658*    863*    864*   1040*   1046*   1049*
DIR     001420           166#    433     511*    556     562     571     582     583*    586     595     600     610     681*
                         694*    707*    735*    745     755*    765     786*    804*    820*    833*    893*    897     900
                        1123    1126    1129
DIR1    001422           167#
DISPLA  001142            61#    219*    919*    936*
DISPRE  000174            44#    219
DOR     001410           160#    685*    698*    711*    720*    742*    760*    780*    783*    790*    803*    808*    819*
                         826*    835*   1122    1129
DOR1    001412           161#    851
DOT     015674           247    1071#
DSWR    = 177570          20#     61     219
DT1     020624            64    1122#
DT10    020702            78    1126#
DT11    020716            80    1127#
DT12    020726            66    1128#
DT13    020744            68    1129#
DT15    020762            82    1130#
DT16    020774            84    1131#
DT5     020636            72    1123#
DT6     020652            74    1124#
DT7     020666            70     76    1125#
DWARF   001440           177#    214*    217*    234     262*    272*    305     421     678     722
EMTVEC= 000030            20#    219*
EM1     015764            64    1074#
EM10    016513            78    1081#
EM11    016562            80    1082#
EM12    016625            66    1083#
EM13    016716            68    1084#
EM14    017005            70    1085#
EM15    017052            82    1086#
EM16    017123            84    1087#
EM2     016074          1075#
EM3     016144          1076#
EM4     016216          1077#
EM5     016262            72    1078#
EM6     016367            74    1079#
EM7     016442            76    1080#
ENDOUT  020607           369     382     391    1118#
ERRTOT  017712           876    1100#
ERRVEC= 000004            20#    219*    283*    301*    302*    431*    437*    438*    440*    441*    919*
EVER    001526           204#    245*    303     314*    315*    317     319     847
EXTMSG  010270           871     873#
FIXADR  003010           246     322     330#    374     396     890
FOUND1  017207           307    1088#
FOUND2  017231           310    1089#
GNS     = ****** U       227     955
GTSWR   = 104407         227     256     411     955#
HT      = 000011          20#    951
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*    688
```

```
                    696*  701   709*  714   726*  727*  737*  741*  743   757*  763   770*  785*
                    789*  794   805*  810   822*  824*  828   834*  838   892*  895   1123  1125
                   1127  1128
ICSR1   001416      164#  464*  535*
IOTRD   014702       42   301   437   440   977#
IOTVEC= 000020       20#  219*
LF    = 000012       20#  951
LOGIC   002520      230   263   280#  886
LOGICO  002514      273   279#
MASKNM  001460      185#  281*  323*  856*  879   929   936
MESGD   017741      881  1101#
MTEST   002274      235   240#  259
MTEST1  002300      232   241#  257   275   375   397   408   918
OCSR    001404      157#  330   684*  697*  710*  725*  729   736*  756*  849   1122  1128
OCSR1   001406      158#
PIRQ  = 177772       20#
PIRQVE= 000240       20#
PRIMEO  015160      240  1057#
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  014110      943   944#
PWRVEC= 000024       20#  219*  943*
RBEG0   001566      213   216   219#
RDCHR = 104411      918   955#
RDLIN = 104412      248   947   955#
RDOCT = 104413      370   383   392   955#
REMAIN  007772      439   680   844#
RESTRT  001534       49   212#
RESVEC= 000010       20#
RUNIT   002512      249*  250*  251   254   258   260   264   267   270   276#
SBR     001424      168#  434   442*  444*  445*  446*  510*  682*  695*  708*  784*  788*  825*
                   1123  1124
SBR1    001426      169#  446*  854
SETUP0  020156      233  1107#
SETUP1  020257      236  1110#
SETUP2  020332      279  1112#
STACK = 001100       20#  219   889
STKLMT= 177774       20#
SWR     001140       61#  219*  227   289   918*  919   936   943*
SWREG   000176       45#  219   227   918
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   020620              902     1120#
TEMP    001442              178#     319*   444*   578*   580    583    590*   595*   738*   740    750*   758*   760
                            761      771*   781*   788    790    801    815*   821*   825    826    835    842*   1131
TEMP1   001444              179#     801*   802*   808
TEMP2   001446              180#     212*   215*   218*   231
TESTER  001554               50      217#
TKVEC  = 000060              20#     918*
TPVEC  = 000064              20#
TRAPVE= 000034               20#     219*
TRFRO   015156              988*    1053#   1130
TRTO    015154              977*     978*    980    1037   1043   1052#  1130
TRTVEC= 000014               20#
TSTR0   001450              181#     237
TSTR2   001452              182#     423*
TSTR4   001454              183#     424
TSTR6   001456              184#
TST1    003424              326      420#    868
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   006664          690     693#
TST43   006752          703     706#
TST44   007040          716     719#
TST45   007134          731     734#
TST46   007250          754#
TST47   007370          775#
TST5    003720          445#
TST50   007612          818#
TST51   007772          845#
TST6    003764          445     446#
TST7    004102          448#
TYPBN = 104406          883     901     955#
TYPDS = 104405          309     871     878     937     955#    1031
TYPE  = 104401          227     233     236     240     247     274     279     307     310     365     369     378     382
                        387     391     871     876     881     884     891     899     902     918     936     937     938
                        939     943     951     952     955#    1029    1034    1036    1038
TYPOC = 104402          368     381     390     898     918     937     955#
TYPON = 104404          955#
TYPOS = 104403          955#    1035    1037
UNITBD  001532          206#    858     928*    932*    1122    1123    1124    1125    1126    1127    1128    1129    1130
                        1131
VADDR   001464          187#    285     853
VADDRO  001462          186#    850
VECIN   020541          388     1117#
VECLST  001466          188#    346     860
VECOFF  001506          196#    347
VTMSG   017756          1029    1102#
VTMSG1  020041          1036    1104#
VTMSG2  020071          1038    1105#
VTMSG3  020011          1034    1103#
WAITO   010442          891     905#
WHICHU  012404          857     928#    936
$APTHD  001000           60#
$ASTAT= ****** U        940
$ATYC   013514          940#
$ATY1   013470          940#
$ATY3   013476          940#    951
$ATY4   013506          936     940#
$AUTOB  001134           61#    227*    228     918
$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     688*
                        689     701*    702     714*    715     729*    730     745*    746     765*    766     794*    795
                        810*    811     828*    829     838*    839     1124    1125    1126    1128    1129
```

```
$BELL   012052            918#
$BIN    014614            952#*
$CDW1   001250             61#    237*    331     367     373*
$CDW2   001252             61#
$CHARC  014536            951#*
$CKSWR  011124            918#    955
$CMTAG  001100             61#    219
$CM3  = 000000             61#
$CNTLC  012056            918#
$CNTLG  012070            918#
$CNTLU  012063            918#
$CPUOP  001216             61#
$CRLF   001165             61#    918     936     937     951
$DBLK   013460            939#
$DEVCT  001200             61#
$DEVM   001246             61#
$DOAGN  010244            871#
$DTBL   013450            939#
$ENDAD  010234             58     227     871#    936
$ENDCT  010202            219     871#
$ENDMG  010253            871#
$ENULL  010250            871#    884
$ENV    001210             61#    227     287     936     940     951
$ENVM   001211             61#    219     940     951
$EOP    010146            300     313     871#
$EOPCT  010174            219*    871#
$ERFLG  001103             61#    919*    936*
$ERMAX  001115             61#    219*    919*
$ERROR  012440            219     936#
$ERRPC  001116             61#    936*    937     1122    1123    1124    1125    1126    1127    1128    1129    1130    1131
$ERRTB  001254             61#    937
$ERRTY  012642            936     937#
$ERTTL  001112             61#    244*    874     877     936*
$ESCAP  001162             61#    219*    919*    936
$ETABL  001210             61#
$ETEND  001254             60      61#
$FATAL  001172             61#    940*
$FFLG   013734            940#*
$FILLC  001156             61#    951
$FILLS  001155             61#    951
$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*    689     699*    702
                          712*    715     721*    724*    730     740*    742     746     761*    762*    766     792*    795
                          806*    811     823*    829     839     1124    1125    1126    1128    1129
$GET42  010224            871#
$GTSWR  011214            918#    955
$HD   = 000000             15
$HIBTS  001000             60#
$HIOCT  014256            947#*
$ICNT   001104             61#    919*
$ILLUP  014102            943#
$INTAG  001135             61#    918*
$ITEMB  001114             61#    936*    937
```

```
$LF     001166          61#   918   936   951
$LFLG   013733         940#*
$LPADR  001106          61#   219*  442*  444*  446*  617*  919*  1051
$LPERR  001110          61#   219*  577*  919*  936
$MADR1  001222          61#
$MADR2  001226          61#
$MADR3  001232          61#
$MADR4  001236          61#
$MAIL   001170          60    61#   219   227   919   936   951
$MAMS1  001220          61#
$MAMS2  001224          61#
$MAMS3  001230          61#
$MAMS4  001234          61#
$MBADR  001002          60#
$MFLG   013732         940#*
$MNEW   012106         918#
$MSGAD  001204          61#   940*
$MSGLG  001206          61#   940*
$MSGTY  001170          61#   940*
$MSWR   012075         918#
$MTYP1  001221          61#
$MTYP2  001225          61#
$MTYP3  001231          61#
$MTYP4  001235          61#
$MXCNT  012402         919#
$NULL   001154          61#   951
$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#  693#  706#  719#  734#  754#  775#
                       818#  845#
$OCNT   013240         938#*
$OMODE  013242         938#*
$OVER   012366         919#
$PASS   001176          61#   219*  243*  871*  919   1027
$PASTM  001006          60#
$PWRAD  014076         943#
$PWRDN  013736         219   943#
$PWRMG  014072         943#
$PWRUP  014010         943#
$QUES   001164          61#   274   918   936   951
$RDCHR  011466         918#  955
$RDDEC= ****** U       955
$RDLIN  011556         918#  955
$RDOCT  014156         947#  955
$RDSZ = 000040         918#
$RTNAD  010246         871#
$R2A  = ****** U       955
$SAVRE= ****** U       955
$SAVR6  014106         943#*
$SCOPE  012120         219   919#
$SETUP= 000137         211#  219   227   871   918   919   936
$STUP = 177777         211#
$SVLAD  012332         919#
$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     693
                        706     719     734     754     775     818     845     871     919     936     943
$SWREG  001212           61#     219
$SWRMK= 000000           22      919
$TESTN  001174           61#     919*
$TIMES  001160           61#     219*    445*    461*    499*    541*    593*    734*    754*    775*    845*    871*    919*
$TKB    001146           61#     403     918
$TKCNT  010570          918#*
$TKINT  010636          226     918#
$TKQEN= 010636          918#
$TKQIN  010572          918#*
$TKQOU  010574          918#*
$TKQSR  010576          918#
$TKS    001144           61#     242*    401     445*    461*    503*    547*    598*    873*    918*
$TKSRV  010706          918#
$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#    690     693#    703
                        706#    716     719#    731     734#    754#    775#    818#    845#
$TPB    001152           61#     951*
$TPFLG  001157           61#     951
$TPS    001150           61#     951
$TRAP   014616          219     955#
$TRAP2  014640          955#
$TRP  = 000014          955#
$TRPAD  014652          955#
$TSTM   001004           60#
$TSTNM  001102           61#     867*    871*    919*    936     991
$TTYIN  012012          918#
$TYPBN  014542          952#    955
$TYPDS  013244          939#    955
$TYPE   014260          222*    223*    224*    225*    940     951#    955
$TYPEC  014472          918     951#
$TYPEX  014540          951#
$TYPOC  013042          938#    955
$TYPON  013056          938#    955
$TYPOS  013016          938#    955
$UNIT   001202           61#    282*    286*    291     295     308     311     314     317     321*    846*    847    1030
                        1131
$UNITM  001010           60#
$USWR   001214           61#
$VECT1  001240           61#     341     351     389     395*
$VECT2  001242           61#
$XTSTR  012136          919#
$$GET4= 000000          871#
$OFILL  013241          938#*
$40CAT= ****** U        919     936
.       = 021021         26#     37      38#     41#     43#     47#     53#     58#     59#     60#     61#    219     871
                        909#    918#    919     936     937#    939#    940#    943     945#    951    1121#
.$ASTA= ****** U        940
.$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   691   704   717   732   748   768   797
        813   831   841   994
ESCAPE   20#
FIXUNT  922#   936
FLOATO  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   693   706   719   734   754   775   818   845
POP      20#   939   940   943   947
PUSH     20#   939   940   943   947
READRW  142#   448   449   450   452   453   454   456   457   458   460
REPORT   20#
RESETO  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   693   706   719   734   754   775   818   845   871
SETPRI   20#   918
SETTRA  955#
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   690   703   716   731   747   767   796   812   830
        840   848  1028
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   693   706   719   734   754   775   818   845
        871   918   919   936   937   938   939   940   943   947   951   952   955
SWRSU    20#   219#
TRMTRP  955#
TYPBIN   20#
TYPDEC   20#   871   937
TYPNAM   20#   227
TYPNUM   20#
TYPOCS   20#  1035  1037
TYPOCT   20#   918   937
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   693   706   719   734   754   775   818   845
$$SET   955#
$$SETM  219#
$$SKIP   20#   422   428   445   448   449   450   452   453   454   456   457   458   460   461
```

```
                468     477     487     496     506     517     527     538     550     573     601     612     690     703     716
                731
.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#     940
.$CATC          10#
.$CMTA          10#      61
.$EOP           10#     871
.$ERRO          10#     936
.$ERRT          12#     937
.$PARM          11#
.$POWE          11#     943
.$RDOC          12#     947
.$READ          11#     918
.$SAVE          11#
.$SCOP          11#     919
.$SPAC          11#
.$SWDO          11#
.$TRAP          11#     955
.$TYPB          12#     952
.$TYPD          12#     939
.$TYPE          10#      11#    951
.$TYPO          10#     938

. ABS.  021021     000     CON   RO    ABS   LCL   I

 ERRORS DETECTED:  0

 CVMNBB,CVMNBB/CRF=CVMNBB
 RUN-TIME: 26 13 1 SECONDS
 RUN-TIME RATIO: 166/41=3.9
 CORE USED:  26K  (51 PAGES)
```