MNCTP MINC-11 SIZER DIAG AH-B101C-MC
CVMNFCO FICHE 1 OF 1

IDENTIFICATION

PRODUCT CODE:	AC-B100C-MC
DIAGNOSTIC CODE:	MAINDEC-11-CVMNF-C
PRODUCT NAME:	CVMNFC MINC-11 SIZER
DATE:	AUG. 1981
MAINTAINER:	DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1.0 ABSTRACT

VERSION "C", INCLUDES SIZING FOR THE MNCTP (TC AMP) WHICH IS AN OPTION, SIMILAR TO THE MNCAM (MUX) AND MNCAG (PREAMP) TO THE MNCAD (A/D). THE PURPOSE OF THE MINC-11 SIZER PROGRAM IS TO VERIFY THE CONFIGURATION OF THE MINC-11 HARDWARE. THE PROGRAM IS TO BE USED TO VERIFY THE CORRECT POSITION OF ANALOG CHANNELS, DEVICE ADDRESS AND VECTOR SWITCHES. THE PROGRAM SIZES THE NUMBER OF MINC-11 OPTIONS AND REPORTS THEIR EXISTANCE, BUS AND VECTOR ADDRESSES. TWO ADDITIONAL REPORTS ARE MADE IF REQUESTED. THE FIRST IS THE MODE OF THE MNCAD (A/D) CHANNELS. THE SECOND IS THE ADDRESS USAGE MAP WHICH IS USEFUL IN DETECTING A INCORRECT ADDRESS SWITCH SETTING. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL VIA 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. MINC-11 SYSTEM

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 ITSE_F AND INFORM THE OPERATOR OF THE OPERATING OPTIONS THAT MAYBE SELECTED.
4. THE OPERATOR SHOULD TYPE "R VMNF??" 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 IF THE OPERATOR WANTS THE REPORT OF THE MODES OF THE MNCAD (A/D) CHANNELS.
3. THE PROGRAM WILL NOW ASK IF THE OPERATOR WANTS THE REPORT OF THE ADDRESS USAGE MAP.
4. THE PROGRAM WILL THEN SIZE THE MINC-11 SYSTEM AND REPORT THE DEVICE BUS AND VECTOR ADDRESSES AND SELECTED REPORTS.

4.1 PROGRAM START

200 STARTING ADDRESS OF THE PROGRAM

5.0 SOFTWARE SWITCH REGISTER

NONE

5.1 PROGRAM CONTROL

THE SIZING MAYBE STOPPED BY TYPING THE ''CONTROL & C'' KEYS. THIS OPERATION WILL STOP THE PROGRAM AND ENABLE THE OPERATOR TO SELECT DIFFERENT PROGRAM REPORT'S.

6.0 ERROR REPORTING

NONE, NO ERROR MESSAGES ARE REPORTED.

7.0 MISCELLANEOUS

7.1 MINC-11 BUS ADDRESS MODIFICATION

MODIFY LOCATION '$BASE' (LOC. 1240) IF BASE BUS ADDRESS IS NOT 171000.

7.2 XXDP/APT NOTES

THIS PROGRAM IS CHAINABLE UNDER XXDP (REQUIRES 8K OR MORE). WHEN RUN IN ''CHAIN'' MODE, THE PROGRAM WILL REQUIRE THE OPERATOR TO PERFORM INITIAL SETUP OPERATIONS. THE SIZER PROGRAM SHOULD NOT BE INCLUDED IN A ''CHAIN'' THAT LOOPS TO THE STARTING FILE. THIS PROGRAM 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 MINC-11 OPTION INTERFACE TESTING

THIS PROGRAM DOES ''AUTO-SIZE'' THE NUMBER OF MINC-11 OPTIONS CONNECTED.
THE PROGRAM WILL SIZE SEQUENTIALLY UP TO:

4	MNCAD (A/D)'S
8	MNCKW (CLOCK)'S
8	MNCAA (D/A)'S
8	MNCDO (DIGITAL OUT)'S
8	MNCDI (DIGITAL IN)'S

EACH MINC-11 OPTION MUST BE CONFIGURED WITH CONTIGUOUS BUS ADDRESSES.

7.5 RESTRICTIONS

ALL USER CONNECTIONS MUST BE REMOVED.

8.0 EXECUTION TIME

DEPENDANT UPON TERMINAL SPEED AND REPORTS SELECTED.

9.0 EXAMPLE REPORT

SEQ 0005

A "TYPICAL" SIZER REPORT FOR A SYSTEM WITH THE FOLLOWING:

```
171000           MNCAD            171020           MNCKW
171060           MNCAA            171160           MNCDI
171260           MNCDO            171420           IBV-11
173000-173776    BOOT-STRAP PROM
176500-176526    SLU 0, 1, AND 2
177170           RX02 DISK SUB-SYSTEM
177560-177566    CONSOLE TERMINAL
************************************************************

DO YOU WANT THE MNCAD (A/D) CHANNEL MODE REPORT ?
                 TYPE "Y" FOR YES = Y
DO YOU WANT THE MEMORY USAGE MAP REPORT ?
                 TYPE "Y" FOR YES = Y
  ;MNCAD AT ADDRESS = 171000    VECTOR = 400
                  0 -  7 SINGLE ENDED
                 10 - 13 DIFFERENTIAL
                 14 - 17 PREAMP
                 20 - 27 TC AMP

   ;MNCKW AT ADDRESS = 171020   VECTOR = 440
   ;MNCAA AT ADDRESS = 171060   VECTOR = ** DOES NOT EXIST **
   ;MNCDI AT ADDRESS = 171160   VECTOR = 120
   ;MNCDO AT ADDRESS = 171260   VECTOR = 340

# OF    ;MNCAD 1  ;MNCKW  1  ;MNCAA  1  ;MNCDI  1  ;MNCDO  1

MNC11 MEMORY USAGE MAP          (EACH BIT = 2 ADDRESSES)
 ADDR        000/400          100/500          200/600          300/700
170000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
170400 0000000000000000 0000000000000000 0000000000000000 0000000000000000
171000 1000100000001100 0000000000001100 0000000000001000 0000000000000000
171400 0000110000000000 0000000000000000 0000000000000000 0000000000000000
172000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
172400 0000000000000000 0000000000000000 0000000000000000 0000000000000000
173000 1111111111111111 1111111111111111 1111111111111111 1111111111111111
173400 1111111111111111 1111111111111111 1111111111111111 1111111111111111
174000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
174400 0000000000000000 0000000000000000 0000000000000000 0000000000000000
175000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
175400 0000000000000000 0000000000000000 0000000000000000 0000000000000000
176000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
176400 0000000000000000 1111110000000000 0000000000000000 0000000000000000
177000 0000000000000000 0000000000000010 0000000000000000 0000000000000000
177400 0000000000000000 0000000000001100 0000000000000000 0000000000000000

END PASS #     1
```

10. LISTING

ATTACHED

```
 12                                .TITLE  MAINDEC-11-CVMNF-C      MINC-11 OPTION SIZER PROGRAM
(1)                                ;*COPYRIGHT (C) 1981
(1)                                ;*DIGITAL EQUIPMENT CORP.
(1)                                ;*MAYNARD, MASS. 01754
(1)                                ;*
(1)                                ;*PROGRAM BY R SHOOP
(1)                                ;*
(1)                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)                                ;*PACKAGE (MAINDEC-11-DZQAC-C4), 31 JULY 1980.
(1)                                ;*
 13                                .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
 14          171000                  ABASE=  171000
 15          000100                          .=100
 16  000100  000104  000200  000002          .WORD   104,200,2       ;B EVENT VECTOR
 17
 18                                           .SBTTL  TRAP CATCHER
 19
 20          000000                          .=0
 21                                           ;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ".+2"
 22                                           ;*AND "JSR PC,R0" SEQUENCE TO CATCH ILLEGAL INTERRUPTS.
 23                                           ;*AND INTERRUPTS TO THE WRONG VECTOR.
 24                                           ;*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
 25                                           ;*VECTORS.
 35
 36          000200                          .=200
 37  000200  000137  001324                  JMP     BEGIN           ;START ADDRESS
```

```
 39                                  .SBTTL  ACT11 HOOKS
(1)
(2)                                  ;;*****************************************************************
(1)                                  ;HOOKS REQUIRED BY ACT11
(1)          000204                          $SVPC=.                 ;SAVE PC
(1)          000046                          .=46
(1)  000046  002654                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)          000052                          .=52
(1)  000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
(1)          000204                          .=$SVPC                 ;; RESTORE PC
 40          001000                  .=1000
 41                                  .SBTTL  APT PARAMETER BLOCK
(1)
(2)                                  ;;*****************************************************************
(1)                                  ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                  ;;*****************************************************************
(1)          001000                          .$X=.   ;;SAVE CURRENT LOCATION
(1)          000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1)  000024  000200                          200     ;;FOR APT START UP
(1)          000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1)  000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
(1)          001000                          .=.$X   ;;RESET LOCATION COUNTER
(2)                                  ;;*****************************************************************
(1)                                  ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)                                  ;INTERFACE SPEC.
(1)
(1)  001000                          $APTHD:
(1)  001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1)  001002  001164                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1)  001004  002260                  $TSTM:  .WORD   1200.   ;;RUN TIM OF LONGEST TEST
(1)  001006  000764                  $PASTM: .WORD   500.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1)  001010  003244                  $UNITM: .WORD   1700.   ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1)  001012  00003?                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
 42
(1)
(2)
(1)                                     ;;*****************************************************************
(1)                                     ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(1)                                     ;*USED IN THE PROGRAM.
(1)
(1)             001100                          .=1100
(1)  001100                             $CMTAG:                         ;;START OF COMMON TAGS
(1)  001100  000000                             .WORD   0
(1)  001102     000                     $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
(1)  001103     000                     $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
(1)  001104  000000                     $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
(1)  001106  000000                     $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
(1)  001110  000000                     $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
(1)  001112  000000                     $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
(1)  001114     000                     $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
(1)  001115     001                     $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
(1)  001116  000000                     $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
(1)  001120  000000                     $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
(1)  001122  000000                     $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
(1)  001124  000000                     $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
(1)  001126  000000                     $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
(1)  001130  000000                             .WORD   0               ;;RESERVED--NOT TO BE USED
(1)  001132  000000                             .WORD   0
(1)  001134     000                     $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
(1)  001135     000                     $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
(1)  001136  000000                             .WORD   0
(1)  001140  177570                     SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
(1)  001142  177570                     DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
(1)  001144  177560                     $TKS:   177560                  ;;TTY KBD STATUS
(1)  001146  177562                     $TKB:   177562                  ;;TTY KBD BUFFER
(1)  001150  177564                     $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
(1)  001152  177566                     $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
(1)  001154     000                     $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
(1)  001155     002                     $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
(1)  001156     012                     $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
(1)  001157     000                     $TPFLG: .BYTE   0               ;;''TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
(1)  001160     077                     $QUES:  .ASCII  /?/             ;;QUESTION MARK
(1)  001161     015                     $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
(1)  001162  000012                     $LF:    .ASCIZ  <12>            ;;LINE FEED
(2)                                     ;;*****************************************************************
(2)                                     .SBTTL  APT MAILBOX-ETABLE
(2)
(3)                                     ;;*****************************************************************
(2)                                     .EVEN
(2)  001164                             $MAIL:                          ;;APT MAILBOX
(2)  001164  000000                     $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
(2)  001166  000000                     $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
(2)  001170  000000                     $TESTN: .WORD   ATESTN          ;;TEST NUMBER
(2)  001172  000000                     $PASS:  .WORD   APASS           ;;PASS COUNT
(2)  001174  000000                     $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
(2)  001176  000000                     $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
(2)  001200  000000                     $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
(2)  001202  000000                     $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
(2)  001204                             $ETABLE:                        ;;APT ENVIRONMENT TABLE
(2)  001204     000                     $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
```

```
(2)  001205    000                $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001206  000000               $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001210  000000               $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001212  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)  001214    000                $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001215    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)  001216  000000               $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                               ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
(2)  001220    000                $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001221    000                $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001222  000000               $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001224    000                $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001225    000                $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001226  000000               $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001230    000                $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001231    000                $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001232  000000               $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001234  000000               $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001236  000000               $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001240  171000               $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001242  000000               $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001244  000000               $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001246                       $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)  001246                         $ERRTB:
 43
 44                                 ;NO ERRORS ARE REPORTED
 45
 46  001246  000020                 KWOFF:  20                      ;OFFSET FROM $BASE VALUE TO KW ADDRESS
 47  001250  000060                 AAOFF:  60                      ;OFFSET FROM $BASE VALUE TO AA ADDRESS
 48  001252  000160                 DIOFF:  160                     ;OFFSET FROM $BASE VALUE TO DI ADDRESS
 49  001254  000260                 DOOFF:  260                     ;OFFSET FROM $BASE VALUE TO DO ADDRESS
 50
 51                                 ;PROGRAM CREATES THESE ADDRESSES FROM THE VALUE OF $BASE PLUS THE OFFSET FOR THAT DEVICE
 52
 53  001256  171000                 ADBASE: 171000
 54  001260  171020                 KWBASE: 171020
 55  001262  171060                 AABASE: 171060
 56  001264  171160                 DIBASE: 171160
 57  001266  171260                 DOBASE: 171260
 58
 59                                 ;COUNT OF EACH UNIT FOUND
 60
 61  001270  000000                 ADCNT:  0
 62  001272  000000                 KWCNT:  0
 63  001274  000000                 AACNT:  0
 64  001276  000000                 DICNT:  0
 65  001300  000000                 DOCNT:  0
 66
 67                                 ;MISC. TEMP LOCATIONS
 68  001302  000000                 VADR:   0
 69  001304  000000                 VECT:   0
 70  001306  000000                 TEMP:   0
 71  001310  000000                 NBEXT:  0
 72  001312  010514                 OUTADR: ADRPOK          ;ADDRESS VALUE FOR MEMORY USAGE
 73  001314  000000                 FLAGMP: 0               ;NON-ZERO INDICATES MEMORY USAGE MAP IS WANTED
 74  001316  000000                 FLAGAD: 0               ;NON-ZERO INDICATES THE A/D CHANNEL MODES BE REPORTED
 75  001320  000000                 SWREG:  0               ;HERE BECAUSE OF SYSMAC
 76  001322  000000                 DISPRE: 0               ;HERE BECAUSE OF SYSMAC
```

```
                                        .SBTTL      INITIAL START-UP,HOUSEKEEPING, AND DIALOGUE
001324  000005                  BEGIN:  RESET
001326  012700  001000                  MOV     #BIT9,R0            ;LOAD DELAY COUNTER
001332  005300                  50$:    DEC     R0                  ;DELAY
001334  001376                          BNE     50$                 ; FOR RESET PULSE TO QUIET THE WORLD
                                        .SBTTL  INITIALIZE THE COMMON TAGS
(1)                                     ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1)     001336  012706  001100          MOV     #$CMTAG,R6          ;;FIRST LOCATION TO BE CLEARED
(1)     001342  005026                  CLR     (R6)+               ;;CLEAR MEMORY LOCATION
(1)     001344  022706  001140          CMP     #SWR,R6 ;;DONE?
(1)     001350  001374                  BNE     .-6                 ;;LOOP BACK IF NO
(1)     001352  012706  001100          MOV     #STACK,SP           ;;SETUP THE STACK POINTER
(1)                                     ;;INITIALIZE A FEW VECTORS
(1)     001356  012737  010436  000034  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1)     001364  012737  000340  000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
(1)     001372  013737  002622  002614  MOV     $ENDCT,$EOPCT       ;;SETUP END-OF-PROGRAM COUNTER
(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)     001400  013746  000004          MOV     @#ERRVEC,-(SP)      ;;SAVE ERROR VECTOR
(2)     001404  012737  001440  000004  MOV     #64$,@#ERRVEC       ;;SET UP ERROR VECTOR
(2)     001412  012737  177570  001140  MOV     #DSWR,SWR           ;;SETUP FOR A HARDWARE SWICH REGISTER
(2)     001420  012737  177570  001142  MOV     #DDISP,DISPLAY      ;;AND A HARDWARE DISPLAY REGISTER
(2)     001426  022777  177777  177504  CMP     #-1,@SWR            ;;TRY TO REFERENCE HARDWARE SWR
(2)     001434  001012                  BNE     66$                 ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                                                                 ;;AND  THE HARDWARE SWR IS NOT = -1
(2)     001436  000403                  BR      65$                 ;;BRANCH IF NO TIMEOUT
(2)     001440  012716  001446  64$:    MOV     #65$,(SP)           ;;SET UP FOR TRAP RETURN
(2)     001444  000002                  RTI
(2)     001446  012737  001320  001140  65$:    MOV     #SWREG,SWR          ;;POINT TO SOFTWARE SWR
(2)     001454  012737  001322  001142          MOV     #DISPREG,DISPLAY
(2)     001462  012637  000004          66$:    MOV     (SP)+,@#ERRVEC      ;;RESTORE ERROR VECTOR
(1)
(2)     001466  005037  001172          CLR     $PASS               ;;CLEAR PASS COUNT
(2)     001472  132737  000200  001205  BITB    #APTSIZE,$ENVM      ;;TEST USER SIZE UNDER APT
(2)     001500  001403                  BEQ     67$                 ;;YES,USE NON-APT SWITCH
(2)     001502  012737  001206  001140  MOV     #$SWREG,SWR         ;;NO,USE APT SWITCH REGISTER
(2)     001510                  67$:
 85     001510  012737  005046  007106  MOV     #5046,$TYPE         ;
 86     001516  012737  012746  007110  MOV     #12746,$TYPE+2
 87     001524  012737  007120  007112  MOV     #$TYPE+12,$TYPE+4
 88     001532  012737  000002  007114  MOV     #RTI,$TYPE+6
 89     001540  004737  006204          JSR     PC,$TKINT           ;INIT THE TKB
 90     001544  013737  001240  001256  MOV     $BASE,ADBASE        ;GET INITIAL BASE
 91     001552  013737  001256  001260  MOV     ADBASE,KWBASE       ;GET KW BASE
 92     001560  063737  001246  001260  ADD     KWOFF,KWBASE        ;UPDATE KW ADDRESS
 93     001566  013737  001256  001262  MOV     ADBASE,AABASE       ;GET AA BASE
 94     001574  063737  001250  001262  ADD     AAOFF,AABASE        ;UPDATE AA ADDRESS
 95     001602  013737  001256  001264  MOV     ADBASE,DIBASE       ;GET DI BASE
 96     001610  063737  001252  001264  ADD     DIOFF,DIBASE        ;UPDATE DI ADDRESS
 97     001616  013737  001256  001266  MOV     ADBASE,DOBASE       ;GET DO BASE
 98     001624  063737  001254  001266  ADD     DOOFF,DOBASE        ;UPDATE DO ADDRESS
 99     001632  005037  001270          CLR     ADCNT               ;CLEAR THE UNIT COUNTERS
100     001636  005037  001272          CLR     KWCNT
101     001642  005037  001274          CLR     AACNT
102     001646  005037  001276          CLR     DICNT
103     001652  005037  001300          CLR     DOCNT
```

```
                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001656  005227  177777                        INC     #-1                     ;;FIRST TIME?
001662  001036                                BNE     68$                     ;;BRANCH IF NO
001664  022737  002654  000042                CMP     #$ENDAD,@#42            ;;ACT-11?
001672  001432                                BEQ     68$                     ;;BRANCH IF YES
001674  104401  001702                        TYPE    ,69$                    ;;TYPE ASCIZ STRING
001700  000427                                BR      68$                     ;;GET OVER THE ASCIZ
                                        ;;69$:  .ASCIZ  <CRLF>#MD-11-CVMNF-C    MINC-11 OPTION SIZER PROGRAM#<CRLF>
001760                                  68$:
001760  105737  001205                        TSTB    $ENVM                   ;CHECK APT MODE?
001764  001042                                BNE     LOOP                    ;BR IF YES
001766  005737  000042                        TST     CHAIN                   ;CHECK XXDP/ACT MODE?
001772  001405                                BEQ     1$                      ;BR IF NOT
001774  104401  005025                        TYPE    ,SETMNC                 ;TELL OPER. ABOUT SETUP STUFF
002000  104407                                RDCHR                           ;GET A CHAR
002002  012600                                MOV     (SP)+,R0                ;READ WHATEVER CHARACTER
002004  000432                                BR      LOOP                    ;RUN TEST WHEN OPER IS READY
002006  104401  004423                  1$:   TYPE    ,QUESAD                 ;ASK OPERATOR ABOUT A/D MODE
002012  104407                                RDCHR                           ;GET OPER INPUT
002014  012600                                MOV     (SP)+,R0                ;
002016  042700  177600                        BIC     #177600,R0              ;MASK OFF JUNK
002022  005037  001316                        CLR     FLAGAD                  ;INIT THE EXPANDED A/D REPORT
002026  122700  000131                        CMPB    #'Y,R0                  ;CHECK IF 'Y'
002032  001002                                BNE     2$                      ;BR IF NOT
002034  005237  001316                        INC     FLAGAD                  ;SET EXPANDED A/D REPORT
002040  104401  004534                  2$:   TYPE    ,QUESMP                 ;ASK OPERATOR ABOUT MEMORY USAGE MAP
002044  104407                                RDCHR                           ;GET OPER INPUT
002046  012600                                MOV     (SP)+,R0                ;
002050  042700  177600                        BIC     #177600,R0              ;MASK OFF JUNK
002054  005037  001314                        CLR     FLAGMP                  ;INIT THE MEMORY USAGE REPORT
002060  122700  000131                        CMPB    #'Y,R0                  ;CHECK IF 'Y'
002064  001002                                BNE     LOOP                    ;BR IF NOT
002066  005237  001314                        INC     FLAGMP                  ;SET MEMORY USAGE FLAG

                                        ;NOW VERIFY THE MNCAD MODULES
002072  004537  002710                  LOOP: JSR     R5,TEST                 ;CHECK THE MNCAD'S
002076  003466                                ADGO                            ;ADDRESS OF INTR. STARTUP
002100  001256                                ADBASE                          ;STARTING A/D ADDRESS
002102  000004                                4                               ;OFFSET TO NEXT UNIT
002104  004160                                ADPRI                           ;A/D MESSAGE POINTER
002106  000004                                4                               ;MAX. # OF A/D'S ON A SYSTEM
002110  001270                                ADCNT                           ;# OF A/D'S FOUND ON SYSTEM

                                        ;NOW VERIFY THE MNCKW MODULES
002112  004537  002710                        JSR     R5,TEST                 ;CHECK THE MNCKW'S
002116  003504                                KWGO                            ;ADDRESS OF INTR. STARTUP
002120  001260                                KWBASE                          ;STARTING K/W ADDRESS
002122  000004                                4                               ;OFFSET TO NEXT UNIT
002124  004172                                KWPRI                           ;KW MESSAGE POINTER
002126  000010                                8.                              ;MAX. # OF KW ON A SYSTEM
002130  001272                                KWCNT                           ;# OF KW FOUND ON SYSTEM
```

```
                                ;NOW VERIFY THE MNCAA MODULES
002132  004537  002710                JSR     R5,TEST         ;CHECK THE MNCAA
002136  000000                        0                       ;NO INTR. STARTUP
002140  001262                        AABASE                  ;STARTING AA ADDRESS
002142  000010                        10                      ;OFFSET TO NEXT UNIT
002144  004204                        AAPRI                   ;AA MESSAGE POINTER
002146  000010                        8.                      ;MAX. # OF AA ON A SYSTEM
002150  001274                        AACNT                   ;# OF AA FOUND ON SYSTEM
                                ;NOW VERIFY THE MNCDI MODULES
002152  004537  002710                JSR     R5,TEST         ;CHECK THE MNCDI
002156  003544                        DIGO                    ;INTR. STARTUP ADDRESS
002160  001264                        DIBASE                  ;STARTING DI ADDRESS
002162  000010                        10                      ;OFFSET TO NEXT UNIT
002164  004216                        DIPRI                   ;DI MESSAGE POINTER
002166  000010                        8.                      ;MAX. # OF DI ON A SYSTEM
002170  001276                        DICNT                   ;# OF DI FOUND ON SYSTEM
                                ;NOW VERIFY THE MNCDO MODULES
002172  004537  002710                JSR     R5,TEST         ;CHECK THE MNCDO
002176  003570                        DOGO                    ;INTR. STARTUP ADDRESS
002200  001266                        DOBASE                  ;STARTING DO ADDRESS
002202  000004                        4                       ;OFFSET TO NEXT UNIT
002204  004230                        DOPRI                   ;DO MESSAGE POINTER
002206  000010                        8.                      ;MAX. # OF DO ON A SYSTEM
002210  001300                        DOCNT                   ;# OF DO FOUND ON SYSTEM

                                ;NOW REPORT THE TOTAL COUNT OF EACH MINC-11 MODULE

002212  104401  001161                TYPE    ,$CRLF          ;FRESH LINE
002216  104401  004271                TYPE    ,NIPRI          ;TELL THE #
002222  104401  004160                TYPE    ,ADPRI          ;A/D
002226  013746  001270                MOV     ADCNT,-(SP)
002232  104403                        TYPOS
002234      001     000               .BYTE   1,0
002236  104401  004172                TYPE    ,KWPRI          ;KW
002242  013746  001272                MOV     KWCNT,-(SP)
002246  104403                        TYPOS
002250      002     000               .BYTE   2,0
002252  104401  004204                TYPE    ,AAPRI          ;AA
002256  013746  001274                MOV     AACNT,-(SP)
002262  104403                        TYPOS
002264      002     000               .BYTE   2,0
002266  104401  004216                TYPE    ,DIPRI          ;DI
002272  013746  001276                MOV     DICNT,-(SP)
002276  104403                        TYPOS
002300      002     000               .BYTE   2,0
002302  104401  004230                TYPE    ,DOPRI          ;DO
002306  013746  001300                MOV     DOCNT,-(SP)
002312  104403                        TYPOS
002314      002     000               .BYTE   2,0
002316  000240                        NOP
002320  000240                        NOP
002322  000240                        NOP
002324  000240                        NOP
```

```
                        ;DETERMINE IF THE MEMORY USAGE MAP SHOULD BE REPORTED
002326  105737  001205           TSTB    $ENVM           ;TEST IF APT
002332  001006                   BNE     1$              ;BR IF YES
002334  005737  000042           TST     @#42            ;TEST IF XXDP
002340  001003                   BNE     1$              ;BR IF YES
002342  005737  001314           TST     FLAGMP          ;TEST IF OPER. ASKED FOR MAP
002346  001002                   BNE     2$              ;BR IF YES
002350  000137  002576   1$:     JMP     $EOP            ;BYPASS REPORT
002354  005037  001314   2$:     CLR     FLAGMP          ;ENSURE ONLY THE 1ST PASS IT'S REPORTED
002360  104401  004635           TYPE    ,MNCMAP         ;INFORM THE OPER. THE HEADER
002364  013746  000004           MOV     ERRVEC,-(SP)    ;SAVE CURRENT BUS TRAP VALUE
002370  012700  170000           MOV     #170000,R0      ;GET 1ST ADDRESS TO MAP
002374  010037  001312           MOV     R0,OUTADR       ;AND SAVE FOR TYPEOUT
002400  012701  100000           MOV     #BIT15,R1       ;PRIME THE ROTATING POINTER
002404  005002                   CLR     R2              ;CLEAR TEMP MASK
002406  012703  010514           MOV     #ADRPOK,R3      ;LOAD MAP BLOCK POINTER
002412  012737  002556  000004   MOV     #4$,ERRVEC      ;LOAD ADDRESS TIMEOUT RETURN
002420  005710           3$:     TST     (R0)            ;TEST THE ADDRESS
002422  050102                   BIS     R1,R2           ;IF NO TIMEOUT-MARK A 1
002424  062700  000002           ADD     #2,R0           ;UPDATE ADDRESS
002430  005710                   TST     (R0)            ;TEST THE ADDRESS
002432  050102                   BIS     R1,R2           ;IF NO TIMEOUT-MARK A 1
002434  062700  000002           ADD     #2,R0           ;UPDATE ADDRESS
002440  000257                   CCC                     ;ENSURE CLEAR CARRY
002442  006001                   ROR     R1              ;MOVE RIGHT
002444  103365                   BCC     3$              ;BR IF MORE BITS LEFT
002446  010223                   MOV     R2,(R3)+        ;SAVE IN MAP BLOCKS
002450  005002                   CLR     R2              ;CLEAR MASK
002452  012701  100000           MOV     #BIT15,R1       ;LOAD ROTATING BIT
002456  022703  010714           CMP     #ADRTOP,R3      ;CHECK IF MAP BLOCKS FILLED
002462  001356                   BNE     3$              ;BR IF NOT
                        ;NOW REPORT THE MEMORY MAP
002464  012700  010514           MOV     #ADRPOK,R0      ;LOAD MEMORY MAP POINTER
002470  104401  001161   5$:     TYPE    ,$CRLF
002474  013746  001312           MOV     OUTADR,-(SP)    ;REPORT ADDRESS
002500  104402                   TYPOC
002502  104401  004335           TYPE    ,SPACE1
002506  004737  002564           JSR     PC,OUTBIN       ;OUTPUT BINARY BITS
002512  004737  002564           JSR     PC,OUTBIN
002516  004737  002564           JSR     PC,OUTBIN
002522  004737  002564           JSR     PC,OUTBIN
002526  062737  000400  001312   ADD     #400,OUTADR     ;UPDATE ADDRESS VALUE
002534  022700  010714           CMP     #ADRTOP,R0      ;TEST IF FINISHED
002540  001353                   BNE     5$              ;BR IF NOT
002542  012637  000004           MOV     (SP)+,ERRVEC    ;RESTORE BUS TRAP VECTOR
002546  104401  001161           TYPE    ,$CRLF
002552  000137  002576           JMP     $EOP
                        ;RETURN TO HERE UPON MEMORY TIME-OUT
002556  062716  000002   4$:     ADD     #2,(SP)         ;ADJUST RETURN ADDRESS
002562  000002                   RTI
                        ;SUBROUTINE TO REPORT I/O FORMAT MESSAGES
002564  012046           OUTBIN: MOV     (R0)+,-(SP)
002566  104406                   TYPBN
002570  104401  004335           TYPE    ,SPACE1         ;INSERT A SPACE
002574  000207                   RTS     PC
```

```
                        .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 LOOP

002576                  $EOP:
002576  000240                  NOP
002600  005237  001172          INC     $PASS           ;;INCREMENT THE PASS NUMBER
002604  042737  100000  001172  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
002612  005327                  DEC     (PC)+           ;;LOOP?
002614  000001          $EOPCT: .WORD   1
002616  003022                  BGT     $DOAGN          ;;YES
002620  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
002622  000001          $ENDCT: .WORD   1
002624  002614                  $EOPCT
002626  104401  002673          TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
002632  013746  001172          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
002636  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
002640  104401  002670          TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
002644  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
002650  001405                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
002652  000005                  RESET                   ;;CLEAR THE WORLD
002654  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
002656  000240                  NOP                     ;;SAVE ROOM
002660  000240                  NOP                     ;;FOR
002662  000240                  NOP                     ;;ACT11
002664                  $DOAGN:
002664  000137                  JMP     @(PC)+          ;;RETURN
002666  002072          $RTNAD: .WORD   LOOP
002670     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
002673     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
002700  050040  051501  020123
002706  000043
```

```
                                ;SUBROUTINE TO VERIFY THE BASE AND VECTOR ADDRESSES

002710  012537  003232          TEST:   MOV     (R5)+,67$       ;GET INTR. STARTUP ADDRESS
002714  013537  001126                  MOV     @(R5)+,$BDDAT   ;GET ADDRESS
002720  012537  001302                  MOV     (R5)+,VADR      ;GET ADDRESS OFFSET
002724  012537  003046                  MOV     (R5)+,70$       ;GET NAME POINTER
002730  012537  001306                  MOV     (R5)+,TEMP      ;GET MAX # OF UNITS
002734  013746  000004                  MOV     @#ERRVEC,-(SP)  ;SAVE ERRVEC
002740  005037  001310                  CLR     NBEXT           ;CLEAR COUNTER
                                ;ADDRESS THE DEVICE TO VERIFY IT EXISTS
002744  012737  003150  000004  1$:     MOV     #2$,ERRVEC      ;SET UP TIME-OUT RETURN
002752  005777  176150                  TST     @$BDDAT         ;ADDRESS THE UNIT
                                ;NOW CAUSE AN INTR. AND FIND THE VECTOR
002756  005037  001304          60$:    CLR     VECT            ;CLEAR VECTOR FLAG
002762  012737  003234  000004          MOV     #IOTRD,ERRVEC   ;SET UP INTELL. TRAP RETURN
002770  005737  003232                  TST     67$             ;TEST IF NON-INTR DEVICE
002774  001414                          BEQ     65$             ;BR IF NOT INTR. DEVICE
002776  004737  003316                  JSR     PC,FIXVCT       ;LOAD VECTOR TRAP
003002  004777  000224                  JSR     PC,@67$         ;GO PRIME THE DEVICE
003006  012700  000000                  MOV     #0,R0           ;PRIME A COUNTER
003012  005046                          CLR     -(SP)           ;LOWER
003014  012746  003022                  MOV     #66$,-(SP)      ;PS
003020  000002                          RTI
003022  005300                  66$:    DEC     R0              ;DELAY
003024  001376                          BNE     66$             ;SOME TIME
003026  005077  176074          65$:    CLR     @$BDDAT         ;LONG ENOUGH DELAY
                                ;REPORT THE ADDRESS AND VECTOR VALUES ON FIRST PASS
003032  005737  001172                  TST     $PASS           ;CHECK IF FIRST
003036  001033                          BNE     73$             ;BR IF NOT
003040  104401  001161                  TYPE    ,$CRLF          ;ENSURE FRESH LINE
003044  104401                          TYPE                    ;REPORT THE NAME
003046  000000                  70$:    0                       ;ADDRESS OF MESSAGE POINTER
003050  104401  004277                  TYPE    ,ATPRI          ;TYPE ADDRESS STARTUP
003054  013746  001126                  MOV     $BDDAT,-(SP)    ;GET ADDRESS
003060  104403                          TYPOS                   ;REPORT IT
003062     006     001                  .BYTE   6,1
003064  104401  004315                  TYPE    ,VTPRI          ;TYPE VECTOR STATUP
003070  005737  001304                  TST     VECT            ;CHECH IF INTR OCCURRED
003074  001003                          BNE     71$             ;BR IF YES
003076  104401  004242                  TYPE    ,NOPRI          ;TELL OPER. NONE
003102  000404                          BR      72$
003104  013746  001304          71$:    MOV     VECT,-(SP)      ;GET VECTOR VALUE
003110  104403                          TYPOS                   ;REPORT IT
003112     003     001                  .BYTE   3,1
                                ;SENSE IF A/D IF SO CHECK OPERATOR INPUT ABOUT CHANNEL MODE REPORT
003114  005737  001316          72$:    TST     FLAGAD          ;IS CHANNEL MODE REPORT, ENABLED?
003120  001402                          BEQ     73$             ;BR IF NOT
003122  004737  003630                  JSR     PC,TCHANL       ;REPORT THE CONFIGURATION
```

```
                                        ;BUMP THE COUNTER AND TEST IF FINISHED?
003126  005237  001310                  73$:    INC     NBEXT           ;INCREMENT UNIT COUNTER
003132  005337  001306                          DEC     TEMP            ;REACHED MAX?
003136  001424                                  BEQ     4$              ;;REACHED MAX NO OF UNIT'S
003140  063737  001302  001126                  ADD     VADR,$BDDAT     ;GET NEXT UNIT
003146  000676                                  BR      1$              ;;TRY NEXT UNIT
                                        ;COME HERE UPON ADDRESS TIME-OUT
003150  022626                          2$:     CMP     (SP)+,(SP)+     ;POP 2 WORDS OFF STACK
003152  005737  001310                          TST     NBEXT           ;CHECK IF ANY UNITS
003156  001014                                  BNE     4$              ;BR IF SOME
003160  005737  001172                          TST     $PASS           ;TEST IF FIRST PASS
003164  001011                                  BNE     4$              ;BR IF NOT FIRST
003166  013737  003046  003202                  MOV     70$,76$         ;GET DEVICE MESSAGE POINTER
003174  104401  001161                          TYPE    ,$CRLF
003200  104401                                  TYPE                    ;TELL OPER. IT'S NOT THERE
003202  000000                          76$:    0
003204  104401  004242                          TYPE    ,NOPRI          ;TELL OPER. NONE FOUND
003210  000240                          4$:     NOP
003212  000240                                  NOP
003214  013735  001310                          MOV     NBEXT,@(R5)+    ;SAVE THE # OF UNITS FOUND
003220  012637  000004                          MOV     (SP)+,ERRVEC    ;RESTORE ERRVEC
003224  005037  001316                          CLR     FLAGAD          ;CLEAR EXPANDED A/D REPORT FLAG
003230  000205                                  RTS     R5
003232  003466                          67$:    ADGO                    ;ADDRESS TO DEVICE PRIMER

                                        ;INTELLIGENT INTERRUPT HANDLER
003234  011637  003314                  IOTRD:  MOV     (SP),TRTO       ;GET ADDRESS
003240  162737  000004  003314                  SUB     #4,TRTO         ;ADJUST VALUE
003246  023727  003314  001000                  CMP     TRTO,#1000      ;CHECK IF VECTOR OR FATAL TRAP
003254  003402                                  BLE     2$              ;BR IF VECTOR
003256  000000                          1$:     HALT    ;FATAL BUS TRAP IN PROGRAM
003260  000776                                  BR      1$
003262  013737  003314  001304          2$:     MOV     TRTO,VECT       ;LOAD VECTOR
003270  005077  175632                          CLR     @$BDDAT         ;CLEAR CURRENT UNIT STATUS
003274  022626                                  CMP     (SP)+,(SP)+
003276  000240                                  NOP
003300  000240                                  NOP
003302  000002                                  RTI                     ;EXIT
003304  000240                                  NOP
003306  000240                                  NOP
003310  000240                                  NOP
003312  000240                                  NOP
003314  000000                          TRTO:   0
```

```
                                                        I 2
352
353  003316  012700  000020          ;SUBROUTINE TO LOAD INTELLIGENT TRAP CATCHER
354  003322  012701  000022  FIXVCT: MOV   #20,R0          ;LOAD STARTING ADDRESS
355  003326  012702  004700          MOV   #22,R1          ;LOAD STARTING POINTER
356  003332  005037  000000          MOV   #4700,R2        ;LOAD 'BAD INSTR''
357  003336  005037  000002          CLR   @#0             ;CLEAR LOC. 0
358  003342  005037  000010          CLR   @#2             ;CLEAR LOC. 2
359  003346  005037  000012          CLR   @#10            ;CLEAR LOC. 10
360  003352  022700  000060  1$:     CLR   @#12            ;CLEAR LOC. 12
361  003356  001432                  CMP   #60,R0          ;TEST IF TKB VECTOR
362  003360  022700  000100          BEQ   3$              ;BR IF YES
363  003364  001425                  CMP   #100,R0         ;TEST IF LINE INTR. <B EVENT>
364  003366  022700  000040          BEQ   2$              ;BR IF YES
365  003372  001422                  CMP   #40,R0          ;TEST FOR XXDP FLAG VECTOR
366  003374  022700  000200          BEQ   2$              ;BR IF YES
367  003400  001421                  CMP   #200,R0         ;TEST IF STARTING ADDRESS
368  003402  022700  000034          BEQ   3$              ;BR IF YES
369  003406  001414                  CMP   #34,R0          ;TEST IF TRAP VECTOR
370  003410  000240                  BEQ   2$              ;BR IF YES
371  003412  000240                  NOP
372  003414  000240                  NOP
373  003416  000240                  NOP
374  003420  000240                  NOP
375  003422  010120                  NOP
376  003424  010220                  MOV   R1,(R0)+        ;LOAD POINTER
377  003426  022121                  MOV   R2,(R0)+        ;LOAD 'BAD''
378  003430  020027  001000          CMP   (R1)+,(R1)+     ;BUMP POINTER
379  003434  001346                  CMP   R0,#1000        ;DONE ?
380  003436  000207                  BNE   1$              ;BR IF NOT
381  003440  022020  2$:     CMP   (R0)+,(R0)+     ;BUMP   RTS   PC              ;RETURN
382  003442  022121                  CMP   (R1)+,(R1)+
383  003444  022020  3$:     CMP   (R0)+,(R0)+
384  003446  022121                  CMP   (R1)+,(R1)+
385  003450  000740                  BR    1$
386  003452  000240                  NOP
387  003454  000240                  NOP
388  003456  000240                  NOP
389  003460  000240                  NOP
390  003462  000240                  NOP
391  003464  000240                  NOP
```

```
                                        ;SUBROUTINE TO PRIME THE MNCAD
003466  012777  000101  175432  ADGO:   MOV     #101,@$BDDAT    ;ENABLE INTR. AND START CONVERSION
003474  000240                  NOP
003476  000240                  NOP
003500  000240                  NOP
003502  000207                  RTS     PC                      ;EXIT
                                        ;SUBROUTINE TO  PRIME THE MNCKW
003504  013700  001126          KWGO:   MOV     $BDDAT,R0       ;GET ADDR.
003510  005200                  INC     R0
003512  005200                  INC     R0                      ;MAKE ADDRESS OF CLOCK PRESET BUFFER
003514  012710  177777          MOV     #-1,(R0)                ;LOAD CLOCK PRESET
003520  012777  000161  175400  MOV     #161,@$BDDAT    ;LOAD RATE, INTR ENABLE AND GO
003526  052777  000400  175372  BIS     #BIT8,@$BDDAT   ;LOAD MAINT CLOCK
003534  000207                  RTS     PC                      ;EXIT
003536  000240                  NOP
003540  000240                  NOP
003542  000240                  NOP
                                        ;SUBROUTINE TO PRIME THE MNCDI
003544  012777  000102  175354  DIGO:   MOV     #102,@$BDDAT    ;LOAD INTR. ENABLE AND MODE
003552  052777  004200  175346  BIS     #4200,@$BDDAT   ;LOAD MAINT. STROBE
003560  000207                  RTS     PC                      ;EXIT
003562  000240                  NOP
003564  000240                  NOP
003566  000240                  NOP
                                        ;SUBROUTINE TO PRIME THE MNCDO
003570  013700  001126          DOGO:   MOV     $BDDAT,R0       ;GET ADDRESS
003574  062700  000003                  ADD     #3,R0   ;MAKE OUTPUT DATA POINTER
003600  105010                  CLRB    (R0)    ;ENABLE MAINT. STROBE
003602  162700  000002          SUB     #2,R0   ;MAKE HIGH BYTE STATUS ADDRESS
003606  112710  000001          MOVB    #BIT0,(R0)      ;MAINT. STROBE
003612  052777  000100  175306  BIS     #BIT6,@$BDDAT   ;ENABLE INTR.
003620  000207                  RTS     PC                      ;EXIT
003622  000240                  NOP
003624  000240                  NOP
003626  000240                  NOP
```

```
                                        ;       *ROUTINE TO TYPE OUT A/D CONFIGURATION
                                        ;LOAD A CODE OF ''0001'' INTO THE GAIN BITS OF CHANNEL 10-77
                                        ;WHEN READ BACK IN THE HIGH 4 BITS OF THE A/D CONVERTED VALUE, THE FOLLOWING
                                        ;INDICATE THE TYPE OF CHANNEL
                                        ;       0000 = SINGLE ENDED MNCAD OR MNCAM CHANNEL
                                        ;       0001 = MNCTP CHANNEL
                                        ;       0010 = DIFFERENTIAL MNCAD OR MNCAM CHANNEL
                                        ;       0011 = NOT USED CODE
                                        ;       0100 THRU 1111 = MNCAG CHANNEL

003630  004737  004122          TCHANL: JSR     PC,LD01CH       ;PRESET MNCTC CHANNELS
003634  005077  175266                  CLR     @$BDDAT         ;CLEAR A/D STATUS
003640  013702  001126                  MOV     $BDDAT,R2       ;GET ADDRESS
003644  062702  000002                  ADD     #2,R2           ;MAKE POINTER TO A/D DATA BUFFER
003650  011201                          MOV     (R2),R1         ;READ A/D BUFFER AND CLEAR A/D DONE FLAG
003652  005037  004120                  CLR     CHB             ;CLEAR MNCAG COUNTER
003656  005001                          CLR     R1              ;INIT R1
003660  104401  001161                  TYPE    ,$CRLF          ;LEAVE A BLANK LINE
003664  104401  004332          2$:     TYPE    ,$SPACE         ;MOVE OVER THE LINE
003670  010146                          MOV     R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
(1) 003672  104403                      TYPOS                   ;;GO TYPE--OCTAL ASCII
(1) 003674     002                      .BYTE   2               ;;TYPE 2 DIGIT(S)
(1) 003675     000                      .BYTE   0               ;;SUPPRESS LEADING ZEROS
003676  104401  004337                  TYPE    ,MDASH          ;TYPE A DASH
003702  005277  175220          3$:     INC     @$BDDAT         ;START CONVERSION
003706  105777  175214          4$:     TSTB    @$BDDAT         ;WAIT FOR DONE
003712  100375                          BPL     4$              ;BR IF NOT
003714  011200                          MOV     (R2),R0         ;GET CONVERTED VALUE
003716  042700  007777                  BIC     #7777,R0        ;IS CHANNEL SINGLE ENDED
003722  001004                          BNE     5$              ;;CHANNEL IS NOT SINGLE ENDED
003724  012737  004343  004044          MOV     #MSE,12$        ;LOAD MESSAGE POINTER
003732  000431                          BR      8$
003734  032700  140000          5$:     BIT     #140000,R0      ;TEST IF MNCAG CHANNEL
003740  001411                          BEQ     6$              ;BR IF NOT
003742  062737  000004  004120          ADD     #4,CHB          ;UPDATE NUMBER OF MNCAG DETECTED
003750  062701  000003                  ADD     #3,R1           ;UPDATE CHANNEL VALUE
003754  012737  004401  004044          MOV     #MPRMP,12$      ;LOAD MESSAGE POINTER
003762  000417                          BR      10$
003764  022700  010000          6$:     CMP     #10000,R0       ;TEST IF MNCTP CHANNEL
003770  001004                          BNE     7$              ;BR IF NOT
003772  012737  004412  004044          MOV     #MTCMP,12$      ;LOAD MESSAGE POINTER
004000  000406                          BR      8$
004002  012737  004362  004044  7$:     MOV     #MDIF,12$       ;LOAD MESSAGE POINTER
004010  062701  000003                  ADD     #3,R1           ;UPDATE CHANNEL VALUE
004014  000402                          BR      10$
004016  062701  000007          8$:     ADD     #7,R1           ;UPDATE CHANNEL VALUE
004022  022701  000100          10$:    CMP     #100,R1         ;IS CHANNEL > LAST POSSIBLE CHANNEL
004026  101002                          BHI     11$             ;;NO
004030  012701  000077                  MOV     #77,R1          ;YES, SET TO LAST CHANNEL
004034                          11$:
(1) 004034  010146                      MOV     R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
(1) 004036  104403                      TYPOS                   ;;GO TYPE--OCTAL ASCII
(1) 004040     002                      .BYTE   2               ;;TYPE 2 DIGIT(S)
(1) 004041     000                      .BYTE   0               ;;SUPPRESS LEADING ZEROS
004042  104401                          TYPE                    ;REPORT THE CHANNEL TYPE
004044  004343          12$:    MSE                             ;POINTER TO MESSAGE
```

```
                                        13$:    INC     R1              ;SET CHANNEL TO NEXT SET OF CHANNELS
                                                CMP     #100,R1         ;DONE?
                                                BEQ     14$             ;;YES
                                                MOV     R1,R0           ;GET CHANNEL
                                                SWAB    R0              ;PUT CHANNEL NUMBER IN HIGH BYTE
                                                BIS     #BIT3,R0        ;SET STATUS ENABLE BIT
                                                MOV     R0,@$BDDAT      ;LOAD INTO A/D STATUS REGISTER
                                                BIT     #BIT1,@$BDDAT   ;IS NON-EXSISTENT CHANNEL BIT SET?
                                                BEQ     2$              ;;NO
                                        14$:    CMP     CHB,#25         ;TEST HOW MANY MNCAG FOUND
                                                BLO     15$             ;BR IF LESS THAN LIMIT
                                                TYPE    ,WOWAGS         ;TELL OPERATOR TOO MANY DETECTED
                                        15$:
                                        TCHANE: RTS     PC              ;EXIT IF DONE
                                        CHB:    0                       ;NUMBER OF MNCAG DETECTED

                                        ;SUBROUTINE TO LOAD A ''0001'' INTO THE GAIN BITS OF CHANNEL 10-77

                                        LD01CH: MOV     $BDDAT,R2       ;LOAD A/D ADDRESS POINTER
                                                INC     R2              ;MAKE POINTER TO HIGH BYTE
                                                MOV     #10,R1          ;LOAD INITIAL CHANNEL
                                        1$:     MOVB    #77,(R2)        ;LOAD ''ESCAPE''
                                                MOVB    #1,(R2)         ;LOAD ''01''
                                                MOVB    R1,(R2)         ;LOAD CHANNEL NUMBER
                                                INC     R1              ;UPDATE TO NEXT CHANNEL
                                                CMP     #100,R1         ;TEST IF DONE ALL CHANNELS
                                                BNE     1$              ;BR IF NOT
                                                RTS     PC              ;EXIT
```

```
                                                ;ASCII MESSAGES
004160  020040  046473  041516  ADPRI:  .ASCIZ  /  ;MNCAD /
004166  042101  000040
004172  020040  046473  041516  KWPRI:  .ASCIZ  /  ;MNCKW /
004200  053513  000040
004204  020040  046473  041516  AAPRI:  .ASCIZ  /  ;MNCAA /
004212  040501  000040
004216  020040  046473  041516  DIPRI:  .ASCIZ  /  ;MNCDI /
004224  044504  000040
004230  020040  046473  041516  DOPRI:  .ASCIZ  /  ;MNCDO /
004236  047504  000040
004242  025040  020052  042040  NOPRI:  .ASCIZ  / **  DOES NOT EXIST **/
004250  042517  020123  047516
004256  020124  054105  051511
004264  020124  025052     000
004271     043  047440  020106  NIPRI:  .ASCIZ  /# OF /
004276     000
004277     101  020124  042101  ATPRI:  .ASCIZ  /AT ADDRESS = /
004304  051104  051505  020123
004312  020075     000
004315     040  020040  042526  VTPRI:  .ASCIZ  /   VECTOR = /
004322  052103  051117  036440
004330  000040
004332  004411     000          SSPACE: .ASCIZ  /                /
004335     040     000          SPACE1: .ASCIZ  / /
004337     040  020055     000  MDASH:  .ASCIZ  / - /
004343     040  044523  043516  MSE:    .ASCIZ  / SINGLE ENDED/<200>
004350  042514  042440  042116
004356  042105  000200
004362  042040  043111  042506  MDIF:   .ASCIZ  / DIFFERENTIAL/<200>
004370  042522  052116  040511
004376  100114     000
004401     040  051120  040505  MPRMP:  .ASCIZ  / PREAMP/<200>
004406  050115  000200
004412  052040  020103  046501  MTCMP:  .ASCIZ  / TC AMP/<200>
004420  100120     000
004423     200  047504  054440  QUESAD: .ASCII  <200>\DO YOU WANT THE MNCAD (A/D) CHANNEL MODE REPORT ?\
004430  052517  053440  047101
004436  020124  044124  020105
004444  047115  040503  020104
004452  040450  042057  020051
004460  044103  047101  042516
004466  020114  047515  042504
004474  051040  050105  051117
004502  020124     077
004505     200  004411  054524          .ASCIZ  <200>\          TYPE ''Y'' FOR YES = \
004512  042520  021040  021131
004520  043040  051117  054440
004526  051505  036440  000040
004534  042200  020117  047531  QUESMP: .ASCII  <200>\DO YOU WANT THE MEMORY USAGE MAP REPORT ?\
004542  020125  040527  052116
004550  052040  042510  046440
004556  046505  051117  020131
004564  051525  043501  020105
004572  040515  020120  042522
```

```
                004600  047520  052122  037440
528             004606  004600  052011  050131          .ASCIZ  <200>\          TYPE 'Y' FOR YES = \
                004614  020105  054442  020042
                004622  047506  020122  042531
                004630  020123  020075     000
529             004635     200  044515  041516  MNCMAP: .ASCII  <200>\MINC-11 MEMORY USAGE MAP  (EACH BIT = 2 ADDRESSES)\<200>
                004642  030455  020061  042515
                004650  047515  054522  052440
                004656  040523  042507  046440
                004664  050101  024011  040505
                004672  044103  041040  052111
                004700  036440  031040  040440
                004706  042104  042522  051523
                004714  051505  100051
530             004720  040440  042104  004522          .ASCII  \ ADDR      000/400          100/500\
                004726  020040  020040  030060
                004734  027460  030064  020060
                004742  020040  020040  020040
                004750  020040  030440  030060
                004756  032457  030060
531             004762  020040  020040  020040          .ASCIZ  \          200/600          300/700\
                004770  020040  020040  030062
                004776  027460  030066  020060
                005004  020040  020040  020040
                005012  020040  031440  030060
                005020  033457  030060     000
532             005025     015     012     012  SETMNC: .BYTE   15,12,12
533             005030  050117  051105  052101          .ASCII  /OPERATOR       -       PLEASE DO THE FOLLOWING:/
                005036  051117  026411  050011
                005044  042514  051501  020105
                005052  047504  052040  042510
                005060  043040  046117  047514
                005066  044527  043516     072
534             005073     015     012     012          .BYTE   15,12,12
535             005076  042522  047515  042526          .ASCII  /REMOVE THE CUSTOMER CONNECTIONS FROM EACH 'MINC-11' OPTION/
                005104  052040  042510  041440
                005112  051525  047524  042515
                005120  020122  047503  047116
                005126  041505  044524  047117
                005134  020123  051106  046517
                005142  042440  041501  020110
                005150  046442  047111  026503
                005156  030461  020042  050117
                005164  044524  047117
536             005170     015     012          .BYTE   15,12
537             005172  047115  040503  020104          .ASCII  \MNCAD (A/D)            SET FRONT PANEL SWITCHES TO THE 'TEST' POSITION\
                005200  040450  042057  004451
                005206  051411  052105  043040
                005214  047522  052116  050040
                005222  047101  046105  051440
                005230  044527  041524  042510
                005236  020123  047524  052040
                005244  042510  021040  042524
                005252  052123  020042  047520
                005260  044523  044524  047117
538             005266     015     012          .BYTE   15,12
```

```
  539  005270  047115  040503  020107          .ASCII  \MNCAG (PREAMP)          SET FRONT PANEL SWITCHES TO THE 'P' POSITION\
       005276  050050  042522  046501
       005304  024520  004411  042523
       005312  020124  051106  047117
       005320  020124  040520  042516
       005326  020114  053523  052111
       005334  044103  051505  052040
       005342  020117  044124  020105
       005350  050042  020042  047520
       005356  044523  044524  047117
  540  005364     015     012          .BYTE   15,12
  541  005366  047115  045503  020127  .ASCII  \MNCKW (CLOCK)           PULL OUT 'ST1' AND 'ST2' SWITCHES\
       005374  041450  047514  045503
       005402  004451  050011  046125
       005410  020114  052517  020124
       005416  051442  030524  020042
       005424  047101  020104  051442
       005432  031124  020042  053523
       005440  052111  044103  051505
  542  005446     015     012          .BYTE   15,12
  543  005450  004411  040411  042116  .ASCII  \                        AND ROTATE FULLY CLOCKWISE\
       005456  051040  052117  052101
       005464  020105  052506  046114
       005472  020131  046103  041517
       005500  053513  051511     105
  544  005505     015     012          .BYTE   15,12
  545  005507     115  041516  044504  .ASCII  \MNCDI (DIGITAL IN)      SET 'DATA' SWITCH TO THE '-' POSITION\
       005514  024040  044504  044507
       005522  040524  020114  047111
       005530  004451  042523  020124
       005536  042042  052101  021101
       005544  051440  044527  041524
       005552  020110  047524  052040
       005560  042510  021040  021055
       005566  050040  051517  052111
       005574  047511     116
  546  005577     015     012          .BYTE   15,12
  547  005601     123  052514  030040  .ASCII  \SLU 0                    INSTALL 'SLU TEST CONNECTOR'\
       005606  004411  044411  051516
       005614  040524  046114  021040
       005622  046123  020125  042524
       005630  052123  041440  047117
       005636  042516  052103  051117
       005644     042
  548  005645     015     012          .BYTE   15,12
  549  005647     123  052514  030440  .ASCII  \SLU 1                    INSTALL 'SLU TEST CONNECTOR'\
       005654  004411  044411  051516
       005662  040524  046114  021040
       005670  046123  020125  042524
       005676  052123  041440  047117
       005704  042516  052103  051117
       005712     042
  550  005713     015     012          .BYTE   15,12
  551  005715     123  052514  031040  .ASCII  \SLU 2                    INSTALL 'SLU TEST CONNECTOR'\
       005722  004411  044411  051516
       005730  040524  046114  021040
```

```
        005736  046123  020125  042524
        005744  052123  041440  047117
        005752  042516  052103  051117
        005760     042
552     005761     015     012     012          .BYTE   15,12,12
553     005764  042504  051120  051505          .ASCIZ  \DEPRESS A KEY ON THE CONSOLE TERMINAL WHEN READY. \
        005772  020123  020101  042513
        006000  020131  047117  052040
        006006  042510  041440  047117
        006014  047523  042514  052040
        006022  051105  044515  040516
        006030  020114  044127  047105
        006036  051040  040505  054504
        006044  020056     000
554     006047     200  044103  041505  WOWAGS: .ASCIZ  <200>\CHECK SYSTEM CONFIGURATION - TOO MANY MNCAG DETECTED\<200>
        006054  020113  054523  052123
        006062  046505  041440  047117
        006070  044506  052507  040522
        006076  044524  047117  026440
        006104  052040  047517  046440
        006112  047101  020131  047115
        006120  040503  020107  042504
        006126  042524  052103  042105
        006134  000200
555                                          .EVEN
556
```

```
    558                                 .SBTTL  TTY INPUT ROUTINE
    (1)
    (2)                                 ;;*****************************************************************
    (1)                                 .ENABL  LSB
    (1)  006136  000000                 $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
    (1)  006140  000000                 $TKQIN: .WORD   0               ;;INPUT POINTER
    (1)  006142  000000                 $TKQOUT: .WORD  0               ;;OUTPUT POINTER
    (1)  006144  000040                 $TKQSRT: .BLKB  32.             ;;TTY KEYBOARD QUEUE
    (1)          006204                 $TKQEND=.
    (1)
    (1)                                 ;*TK INITIALIZE ROUTINE
    (1)                                 ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
    (1)                                 ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
    (1)                                 ;
    (1)                                 ;*CALL:
    (1)                                 ;*      JSR     PC,$TKINT
    (1)                                 ;*      RETURN
    (1)                                 ;
    (1)  006204  005037  006136         $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
    (1)  006210  012737  006144  006140         MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
    (1)  006216  013737  006140  006142         MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
    (1)  006224  012737  006254  000060         MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
    (1)  006232  012737  000200  000062         MOV     #200,@#TKVEC+2  ;;'BR'' LEVEL 4
    (1)  006240  005777  172702                 TST     @$TKB           ;;CLEAR DONE FLAG
    (1)  006244  012777  000100  172672         MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
    (1)  006252  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 (BEGIN)
    (1)                                 ;
    (1)  006254  117746  172666         $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
    (1)  006260  042716  177600                 BIC     #^C177,(SP)     ;;STRIP THE JUNK
    (1)  006264  021627  000003                 CMP     (SP),#3         ;;IS IT A CONTROL C?
    (1)  006270  001007                         BNE     1$              ;;BRANCH IF NO
    (1)  006272  104401  007044                 TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
    (1)  006276  004737  006204                 JSR     PC,$TKINT       ;;INIT THE KEYBOARD
    (1)  006302  005726                         TST     (SP)+           ;;CLEAN UP STACK
    (1)  006304  000137  001324                 JMP     BEGIN           ;;CONTROL C RESTART
    (1)
    (1)  006310                         1$:
    (1)  006310  022737  000040  006136         CMP     #32.,$TKCNT     ;;IS THE QUEUE FULL?
    (1)  006316  001004                         BNE     3$              ;;BRANCH IF NO
    (1)  006320  104401  007040                 TYPE    ,$BELL          ;;RING THE TTY BELL
    (1)  006324  005726                         TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
    (1)  006326  000451                         BR      5$              ;;EXIT
    (1)  006330  021627  000023         3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
    (1)  006334  001021                         BNE     32$             ;;BRANCH IF NO
    (1)  006336  005077  172602                 CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
    (1)  006342  005726                         TST     (SP)+           ;;CLEAN CHAR OFF STACK
    (1)  006344  105777  172574         31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
    (1)  006350  100375                         BPL     31$             ;;LOOP UNTIL ITS THERE
    (1)  006352  117746  172570                 MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
```

```
(1)  006356  042716  177600                BIC     #^C177,(SP)       ;;MAKE IT 7-BIT ASCII
(1)  006362  022627  000021                CMP     (SP)+,#21         ;;IS IT A CONTROL-Q?
(1)  006366  001366                        BNE     31$               ;;BRANCH IF NO
(1)  006370  012777  000100  172546        MOV     #100,@$TKS        ;;REENABLE TTY KEYBOARD INTERRUPTS
(1)  006376  000002                        RTI                       ;;RETURN
(1)  006400  005237  006136          32$:  INC     $TKCNT            ;;COUNT THIS CHARACTER
(1)  006404  021627  000140                CMP     (SP),#140         ;;IS IT UPPER CASE?
(1)  006410  002405                        BLT     4$                ;;BRANCH IF YES
(1)  006412  021627  000175                CMP     (SP),#175         ;;IS IT A SPECIAL CHAR?
(1)  006416  003002                        BGT     4$                ;;BRANCH IF YES
(1)  006420  042716  000040                BIC     #40,(SP)          ;;MAKE IT UPPER CASE
(1)  006424  112677  177510          4$:   MOVB    (SP)+,@$TKQIN     ;;AND PUT IT IN QUEUE
(1)  006430  005237  006140                INC     $TKQIN            ;;UPDATE THE POINTER
(1)  006434  023727  006140  006204        CMP     $TKQIN,#$TKQEND   ;;GO OFF THE END?
(1)  006442  001003                        BNE     5$                ;;BRANCH IF NO
(1)  006444  012737  006144  006140        MOV     #$TKQSRT,$TKQIN   ;;RESET THE POINTER
(1)  006452  000002                  5$:   RTI                       ;;RETURN
(1)
(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)  006454  011646          $RDCHR: MOV     (SP),-(SP)        ;;PUSH DOWN THE PC AND
(1)  006456  016666  000004  000002  MOV     4(SP),2(SP)       ;;THE PS
(1)  006464  005066  000004          CLR     4(SP)             ;;GET READY FOR A CHARACTER
(2)  006470  005046                  CLR     -(SP)             ;;PUT NEW PS ON STACK
(2)  006472  012746  006500          MOV     #64$,-(SP)        ;;PUT NEW PC ON  STACK
(2)  006476  000002                  RTI                       ;;POP NEW PC AND PS
(2)  006500                    64$:
(1)  006500  005737  006136    1$:   TST     $TKCNT            ;;WAIT ON A CHARACTER
(1)  006504  001775                  BEQ     1$
(1)  006506  005337  006136          DEC     $TKCNT            ;;DECREMENT THE COUNTER
(1)  006512  117766  177424  000004  MOVB    @$TKQOUT,4(SP)    ;;GET ONE CHARACTER
(1)  006520  005237  006142          INC     $TKQOUT           ;;UPDATE THE POINTER
(1)  006524  023727  006142  006204  CMP     $TKQOUT,#$TKQEND  ;;DID IT GO OFF OF THE END?
(1)  006532  001003                  BNE     2$                ;;BRANCH IF NO
(1)  006534  012737  006144  006142  MOV     #$TKQSRT,$TKQOUT  ;;RESET THE POINTER
(1)  006542  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)  006544  010346          $RDLIN: MOV     R3,-(SP)          ;;SAVE R3
(1)  006546  005046                  CLR     -(SP)             ;;CLEAR THE RUBOUT KEY
(1)  006550  012703  007000    1$:   MOV     #$TTYIN,R3        ;;GET ADDRESS
(1)  006554  022703  007040    2$:   CMP     #$TTYIN+32.,R3    ;;BUFFER FULL?
```

```
(1)  006560  101456                          BLOS    4$              ;;BR IF YES
(1)  006562  104407                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  006564  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  006566  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  006572  001022                          BNE     5$              ;;BR IF NO
(1)  006574  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  006576  001007                          BNE     6$              ;;BR IF NO
(1)  006600  112737  000134  006776          MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  006606  104401  006776                  TYPE    ,9$
(1)  006612  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  006616  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
(1)  006620  020327  007000                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  006624  103434                          BLO     4$              ;;BR IF YES
(1)  006626  111337  006776                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  006632  104401  006776                  TYPE    ,9$             ;;GO TYPE
(1)  006636  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
(1)  006640  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
(1)  006642  001406                          BEQ     7$              ;;BR IF NO
(1)  006644  112737  000134  006776          MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  006652  104401  006776                  TYPE    ,9$
(1)  006656  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  006660  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  006664  001003                          BNE     8$              ;;BR IF NO
(1)  006666  104401  007051                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U''
(1)  006672  000726                          BR      1$              ;;GO START OVER
(1)  006674  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A ''^R''?
(1)  006700  001011                          BNE     3$              ;;BRANCH IF NO
(1)  006702  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
(1)  006704  104401  001161                  TYPE    ,$CRLF          ;;TYPE A ''CR'' & 'LF''
(1)  006710  104401  007000                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  006714  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  006716  104401  001160          4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  006722  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  006724  111337  006776          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  006730  104401  006776                  TYPE    ,9$
(1)  006734  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  006740  001305                          BNE     2$              ;;LOOP IF NOT RETURN
(1)  006742  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  006746  104401  001162                  TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  006752  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  006754  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  006756  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  006760  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  006766  012766  007000  000004          MOV     #$TTYIN,4(SP)
(1)  006774  000002                          RTI                     ;;RETURN
(1)  006776     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  006777     000                          .BYTE   0               ;;TERMINATOR
(1)  007000  000040          $TTYIN: .BLKB   32.                     ;;RESERVE 32. BYTES FOR TTY INPUT
(1)  007040  177607  000377  $BELL:  .ASCIZ  <207><377><377>        ;;CODE FOR BELL
(1)  007044  041536  005015     000  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL ''C''
(1)  007051     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL ''U''
(1)  007056  043536  005015     000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL ''G''
(1)  007063     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  007070  036440  000040
(1)  007074  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
(1)  007102  020075     000
```

```
(1)          007106                    .EVEN
559                                    .SBTTL  TYPE ROUTINE
(1)
(2)                                    ;;*****************************************************************
(1)                                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)                                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)                                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)                                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)                                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)                                    ;*
(1)                                    ;*CALL:
(1)                                    ;*1) USING A TRAP INSTRUCTION
(1)                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)                                    ;*OR
(1)                                    ;*      TYPE
(1)                                    ;*      MESADR
(1)                                    ;*
(1)
(1)  007106  105737  001157            $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
(1)  007112  100002                            BPL     1$              ;;BR IF YES
(1)  007114  000000                            HALT                    ;;HALT HERE IF NO TERMINAL
(1)  007116  000430                            BR      3$              ;;LEAVE
(1)  007120  010046                    1$:     MOV     R0,-(SP)        ;;SAVE R0
(1)  007122  017600  000002                    MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
(1)  007126  122737  000001  001204            CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  007134  001011                            BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
(1)  007136  132737  000100  001205            BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1)  007144  001405                            BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
(1)  007146  010037  007156                    MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
(1)  007152  004737  010176                    JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
(1)  007156  000000                    61$:    .WORD   0               ;;MESSAGE ADDRESS
(1)  007160  132737  000040  001205    62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
(1)  007166  001003                            BNE     60$             ;;YES,SKIP TYPE OUT
(1)  007170  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1)  007172  001005                            BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
(1)  007174  005726                            TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
(1)  007176  012600                    60$:    MOV     (SP)+,R0        ;;RESTORE R0
(1)  007200  062716  000002            3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
(1)  007204  000002                            RTI                     ;;RETURN
(1)  007206  122716  000011            4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
(1)  007212  001430                            BEQ     8$
(1)  007214  122716  000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
(1)  007220  001006                            BNE     5$
(1)  007222  005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
(1)  007224  104401                            TYPE                    ;;TYPE A CR AND LF
(1)  007226  001161                            $CRLF
(1)  007230  105037  007436                    CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
(1)  007234  000755                            BR      2$              ;;GET NEXT CHARACTER
(1)  007236  004737  007320            5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
(1)  007242  123726  001156            6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
(1)  007246  001350                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
(1)  007250  013746  001154                    MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
(1)                                            ;;AND THE NULL CHAR.
(1)  007254  105366  000001            7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
(1)  007260  002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1)  007262  004737  007320                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
```

```
(1)  007266  105337  007436            DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  007272  000770                    BR      7$              ;;LOOP
(1)
(1)                                ;HORIZONTAL TAB PROCESSOR
(1)
(1)  007274  112716  000040       8$:  MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  007300  004737  007320       9$:  JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  007304  132737  000007  007436    BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  007312  001372                    BNE     9$              ;;TAB STOP
(1)  007314  005726                    TST     (SP)+           ;;POP SPACE OFF STACK
(1)  007316  000724                    BR      2$              ;;GET NEXT CHARACTER
(1)  007320                        $TYPEC:
(1)  007320  105777  171620            TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
(1)  007324  100022                    BPL     10$             ;;BR IF NOT                     ;MJD001
(1)  007326  017746  171614            MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
(1)  007332  042716  177600            BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
(1)  007336  122716  000023            CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
(1)  007342  001012                    BNE     102$            ;;BR IF NOT                     ;MJD001
(1)  007344                        101$:                                                       ;MJD001
(1)  007344  105777  171574            TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
(1)  007350  100375                    BPL     101$                                            ;MJD001
(1)  007352  117716  171570            MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
(1)  007356  042716  177600            BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
(1)  007362  122716  000021            CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
(1)  007366  001366                    BNE     101$            ;;BR IF NOT                     ;MJD001
(1)  007370                        102$:                                                       ;MJD001
(1)  007370  005726                    TST     (SP)+           ;;FIX STACK                     ;MJD001
(1)  007372                        10$:                                                        ;MJD001
(1)  007372  105777  171552            TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  007376  100375                    BPL     10$                                             ;MJD001
(1)  007400  116677  000002  171544    MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  007406  122766  000015  000002    CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  007414  001003                    BNE     1$              ;;BRANCH IF NO
(1)  007416  105037  007436            CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  007422  000406                    BR      $TYPEX          ;;EXIT
(1)  007424  122766  000012  000002  1$:  CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  007432  001402                    BEQ     $TYPEX          ;;BRANCH IF YES
(1)  007434  105227                    INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  007436  000000            $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  007440  000207            $TYPEX: RTS     PC
(1)
```

```
                                        .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                        ;;**********************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
007442  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
007446  116637  000001  007665          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
007454  112637  007667                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
007460  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
007464  000406                          BR      $TYPON
007466  112737  000001  007665  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
007474  112737  000006  007667          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
007502  112737  000005  007664  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
007510  010346                          MOV     R3,-(SP)        ;;SAVE R3
007512  010446                          MOV     R4,-(SP)        ;;SAVE R4
007514  010546                          MOV     R5,-(SP)        ;;SAVE R5
007516  113704  007667                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
007522  005404                          NEG     R4
007524  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
007530  110437  007666                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
007534  113704  007665                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
007540  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
007544  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
007546  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
007550  000404                          BR      3$              ;;GO DO MSB
007552  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
007554  006105                          ROL     R5
007556  006105                          ROL     R5
007560  010503                          MOV     R5,R3
007562  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
007564  105337  007666                  DECB    $OMODE          ;;TYPE THIS DIGIT?
007570  100016                          BPL     7$              ;;BR IF NO
007572  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
007576  001002                          BNE     4$              ;;TEST FOR 0
007600  005704                          TST     R4              ;;SUPPRESS THIS 0?
007602  001403                          BEQ     5$              ;;BR IF YES
```

```
(1)  007604  005204                    4$:   INC   R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  007606  052703  000060                  BIS   #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  007612  052703  000040            5$:   BIS   #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  007616  110337  007662                  MOVB  R3,8$           ;;SAVE FOR TYPING
(1)  007622  104401  007662                  TYPE  ,8$             ;;GO TYPE THIS DIGIT
(1)  007626  105337  007664            7$:   DECB  $OCNT           ;;COUNT BY 1
(1)  007632  003347                          BGT   2$              ;;BR IF MORE TO DO
(1)  007634  002402                          BLT   6$              ;;BR IF DONE
(1)  007636  005204                          INC   R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  007640  000744                          BR    2$              ;;GO DO THE LAST DIGIT
(1)  007642  012605                    6$:   MOV   (SP)+,R5        ;;RESTORE R5
(1)  007644  012604                          MOV   (SP)+,R4        ;;RESTORE R4
(1)  007646  012603                          MOV   (SP)+,R3        ;;RESTORE R3
(1)  007650  016666  000002  000004          MOV   2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  007656  012616                          MOV   (SP)+,(SP)
(1)  007660  000002                          RTI                   ;;RETURN
(1)  007662     000                   8$:   .BYTE 0               ;;STORAGE FOR ASCII DIGIT
(1)  007663     000                         .BYTE 0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  007664     000                   $OCNT:.BYTE 0               ;;OCTAL DIGIT COUNTER
(1)  007665     000                   $OFILL:.BYTE 0              ;;ZERO FILL SWITCH
(1)  007666  000000                   $OMODE:.WORD 0              ;;NUMBER OF DIGITS TO TYPE
 562                                  .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)  007670                           $TYPDS:
(3)  007670  010046                         MOV   R0,-(SP)        ;;PUSH R0 ON STACK
(3)  007672  010146                         MOV   R1,-(SP)        ;;PUSH R1 ON STACK
(3)  007674  010246                         MOV   R2,-(SP)        ;;PUSH R2 ON STACK
(3)  007676  010346                         MOV   R3,-(SP)        ;;PUSH R3 ON STACK
(3)  007700  010546                         MOV   R5,-(SP)        ;;PUSH R5 ON STACK
(1)  007702  012746  020200                 MOV   #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  007706  016605  000020                 MOV   20(SP),R5       ;;GET THE INPUT NUMBER
(1)  007712  100004                         BPL   1$              ;;BR IF INPUT IS POS.
(1)  007714  005405                         NEG   R5              ;;MAKE THE BINARY NUMBER POS.
(1)  007716  112766  000055  000001         MOVB  #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  007724  005000                   1$:   CLR   R0              ;;ZERO THE CONSTANTS INDEX
(1)  007726  012703  010104                 MOV   #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  007732  112723  000040                 MOVB  #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  007736  005002                   2$:   CLR   R2              ;;CLEAR THE BCD NUMBER
(1)  007740  016001  010074                 MOV   $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  007744  160105                   3$:   SUB   R1,R5           ;;FORM THIS BCD DIGIT
(1)  007746  002402                         BLT   4$              ;;BR IF DONE
(1)  007750  005202                         INC   R2              ;;INCREASE THE BCD DIGIT BY 1
(1)  007752  000774                         BR    3$
(1)  007754  060105                   4$:   ADD   R1,R5           ;;ADD BACK THE CONSTANT
(1)  007756  005702                         TST   R2              ;;CHECK IF BCD DIGIT=0
(1)  007760  001002                         BNE   5$              ;;FALL THROUGH IF 0
```

```
    (1)  007762  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
    (1)  007764  100407                          BMI     7$              ;;BR IF YES
    (1)  007766  106316                  5$:     ASLB    (SP)            ;;MSD?
    (1)  007770  103003                          BCC     6$              ;;BR IF NO
    (1)  007772  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
    (1)  010000  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
    (1)  010004  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
    (1)  010010  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
    (1)  010012  005720                          TST     (R0)+           ;;JUST INCREMENTING
    (1)  010014  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
    (1)  010020  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
    (1)  010022  003002                          BGT     8$              ;;GO TO EXIT
    (1)  010024  010502                          MOV     R5,R2           ;;GET THE LSD
    (1)  010026  000764                          BR      6$              ;;GO CHANGE TO ASCII
    (1)  010030  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
    (1)  010032  100003                          BPL     9$              ;;BR IF NO
    (1)  010034  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
    (1)  010042  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
    (3)  010044  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
    (3)  010046  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
    (3)  010050  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
    (3)  010052  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
    (3)  010054  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
    (1)  010056  104401  010104                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
    (1)  010062  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
    (1)  010070  012616                          MOV     (SP)+,(SP)
    (1)  010072  000002                          RTI                     ;;RETURN TO USER
    (1)  010074  023420                  $DTBL:  10000.
    (1)  010076  001750                          1000.
    (1)  010100  000144                          100.
    (1)  010102  000012                          10.
    (1)  010104  000004                  $DBLK:  .BLKW   4
563                                      .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)  010114  010146                  $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
    (1)  010116  016601  000006                  MOV     6(SP),R1        ;;GET THE INPUT NUMBER
    (1)  010122  000261                          SEC                     ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
    (1)  010124  112737  000060  010166  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII '0'.
    (1)  010132  006101                          ROL     R1              ;;GET THIS BIT
    (1)  010134  001406                          BEQ     2$              ;;DONE?
    (1)  010136  105537  010166                  ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
    (1)  010142  104401  010166                  TYPE    ,$BIN           ;;GO TYPE THIS BIT
    (1)  010146  000241                          CLC                     ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
    (1)  010150  000765                          BR      1$              ;;GO DO THE NEXT BIT
    (1)  010152  012601                  2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
    (1)  010154  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
    (1)  010162  012616                          MOV     (SP)+,(SP)
    (1)  010164  000002                          RTI                     ;;RETURN TO USER
    (1)  010166     000     000          $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
```

```
                        .SBTTL  APT COMMUNICATIONS ROUTINE
                        ;;*****************************************************************
(1)  010170 112737 000001 010434 $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  010176 112737 000001 010432 $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  010204 000403               BR      $ATYC
(1)  010206 112737 000001 010434 $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  010214                      $ATYC:
(3)  010214 010046               MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  010216 010146               MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  010220 105737 010432        TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  010224 001450               BEQ     5$              ;;IF NOT:  BR
(1)  010226 122737 000001 001204 CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  010234 001031               BNE     3$              ;;IF NOT:  BR
(1)  010236 132737 000100 001205 BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  010244 001425               BEQ     3$              ;;IF NOT:  BR
(1)  010246 017600 000004        MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  010252 062766 000002 000004 ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  010260 005737 001164        1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  010264 001375               BNE     1$              ;;IF NOT:  WAIT
(1)  010266 010037 001200        MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  010272 105720               2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  010274 001376               BNE     2$
(1)  010276 163700 001200        SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  010302 006200               ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  010304 010037 001202        MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  010310 012737 000004 001164 MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  010316 000413               BR      5$
(1)  010320 017637 000004 010344 3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  010326 062766 000002 000004 ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  010334 013746 177776        MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  010340 004737 007106        JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  010344 000000               4$:     .WORD   0
(1)  010346                      5$:
(1)  010346 105737 010434        10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  010352 001416               BEQ     12$             ;;IF NOT:  BR
(1)  010354 005737 001204        TST     $ENV            ;;RUNNING UNDER APT?
(1)  010360 001413               BEQ     12$             ;;IF NOT:  BR
(1)  010362 005737 001164        11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  010366 001375               BNE     11$             ;;IF NOT:  WAIT
(1)  010370 017637 000004 001166 MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  010376 062766 000002 000004 ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  010404 005237 001164        INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  010410 105037 010434        12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  010414 105037 010433        CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  010420 105037 010432        CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  010424 012601               MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  010426 012600               MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  010430 000207               RTS     PC              ;;RETURN
(1)  010432    000               $MFLG:  .BYTE   0       ;;MESSG. FLAG
(1)  010433    000               $LFLG:  .BYTE   0       ;;LOG FLAG
(1)  010434    000               $FFLG:  .BYTE   0       ;;FATAL FLAG
(1)          010436                    .EVEN
(1)          000200             APTSIZE=200
(1)          000001             APTENV=001
(1)          000100             APTSPOOL=100
```

```
(1)          00004C                  APTCSUP=040
```

```
                                        .SBTTL  TRAP DECODER

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

010436  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
010440  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
010444  005740                          TST     -(R0)           ;;BACKUP BY 2
010446  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
010450  006300                          ASL     R0              ;;POSITION FOR INDEXING
010452  016000  010472                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
010456  000200                          RTS     R0              ;;GO TO ROUTINE


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

010460  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
010462  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
010470  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
010472  010460                  $TRPAD: .WORD   $TRAP2
                                        $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
                                        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
                                        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
                                        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
                                        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
                                        $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER


                                        $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
                                        $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE

                                ;MEMORY USAGE MAP BLOCK
010514  000100                  ADRPOK: .BLKW   100
010714  000000                  ADRTOP: 0                       ;LAST LOCATION USED BY PROGRAM

                                        .END
000001
```

```
AABASE  001262            55#     93*     94*    151
AACNT   001274            63#    101*    155     185
AAOFF   001250            47#     94
AAPRI   004204           153     184     511#
ABASE = 171000            14#     42
ACDW1 = 000000            42
ACDW2 = 000000            42
ACPUOP= 000000            42
ADBASE  001256            53#     90*     91      93      95      97     134
ADCNT   001270            61#     99*    138     177
ADDW0 = 000000            42
ADDW1 = 000000            42
ADDW10= 000000            42
ADDW11= 000000            42
ADDW12= 000000            42
ADDW13= 000000            42
ADDW14= 000000            42
ADDW15= 000000            42
ADDW2 = 000000            42
ADDW3 = 000000            42
ADDW4 = 000000            42
ADDW5 = 000000            42
ADDW6 = 000000            42
ADDW7 = 000000            42
ADDW8 = 000000            42
ADDW9 = 000000            42
ADEVCT= 000000            42
ADEVM = 000000            42
ADGO    003466           133     331     394#
ADPRI   004160           136     176     509#
ADRPOK  010514            72     216     233     569#
ADRTOP  010714           230     243     570#
AENV  = 000000            42
AENVM = 000000            42
AFATAL= 000000            42
AMADR1= 000000            42
AMADR2= 000000            42
AMADR3= 000000            42
AMADR4= 000000            42
AMAMS1= 000000            42
AMAMS2= 000000            42
AMAMS3= 000000            42
AMAMS4= 000C00            42
AMSGAD= 000000            42
AMSGLG= 000000            42
AMSGTY= 000000            42
AMTYP1= 000000            42
AMTYP2= 000000            42
AMTYP3= 000000            42
AMTYP4= 000000            42
APASS = 000000            42
APRIOR= 000000            42
APTCSU= 000040           559     564#
APTENV= 000001           559     564#
APTSIZ= 000200            84     564#
APTSPO= 000100           559     564#
```

```
ASWREG= 000000            42
ATESTN= 000000            42
ATPRI   004277           291     516#
AUNIT = 000000            42
AUSWR = 000000            42
AVECT1= 000000            42
AVECT2= 000000            42
BEGIN   001324            37      80#    558
BIT0  = 000001            13#    422
BIT00 = 000001            13#
BIT01 = 000002            13#
BIT02 = 000004            13#
BIT03 = 000010            13#
BIT04 = 000020            13#
BIT05 = 000040            13#
BIT06 = 000100            13#
BIT07 = 000200            13#
BIT08 = 000400            13#
BIT09 = 001000            13#
BIT1  = 000002            13#    485
BIT10 = 002000            13#
BIT11 = 004000            13#
BIT12 = 010000            13#
BIT13 = 020000            13#
BIT14 = 040000            13#
BIT15 = 100000            13#    214     229
BIT2  = 000004            13#
BIT3  = 000010            13#    483
BIT4  = 000020            13#
BIT5  = 000040            13#
BIT6  = 000100            13#    423
BIT7  = 000200            13#
BIT8  = 000400            13#    405
BIT9  = 001000            13#     81
BPTVEC= 000014            13#
CHAIN   000042            33#    108
CHB     004120           444*    460*   487     492#
CR    = 000015            13#    559
CRLF  = 000200            13#    105     559
DDISP = 177570            13#     42      84
DIBASE  001264            56#     95*     96*    159
DICNT   001276            64#    102*    163     189
DIGO    003544           158     411#
DIOFF   001252            48#     96
DIPRI   004216           161     188     512#
DISPLA  001142            42#     84*
DISPRE  001322            76#     84
DOBASE  001266            57#     97*     98*    167
DOCNT   001300            65#    103*    171     193
DOGO    003570           166     418#
DOOFF   001254            49#     98
DOPRI   004230           169     192     513#
DSWR  = 177570            13#     42      84
EMTVEC= 000030            13#
ERRVEC= 000004            13#     84*    211     217*    245*    266     269*    273*    328*
FIXVCT  003316           276     353#
```

```
FLAGAD  001316            74#    118*   121*   304    329*
FLAGMP  001314            73#    126*   129*   206    209*
GNS   = ****** U          105    566
HT    = 000011             13#   559
IOTRD   003234           273     334#
IOTVEC= 000020             13#
KWBASE  001260             54#    91*    92*   142
KWCNT   001272             62#   100*   146    181
KWGO    003504           141     400#
KWOFF   001246             46#    92
KWPRI   004172           144     180    510#
LD01CH  004122           439     496#
LF    = 000012             13#   559
LOOP    002072           107     113    128    132#   257
MDASH   004337           449     520#
MDIF    004362           468     522#
MNCMAP  004635           210     529#
MPRMP   004401           462     523#
MSE     004343           456     477    521#
MTCMP   004412           466     524#
NBEXT   001310             71#   267*   309*   316    327
NIPRI   004271           175     515#
NOPRI   004242           298     324    514#
OUTADR  001312             72#   213*   235    242*
OUTBIN  002564           238     239    240    241    252#
PIRQ  = 177772             13#
PIRQVE= 000240             13#
PR0   = 000000             13#
PR1   = 000040             13#
PR2   = 000100             13#
PR3   = 000140             13#
PR4   = 000200             13#
PR5   = 000240             13#
PR6   = 000300             13#
PR7   = 000340             13#
PS    = 177776             13#
PSW   = 177776             13#
PWRVEC= 000024             13#
QUESAD  004423           114     525#
QUESMP  004534           122     527#
RDCHR = 104407           111     115    123    558    566#
RDLIN = 104410           566#
RESVEC= 000C10             13#
SETMNC  005025           110     532#
SPACE1  004335           237     254    519#
SSPACE  004332           447     518#
STACK = 001100             13#    84
STKLMT= 177774             13#
SWR     001140             42#    84*
SWREG   001320             75#    84
SW0   = 000001             13#
SW00  = 000001             13#
SW01  = 000002             13#
SW02  = 000004             13#
SW03  = 000010             13#
SW04  = 000020             13#
```

```
SW05   = 000040            13#
SW06   = 000100            13#
SW07   = 000200            13#
SW08   = 000400            13#
SW09   = 001000            13#
SW1    = 000002            13#
SW10   = 002000            13#
SW11   = 004000            13#
SW12   = 010000            13#
SW13   = 020000            13#
SW14   = 040000            13#
SW15   = 100000            13#
SW2    = 000004            13#
SW3    = 000010            13#
SW4    = 000020            13#
SW5    = 000040            13#
SW6    = 000100            13#
SW7    = 000200            13#
SW8    = 000400            13#
SW9    = 001000            13#
TBITVE= 000014             13#
TCHANE  004116            491#
TCHANL  003630            306     439#
TEMP    001306             70#    265*    310*
TEST    002710            132     140     149     157     165     261#
TKVEC  = 000060            13#    558*
TPVEC  = 000064            13#
TRAPVE= 000034             13#     84*
TRTO    003314            334*    335*    336     340     350#
TRTVEC= 000014             13#
TYPBN  = 104406           253     566#
TYPDS  = 104405           257     566#
TYPE   = 104401           105     110     114     122     174     175     176     180     184     188     192     210     234
                          237     246     254     257     288     289     291     295     298     321     322     324     446
                          447     449     476     489     558     559     561     562     563     566#
TYPOC  = 104402           236     566#
TYPON  = 104404           566#
TYPOS  = 104403           178     182     186     190     194     293     301     448     475     566#
VADR    001302             68#    263*    312
VECT    001304             69#    272*    296     300     340*
VTPRI   004315            295     517#
WOWAGS  006047            489     554#
$APTHD  001C00             41#
$ASTAT= ****** U          564
$ATYC   010214            564#
$ATY1   010170            564#
$ATY3   010176            559     564#
$ATY4   010206            564#
$AUTOB  001134             42#
$BASE   001240             42#     90
$BDADR  001122             42#
$BDDAT  001126             42#    262*    270     284*    292     312*    341*    394*    400     404*    405*    411*    412*
                          418     423*    440*    441     450*    451     484*    485     496
$BELL   007040            558#
$BIN    010166            563#*
$CDW1   001244             42#
```

```
$CHARC  007436            559#*
$CKSWR= ****** U          566
$CMTAG  001100             42#     84
$CM3  = 000000             42#
$CNTLC  007044            558#
$CNTLG  007056            558#
$CNTLU  007051            558#
$CPUOP  001212             42#
$CRLF   001161             42#    174     234     246     288     321     446     558     559
$DBLK   010104            562#
$DEVCT  001174             42#
$DEVM   001242             42#
$DOAGN  002664            257#
$DTBL   010074            562#
$ENDAD  002654             39     105     257#
$ENDCT  002622             84     257#
$ENDMG  002673            257#
$ENULL  002670            257#
$ENV    001204             42#    559     564
$ENVM   001205             42#     84     106     202     559     564
$EOP    002576            208     247     257#
$EOPCT  002614             84*    257#
$ERFLG  001103             42#
$ERMAX  001115             42#
$ERRPC  001116             42#
$ERRTB  001246             42#
$ERTTL  001112             42#
$ETABL  001204             42#
$ETEND  001246             41      42#
$FATAL  001166             42#    564*
$FFLG   010434            564#*
$FILLC  001156             42#    559
$FILLS  001155             42#    559
$GDADR  001120             42#
$GDDAT  001124             42#
$GET42  002644            257#
$GTSWR= ****** U          566
$HD   = 000000             12
$HIBTS  001000             41#
$ICNT   001104             42#
$INTAG  001135             42#
$ITEMB  001114             42#
$LF     001162             42#    558     559
$LFLG   010433            564#*
$LPADR  001106             42#
$LPERR  001110             42#
$MADR1  001216             42#
$MADR2  001222             42#
$MADR3  001226             42#
$MADR4  001232             42#
$MAIL   001164             41      42#     84     559
$MAMS1  001214             42#
$MAMS2  001220             42#
$MAMS3  001224             42#
$MAMS4  001230             42#
$MBADR  001002             41#
```

```
$MFLG   010432            564#*
$MNEW   007074            558#
$MSGAD  001200             42#    564*
$MSGLG  001202             42#    564*
$MSGTY  001164             42#    564*
$MSWR   007063            558#
$MTYP1  001215             42#
$MTYP2  001221             42#
$MTYP3  001225             42#
$MTYP4  001231             42#
$NULL   001154             42#    559
$OCNT   007664            561#*
$OMODE  007666            561#*
$PASS   001172             42#     84*   257*    286     318
$PASTM  001006             41#
$QUES   001160             42#    558    559
$RDCHR  006454            558#    566
$RDDEC= ****** U          566
$RDLIN  006544            558#    566
$RDOCT= ****** U          566
$RDSZ = 000040            558#
$RTNAD  002666            257#
$R2A  = ****** U          566
$SAVRE= ****** U          566
$SETUP= 000024             77#     84    105     257     558
$STUP = 177777             77#
$SVPC = 000204             39#
$SWR  = 000000              5#     12     42      84     257
$SWREG  001206             42#     84
$TESTN  001170             42#
$TKB    001146             42#    558    559
$TKCNT  006136            558#*
$TKINT  006204             89     558#
$TKQEN= 006204            558#
$TKQIN  006140            558#*
$TKQOU  006142            558#*
$TKQSR  006144            558#
$TKS    001144             42#    558*   559
$TKSRV  006254            558#
$TN   = 000001              6#     12
$TPB    001152             42#    559*
$TPFLG  001157             42#    559
$TPS    001150             42#    559
$TRAP   010436             84     566#
$TRAP2  010460            566#
$TRP  = 000011            566#
$TRPAD  010472            566#
$TSTM   001004             41#
$TSTNM  001102             42#
$TTYIN  007000            558#
$TYPBN  010114            563#    566
$TYPDS  007670            562#    566
$TYPE   007106             85*    86*    87*    88*    559#   564    566
$TYPEC  007320            559#
$TYPEX  007440            559#
$TYPOC  007466            561#    566
```

```
$TYPON  007502            561#    566
$TYPOS  007442            561#    566
$UNIT   001176             42#
$UNITM  001010             41#
$USWR   001210             42#
$VECT1  001234             42#
$VECT2  001236             42#
$XOFF = 000023            559
$XON  = 000021            559
$$GET4= 000000            257#
$OFILL  007665            561#*
.     = 010716             15#     20#     31      32#     36#     39#     40#     41#     42#     84      105#    257     558#
                          559     562#    564#    569#
.$ASTA= ****** U          564
.$X   = 001000             41#
```

```
ASCIZ       8#
COMMEN     13#
ENDCOM     13#
ERROR      13#
ESCAPE     13#
GETPRI     13#
GETSWR     13#   105#
MULT       13#
NEWTST     13#
POP        13#   562    564
PUSH       13#   562    564
REPORT     13#
SCOPE      13#
SETPRI     13#   558
SETTRA    566#
SETUP      13#    84
SKIP       13#   311    313    455    473    480    486
SLASH      13#
SPACE      13#
STARS      13#    39     41     42    257    558    559    561    562    563    564    566
SWRSU      13#    84#
TRMTRP    566#
TYPBIN     13#
TYPDEC     13#   257
TYPNAM     13#   105
TYPNUM     13#
TYPOCS     13#   448    475
TYPOCT     13#
TYPTXT     13#
$$CMRE     42#
$$CMTM     42#
$$ESCA     13#
$$NEWT     13#
$$SET     566#
$$SETM     84#
$$SKIP     13#
.EQUAT      7#    13
.HEADE      7#    12
.SETUP      9#    77
.SWRHI      9#
.$ACT1     10#    39
.$APTB     10#    42#
.$APTH     10#    41
.$APTY     10#   564
.$CATC      7#
.$CMTA      7#    42
.$EOP       7#   257
.$ERRO      7#
.$ERRT      9#
.$PARM      8#
.$POWE      8#
.$RAND     10#
.$RDOC     10#
.$READ      8#   558
.$SAVE      8#
.$SCOP      8#
```

```
.$SPAC      9#
.$SWDO      9#
.$TRAP      9#    566
.$TYPB      8#    563
.$TYPD     10#    562
.$TYPE      9#    559
.$TYPO      8#    561


. ABS.  010716     000     OVR   RW    ABS   GBL   D


 ERRORS DETECTED:  0

 CVMNFC,CVMNFC/CRF=CVMNFC
 RUN-TIME: 17 4 .6 SECONDS
 RUN-TIME RATIO: 374/22=16.3
 CORE USED:  25K  (49 PAGES)
```