M9312

BOOTSTRAP TERMINATOR 8K

CZM9BB0

AH-E058B-MC

FICHE 1 OF 1

AUG 1978

digital

MADE IN USA

```
        REM '
```

IDENTIFICATION

PRODUCT CODE: AC-E057B-MC

PRODUCT NAME CZM9BB0 M9312 BOOT TERMR 8K

DATE JUNE, 1978

MAINTAINER DIAGNOSTIC GROUP

HISTORY SECTION

CZM98A0 WAS RELEASED MARCH, 1978
CZM98B0 WAS RELEASED JUNE, 1978

REVISION B WAS CREATED TO PROVIDE THE FOLLOWING ENHANCEMENTS·

1 PROPERLY CHECK THE BOOT ROM'S ALPHABETIC SEQUENCE AND, IF NOT IN CORRECT SEQUENCE, PRINT THE CORRECT SEQUENCE AS AN ERROR MESSAGE. ALSO CHECK FOR NO HOLES AND CHECK FOR ROM IN SOCKET #2 IF 11/60 AND ONLY ONE ROM EXISTS, ELSE PRINT THE CORRECT SEQUENCE

2 THE DIAGNOSTIC CANNOT DETERMINE THE DEVICE CODE FOR A CONTINUATION ROM THEREFORE, CONTINUATION ROMS ARE TREATED AS EXTENSIONS OF THE PRECEEDING DEVICE CODE ROM ILLEGAL PLACEMENT OF CONTINUATION ROMS ARE REPORTED IN ERROR MESSAGES. A DUPLICATE DEVICE ROM IS ALSO REPORTED IN AN ERROR MESSAGE.

COMMENT FIELDS BEGINNING WITH THE CHARACTERS ";+" IDENTIFY ALL LINES ADDED OR MODIFIED BY REV B0

```
;
 REM ;
```

1 0 ABSTRACT

THIS PROGRAM VERIFIES THE ROM INFORMATION FOR THE M9312 BOOTSTRAP TERMINATOR IT HAS TWO MODES OF OPERATION, STAND-ALONE MODE WHICH REQUIRES OPERATOR INTERVENTION AND APT-MODE

2 0 REQUIREMENTS

2 1 HARDWARE

ANY PDP-11 UNIBUS PROCESSOR WITH CONSOLE TERMINAL AND/OR HARDWARE SWITCH REGISTER
M9312 BOOT STRAP TERMINATOR
4K MEMORY

2.2 SOFTWARE

THIS PROGRAM REQUIRES THAT THE CORRECT OPERATION OF THE PROCESSOR, MEMORY AND CONSOLE TERMINAL HAVE BEEN VERIFIED BY THE APPROPRIATE DIAGNOSTICS

3 0 LOADING AND STARTING PROCEDURES

3 1 LOAD THE PROGRAM BY ANY OF THE STANDARD PROCEDURES FOR ABSOLUTE PROGRAM FORMATS

3 2 STARTING ADDRESS

200- DO CRC VERIFICATION AND SIZING

3 3 RESTART ADDRESS

204- RESTART WITHOUT SIZING

3 4 SPECIAL ENVIRONMENTS

THIS PROGRAM IS APT COMPATIBLE SEE SECTION FOR ETABLE SET-UP
FOLLOW STANDARD APT PROCEDURES FOR LOADING AND STARTING

4 0 OPERATING PROCEDURES

4 1 OPERATIONAL SWITCH SETTINGS

THE PROGRAM IS DESIGNED TO USE THE HARDWARE SWITCH REGISTER,
HOWEVER IF THIS REGISTER IS NOT AVAILABLE THE PROGRAM WILL USE
LOCATION 176 AS THE SWITCH REGISTER

SW 15=1 OR UP-- HALT ON ERROR
SW 13=1 OR UP-- INHIBIT ERROR TYPEOUTS
SW 10=1 OR UP-- BELL ON ERROR

4 2 EXECUTION TIMES

EXECUTION TIME IS DEPENDENT ON THE CONSOLE TERMINAL DURING THE
FIRST PASS. ALL OTHER PASSES TAKE LESS THEN 1 SECOND

4 3 APT PROCEDURES

THERE ARE TWO CHOICES WHEN RUNNING UNDER APT IF THE SIZE BIT
(BIT 7-$ENVM) IS CLEAR THE PROGRAM WILL OPERATE IN NORMAL
STAND-ALONE MODE IF THE SIZE BIT IS SET THE PROGRAM WILL
COMPARE PARAMETERS FROM THE BOARD TO THE CONTENTS OF THE ETABLE
TO USE THE APT COMPARSION FEATURE THE ETABLE MUST BE SET UP IN
THIS ORDER,

$ENV	DON'T CARE
$ENVM	200 OR 240

```
                $SWREG          DON'T CARE
                $USWR.          NOT USED
                $CPUOP          NOT USED
                $MAMS1          NOT USED
                $MTYP1          NOT USED
                $MADR1          NOT USED
                $MAMS2.         NOT USED
                $MTYP2:         NOT USED
                $MADR2          NOT USED
                $MAMS3          NOT USED
                $MTYP3          NOT USED
                $MADR3          NOT USED
                $MAMS4          NOT USED
                $MAMS4          NOT USED
                $MTYP4          NOT USED
                $MADR4          NOT USED
                $VECT1          NOT USED
                $VECT2          NOT USED
                $BASE           PSEUDO POWER-FAIL VECTOR ADDRESS
                $DEVM           NOT USED
                $CDW1.          CONTENTS OF ADDRESS IN $BASE
                $CDW2.          NOT USED
                $DDW0:          DEVICE CODES EXPECTED
                                FIRST LETTER/SECOND LETTER
                DDW15.
```

5 0 SUBROUTINE ABSTRACTS

5 1 NOROMS

THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND DURING SIZING
IT DOES A READ OF ALL BOOTSTRAP ROM ADDRESSES AND COMPARES THE
CONTENTS TO A KNOWN EXPECTED VALUE

5 2 CHECKS

THIS ROUTINE SETS UP THE FIRST, LAST AND EXCEPTION ADDRESSES FOR
THE "CALSUM" SUBROUTINE IT RECIEVES THE CALCULATED CHECKSUM FROM
"CALSUM" AND COMPARES IT AGAINST THE GOOD CHECKSUM TO DETERMINE
CRC ERRORS

5 3 CALSUM

THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF EACH ROM IT
RECEIVES THE FIRST ADDRESS TO BE CHECKED(FIRSTA), THE LAST
ADDRESS TO BE CHECKED(LASTAD) AND THE EXCEPTION ADDRESS (EXCADD)
FROM THE "CHECKS" MODULE AND RETURNS TO IT THE CHECKSUM IN R4

5 4 PROMP

THIS ROUTINE PROCESSES THE ROM PARAMETERS IT CHECKS THE SIZE/ DON'T SIZE BIT IN THE APT ETABLE AND EITHER FORMATS THE SIZING MESSAGES OR COMPARES THE SIZING INFORMATION TO THE APT ETABLE DATA

5 5 DEVCOD

THIS ROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE ADDRESS IN WHICH IT WAS FOUND BACK TO THE "PROMP" MODULE

5 6 PPFVAR

THIS ROUTINE DETERMINES THE PSEUDO POWER-FAIL VECTOR ADDRESS AND PASSES IT AND ITS CONTENTS TO THE "PROMP" MODULE

5 7 PUTMES

THIS ROUTINE FORMATS THE PARAMETER AND POWER-FAIL ADDRESS MESSAGES.

5 8 ERRHAN

THIS ROUTINE FORMATS ALL ERROR MESSAGES AND CALLS THE TYPE ROUTINE TO OUTPUT THEM IT ALSO USES THE OPERATIONAL SWITCHES TO DETERMINE WHETHER TO OUTPUT THE MESSAGE, OR HALT

5 9 FILBUF

THIS ROUTINE FILLS THE MESSAGE BUFFER WITH ASCII CHARACTERS IT RECIEVES THE ADDRESS OF THE ASCII IN R5

5 10 OCASC

THIS ROUTINE TAKES A SIXTEEN BIT BINARY NUMBER AND CONVERTS IT TO 6 ASCII CHARACTERS

5.11 OCADD

THIS ROUTINE IS USED BY THE "PUTMES" MODULE WHEN THE DATA IN R3 IS NOT IN THE RIGHT MODE TO BE HANDLED BY THE "OCASC" MODULE IT MOVES THE ADDRESSING MODE OF R3 UP ONE LEVEL OF DEFERMENT

6 0 RELIABILITY//AVAILABILITY/SERVICEABILITY

WHEN RUNNING IN ANY ENVIRONMENT BUT APT THERE IS ONLY ONE ERROR DETECTED BY THE PROGRAM THIS ERROR IS A CHECKSUM ERROR THE MESSAGE WILL BE

CRC ERROR IN ROM EXX

THE FOLLOWING ERROR MESSAGES WERE ADDED IN REV BO

1 A CONTINUATION ROM IS INCORRECTLY LOCATED IN ROM X(EXX)
2 A CONTINUATION ROM IS MISSING FOR DEVICE CODE XX
3 THERE IS A DUPLICATE ROM WITH DEVICE CODE XX
4 ROM SEQUENCE IS INCORRECT AS PER INSTALLATION PROCEDURE

SEQUENCE SHOULD BE

ROM 1(E35) XX
ROM 2(E33) XX
ROM 3(E34) XX
ROM 4(E32) XX

WHEN RUNNING IN THE APT ENVIRONMENT THREE OTHER ERRORS MAY OCCUR

1 AN ERROR WILL OCCUR WHEN THE DEVICE CODES IN THE ETABLE DO NOT MATCH THE DEVICE CODES FOUND IN THE ROMS THE ERROR MESSAGE WILL BE EITHER

1 COULD NOT FIND DEVICE CODE XX

2. FOUND UNEXPECTED DEVICE CODE XX

THE FIRST MESSAGE WILL BE PASSED TO APT IF A DEVICE CODE LISTED IN THE ETABLE CANNOT BE FOUND IN THE EXISTING ROMS THE SECOND MESSAGE WILL BE SENT IF A DEVICE CODE NOT LISTED IN THE ETABLE IS FOUND IN A ROM.

2 THE SECOND ERROR WILL BE IF THE PSEUDO POWER-FAIL VECTOR ADDRESS IN THE ETABLE DOES NOT MATCH THE ADDRESS DETERMINED BY THE PROGRAM. TO BE IN THE BOARDS' SWITCHES THE MESSAGE WILL BE:

POWER-FAIL VECTOR

EXPECTED	RECEIVED
--------	--------

WHERE EXPECTED IS THE CONTENTS OF THE ETABLE AND RECEIVED IS THE VALUE FOUND BY THE PROGRAM.

3 THE THIRD ERROR WILL BE IF THE CONTENTS OF BOARD'S PSEUDO POWER-FAIL VECTOR ADDRESS DOES NOT MATCH THE VALUE EXPECTED FROM THE ETABLE THE MESSAGE WILL BE

POWER-FAIL DATA ERROR
EXPECTED RECEIVED
-------- --------

```
                                ENABLE ABS
                                LIST   ME
                                NLIST  MC,MD,CND
                                TITLE  CZM9BBO M9312 BOOT TERMR 8K
                               ,*COPYRIGHT (C) 1978
                               ,*DIGITAL EQUIPMENT CORP
                               ,*MAYNARD, MASS. 01754
                               ,*
                               ,*PROGRAM BY BARRY G. IRRGANG
                               ,*
                               ,*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                               ,*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977
                               ,*
000001                         $TN=1
160000                         $SWR=160000      ,,HALT ON ERROR, LOOP ON TEST, INHIBIT .RROR TYPOUT
                                SBTTL  OPERATIONAL SWITCH SETTINGS
                               ,*
                               ,*      SWITCH                  USE
                               ,*      ------          --------------------
                               ,*        15            HALT ON ERROR
                               ,*        14            LOOP ON TEST
                               ,*        13            INHIBIT ERROR TYPEOUTS
                                SBTTL  BASIC DEFINITIONS

                               ,*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                         STACK=  1100
                                EQUIV  EMT,ERROR        ,,BASIC DEFINITION OF ERROR CALL
                                EQUIV  IOT,SCOPE        ,,BASIC DEFINITION OF SCOPE CALL

                               ,*MISCELLANEOUS DEFINITIONS
000011                         HT=     11              ,,CODE FOR HORIZONTAL TAB
000012                         LF=     12              ,,CODE FOR LINE FEED
000015                         CR=     15              ,,CODE FOR CARRIAGE RETURN
000200                         CRLF=   200             ,,CODE FOR CARRIAGE RETURN-LINE FEED
177776                         PS=     177776          ;,PROCESSOR STATUS WORD
                                EQUIV  PS,PSW
177774                         STKLMT= 177774          ,,STACK LIMIT REGISTER
177772                         PIRQ=   177772          ,,PROGRAM INTERRUPT REQUEST REGISTER
177570                         DSWR=   177570          ,,HARDWARE SWITCH REGISTER
177570                         DDISP=  177570          ,,HARDWARE DISPLAY REGISTER

                               ,*GENERAL PURPOSE REGISTER DEFINIT.ONS
000000                         R0=     %0              ,,GENERAL REGISTER
000001                         R1=     %1              ,,GENERAL REGISTER
000002                         R2=     %2              ,,GENERAL REGISTER
000003                         R3=     %3              ,;GENERAL REGISTER
000004                         R4=     %4              ,;GENERAL REGISTER
000005                         R5=     %5              ;,GENERAL REGISTER
000006                         R6=     %6              ,,GENERAL REGISTER
000007                         R7=     %7              ,,GENERAL REGISTER
000006                         SP=     %6              ,,STACK POINTER
000007                         PC=     %7              ,,PROGRAM COUNTER

                               ,*PRIORITY LEVEL DEFINITIONS
000000                         PR0=    0               ,,PRIORITY LEVEL 0
```

```
                        PR1=    40              ,;PRIORITY LEVEL 1
                        PR2=    100             ;,PRIORITY LEVEL 2
                        PR3=    140             ,,PRIORITY LEVEL 3
                        PR4=    200             ;,PRIORITY LEVEL 4
                        PR5=    240             ,,PRIORITY LEVEL 5
                        PR6=    300             ,,PRIORITY LEVEL 6
                        PR7=    340             ,,PRIORITY LEVEL 7

                        ,*"SWITCH REGISTER" SWITCH DEFINITIONS
                        SW15=   100000
                        SW14=   40000
                        SW13=   20000
                        SW12=   10000
                        SW11=   4000
                        SW10=   2000
                        SW09=   1000
                        SW08=   400
                        SW07=   200
                        SW06=   100
                        SW05=   40
                        SW04=   20
                        SW03=   10
                        SW02=   4
                        SW01=   2
                        SW00=   1
                         EQUIV  SW09,SW9
                         EQUIV  SW08,SW8
                         EQUIV  SW07,SW7
                         EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                         EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                         EQUIV  SW01,SW1
                         EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
                        .EQUIV  BIT09,BIT9
                         EQUIV  BIT08,BIT8
                         EQUIV  BIT07,BIT7
```

```
                        EQUIV  BIT06,BIT6
                        EQUIV  BIT05,BIT5
                        EQUIV  BIT04,BIT4
                        EQUIV  BIT03,BIT3
                        EQUIV  BIT02,BIT2
                        EQUIV  BIT01,BIT1
                        EQUIV  BIT00,BIT0

                       ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                 ERRVEC= 4              ;;TIME OUT AND OTHER ERRORS
000010                 RESVEC= 10             ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                 TBITVEC=14             ;;"T" BIT
000014                 TRTVEC= 14             ;;TRACE TRAP
000014                 BPTVEC= 14             ;;BREAKPOINT TRAP (BPT)
000020                 IOTVEC= 20             ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                 PWRVEC= 24             ;;POWER FAIL
000030                 EMTVEC= 30             ;;EMULATOR TRAP (EMT) **ERROR**
000034                 TRAPVEC=34             ;;"TRAP" TRAP
000060                 TKVEC=  60             ;;TTY KEYBOARD VECTOR
000064                 TPVEC=  64             ;;TTY PRINTER VECTOR
000240                 PIRQVEC=240            ;;PROGRAM INTERRUPT REQUEST VECTOR
                       ;;*************************************************************
                       ;*              PROGRAM EQUATES
                       ;;*************************************************************
000001                         APTER1 =  1
000002                         APTER2 =  2
000004                         APTER3 =  4
000010                         APTER4 = 10
000020                         CRCERR = 20
000040                         PFERR  = 40
000100                         NOROME =100
000200                         SEQERR =200
000400                         CONONE =400     ;+
001000                         CONTWO=1000     ;+
002000                         DUPERR=2000     ;+

000000                         .=0
                        .SBTTL  TRAP CATCHER

000000                         .=0
                       ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A " +2,HALT"
                       ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                       ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
000174                         .=174
000174  000000        DISPREG: .WORD  0       ;;SOFTWARE DISPLAY REGISTER
000176  000000        SWREG:   .WORD  0       ;;SOFTWARE SWITCH REGISTER
                        SBTTL  STARTING ADDRESS(ES)
000200  000137  001400        JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204                         .=204
000204  000167  001656        JMP     RSTART

                       .SBTTL  ACT11 HOOKS

                       ;;*************************************************************
                       ;HOOKS REQUIRED BY ACT11
000210                         $SVPC=.                 ;SAVE PC
```

```
                 000046                          =46
000046  002174                          $ENDAD                  ,,1)SET LOC 46 TO ADDRESS OF $ENDAD IN  $EOP
                 000052                          =52
000052  000000                          WORD   0                ,,2)SET LOC 52 TO ZERO
                 000210                          =$SVPC                  ,, RESTORE PC

                 001100                          =1100
001100     000  $AUTOB  .BYTE   0
001101     000  $INTAG:  BYTE   0
001102  000000          .WORD   0
001104  177570  SWR      WORD   DSWR
001106  177570  DISPLAY:  WORD  DDISP
001110  000000  ROMERR:  0
001112  000000  MESSAG   0
001114  173000  FIRSTB:  173000
001116  000000  ROMFIN:  0
001120  000000  ERRCNT   0

                 $BTTL  APT PARAMETER BLOCK

                ,,*****************************************************************
                ,SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                ,,*****************************************************************
                 001122         .$X=.    ,,SAVE CURRENT LOCATION
                 000024         .=24     ;,SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                  200      ;,FOR APT START UP
                 000044          =44     ;,POINT TO APT INDIRECT ADDRESS PNTR
000044  001122                  $APTHDR ,,POINT TO APT HEADER BLOCK
                 001122          = $X    ,,RESET LOCATION COUNTER
                ,,*****************************************************************
                ,SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                ,INTERFACE SPEC

001122                          $APTHD
001122  000000  $HIBTS  .WORD   0        ,;TWO HIGH BITS OF 18 BIT MAILBOX ADDR
001124  001136  $MBADR:  .WORD   $MAIL   ;,ADDRESS OF APT MAILBOX (BITS 0-15)
001126  000002  $TSTM:   .WORD   2       ,,RUN TIM OF LONGEST TEST
001130  000002  $PASTM:  .WORD   2       ,,RUN TIME IN SECS  OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001132  000000  $UNITM:  .WORD   0       ;,ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001134  000052          .WORD   $ETEND-$MAIL/2 ,,LENGTH MAILBOX-ETABLE(WORDS)
                 $BTTL  APT MAILBOX-ETABLE

                ,,*****************************************************************
                 EVEN
001136  $MAIL:                           ,,APT MAILBOX
001136  000000  $MSGTY:  .WORD   AMSGTY  ,,MESSAGE TYPE CODE
001140  000000  $FATAL:  .WORD   AFATAL  ;;FATAL ERROR NUMBER
001142  000000  $TESTN:  .WORD   ATESTN  ;;TEST NUMBER
001144  000000  $PASS:   .WORD   APASS   ,,PASS COUNT
001146  000000  $DEVCT:  .WORD   ADEVCT  ,,DEVICE COUNT
001150  000000  $UNIT:   .WORD   AUNIT   ,;I/O UNIT NUMBER
001152  000000  $MSGAD:  .WORD   AMSGAD  ;;MESSAGE ADDRESS
001154  000000  $MSGLG:  .WORD   AMSGLG  ,,MESSAGE LENGTH
001156  $ETABLE:                         ,;APT ENVIRONMENT TABLE
001156     000  $ENV     BYTE   AENV     ,,ENVIRONMENT BYTE
001157     000  $ENVM    BYTE   AENVM    ,,ENVIRONMENT MODE BITS
```

```
                                $SWREG:  WORD   ASWREG  ,;APT SWITCH REGISTER
                                $USWR    WORD   AUSWR   ,,USER SWITCHES
                                $CPUOP:  WORD   ACPUOP  ;,CPU TYPE,OPTIONS
                                ,*                      BITS 15-11=CPU TYPE
                                ,*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ,*                              11/70=06,PDQ=07,Q=10
                                ,*                      BIT *0=REAL TIME CLOCK
                                ,*                      BIT  9=FLOATING POINT PROCESSOR
                                ,*                      BIT  8=MEMORY MANAGEMENT
001166  000                   $MAMS1   BYTE   AMAMS1  ,,HIGH ADDRESS,M.S.  BYTE
001167  000                   $MTYP1   BYTE   AMTYP1  ;,MEM.  TYPE,BLK#1
                                ,*                      MEM.TYPE BYTE    --  (HIGH BYTE)
                                ,*                              900 NSEC CORE=001
                                ,*                              300 NSEC BIPOLAR=002
                                ,*                              500 NSEC MOS=003
001170  000000                $MADR1   WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ,*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001172  000                   $MAMS2   BYTE   AMAMS2  ;;HIGH ADDRESS,M.S.  BYTE
001173  000                   $MTYP2:  BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001174  000000                $MADR2.  WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001176  000                   $MAMS3:  BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001177  000                   $MTYP3:  BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001200  000000                $MADR3:  WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001202  000                   $MAMS4:  BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001203  000                   $MTYP4   BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001204  000000                $MADR4:  WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001206  000000                $VECT1:  WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001210  000000                $VECT2:  WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001212  000000                $BASE    WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001214  000000                $DEVM    WORD   ADEVM   ;;DEVICE MAP
001216  000000                $CDW1:   WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001220  000000                $CDW2.   WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001222  000000                $DDW0.   WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001224  000000                $DDW1:   WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001226  000000                $DDW2.   WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001230  000000                $DDW3    WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001232  000000                $DDW4    WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001234  000000                $DDW5:   WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001236  000000                $DDW6:   WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001240  000000                $DDW7    WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001242  000000                $DDW8    WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001244  000000                $DDW9:   WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001246  000000                $DDW10:  WORD   ADDW10  ,;DEVICE DESCRIPTOR WORD#10
001250  000000                $DDW11.  WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001252  000000                $DDW12:  WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001254  000000                $DDW13:  WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001256  000000                $DDW14:  WORD   ADDW14  ;,DEVICE DESCRIPTOR WORD#14
001260  000000                $DDW15   WORD   ADDW15  ;,DEVICE DESCRIPTOR WORD#15

001262                        $ETEND

        001400                          =1400
001400                        START
                                 SBTTL  INITIALIZE THE COMMON TAGS
001400  012706  001100                  MOV     #STACK,SP       ,,SETUP THE STACK POINTER
```

```
                                        ;;INITIALIZE A FEW VECTORS
001404  012737  006412  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001412  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001420  005067  177520                  CLR     $PASS            ;;CLEAR THE PASS COUNT
001424  016767  000520  000510          MOV     $ENDCT,$EOPCT    ;;SETUP END-OF-PROGRAM COUNTER
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER.  IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001432  013746  000004                  MOV     @#ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
001436  012737  001472  000004          MOV     #64$,@#ERRVEC    ;;SET UP ERROR VECTOR
001444  012767  177570  177432          MOV     #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
001452  012767  177570  177426          MOV     #DDISP,DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
001460  022777  177777  177416          CMP     #-1,@SWR         ;;TRY TO REFERENCE HARDWARE SWR
001466  001012                          BNE     66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                 ;;AND  THE HARDWARE SWR IS NOT = -1
001470  000403                          BR      65$              ;;BRANCH IF NO TIMEOUT
001472  012716  001500          64$     MOV     #65$,(SP)        ;;SET UP FOR TRAP RETURN
001476  000002                          RTI
001500  012767  000176  177376  65$     MOV     #SWREG,SWR       ;;POINT TO SOFTWARE SWR
001506  012767  000174  177372          MOV     #DISPREG,DISPLAY
001514  012637  000004          66$     MOV     (SP)+,@#ERRVEC   ;;RESTORE ERROR VECTOR

001520  005067  177420                  CLR     $PASS            ;;CLEAR PASS COUNT
001524  132767  000200  177425          BITB    #APTSIZE,$ENVM   ;;TEST USER SIZE UNDER APT
001532  001403                          BEQ     67$              ;;YES, USE NON-APT SWITCH
001534  012767  001160  177342          MOV     #$SWREG,SWR      ;;NO, USE APT SWITCH REGISTER
001542                          67$
                                 SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001542  005227  177777                  INC     #-1              ;;FIRST TIME?
001546  001046                          BNE     68$              ;;BRANCH IF NO
001550  022737  002174  000042          CMP     #$ENDAD,@#42     ;;ACT-11?
001556  001442                          BEQ     68$              ;;BRANCH IF YES
001560  104401  001626                  TYPE    ,69$             ;;TYPE ASCIZ STRING
                                 SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001564  005737  000042                  TST     @#42             ;;ARE WE RUNNING UNDER XXDP/ACT?
001570  001012                          BNE     70$              ;;BRANCH IF YES
001572  126727  177360  000001          CMPB    $ENV,#1          ;;ARE WE RUNNING UNDER APT?
001600  001406                          BEQ     70$              ;;BRANCH IF YES
001602  026727  177276  000176          CMP     SWR,#SWREG       ;;SOFTWARE SWITCH REG SELECTED?
001610  001005                          BNE     71$              ;;BRANCH IF NO
001612  104405                          GTSWR                    ;;GET SOFT-SWR SETTINGS
001616  000403                          BR      71$
001616  112767  000001  177254  70$     MOVB    #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
001624                          71$
001624  000417                          BR      68$              ;;GET OVER THE ASCIZ
                                ,,69$    ASCIZ  <CRLF>*CZM9BB0 M9312 BOOT TERMR 8K*<CRLF>
001664                          68$
001664  012700  007434                  MOV     #BUF1,          R0              ,CLR SIZING BUFFERS
001670  005020                  8$      CLR     (R0)+
001672  020027  007520                  CMP     R0,             #OCTBUF         ,HAVE WE CLEARED THE WHOLE
001676  002774                          BLT     8$                              ,BUFFER, NO, GO BACK
001700  005037  001116                  CLR     @#ROMFIN                        ,INITIALIZE ROM FOUND INDICATORS
001704  005037  003770                  CLR     @#TIMES                         ,INITIALIZE ENTRY COUNTER FOR PUTMES SUB
001710  004767  006714                  JSR     PC,     CLEAR                   ,+CLEAR SOME MORE LOCATIONS
001714  013746  000004                  MOV     @#ERRVEC        -(R6)           ,SAVE CONTENTS OF LOCATION 4
001720  012737  001762  000004          MOV     #2$             @#ERRVEC        ,+SET UP FOR POSSIBLE TRAP
```

```
674  001726  122737  177777  165000          CMPB    #-1,        @#165000    ;IF CONTENTS = -1, OR TRAP
675  001734  001414                          BEQ     3$                      ;THEN CPU ROM NOT PLUGGED IN
676  001736  012700  165000                  MOV     #165000,    R0          ;GET FIRST ADDRESS OF ROM SPACE
677  001742  005720                  1$      TST     (R0)+                   ;IF THIS INSTRUCTION TRAPS CPU ROM
678                                                                          ;NOT PLUGGED IN
679  001744  020027  166000                  CMP     R0,         #166000     ;HAVE WE CHECKED ALL ADDRESSES
680  001750  001374                          BNE     1$                      ;IF NO THEN CONTINUE LOOPING
681  001752  012737  000001  001116          MOV     #1,         @#ROMFIN    ;IF NO TRAPS OR LOW BYTE OF FIRST
682                                                                          ;ADDRESS NOT EQUAL -1 ASSUME ROM PRESENT
683  001760  000402                          BR      3$                      ;NO TRAP JUST RESTORE ERRVEL
684  001762  062706  000004          2$:     ADD     #4,         R6          ;+IF TRAP CLEAN OFF PC, PSW
685  001766  012637  000004          3$:     MOV     (R6)+,      @#ERRVEC    ;RESTORE ERRVEC
686  001772  012700  000002                  MOV     #2,         R0          ;INITIALIZE ROM FOUND INDICATOR
687  001776  013737  001114  003324          MOV     @#FIRSTB,   @#TESTAD    ;GET FIRST BOOT
688  002004  132777  000200  001312  5$:     BITB    #200,       @TESTAD     ;IF LOW BYTE HAS BIT 7 SET
689  002012  001411                          BEQ     7$                      ;DO NOT SET ROM FOUND INDICATOR
690  002014  022777  177776  001302          CMP     #-2,        @TESTAD     ;UNLESS, CONTENTS OF ADDRESS
691  002022  001011                          BNE     4$                      ;EQUAL TO -2 (CONTINUATION ROM)
692  002024  050067  007774                  BIS     R0,     CONFIN          ;+SET CONTINUATION ROM FOUND INDICATOR
693  002030  050067  177062                  BIS     R0,     ROMFIN          ;+SET ROM FOUND INDICATOR
694  002034  000404                          BR      4$                      ;+CONTINUE
695  002036  050037  001116          7$      BIS     R0,         @#ROMFIN    ;SET ROM FOUND INDICATOR
696  002042  050067  007760                  BIS     R0,     DEVFIN          ;+SET DEVICE ROM FOUND INDICATOR
697  002046  062737  000200  003324  4$:     ADD     #200,       @#TESTAD    ;UPDATE TEST ADDRESS TO NEXT ROM
698  002054  006100                          ROL     R0                      ;UPDATE ROM FOUND INDICATOR
699  002056  022737  174000  003324          CMP     #174000,    @#TESTAD    ;HAVE CHECKED ALL THE ROMS
700  002064  001347                          BNE     5$                      ;IF NO CONTINUE CHECKING
701  002066  005737  001116          RSTART  TST     @#ROMFIN                ,ARE THERE ANY ROMS AVAILABLE
702  002072  001003                          BNE     6$                      ;IF YES GO TO ROM TEST
703  002074  004767  000126                  JSR     PC,         NOROMS      ;IF NO GO TEST NO ROMS
704  002100  000411                          BR      $EOP                    ,GO TO END OF PASS
705  002102  004767  000202          6$      JSR     PC,         CHECKS      ;GO CALCULATE CHECKSUMS
706  002106  005737  001144                  TST     @#$PASS                 ,ARE WE ON 1ST PASS
707  002112  001004                          BNE     $EOP                    ,IF NO SKIP PROCESS OF PARAMETERS
708  002114  004767  000504                  JSR     PC,         PROMP       ,GO PROCESS ROM PARAMETERS
709  002120  004767  007004                  JSR     PC,         SEQTST      ,GO CHECK SEQUENCE OF DEVICE CODES
710                                   SBTTL   END OF PASS ROUTINE
711
712                                  ,;**********************************************************************
713                                  ;*INCREMENT THE PASS NUMBER ($PASS)
714                                  ,*TYPE "END PASS"
715                                  ,*IF THERES A MONITOR GO TO IT
716                                  ,*IF THERE ISN'T JUMP TO RSTART
717                                  ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
718                                  ,*SENDMG CAN BE CHANGED TO 7
719
720  002124                          $EOP
721  002124  000240                          NOP
722  002126  005267  177012                  INC     $PASS               ,,INCREMENT THE PASS NUMBER
723  002132  042767  100000  177004          BIC     #100000,$PASS       ,,DON'T ALLOW A NEG  NUMBER
724  002140  005327                          DEC     (PC)+               ,,LOOP?
725  002142  000001                  $EOPCT   WORD    1
726  002144  003017                          BGT     $DOAGN              ,,YES
727  002146  012737                          MOV     (PC)+,@(PC)+        ,,RESTORE COUNTER
728  002150  000001                  $ENDCT   WORD    1
729  002152  002142                          $EOPCT
```

```
                                        TYPE    ,$ENDMG         ,,TYPE "END PASS"
                                        TYPE    ,$ENULL         ,,TYPE A NULL CHARACTER
                        $GET42          MOV     @#42,R0         ,,GET MONITOR ADDRESS
                                        BEQ     $DOAGN          ,,BRANCH IF NO MONITOR
                                        RESET                   ;,CLEAR THE WORLD
                        $ENDAD          JSR     PC,(R0)         ;;GO TO MONITOR
                                        NOP                     ;;SAVE ROOM
                                        NOP                     ;;FOR
                                        NOP                     ,,ACT11
                        $DOAGN
                                        JMP     @(PC)+          ,,RETURN
                        $RTNAD          WORD    RSTART
                        $ENULL          BYTE    -1,-1,0         ,,NULL CHARACTER STRING
                        $ENDMG          .ASCIZ  <15><12>/END PASS/

                        ;NO ROMS TEST
                        ,THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND
                        ,DURING SIZING  IT DOES A READ OF ALL BOOTSTRAP ROM
                        ,ADDRESSES AND COMPARES THE CONTENTS TO A KNOWN
                        ,EXPECTED VALUE

                        NOROMS  MOV     #173000,        R3              ,GET FIRST ADDRESS TO BE READ
                        1$      MOV     @#NODATA,       R4              ,GET ADDRESS OF EXPECTED VALUE
                                CMPB    (R3),           R4              ,+DOES RECIEVED VALUE EQUAL EXPECTED
                                BEQ     2$                              ,IF YES CONTINUE TESTING
                                MOV     (R3),   R4                      ,+GET BAD VALUE
                                BIS     #NOROME,        @#ROMERR        ,IF NO SET ERROR INDICATER
                                JSR     PC,             ERRHAN          ,REPORT ERROR
                        2$      ADD     #2,     R3                      ,+GET TO NEXT EVEN ADDRESS
                                CMP     #173024,        R3              ,+AT A VECTOR ADDRESS?
                                BEQ     2$                              ,+BRANCH IF YES
                                CMP     #173224,        R3              ,+AT A VECTOR ADDRESS?
                                BEQ     2$                              ,+BRANCH IF YES
                                CMP     #174000,        R3              ,HAVE ALL ADDRESSES BEEN TESTED
                                BGT     1$                              ,IF NO GO TEST THIS ADDRESS
                                RTS     PC

                        NODATA  WORD    777                     ,+A HOLE LOOKS LIKE XXX777


                        ,CHECKSUM ROMS MODULE
                        ,THIS ROUTINE DOES A CHECKSUM OF ALL ROMS PRESENT

                        CHECKS  MOV     #1,             @#ROMCNT        ,INITIALIZE ROM COUNTER
                                BIT     @#ROMCNT,       @#ROMFIN        ,IS DIAGNOSTIC ROM PRESENT
                                BEQ     1$                              ,IF NO GO CHECKSUM BOOT ROMS
                                MOV     #165775,        @#LASTAD        ,SET UP LAST ADDRESS TO BE SUMMED
                                MOV     #0,             @#EXCADD        ,SET UP EXCEPTION ADDRESS
                                MOV     #165000,        @#FIRSTA        ,SET UP FIRST ADDRESS TO BE SUMMED
                                JSR     PC,             CALSUM          ,GO CALCULATE CHECKSUM
                                MOV     @#165776,       R3              ,GET EXPECTED CHECKSUM
```

```
786  002360  020304                          CMP   R3,          R4          ;COMPARE CHECKSUMS
787  002362  001405                          BEQ   1$                        ;IF CHECKSUMS COMPARE CONTINUE
788  002364  052737  000020  001110          BIS   #CRCERR,     @#ROMERR    ;IF ERROR SET INDICATER
789  002372  004767  001374                  JSR   PC,          ERRHAN      ;REPORT ERROR
790  002376  012737  173000  002616  1$      MOV   #173000,     @#FIRSTA    ;SET UP FIRST ADDRESS TO BE SUMMED
791  002404  012737  173024  002620          MOV   #173024,     @#EXCADD    ;SET UP EXCEPTION ADDRESS
792  002412  012737  173175  002622          MOV   #173175,     @#LASTAD    ;SET UP LAST ADDRESS TO BE SUMMED
793  002420  012702  173176                  MOV   #173176,     R2          ;GET ADDRESS OF GOOD DATA
794  002424  006137  002524          2$      ROL   @#ROMCNT                  ;UPDATE ROM COUNTER
795  002430  033737  002524  001116          BIT   @#ROMCNT,    @#ROMFIN    ,IS ROM PRESENT
796  002436  001412                          BEQ   3$                        ,IF NO, GO UPDATE TO NEXT ROM
797  002440  004767  000062                  JSR   PC,          CALSUM      ,IF YES GO CALCULATE CHECKSUM
798  002444  011203                          MOV   (R2),        R3          ;GET EXPECTED CHECKSUM
799  002446  020304                          CMP   R3,          R4          ,COMPARE CHECKSUMS
800  002450  001405                          BEQ   3$                        ,IF CHECKSUMS EQUAL CONTINUE
801  002452  052737  000020  001110          BIS   #CRCERR,     @#ROMERR    ,IF ERROR SET INDICATER
802  002460  004767  001306                  JSR   PC,          ERRHAN      ,REPORT ERROR
803  002464  062737  000200  002616  3$      ADD   #200,        @#FIRSTA    ,UPDATE FIRST ADDRESS
804  002472  062737  000200  002620          ADD   #200,        @#EXCADD    ,UPDATE EXCEPTION ADDRESS
805  002500  062737  000200  002622          ADD   #200,        @#LASTAD    ;UPDATE LAST ADDRESS
806  002506  062702  000200                  ADD   #200,        R2          ,UPDATE ADDRESS OF GOOD DATA
807  002512  022737  174000  002616          CMP   #174000,     @#FIRSTA    ,HAVE ALL ROMS BEEN CHECKED
808  002520  001341                          BNE   2$                        ,IF NO GO CHECKSUM NEXT ONE
809  002522  000207                          RTS   PC                        ,IF YES RETURN
810
811  002524  000000                  ROMCNT  0
812
813
814                                  ,CALCULATE CHECKSUMS MODULE
815                                  ,THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF THE CONTENTS
816                                  ,OF EVERY LOCATION EXCEPT THE EXCEPTION ADDRESS AND LAST ADDRESS
817                                  ,OF EVERY ROM
818
819
820  002526  010246                  CALSUM  MOV   R2,          -(SP)       ,SAVE R2
821  002530  013700  002616                  MOV   @#FIRSTA,    R0          ,GET STARTING ADDRESS
822  002534  005004                          CLR   R4                        ,INITIALIZE CRC WORD
823  002536  012005                  LOOP    MOV   (R0)+,       R5          ,GET A BYTE
824  002540  012702  000020                  MOV   #16 ,        R2          ,SET BYTE COUNT
825  002544  000241                  CRCLOP  CLC                             ,THE NEXT NINE LINES
826  002546  006004                          ROR   R4                        ,DO THE MATH CALCULATIONS
827  002550  006005                          ROR   R5
828  002552  102006                          BVC   1$
829  002554  012701  120001                  MOV   #120001,     R1
830  002560  040401                          BIC   R4,          R1
831  002562  042704  120001                  BIC   #120001,     R4
832  002566  050104                          BIS   R1,          R4
833  002570  005302                  1$      DEC   R2
834  002572  003364                          BGT   CRCLOP
835  002574  020037  002620                  CMP   R0,          @#EXCADD    ,IS NEXT ADDRESS AN EXCEPTION ADDRESS
836  002600  001001                          BNE   2$                        ,IF NO TEST IT
837  002602  005720                          TST   (R0)+                     ,IF YES SKIP ADDRESS
838  002604  020037  002622          2$      CMP   R0,          @#LASTAD    ,HAVE ALL LOCATIONS BEEN SUMMED
839  002610  101752                          BLOS  LOOP                      ,IF NO CONTINUE
840  002612  012602                          MOV   (SP)+,       R2          ,RESTORE R2
841  002614  000207                          RTS   PC                        ,IF YES RETURN
```

```
                                FIRSTA  0
                                EXCADD  0
                                LASTAD  0


                                ;PROCESS ROM PARAMETERS MODULE
                                ;THIS ROUTINE DETERMINES IF SIZING IS TO BE DONE   IF IT IS THE
                                ;MODULE WILL GET THE PARAMETERS FROM THE DEVCOD AND PPFVAR
                                ;MODULES AND ASSEMBLE MESSAGES TO BE OUTPUT   IF SIZING IS
                                ;NOT TO BE DONE THIS MODULE WILL TAKE THE PARAMETERS RECEIVED
                                ;AND COMPARE THEM TO THE VALUES SUPPLIED IN THE ETABLE

002624  105737  001157    PROMP   TSTB    @#SENVM                 ;IF BIT IS CLEAR SIZE
002630  100424                    BMI     1$                      ;IF BIT IS SET DON'T SIZE
002632  004767  000256            JSR     PC,             DEVCOD  ;CALL DEVICE CODE MODULE
002636  004767  000576            JSR     PC,             PUTMES  ;FORMAT PARAMETER MESSAGE
002642  007530                    MES1
002644  004767  000460            JSR     PC,             PPFVAR  ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
002650  032737  000040  001110    BIT     #PFERR,         @#ROMERR ;WAS THERE AN ERR
002656  001403                    BEQ     10$                     ;IF NO GO FORMAT MESSAGE
002660  004767  001106            JSR     PC,             ERRHAN  ;IF YES GO REPORT MESSAGE
002664  000512                    BR      9$
002666  004767  000546    10$     JSR     PC,             PUTMES  ;FORMAT PSEUDO POWER-FAIL VECTOR
002672  010130                    MES2                            ;ADDRESS MESSAGE
002674  005237  001112            INC     @#MESSAG                ;SET MESSAGE INDICATOR
002700  000504                    BR      9$                      ;GO TO EXIT
002702  004767  000206    1$      JSR     PC,             DEVCOD  ;CALL GET DEVICE CODE MODULE
002706  012703  007434            MOV     #BUF1,          R3      ;GET BOARD DEVICE CODES
002712  012704  001222    2$      MOV     #$DDWO,         R4      ;GET ETABLE DEVICE CODE PARAMETERS
002716  062703  000002            ADD     #2,             R3      ;GET TO BOARD DEVICE CODE
002722  012402    3$      MOV     (R4)+,          R2      ;GET ETABLE DEVICE CODE
002724  000302                    SWAB    R2                      ;REFORMAT ASCII
002726  020213                    CMP     R2,             (R3)    ;DO THE TWO DEVICE CODES COMPARE
002730  001407                    BEQ     4$                      ;IF YES GET NEXT BOARD DEVICE CODE
002732  005714                    TST     (R4)                    ;IF NO HAVE WE CHECKED THE WHOLE ETABLE
002734  001372                    BNE     3$                      ;IF WE HAVEN'T CHECK NEXT ETABLE ENTRY
002736  052737  000001  001110    BIS     #APTER1,        @#ROMERR ;IF WE HAVE THEN DEVICE CODE ON BOARD
002744  004767  001022            JSR     PC,             ERRHAN  ;DOES NOT  EXIST IN ETABLE
002750  062703  000002    4$      ADD     #2,             R3      ;UPDATE TO NEXT ADDRESS
002754  005713                    TST     (R3)                    ;IF CONTENTS EQUALS ZERO WE'RE DONE
002756  001355                    BNE     2$                      ;IF CONTENTS NO EQUAL ZERO CONTINUE
002760  012703  001222            MOV     #$DDWO,         R3      ;GET BOARD DEVICE CODES
002764  012704  007432    5$      MOV     #BUF1-2,        R4      ;GET ETABLE DEVICE CODES
002770  000313                    SWAB    (R3)                    ;FORMAT ASCII
002772  062704  000004    6$      ADD     #4,             R4      ;GET BOARD DEVICE CODE
002776  021314                    CMP     (R3),           (R4)    ;DO THE TWO DEVICE CODES COMPARE
003000  001410                    BEQ     7$                      ;IF YES GET NEXT ETABLE DEVICE CODE
003002  005714                    TST     (R4)                    ;IF NO HAVE WE CHECKED ALL THE BOARD
003004  001372                    BNE     6$                      ;DEVICE CODES, IF NO CONTINUE
003006  000313                    SWAB    (R3)                    ;PUT ASCII IN RIGHT ORDER FOR OUTPUT
003010  052737  000002  001110    BIS     #APTER2,        @#ROMERR ;IF YES ETABLE DEVICE CODE NOT
003016  004767  000750            JSR     PC,             ERRHAN  ;NOT ON BOARD

003022  062703  000002    7$      ADD     #2,             R3
```

```
898  003026  005713                        TST     (R3)                    ,IF CONTENTS EQUAL ZERO-DONE
899  003030  001355                        BNE     5$                      ,IF CONTENTS NO EQUAL ZERO-CONTINUE
900  003032  004767  000272                JSR     PC,         PPFVAR      ,GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
901  003036  013704  001212                MOV     @#$BASE,    R4          ,GET ETABLE PPFVA
902  003042  013703  007514                MOV     @#BUF2,     R3          ,GET BOARD PPFVA
903  003046  020403                        CMP     R4,         R3          ,DO THE TWO PARAMETERS COMPARE
904  003050  001405                        BEQ     8$                      ,IF YES CONTINUE
905  003052  052737  000004  001110        BIS     #APTER3,    @#ROMERR    ,SET APT ERROR INDICATOR
906  003060  004767  000706                JSR     PC,         ERRHAN      ,GO TO ERROR ROUTINE
907  003064  013704  001216          8$    MOV     @#$CDW1,    R4          ,GET ETABLE DATA
908  003070  013703  007516                MOV     @#BUF2+2,   R3          ,GET BOARD DATA
909  003074  020403                        CMP     R4,         R3          ,DO THE TWO PARAMETER COMPARE
910  003076  001405                        BEQ     9$                      ,IF YES THEN DONE
911  003100  052737  000010  001110        BIS     #APTER4,    @#ROMERR    ,SET APT ERROR INDICATOR
912  003106  004767  000660                JSR     PC,         ERRHAN      ,GO TO ERROR ROUTINE
913  003112  000207                  9$    RTS     PC
914
915                                  ,GET DEVICE CODES MODULE
916                                  ,THIS SUBROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE
917                                  ,ADDRESS IN WHICH IT WAS FOUND BACK TO THE CALLING ROUTINE
918                                  ,DATA IS STORED IN BUFFER "BUF1" IN THIS FORMAT
919                                  ,
920                                  ,       BUF1    ADDRESS OF FIRST DEVICE CODE
921                                  ,               DEVICE CODE
922                                  ,               ADDRESS OF SECOND DEVICE CODE
923                                  ,               DEVICE CODE
924                                  ,
925                                  ,
926                                  ,
927                                  ,
928                                  ,               ADDRESS OF NTH DEVICE CODE
929                                  ,               DEVICE CODE
930                                  ,
931                                  ,+CONTINUATION ROM DATA IS STORED IN "BUF1' AS FOLLOWS
932                                  ,+
933                                  ,+
934                                  ,+
935                                  ,+              CONTINUATION CHIP IDENTIFIER (ALWAYS A +1)
936                                  ,+              DEVICE CODE (PREVIOUS ROM'S DEVICE CODE,OR -1 IF ILLEGAL ROM)
937                                  ,+
938                                  ,+
939                                  ,+
940                                  ,IF DIAGNOSTIC ROM PRESENT IT WILL BE THE FIRST DEVICE CODE
941                                  ,THERE IS ROOM FOR UP TO 12 DEVICE CODES IN TEMP
942  003114  012700  007434          DEVCOD  MOV     #BUF1,              R0
943  003120  012701  000001                  MOV     #1,                 R1              ,INITIALIZE ROM POINTER
944  003124  030137  001116                  BIT     R1,                 @#ROMFIN        ,IS DIAGNOSTIC ROM PRESENT
945  003130  001411                          BEQ     1$                                  ,IF NO GO TEST BOOT ROMS
946  003132  012737  165774  003324          MOV     #165774,            @#TESTAD        ,GET ADDRESS OF DIAG. ROM DEVICE CODE
947  003140  013720  003324                  MOV     @#TESTAD,           (R0)+           ,STORE ADDRESS OF DEVICE CODE
948  003144  017720  000154                  MOV     @TESTAD,            (R0)+           ,STORE DEVICE CODE
949  003150  005037  003326                  CLR     @#DAFLAG                            ,CLR DATA TYPE FLAG
950  003154  013737  001114  003324  1$      MOV     @#FIRSTB,           @#TESTAD        ,GET ADDRESS OF FIRST BOOT ROM
951  003162  006101                          ROL     R1                                  ,UPDATE POiNTER TO NEXT ROM
952  003164  030167  006636            3$    BIT     R1,                 DEVFIN          ,+IS DEVICE ROM PRESENT
953  003170  001007                          BNE     4$                                  ,IF YES GO GET PARAMETERS
```

```
  954  003172  004767  005450                    JSR     PC,     CON1                    ,+CHECK FOR CONTINUATION CHIP
  955  003176  062737  000200  003324            ADD     #200,           @#TESTAD        ,IF NO UPDATE TEST ADDRESS
  956  003204  006101                            ROL     R1                              ;UPDATE POINTER TO NEXT ROM
  957  003206  000441                            BR      9$                              ;GO SEE IF ALL ROM CHECKED
  958  003210  005737  003326            4$      TST     @#DAFLAG                        ;IF DATAFLAG=0 THEN DATA IS DEVICE CODE
  959  003214  001010                            BNE     5$                              ;IF DATAFLAG=1 THE DATA IS OFFSET TO NEX
  960  003216  013720  003324                    MOV     @#TESTAD,       (R0)+           ;STORE ADDRESS OF DEVICE CODE
  961  003222  017720  000076                    MOV     @TESTAD,        (R0)+           ;STORE DEVICE CODE
  962  003226  062737  000002  003324            ADD     #2,             @#TESTAD        ;UPDATE TEST ADDRESS
  963  003234  000424                            BR      8$                              ;GET OVER SOME CODE
  964  003236  013702  003324            5$      MOV     @#TESTAD,       R2              ;SAVE OLD TEST ADDRESS
  965  003242  067737  000056  003324            ADD     @TESTAD,        @#TESTAD        ,UPDATE TO NEW TEST ADDRESS
  966  003250  013703  003324                    MOV     @#TESTAD,       R3              ;GET THE NEW ADDRESS
  967  003254  042702  170177                    BIC     #170177,        R2              ,+SAVE ONLY BITS 7,8,9,10,11
  968  003260  042703  170177                    BIC     #170177,        R3              ,+IN R3 ALSO
  969  003264  160203                            SUB     R2,             R3              ,CALCULATE DISTANCE BETWEEN  ADD
  970  003266  005703                    6$      TST     R3                              ,IS R3 EQUAL TO 0
  971  003270  001406                            BEQ     8$                              ,IF YES THEN DONE
  972  003272  162703  000200                    SUB     #200,           R3              ,IF NO THEN MOVE POINTER
  973  003276  006101                            ROL     R1                              ,ONE BIT FOR EVERY 200
  974  003300  004767  005370                    JSR     PC,     CON2                    ,+CHECK FOR CONTINUATION CHIP
  975  003304  000770                            BR      6$
  976  003306  005137  003326            8$      COM     @#DAFLAG                        ,CHANGE DATA FLAG TO RIGHT DATA TYPE
  977  003312  023727  003324  174000    9$      CMP     @#TESTAD,       #174000         ,HAVE WE CHECKED ALL THE ROM
  978  003320  002721                            BLT     3$                              ,IF NO CONTINUE
  979  003322  000207                            RTS     PC                              ,IF YES RETURN
  980
  981  003324  000000                    TESTAD  0
  982  003326  000000                    DAFLAG  0
  983
  984                                    ,GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE
  985                                    ,THIS SUBROUTINE TESTS LOCATIONS 173024 AND 173224 TO DETERMINE
  986                                    ,WHICH VECTOR WILL BE USED IF POWER-FAIL OPTION ENABLED ON THE
  987                                    ,BOARD   THE OPTION MUST BE ENABLED AND AT LEAST ONE ADDRESS
  988                                    ,SWITCH MUST BE "ON" OR AN ERROR WILL BE DETECTED
  989                                    ,THE DATA WILL BE RETURNED IN "BUF2" IN THE FORMAT
  990                                    ,
  991                                    ,       BUF2    PSEUDO POWER-FAIL VECTOR ADDRESS
  992                                    ,               CONTENTS OF VECTOR ADDRESS
  993
  994  003330  032737  000777  173024    PPFVAR  BIT     #777,           @#173024        ,TEST IF LOCATION 173024 SELECTED
  995  003336  001414                            BEQ     1$                              ,IF NOT THEN GO TEST LOCATION 173224
  996  003340  012767  003370  006452            MOV     #1$,    RETURN                  ,+SET UP AN ALTERNATE RETURN
  997  003346  004767  005366                    JSR     PC,     HOLCK1                  ,+CHECK FOR A HOLE
  998  003352  012737  173024  007514            MOV     #173024,        @#BUF2          ,IF IT IS THEN STORE ADDRESS
  999  003360  013737  173024  007516            MOV     @#173024,       @#BUF2+2        ,STORE CONTENTS OF LOCATION 173024
 1000  003366  000423                            BR      3$                              ,GO TO RETURN
 1001  003370  032737  000777  173224    1$      BIT     #777,           @#173224        ,TEST IF LOCATION 173224 SELECTED
 1002  003376  001414                            BEQ     2$                              ,IF NOT THEN SET ERROR INDICATOR
 1003  003400  012767  003430  006412            MOV     #2$,    RETURN                  ,+SET UP AN ALTERNATE RETURN
 1004  003406  004767  005340                    JSR     PC,     HOLCK2                  ,+CHECK FOR A HOLE
 1005  003412  012737  173224  007514            MOV     #173224,        @#BUF2          ,IF IT IS THEN STORE ADDRESS
 1006  003420  013737  173224  007516            MOV     @#173224,       @#BUF2+2        ,STORE CONTENTS OF VECTOR
 1007  003426  000403                            BR      3$                              ,GET OVER ERROR
 1008  003430  052737  000040  001110    2$:     BIS     #PFERR,         @#ROMERR        ,SET ERROR INDICATOR
 1009  003436  000207                    3$.     RTS     PC
```

```
1010
1011
1012
1013
1014                            ;PUT MESSAGE IN BUFFER ROUTINE
1015                            ;THIS SUBROUTINE FORMATS THE PARAMETER AND POWER-FAIL MESSAGES
1016
1017  003440  017604  000000   PUTMES  MOV     @(R6),          R4              ;GET MESSAGE BUFFER ADDRESS
1018  003444  005737  003770           TST     @#TIMES                         ;IS THIS FIRST TIME THROUGH
1019  003450  001110                   BNE     3$                              ;IF NOT FIRST TIME THEN FORMAT POWER-
1020                                                                           ;FAIL MESSAGE
1021  003452  005237  003770           INC     @#TIMES                         ;TOGGLE WATCHDOG
1022  003456  012703  007434           MOV     #BUF1,          R3              ;GET DATA BUFFER ADDRESS
1023  003462  022713  165774           CMP     #165774,        (R3)            ;IS DIAGNOSTIC ROM PRESENT
1024  003466  001012                   BNE     1$                              ;IF NOT DON'T FORMAT DIAG. ROM MESSAGE
1025  003470  012705  006512           MOV     #DRHEAD,        R5              ;IF IT IS GET DIAG. ROM MESSAGE HEADER
1026  003474  004767  000662           JSR     PC,             FILBUF          ;GO PUT HEADER IN MESSAGE BUFFER
1027  003500  000015                   CR
1028  003502  062703  000002           ADD     #2,             R3              ;SKIP OVER ADDRESS OF ASCII
1029  003506  000313                   SWAB    (R3)                            ;FORMAT ASCII FOR MESSAGE
1030  003510  112324                   MOVB    (R3)+,          (R4)+           ;PUT ASCII IN MESSAGE BUFFER
1031  003512  112324                   MOVB    (R3)+,          (R4)+
1032  003514  012705  006533   1$      MOV     #BRHEAD,        R5
1033  003520  004767  000636           JSR     PC,             FILBUF          ;GO PUT BOOT ROM HEADER IN MESS  BUF
1034  003524  000015                   CR
1035  003526  012701  000001           MOV     #1,     R1                      ;+POINT TO DIAGNOSTIC ROM
1036  003532  012702  012040           MOV     #MESTAB,        R2              ;+POINT TO MSG TABLE
1037  003536  012767  003552  006254   MOV     #2$,    RETURN                  ;+SET UP AN ALTERNATE RETURN
1038  003544  012767  003736  006264   MOV     #5$,    FINISH                  ;+SET UP ANOTHER ALTERNATE RETURN
1039  003552  112724  000015   2$      MOVB    #CR,            (R4)+           ;PUT A CR/LF HERE
1040  003556  112724  000012           MOVB    #LF,            (R4)+
1041  003562  004767  005222           JSR     PC,     ROMTYP                  ;+FIND THE ROM TYPE
1042  003566  062713  000004           ADD     #4,             (R3)            ;GET FIRST ENTRY POINT
1043  003572  004767  000732           JSR     PC,             OCADD           ;GO CONVERT OCTAL TO ASCII
1044  003576  004767  000560           JSR     PC,             FILBUF          ;GO PUT ENTRY POINT IN MESSAGE BUF
1045  003602  000011                   HT
1046  003604  112724  000040           MOVB    #40,            (R4)+
1047  003610  112724  000040           MOVB    #40,            (R4)+
1048  003614  112724  000040           MOVB    #40,            (R4)+
1049  003620  062713  000002           ADD     #2,             (R3)            ;GET SECOND ENTRY POINT
1050  003624  004767  000700           JSR     PC,             OCADD           ;GO CONVERT OCTAL TO ASCII
1051  003630  004767  000526           JSR     PC,             FILBUF          ;GO PUT ENTRY POINT IN MESSAGE BUF
1052  003634  000011                   HT
1053  003636  112724  000040           MOVB    #40,            (R4)+
1054  003642  112724  000040           MOVB    #40,            (R4)+
1055  003646  112724  000040           MOVB    #40,            (R4)+
1056  003652  112724  000011           MOVB    #HT,            (R4)+           ;PUT TAB IN HERE
1057  003656  062703  000002           ADD     #2,             R3              ;UPDATE DATA BUFFER ADDRESS
1058  003662  000313                   SWAB    (R3)                            ;FORMAT ASCII FOR MESSAGE
1059  003664  112324                   MOVB    (R3)+,          (R4)+           ;MOV ASCII TO MES1
1060  003666  112324                   MOVB    (R3)+,          (R4)+
1061  003670  000730                   BR      2$                              ;GO BACK TO START OF LOOP
1062  003672  012703  007514   3$      MOV     #BUF2,          R3              ;GET DATA BUFFER ADDRESS
1063  003676  012705  006670           MOV     #PFHEAD,        R5              ;GET POWER-FAIL HEADER ADDRESS
1064  003702  004767  000454           JSR     PC,             FILBUF          ;GO PUT POWER-FAIL HEADER IN MESSAGE BUF
1065  003706  000015                   CR
```

```
1066  003710  022703  007520          4$    CMP   #BUF2+4,     R3          ,ARE WE DONE
1067  003714  001410                        BEQ   5$                       ,IF YES THEN GO RETURN
1068  003716  004767  000606                JSR   PC,          OCADD       ,GO CONVERT OCTAL TO ASCII
1069  003722  004767  000434                JSR   PC,          FILBUF
1070  003726  000011                        HT
1071  003730  062703  000002                ADD   #2,          R3          ,UPDATE DATA BUFFER ADDRESS
1072  003734  000765                        BR    4$                       ,GO BACK TO START OF LOOP
1073  003736  112724  000015          5$    MOVB  #CR,         (R4)+       ,PUT A CR/LF AT END OF MESSAGE
1074  003742  112724  000012                MOVB  #LF,         (R4)+
1075  003746  105024                        CLRB  (R4)+                    ,PUT ZERO TERMINATOR AT END OF MESSAGE
1076  003750  017667  000000  000002        MOV   @(R6),       6$          ,GET MESSAGE BUFFER ADDRESS
1077  003756  104401                        TYPE
1078  003760  000000                6$     WORD  0
1079  003762  062716  000002                ADD   #2,          (R6)        ,GET OVER MESSAGE BUFFER ADDRESS
1080  003766  000207                        RTS   PC
1081  003770  000000                TIMES   0
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091                                ,ERROR HANDLER ROUTINE
1092                                ,THIS SUBROUTINE FORMATS THE ERROR MESSAGES THE TYPES THEM
1093                                ,OUT
1094  003772  104411                ERRHAN: SAVREG
1095  003774  012704  010230                MOV   #ERRMSG,     R4
1096  004000  005237  001120                INC   @#ERRCNT                 ,INCREMENT ERROR COUNTER
1097  004004  032777  002000  175072        BIT   #BIT10,      @SWR        ,BELL ON ERROR
1098  004012  001402                        BEQ   1$                       ,BRANCH IF NO
1099  004014  104401                        TYPE
1100  004016  006506                        BELL
1101  004020  032777  020000  175056  1$    BIT   #BIT13,      @SWR        ,INHIBIT ERROR TYPEOUT
1102  004026  001131                        BNE   10$
1103  004030  013700  001110                MOV   @#ROMERR,    R0          ,GET ERROR CODE
1104  004034  013767  001110  000236        MOV   @#ROMERR,    9$          ,SAVE ERROR CODE FOR APT
1105  004042  012701  004334                MOV   #EHEADT,     R1          ,GET ERROR HEADER TABLE
1106  004046  000241                        CLC                            ,C-BIT USED TO STOP STEPPING THROUGH TAB
1107  004050  006000                2$     ROR   R0                       ,ROTATE ERROR CODES
1108  004052  103403                        BCS   3$                       ,IF C-BIT SET STOP STEPPING
1109  004054  062701  000002                ADD   #2,          R1          ,IF C-BIT CLEAR STEP TO NEXT HEADER
1110  004060  000773                        BR    2$                       ,GO STEP
1111  004062  011105                3$     MOV   (R1),        R5          ,PUT HEADER ADDRESS IN REGISTER
1112  004064  004767  000272                JSR   PC,          FILBUF      ,GO PUT HEADER IN ERROR MESSAGE BUF
1113  004070  000015                        CR
1114  004072  032767  000040  175010        BIT   #PFERR, ROMERR           ,+IS IT A "PPFVAR" ERROR?
1115  004100  001063                        BNE   8$                       ,+BRANCH IF YES
1116  004102  032767  003600  175000        BIT   #3600, ROMERR            ,+IS IT A "SEQTST" ERROR?
1117  004110  001411                        BEQ   14$                      ,+BRANCH IF NO
1118  004112  016767  005714  000010        MOV   ARG2,   13$              ,+PUT #CR OR #HT AT 13$
1119  004120  016705  005704                MOV   ARG1,   R5               ,+ARG1 CONTAINS ADR  OF MSG
1120  004124  004767  000232                JSR   PC,     FILBUF           ,+PUT DATA INTO BUFFER
1121  004130  000000                13$    WORD  0                        ,+CONTAINS CR OR HT
```

```
1122  004132  000446                          BR      8$                      ;+
1123  004134  032737  000023  001110  14$     BIT     #23,    @#ROMERR        ;+IS ERROR A CRC, APTER1 OR APTER2
1124  004142  001424                          BEQ     7$                      ;IF NO GO FORMAT APTER3, APTER4
1125  004144  032737  000020  001110          BIT     #20,    @#ROMERR        ;IS ERROR CRC
1126  004152  001415                          BEQ     6$                      ;IF NO FORMAT APTER1, APTER2
1127  004154  013700  002524                  MOV     @#ROMCNT,       R0      ;GET ROM COUNTER
1128  004160  012701  006474                  MOV     #ROMNUM,        R1      ;GET ROM NUMBER TABLE
1129  004164  000241                          CLC                             ;C-BIT USED TO STOP STEPPING THROUGH TAB
1130  004166  006000                  4$      ROR     R0                      ;ROTATE ROM NUMBER
1131  004170  103403                          BCS     5$                      ;IF C-BIT SET STOP STEPPING
1132  004172  062701  000002                  ADD     #2,             R1      ;IF C-BIT CLEAR STEP TO NEXT HEADER
1133  004176  000773                          BR      4$                      ;CONTINUE STEPPING
1134  004200  112124                  5$      MOVB    (R1)+,          (R4)+   ;PUT ROM NUMBER IN ERROR MESSAGE BUF
1135  004202  112124                          MOVB    (R1)+,          (R4)+
1136  004204  000421                          BR      8$                      ;GO TO END OF ROUTINE
1137  004206  112324                  6$      MOVB    (R3)+,          (R4)+   ;PUT BAD DEVICE CODE IN ERROR MESS  BUF
1138  004210  112324                          MOVB    (R3)+,          (R4)+
1139  004212  000416                          BR      8$                      ;GO TO END OF ROUTINE
1140  004214  016603  000006          7$      MOV     6(R6),          R3      ;GET SAVED EXPECTED VALUE
1141  004220  004767  000200                  JSR     PC,             OCASC   ;GO COVERT OCTAL TO ASCII
1142  004224  004767  000132                  JSR     PC,             FILBUF  ;GO PUT DATA IN ERROR MESSAGE BUF
1143  004230  000011                          HT
1144  004232  016603  000010                  MOV     10(R6),         R3      ;GET SAVED RECEIVED VALUE
1145  004236  004767  000162                  JSR     PC,             OCASC   ;GO COVERT OCTAL TO ASCII
1146  004242  004767  000114                  JSR     PC,             FILBUF  ;GO PUT DATA IN ERROR MESSAGE BUFFER
1147  004246  000011                          HT
1148  004250  112724  000015          8$      MOVB    #CR,            (R4)+   ;PUT CR/LF AT END OF MESSAGE
1149  004254  112724  000012                  MOVB    #LF,            (R4)+
1150  004260  112724  000000                  MOVB    #0,             (R4)+   ;PUT TERMINATOR AT END OF MESSAGE
1151  004264  005767  174666                  TST     $ENV                    ;TST IF ON APT
1152  004270  001405                          BEQ     11$                     ;IF NO BRANCH
1153  004272  004767  000544                  JSR     PC,             $ATY1   ;GO REPORT ERROR
1154  004276  010230                          ERRMSG
1155  004300  000000                  9$       WORD    0
1156  004302  000000                          HALT                            ;THEN HALT
1157  004304  004767  000540          11$     JSR     PC,             $ATY3   ;IF NOT ON APT JUST REPORT ERROR
1158  004310  010230                          ERRMSG
1159  004312  032777  100000  174564  10$     BIT     #BIT15,         @SWR    ;IS HALT ON ERROR SET
1160  004320  001401                          BEQ     12$                     ;IF NO SKIP OVER HALT
1161  004322  000000                          HALT
1162  004324  005037  001110          12$.    CLR     @#ROMERR                ;CLEAR ERROR FLAGS
1163  004330  104412                          RESREG
1164  004332  000207                          RTS     PC
1165
1166
1167  004334  006771                  EHEADT  ER1MSG
1168  004336  006735                          ER2MSG
1169  004340  007027                          ER3MSG
1170  004342  007105                          ER4MSG
1171  004344  007161                          CRCMSG
1172  004346  007205                          PFMSG
1173  004350  007263                          NOROMM
1174  004352  007341                          SEQMSG                          ;+
1175  004354  012076                          ERMES1                          ;+
1176  004356  012155                          ERMES2                          ;+
1177  004360  012235                          ERMES3                          ;+
```

```
1178
1179
1180
1181                                    ,FILL BUFFER ROUTINE
1182                                    ,THIS SUBROUTINE FILLS THE MESSAGE BUFFER WILL ASCII
1183                                    ,CHARACTERS.
1184
1185  004362  022776  000011  000000  FILBUF. CMP     #HT,            @(R6)           ,IS FIRST CHARACTER A TAB OR CR
1186  004370  001003                          BNE     1$                              ,IF CR THEN GO PUT CR/LF IN BUFFER
1187  004372  112724  000011                  MOVB    #HT,            (R4)+           ,IF TAB THEN PUT TAB IN BUFFER
1188  004376  000404                          BR      2$                              ,GET OVER NEXT LINE
1189  004400  112724  000015          1$      MOVB    #CR,            (R4)+           ,MOV CR/LF TO BUFFER
1190  004404  112724  000012                  MOVB    #LF,            (R4)+
1191  004410  112524                  2$      MOVB    (R5)+,          (R4)+           ,PUT A CHARACTER IN MESSAGE BUFFER
1192  004412  105715                          TSTB    (R5)                            ,IS NEXT CHARACTER ZERO
1193  004414  001375                          BNE     2$                              ,IF NOT PUT IT IN MESSAGE BUFFER AND GET
1194  004416  062716  000002                  ADD     #2,             (R6)            ,UPDATE RETURN POINTER TO GET OVER CHARA
1195  004422  000207                          RTS     PC                              ,THEN RETURN
1196
1197
1198
1199
1200                                    ,OCTAL TO ASCII CONVERSION ROUTINE
1201                                    ,THIS SUBROUTINE TAKES A SIXTEEN BIT OCTAL NUMBER AND
1202                                    ,CONVERTS IT TO 6 ASCII CHARACTERS
1203
1204  004424  012700  007520          OCASC   MOV     #OCTBUF,        R0              ,GET BUFFER ADDRESS
1205  004430  005020                          CLR     (R0)+                           ,CLEAR BUFFER
1206  004432  005020                          CLR     (R0)+
1207  004434  005020                          CLR     (R0)+
1208  004436  005020                          CLR     (R0)+
1209  004440  012700  007520                  MOV     #OCTBUF,        R0              ,GET BUFFER ADDRESS
1210  004444  010337  004526                  MOV     R3,             @#TEMP          ,GET OCTAL NUMBER
1211  004450  000241                          CLC                                     ,CLEAR CARRY
1212  004452  006137  004526                  ROL     @#TEMP                          ,ROTATE BIT INTO CARRY BIT
1213  004456  006110                          ROL     (R0)                            ,ROTATE CARRY BIT INTO BUFFER
1214  004460  152710  000060          1$      BISB    #60,            (R0)            ,MAKE IT ASCII
1215  004464  005200                          INC     R0                              ,UPDATE BUFFER ADDRESS
1216  004466  020027  007526                  CMP     R0,             #OCTBUF+6       ,HAVE WE CONVERTED ALL THE NUMBER
1217  004472  001003                          BNE     2$                              ,IF NO CONTINUE
1218  004474  012705  007520                  MOV     #OCTBUF,        R5              ,IF YES PUT BUFFER ADDRESS IN REGISTER
1219  004500  000207                          RTS     PC                              ,RETURN
1220  004502  006137  004526          2$      ROL     @#TEMP                          ,ROTATE BIT INTO CARRY BIT
1221  004506  106110                          ROLB    (R0)                            ,ROTATE CARRY BIT INTO BUFFER
1222  004510  006137  004526                  ROL     @#TEMP                          ,
1223  004514  106110                          ROLB    (R0)                            ,
1224  004516  006137  004526                  ROL     @#TEMP                          ,
1225  004522  106110                          ROLB    (R0)                            ,
1226  004524  000755                          BR      1$                              ,GO TO START OF LOOP
1227  004526  000000          TEMP    0
1228
1229
1230
1231                                    ;THIS ROUTINE IS CALLED BY THE PUT MESSAGE ROUTINE TO GET THE RIGHT
1232                                    ,VALUE IN R3 SO THE OCTAL TO ASCII ROUTINE GETS THE RIGHT NUMBER
1233
```

```
1234  004530  010346           OCADD   MOV     R3,             -(R6)           ,SAVE THE VALUE OF R3
1235  004532  011303                   MOV     (R3),           R3              ,PUT THE DATA TO BE CONVERTED IN R3
1236  004534  004767  177664           JSR     PC,             OCASC           ;GO CONVERT OCTAL TO ASCII
1237  004540  012603                   MOV     (R6)+,          R3              ,RESTORE R3
1238  004542  000207                   RTS     PC                              ,RETURN
1239
1240
1241
1242
1243
1244                                    SBTTL   TYPE ROUTINE
1245
1246                                   ,,*************************************************************************
1247                                   ,*ROUTINE TO TYPE ASCIZ MESSAGE  MESSAGE MUST TERMINATE WITH A 0 BYTE
1248                                   ,*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED
1249                                   ,*NOTE1·        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER
1250                                   ,*NOTE2         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED
1251                                   ,*NOTE3         $FILLC CONTAINS THE CHARACTER TO FILL AFTER
1252                                   ,*
1253                                   ,*CALL
1254                                   ,*1) USING A TRAP INSTRUCTION
1255                                   ,*      TYPE    ,MESADR         ,,MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1256                                   ,*OR
1257                                   ,*      TYPE
1258                                   ,*      MESADR
1259                                   ,*
1260
1261  004544  105767  000265  $TYPE    TSTB    $TPFLG          ,;IS THERE A TERMINAL?
1262  004550  100002                   BPL     1$              ,,BR IF YES
1263  004552  000000                   HALT                    ,,HALT HERE IF NO TERMINAL
1264  004554  000430                   BR      3$              ,,LEAVE
1265  004556  010046           1$      MOV     R0,-(SP)        ;;SAVE R0
1266  004560  017600  000002           MOV     @2(SP),R0       ,;GET ADDRESS OF ASCIZ STRING
1267  004564  122767  000001  174364   CMPB    #APTENV,$ENV    ;,RUNNING IN APT MODE
1268  004572  001011                   BNE     62$             ,,NO, GO CHECK FOR APT CONSOLE
1269  004574  132767  000100  174355   BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
1270  004602  001405                   BEQ     62$             ;,NO, GO CHECK FOR CONSOLE
1271  004604  010067  000004           MOV     R0,61$          ;,SETUP MESSAGE ADDRESS FOR APT
1272  004610  004767  000234           JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
1273  004614  000000           61$     .WORD   0               ,;MESSAGE ADDRESS
1274  004616  132767  000040  174333   62$     BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
1275  004624  001003                   BNE     60$             ,;YES, SKIP TYPE OUT
1276  004626  112046           2$·     MOVB    (R0)+,-(SP)     ,,PUSH CHARACTER TO BE TYPED ONTO STACK
1277  004630  001005                   BNE     4$              ,,BR IF IT ISN'T THE TERMIN TOR
1278  004632  005726                   TST     (SP)+           ,;IF TERMINATOR POP IT OFF THE STACK
1279  004634  012600           60$·    MOV     (SP)+,R0        ,,RESTORE R0
1280  004636  062716  000002   3$      ADD     #2,(SP)         ,,ADJUST RETURN PC
1281  004642  000002                   RTI                     ,,RETURN
1282  004644  122716  000011   4$      CMPB    #HT,(SP)        ,,BRANCH IF <HT>
1283  004650  001430                   BEQ     8$
1284  004652  122716  000200           CMPB    #CRLF,(SP)      ,,BRANCH IF NOT <CRLF>
1285  004656  001006                   BNE     5$
1286  004660  005726                   TST     (SP)+           ,,POP  <CR><LF> EQUIV
1287  004662  104401                   TYPE                    ,,TYPE A CR AND LF
1288  004664  005037                   $CRLF
1289  004666  105067  000130           CLRB    $CHARCNT        ,,CLEAR CHARACTER COUNT
```

```
1290  004672  000755                        BR      2$              ;;GET NEXT CHARACTER
1291  004674  004767  000056        5$      JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1292  004700  126726  000130        6$      CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS ?
1293  004704  001350                        BNE     2$              ;;IF NO GO GET NEXT CHAR
1294  004706  016746  000120                MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1295                                                                ;;AND THE NULL CHAR.
1296  004712  105366  000001        7$      DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1297  004716  002770                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1298  004720  004767  000032                JSR     PC,$TYPEC       ;;GO TYPE A NULL
1299  004724  105367  000072                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
1300  004730  000770                        BR      7$              ;;LOOP
1301
1302                                  ;HORIZONTAL TAB PROCESSOR
1303
1304  004732  112716  000040        8$      MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1305  004736  004767  000014        9$      JSR     PC,$TYPEC       ;;TYPE A SPACE
1306  004742  132767  000007  000052        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1307  004750  001372                        BNE     9$              ;;TAB STOP
1308  004752  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
1309  004754  000724                        BR      2$              ;;GET NEXT CHARACTER
1310  004756  105777  000044        $TYPEC  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
1311  004762  100375                        BPL     $TYPEC
1312  004764  116677  000002  000036        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG
1313  004772  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1314  005000  001003                        BNE     1$              ;;BRANCH IF NO
1315  005002  105067  000014                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1316  005006  000406                        BR      $TYPEX          ;;EXIT
1317  005010  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1318  005016  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
1319  005020  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
1320  005022  000000        $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
1321  005024  000207        $TYPEX: RTS     PC
1322
1323  005026  177564        $TPS:   .WORD   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
1324  005030  177566        $TPB:   .WORD   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
1325  005032     000        $NULL:  .BYTE   0                       ;;CONTAINS NULL CHARACTER FOR FILLS
1326  005033     002        $FILLS: .BYTE   2                       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
1327  005034     012        $FILLC: .BYTE   12                      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
1328  005035     000        $TPFLG: .BYTE   0                       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
1329  005036     077        $QUES:  .ASCII  "?"                     ;;QUESTION MARK
1330  005037     015        $CRLF:  .ASCII  <15>                    ;;CARRAIGE RETURN
1331  005040  000012        $LF:    .ASCIZ  <12>                    ;;LINEFEED
1332                                .SBTTL  APT COMMUNICATIONS ROUTINE
1333
1334                                ;;**************************************************************************
1335  005042  112767  000001  000236  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
1336  005050  112767  000001  000226  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
1337  005056  000403                        BR      $ATYC
1338  005060  112767  000001  000220  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
1339  005066                        $ATYC:
1340  005066  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1341  005070  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1342  005072  105767  000206                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
1343  005076  001450                        BEQ     5$              ;;IF NOT:  BR
1344  005100  122767  000001  174050        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
1345  005106  001031                        BNE     3$              ;;IF NOT   BR
```

```
1346  005110  132767  000100  174041          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
1347  005116  001425                          BEQ     3$              ;;IF NOT:  BR
1348  005120  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
1349  005124  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
1350  005132  005767  174000          1$      TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
1351  005136  001375                          BNE     1$              ;;IF NOT:  WAIT
1352  005140  010067  174006                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
1353  005144  105720                  2$      TSTB    (R0)+           ;;FIND END OF MESSAGE
1354  005146  001376                          BNE     2$
1355  005150  166700  173776                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
1356  005154  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
1357  005156  010067  173772                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
1358  005162  012767  000004  173746          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
1359  005170  000413                          BR      5$
1360  005172  017667  000004  000016  3$      MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
1361  005200  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
1362  005206  016746  172564                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
1363  005212  004767  177326                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
1364  005216  000000                  4$      .WORD   0
1365  005220                           5$
1366  005220  105767  000062          10$     TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
1367  005224  001416                          BEQ     12$             ;;IF NOT:  BR
1368  005226  005767  173724                  TST     $ENV            ;;RUNNING UNDER APT?
1369  005232  001413                          BEQ     12$             ;;IF NOT:  BR
1370  005234  005767  173676          11$     TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
1371  005240  001375                          BNE     11$             ;;IF NOT:  WAIT
1372  005242  017667  000004  173670          MOV     @4(SP),$FATAL   ;;GET ERROR #
1373  005250  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR
1374  005256  005267  173654                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
1375  005262  105067  000020          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
1376  005266  105067  000013                  CLRB    $LFLG           ;;CLEAR LOG FLAG
1377  005272  105067  000006                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
1378  005276  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
1379  005300  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1380  005302  000207                          RTS     PC              ;;RETURN
1381  005304     000          $MFLG:  .BYTE   0               ;;MESSG  FLAG
1382  005305     000          $LFLG:  .BYTE   0               ;;LOG FLAG
1383  005306     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
1384          005310                  .EVEN
1385          000200          APTSIZE=200
1386          000001          APTENV=001
1387          000100          APTSPOOL=100
1388          000040          APTCSUP=040
1389                           .SBTTL  TTY INPUT ROUTINE
1390
1391                           ;;**************************************************************
1392  005310  177560          $TKS    .WORD   177560          ;;TTY KBD STATUS
1393  005312  177562          $TKB    .WORD   177562          ;;TTY KBD BUFFER
1394                           .ENABL  LSB
1395
1396                           ;;**************************************************************
1397                           ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE
1398                           ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1399                           ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1400                           ;*WHEN OPERATING IN TTY FLAG MODE
1401  005314  022767  000176  173562  $CKSWR   CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
```

```
1402  005322  001074                        BNE     15$             ;;BRANCH IF NO
1403  005324  105777  177760                TSTB    @$TKS           ;;CHAR THERE?
1404  005330  100071                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
1405  005332  117746  177754                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
1406  005336  042716  177600                BIC     #C177,(SP)      ;;STRIP-OFF THE ASCII
1407  005342  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
1408  005346  001062                        BNE     15$             ;;NO, RETURN TO USER
1409  005350  126727  173524  000001        CMPB    $AUTOB,#1       ;,ARE WE RUNNING IN AUTO-MODE?
1410  005356  001456                        BEQ     15$             ;;BRANCH IF YES
1411
1412  005360  104401  006041                TYPE    ,$CNTLG         ;,ECHO THE CONTROL-G ( G)
1413  005364  104401  006046        $GTSWR. TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
1414  005370  016746  172602                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
1415  005374  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1416  005376  104401  006057                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
1417  005402  005046          19$   CLR     -(SP)           ;;CLEAR COUNTER
1418  005404  005046                CLR     -(SP)           ;;THE NEW SWR
1419  005406  105777  177676    7$  TSTB    @$TKS           ;;CHAR THERE?
1420  005412  100375                BPL     7$              ;,IF NOT TRY AGAIN
1421
1422  005414  117746  177672                MOVB    @$TKB,-(SP)     ;,PICK UP CHAR
1423  005420  042716  177600                BIC     #C177,(SP)      ;,MAKE IT 7-BIT ASCII
1424
1425
1426
1427  005424  021627  000025    9$  CMP     (SP),#25        ;,IS IT A CONTROL-U?
1428  005430  001005                BNE     10$             ;,BRANCH IF NOT
1429  005432  104401  006034                TYPE    ,$CNTLU         ;,YES, ECHO CONTROL-U ( U)
1430  005436  062706  000006    20$ ADD     #6,SP           ;,IGNORE PREVIOUS INPUT
1431  005442  000757                BR      19$             ;,LET'S TRY IT AGAIN
1432
1433
1434  005444  021627  000015    10$ CMP     (SP),#15        ;,IS IT A <CR>?
1435  005450  001022                BNE     16$             ;,BRANCH IF NO
1436  005452  005766  000004                TST     4(SP)           ;,YES, IS IT THE FIRST CHAR?
1437  005456  001403                BEQ     11$             ;,BRANCH IF YES
1438  005460  016677  000002  173416        MOV     2(SP),@SWR      ;;SAVE NEW SWR
1439  005466  062706  000006    11$ ADD     #6,SP           ;,CLEAR UP STACK
1440  005472  104401  005037    14$ TYPE    ,$CRLF          ;,ECHO <CR> AND <LF>
1441  005476  126727  173377  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
1442  005504  001003                BNE     15$             ;;BRANCH IF NOT
1443  005506  012777  000100  177574        MOV     #100,@$TKS      ;,RE-ENABLE TTY KBD INTERRUPTS
1444  005514  000002    15$ RTI             ;,RETURN
1445  005516  004767  177234    16$ JSR     PC,$TYPEC       ;;ECHO CHAR
1446  005522  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
1447  005526  002420                BLT     18$             ;;BRANCH IF YES
1448  005530  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
1449  005534  003015                BGT     18$             ;;BRANCH IF YES
1450  005536  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
1451  005542  005766  000002                TST     2(SP)           ;,IS THIS THE FIRST CHAR
1452  005546  001403                BEQ     17$             ;,BRANCH IF YES
1453  005550  006316                ASL     (SP)            ;,NO, SHIFT PRESENT
1454  005552  006316                ASL     (SP)            ;;    CHAR OVER TO MAKE
1455  005554  006316                ASL     (SP)            ;,    ROOM FOR NEW ONE
1456  005556  005266  000002    17$ INC     2(SP)           ;,KEEP COUNT OF CHAR
1457  005562  056616  177776                BIS     -2(SP),(SP)     ;,SET IN NEW CHAR
```

```
1458  005566  000707                          BR      7$              ;;GET THE NEXT ONE
1459  005570  104401  005036          18$     TYPE    ;$QUES          ;;TYPE ?<CR><LF>
1460  005574  000720                          BR      20$             ;;SIMULATE CONTROL-U
1461                                  DSABL   LSB
1462
1463
1464                                  ;;**********************************************************************
1465                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1466                                  ;*CALL
1467                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
1468                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
1469                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
1470                                  ;
1471
1472  005576  011646                  $RDCHR  MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
1473  005600  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
1474  005606  105777  177476          1$      TSTB    @$TKS           ;;WAIT FOR
1475  005612  100375                          BPL     1$              ;;A CHARACTER
1476  005614  117766  177472  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
1477  005622  042766  177600  000004          BIC     #^C(177),4(SP)  ;;GET RID OF JUNK IF ANY
1478  005630  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
1479  005636  001013                          BNE     3$              ;;BRANCH IF NO
1480  005640  105777  177444          2$      TSTB    @$TKS           ;;WAIT FOR A CHARACTER
1481  005644  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
1482  005646  117746  177440                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
1483  005652  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1484  005656  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
1485  005662  001366                          BNE     2$              ;;IF NOT DISCARD IT
1486  005664  000750                          BR      1$              ;;YES, RESUME
1487  005666  026627  000004  000140  3$      CMP     4(SP),#140      ;;IS IT UPPER CASE?
1488  005674  002407                          BLT     4$              ;;BRANCH IF YES
1489  005676  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
1490  005704  003003                          BGT     4$              ;;BRANCH IF YES
1491  005706  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
1492  005714  000002                  4$.     RTI                     ;;GO BACK TO USER
1493                                  ;;**********************************************************************
1494                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1495                                  ;*CALL
1496                                  ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
1497                                  ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1498                                  ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
1499
1500  005716  010346                  $RDLIN  MOV     R3,-(SP)        ;;SAVE R3
1501  005720  012703  006024          1$      MOV     #$TTYIN,R3      ;;GET ADDRESS
1502  005724  022703  006034          2$      CMP     #$TTYIN+8 ,R3   ;;BUFFER FULL?
1503  005730  101405                          BLOS    4$              ;;BR IF YES
1504  005732  104407                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
1505  005734  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
1506  005736  122713  000177          10$     CMPB    #177,(R3)       ;;IS IT A RUBOUT
1507  005742  001003                          BNE     3$              ;;SKIP IF NOT
1508  005744  104401  005036          4$      TYPE    ,$QUES          ;;TYPE A '?'
1509  005750  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
1510  005752  111367  000044          3$      MOVB    (R3),9$         ;;ECHO THE CHARACTER
1511  005756  104401  006022                  TYPE    ,9$
1512  005762  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
1513  005766  001356                          BNE     2$              ;;LOOP IF NOT RETURN
```

```
1514  005770  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
1515  005774  104401  005040                  TYPE    ,$LF            ;;TYPE A LINE FEED
1516  006000  012603                          MOV     (SP)+,R3        ;,RESTORE R3
1517  006002  011646                          MOV     (SP),-(SP)      ;,ADJUST THE STACK AND PUT ADDRESS OF THE
1518  006004  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
1519  00601Z  012766  006024  000004          MOV     #$TTYIN,4(SP)
1520  006020  000002                          RTI                     ;;RETURN
1521  006022     000                  9$       BYTE    0               ;;STORAGE FOR ASCII CHAR  TO TYPE
1522  006023     000                           BYTE    0               ;;TERMINATOR
1523  006024  000010              $TTYIN:  .BLKB   8.                ;,RESERVE 8 BYTES FOR TTY INPUT
1524  006034  052536  005015     000  $CNTLU:  .ASCIZ  / U/<15><12>   ;;CONTROL "U"
1525  006041     136  006507  000012  $CNTLG.  .ASCIZ  / G/<15><12>   ;,CONTROL "G"
1526  006046  005015  053523  020122  $MSWR    .ASCIZ  <15><12>/SWR = /
1527  006054  020075     000
1528  006057     040  047040  053505  $MNEW    .ASCIZ  /  NEW = /
1529  006064  036440  000040
1530                                    SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1531
1532                                  ;;*****************************************************************
1533                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1534                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT
1535                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1536                                  ;*CALL.
1537                                  ;*      MOV     NUM,-(SP)       ;,NUMBER TO BE TYPED
1538                                  ;*      TYPOS                   ;,CALL FOR TYPEOUT
1539                                  ;*       BYTE   N               ;,N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1540                                  ;*      .BYTE   M               ;,M=1 OR 0
1541                                  ;*                                      ;,1=TYPE LEADING ZEROS
1542                                  ;*                                      ;,0=SUPPRESS LEADING ZEROS
1543                                  ;*
1544                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1545                                  ;*$TYPOS OR $TYPOC
1546                                  ;*CALL.
1547                                  ;*      MOV     NUM,-(SP)       ;,NUMBER TO BE TYPED
1548                                  ;*      TYPON                   ;,CALL FOR TYPEOUT
1549                                  ;*
1550                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1551                                  ;*CALL
1552                                  ;*      MOV     NUM,-(SP)       ;,NUMBER TO BE TYPED
1553                                  ;*      TYPOC                   ;,CALL FOR TYPEOUT
1554
1555  006070  017646  000000          $TYPOS.  MOV     @(SP),-(SP)    ;,PICKUP THE MODE
1556  006074  116667  000001  000211           MOVB    1(SP),$OFILL   ;,LOAD ZERO FILL SWITCH
1557  006102  112667  000207                  MOVB    (SP)+,$OMODE+1  ;,NUMBER OF DIGITS TO TYPE
1558  006106  062716  000002                  ADD     #2,(SP)         ;,ADJUST RETURN ADDRESS
1559  006112  000406                          BR      $TYPON
1560  006114  112767  000001  000171  $TYPOC:  MOVB    #1,$OFILL      ;,SET THE ZERO FILL SWITCH
1561  006122  112767  000006  000165          MOVB    #6,$OMODE+1     ;,SET FOR SIX(6) DIGITS
1562  006130  112767  000005  000154  $TYPON.  MOVB    #5,$OCNT        ;,SET THE ITERATION COUNT
1563  006136  010346                          MOV     R3,-(SP)        ;,SAVE R3
1564  006140  010446                          MOV     R4,-(SP)        ;,SAVE R4
1565  006142  010546                          MOV     R5,-(SP)        ;,SAVE R5
1566  006144  116704  000145                  MOVB    $OMODE+1,R4     ;,GET THE NUMBER OF DIGITS TO TYPE
1567  006150  005404                          NEG     R4
1568  006152  062704  000006                  ADD     #6,R4           ;,SUBTRACT IT FOR MAX  ALLOWED
1569  006156  110467  000132                  MOVB    R4,$OMODE       ;,SAVE IT FOR USE
```

```
1570  006162  116704  000125          MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
1571  006166  016605  000012          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
1572  006172  005003                  CLR     R3              ;;CLEAR THE OUTPUT WORD
1573  006174  006105              1$  ROL     R5              ;;ROTATE MSB INTO "C"
1574  006176  000404                  BR      3$              ;;GO DO MSB
1575  006200  006105              2$  ROL     R5              ;;FORM THIS DIGIT
1576  006202  006105                  ROL     R5
1577  006204  006105                  ROL     R5
1578  006206  010503                  MOV     R5,R3
1579  006210  006103              3$  ROL     R3              ;;GET LSB OF THIS DIGIT
1580  006212  105367  000076          DECB    $OMODE          ;;TYPE THIS DIGIT?
1581  006216  100016                  BPL     7$              ;;BR IF NO
1582  006220  042703  177770          BIC     #177770,R3      ;;GET RID OF JUNK
1583  006224  001002                  BNE     4$              ;;TEST FOR 0
1584  006226  005704                  TST     R4              ;;SUPPRESS THIS 0?
1585  006230  001403                  BEQ     5$              ;;BR IF YES
1586  006232  005204              4$  INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
1587  006234  052703  000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
1588  006240  052703  000040      5$  BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
1589  006244  110367  000040          MOVB    R3,8$           ;;SAVE FOR TYPING
1590  006250  104401  006310          TYPE    ,8$             ;;GO TYPE THIS DIGIT
1591  006254  105367  000032      7$  DECB    $OCNT           ;;COUNT BY 1
1592  006260  003347                  BGT     2$              ;;BR IF MORE TO DO
1593  006262  002402                  BLT     6$              ;;BR IF DONE
1594  006264  005204                  INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
1595  006266  000744                  BR      2$              ;;GO DO THE LAST DIGIT
1596  006270  012605              6$  MOV     (SP)+,R5        ;;RESTORE R5
1597  006272  012604                  MOV     (SP)+,R4        ;;RESTORE R4
1598  006274  012603                  MOV     (SP)+,R3        ;;RESTORE R3
1599  006276  016666  000002  000004  MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
1600  006304  012616                  MOV     (SP)+,(SP)
1601  006306  000002                  RTI                     ;;RETURN
1602  006310     000              8$   BYTE    0               ;;STORAGE FOR ASCII DIGIT
1603  006311     000                   BYTE    0               ;;TERMINATOR FOR TYPE ROUTINE
1604  006312     000          $OCNT    BYTE    0               ;;OCTAL DIGIT COUNTER
1605  006313     000          $OFILL   BYTE    0               ;;ZERO FILL SWITCH
1606  006314  000000          $OMODE   .WORD   0               ;;NUMBER OF DIGITS TO TYPE
1607                           .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
1608
1609                           ;;**************************************************************************
1610                           ;*SAVE R0-R5
1611                           ;*CALL
1612                           ;*      SAVREG
1613                           ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE
1614                           ;*
1615                           ;*TOP---(+16)
1616                           ;* +2---(+18)
1617                           ;* +4---R5
1618                           ;* +6---R4
1619                           ;* +8---R3
1620                           ;*+10---R2
1621                           ;*+12---R1
1622                           ;*+14---R0
1623
1624  006316                   $SAVREG
1625  006316  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
```

```
1626  006320  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1627  006322  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1628  006324  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1629  006326  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
1630  006330  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1631  006332  016646  000022                MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
1632  006336  016646  000022                MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
1633  006342  016646  000022                MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
1634  006346  016646  000022                MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
1635  006352  000002                        RTI
1636
1637                                  ;*RESTORE R0-R5
1638                                  ;*CALL
1639                                  ;*      RESREG
1640  006354                          $RESREG
1641  006354  012666  000022                MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
1642  006360  012666  000022                MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
1643  006364  012666  000022                MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
1644  006370  012666  000022                MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
1645  006374  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
1646  006376  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
1647  006400  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
1648  006402  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
1649  006404  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
1650  006406  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
1651  006410  000002                        RTI
1652                                   SBTTL  TRAP DECODER
1653
1654                                  ;;*******************************************************************
1655                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1656                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1657                                  ;*OF THE DESIRED ROUTINE.  THEN USING THE ADDRESS OBTAINED IT WILL
1658                                  ;*GO TO THAT ROUTINE
1659
1660  006412  010046                  $TRAP   MOV     R0,-(SP)        ;;SAVE R0
1661  006414  016600  000002                 MOV     2(SP),R0        ;;GET TRAP ADDRESS
1662  006420  005740                         TST     -(R0)           ;;BACKUP BY 2
1663  006422  111000                         MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
1664  006424  006300                         ASL     R0              ;;POSITION FOR INDEXING
1665  006426  016000  006446                 MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
1666  006432  000200                         RTS     R0              ;;GO TO ROUTINE
1667
1668
1669                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
1670
1671  006434  011646                  $TRAP2  MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
1672  006436  016666  000004  000002         MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
1673  006444  000002                         RTI                     ;;RESTORE THE PSW
1674
1675                                   SBTTL  TRAP TABLE
1676
1677                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1678                                  ;*BY THE "TRAP" INSTRUCTION
1679
1680                                  ;       ROUTINE
1681                                  ;       -------
```

```
1682  006446  006434                  $TRPAD   WORD   $TRAP2
1683  006450  004544                           $TYPE  ,,CALL=TYPE    TRAP+1(104401)  TTY TYPEOUT ROUTINE
1684  006452  006114                           $TYPOC ,,CALL=TYPOC   TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1685  006454  006070                           $TYPOS ,,CALL=TYPOS   TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
1686  006456  006130                           $TYPON ,,CALL=TYPON   TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
1687
1688  006460  005364                           $GTSWR ,,CALL=GTSWR   TRAP+5(104405)  GET SOFT-SWR SETTING
1689
1690  006462  005314                           $CKSWR ,,CALL=CKSWR   TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
1691  006464  005576                           $RDCHR ,,CALL=RDCHR   TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
1692  006466  005716                           $RDLIN ,,CALL=RDLIN   TRAP+10(104410) TTY TYPEIN STRING ROUTINE
1693  006470  006316                           $SAVREG,,CALL=SAVREG  TRAP+11(104411) SAVE R0-R5 ROUTINE
1694  006472  006354                           $RESREG,,CALL=RESREG  TRAP+12(104412) RESTORE R0-R5 ROUTINE
1695
1696
1697                                  ;MESSAGES
1698
1699
1700  006474  030062                  ROMNUM   ASCII  /20/
1701  006476  032463                           ASCII  /35/
1702  006500  031463                           ASCII  /33/
1703  006502  032063                           ASCII  /34/
1704  006504  031063                           ASCII  /32/
1705  006506  177607  000377          BELL     ASCIZ  <207><377><377>
1706
1707
1708  006512  044504  043501  020056  DRHEAD   ASCIZ  /DIAG  ROM (E20) /
1709  006520  047522  020115  042450
1710  006526  030062  020051     000
1711  006533     012  047502  052117  BRHEAD   ASCII  <12>/BOOTSTRAP ROM ENTRY POINTS AND DEVICE CODES/<15><12>
1712  006540  052123  040522  020120
1713  006546  047522  020115  047105
1714  006554  051124  020131  047520
1715  006562  047111  051524  040440
1716  006570  042116  042040  053105
1717  006576  041511  020105  047503
1718  006604  042504  006523     012
1719  006611     114  041517  020056           ASCIZ  /LOC             NO DIAG        RUN DIAG        DEVICE CODE/
1720  006616  020040  020040  020040
1721  006624  020040  004440  047040
1722  006632  020117  044504  043501
1723  006640  004456  052522  020116
1724  006646  044504  043501  004456
1725  006654  042504  044526  042503
1726  006662  041440  042117  000105
1727  006670  051520  052506  047504  PFHEAD   ASCIZ  @PSEUDO POWER-FAIL VECTOR ADR /NEW PC@
1728  006676  050040  053517  051105
1729  006704  043066  044501  020114
1730  006712  042526  052103  051117
1731  006720  040440  051104  027456
1732  006726  042516  020127  041520
1733  006734     000
1734  006735     103  052517  042114  ER2MSG   ASCIZ  /COULD NOT FIND DEVICE CODE /
1735  006742  047040  052117  043040
1736  006750  047111  020104  042504
1737  006756  044526  042503  041440
```

```
1738  006764  042117  020105     000
1739  006771     106  052517  042116  ER1MSG   ASCIZ  /FOUND UNEXPECTED DEVICE CODE /
1740  006776  052440  042516  050130
1741  007004  041505  042524  020104
1742  007012  042504  044526  042503
1743  007020  041440  042117  020105
1744  007026     000
1745  007027     120  053517  051105  ER3MSG   ASCII  /POWER-FAIL VECTOR ERROR/<15><12>
1746  007034  043055  044501  020114
1747  007042  042526  052103  051117
1748  007050  042440  051122  051117
1749  007056  005015
1750  007060  042411  050130  041505          ASCIZ  /       EXPECTED        RECIEVED/<15><12>
1751  007066  042524  004504  042522
1752  007074  044503  053105  042105
1753  007102  005015     000
1754  007105     120  053517  051105  ER4MSG   ASCII  /POWER-FAIL DATA ERROR/<15><12>
1755  007112  043055  044501  020114
1756  007120  040504  040524  042440
1757  007126  051122  051117  005015
1758  007134  042411  050130  041505          ASCIZ  /       EXPECTED        RECIEVED/<15><12>
1759  007142  042524  004504  042522
1760  007150  044503  053105  042105
1761  007156  005015     000
1762  007161     103  041522  042440  CRCMSG   ASCIZ  /CRC ERROR IN ROM E-/
1763  007166  051122  051117  044440
1764  007174  020116  047522  020115
1765  007202  026505     000
1766  007205     103  052517  042114  PFMSG    ASCIZ  /COULD NOT DETERMINE POWER-FAIL VECTOR ADDRESS/
1767  007212  047040  052117  042040
1768  007220  052105  051105  044515
1769  007226  042516  050040  053517
1770  007234  051105  043055  044501
1771  007242  020114  042526  052103
1772  007250  051117  040440  042104
1773  007256  042522  051523     000
1774  007263     116  020117  047522  NOROMM   ASCII  /NO ROMS TEST ERROR/<15><12>
1775  007270  051515  052040  051505
1776  007276  020124  051105  047522
1777  007304  006522     012
1778  007307     040  020040  020040          ASCIZ  /         VALUE   ADDRESS/<15><12>
1779  007314  020040  053040  046101
1780  007322  042526  020040  040440
1781  007330  042104  042522  051523
1782  007336  005015     000
1783  007341     015  051012  046517  SEQMSG   ASCII  <15><12>/ROM SEQUENCE IS INCORRECT AS PER /
1784  007346  051440  050505  042525
1785  007354  041516  020105  051511
1786  007362  044440  041516  051117
1787  007370  042522  052103  040440
1788  007376  020123  042520  020122
1789  007404  047111  052123  046101          ASCIZ  /INSTALLATION PROCEDURE /
1790  007412  040514  044524  047117
1791  007420  050040  047522  042503
1792  007426  052504  042522  000056
1793                                  EVEN
```

```
1794
1795
1796                                    ;BUFFERS
1797
1798  007434  000030                    BUF1
1799  007434  000000                            WORD    0
1800  007436  000000                            WORD    0
1801  007440  000000                            WORD    0
1802  007442  000000                            WORD    0
1803  007444  000000                            WORD    0
1804  007446  000000                            WORD    0
1805  007450  000000                            WORD    0
1806  007452  000000                            WORD    0
1807  007454  000000                            WORD    0
1808  007456  000000                            WORD    0
1809  007460  000000                            WORD    0
1810  007462  000000                            WORD    0
1811  007464  000000                            WORD    0
1812  007466  000000                            WORD    0
1813  007470  000000                            WORD    0
1814  007472  000000                            WORD    0
1815  007474  000000                            WORD    0
1816  007476  000000                            WORD    0
1817  007500  000000                            WORD    0
1818  007502  000000                            WORD    0
1819  007504  000000                            WORD    0
1820  007506  000000                            WORD    0
1821  007510  000000                            WORD    0
1822  007512  000000                            WORD    0
1823  007514  000000                    BUF2    WORD    0
1824  007516  000000                            WORD    0
1825  007520  000010                    OCTBUF  .BLKB   10
1826  007530  000400                    MES1    .BLKB   400
1827  010130  000100                    MES2    .BLKB   100
1828  010230  000400                    ERRMSG  BLKB    400                     ;+
1829
1830                                    ;+
1831                                    ;+THE FOLLOWING CODE WAS ADDED IN REV B0
1832                                    ;+
1833                                    ;+"CLEAR" CLEARS THE NEW LABEL AND BUFFER LOCATIONS
1834                                    ;+
1835                                    ;+"CON1" AND "CON2" FILL "BUF1" WITH THE CONTINUATION ROM DATA AND
1836                                    ;+SET ERROR FLAGS WHEN NECESSARY
1837                                    ;+
1838                                    ;+"HOLCK1" AND "HOLCK2" CHECK FOR HOLES TO VERIFY THE POWER FAIL
1839                                    ;+VECTOR ADDRESS, ALSO SET 11/60 INDICATOR WHEN APPLICABLE
1840                                    ;+
1841                                    ;+"ROMTYP" AND "ROMNM" FILL THE MESSAGE BUFFER WITH A CONTINUATION ROM
1842                                    ;+MESSAGE AND WITH A ROM IDENTIFICATION MESSAGE
1843                                    ;+
1844
1845  010630  012700  011774    CLEAR   MOV     #SEQBUF,        R0      ;+SET UP TO CLEAR LOCATIONS
1846  010634  005020            1$      CLR     (R0)+                   ;+
1847  010636  020027  012036            CMP     R0,     #FINISH         ;+ALL CLEARED?
1848  010642  101774                    BLOS    1$                      ;+BRANCH IF NO
1849  010644  000207                    RTS     PC                      ;+
```

```
1850  010646  030167  001152          CON1    BIT     R1,     CONFIN          ,+CONTINUATION CHIP?
1851  010652  001001                          BNE     1$                      ,+BRANCH IF YES
1852  010654  000207                          RTS     PC                      ,+
1853  010656  050167  001132          1$      BIS     R1,     CONER1          ,+CHIP DOES NOT BELONG HERE-SET ERROR FLAG
1854  010662  012720  000001                  MOV     #1,     (R0)+           ,+SET CONTINUATION ROM INDICATOR
1855  010666  012720  177777                  MOV     #-1,    (R0)+           ,+ILLEGAL ROM'S DEVICE CODE =-1
1856  010672  000207                          RTS     PC                      ,+
1857  010674  005703                  CON2    TST     R3                      ;+POINTING OVER THIS CHIP?
1858  010676  001001                          BNE     1$                      ,+BRANCH IF YES
1859  010700  000207                          RTS     PC                      ,+
1860  010702  030167  001116          1$      BIT     R1,     CONFIN          ,+CONTINUATION CHIP?
1861  010706  001005                          BNE     2$                      ,+BRANCH IF YES
1862  010710  014067  001102                  MOV     -(R0),  CONER2          ,+CHIP IS MISSING-SAVE DEVICE CODE
1863  010714  062700  000002                  ADD     #2,     R0              ,+
1864  010720  000207                          RTS     PC                      ,+
1865  010722  012710  000001          2$      MOV     #1,     (R0)            ,+SET CONTINUATION CHIP INDICATOR
1866  010726  014002                          MOV     -(R0),  R2              ,+GET PREVIOUS DEVICE CODE
1867  010730  062700  000004                  ADD     #4,     R0              ,+POINT TO CORRECT LOCATION
1868  010734  010220                          MOV     R2,     (R0)+           ,+PUT IN DEVICE CODE
1869  010736  000207                          RTS     PC                      ,+
1870  010740  032767  000002  170150  HOLCK1. BIT     #2,     ROMFIN          ,+A CHIP IN SOCKET #1?
1871  010746  001415                          BEQ     END                     ,+BRANCH IF NO
1872  010750  000207                          RTS     PC                      ,+
1873  010752  032767  000004  170136  HOLCK2  BIT     #4,     ROMFIN          ,+A CHIP IN SOCKET #2?
1874  010760  001401                          BEQ     1$                      ,+BRANCH IF NO
1875  010762  000207                          RTS     PC                      ,+
1876  010764  122737  000777  173224  1$      CMPB    #777,   @#173224        ,+ARE SWITCHES SET?
1877  010772  001403                          BEQ     END                     ,+BRANCH IF NO
1878  010774  052767  000002  001020          BIS     #2,     FLAG            ,+SET 11/60 INDICATOR
1879  011002  016716  001012          END     MOV     RETURN, (R6)            ,+SET UP ALTERNATE RETURN
1880  011006  000207                          RTS     PC                      ,+
1881  011010  005713                  ROMTYP. TST     (R3)                    ,+ARE WE FINISHED?
1882  011012  001003                          BNE     1$                      ,+BRANCH IF NO
1883  011014  016716  001016                  MOV     FINISH, (R6)            ,+SET UP ALTERNATE RETURN
1884  011020  000207                          RTS     PC                      ;+
1885  011022  022713  000001          1$      CMP     #1,     (R3)            ,+CONTINUATION CHIP?
1886  011026  001401                          BEQ     2$                      ,+BRANCH IF YES
1887  011030  000414                          BR      ROMNM                   ,+ITS A DEVICE CHIP-FIND SOCKET #
1888  011032  004767  000024          2$      JSR     PC,     ROMNM           ;+FIND SOCKET #
1889  011036  012705  012050                  MOV     #CONMES,        R5      ,+LOAD MSG INTO BUFFER
1890  011042  004767  173314                  JSR     PC,     FILBUF          ,+
1891  011046  000011                          HT                              ,+
1892  011050  062703  000004                  ADD     #4,     R3              ,+POP OVER CONTINUATION CHIP DATA
1893  011054  016716  000740                  MOV     RETURN, (R6)            ,+SET UP ALTERNATE RETURN
1894  011060  000207                          RTS     PC                      ,+
1895  011062  032713  000170          ROMNM   BIT     #170,   (R3)            ,+BEGINNING A NEW ROM?
1896  011066  001015                          BNE     3$                      ,+BRANCH IF NOT
1897  011070  000241                  1$      CLC                             ,+
1898  011072  006101                          ROL     R1                      ,+POINT TO NEXT SOCKET
1899  011074  030167  170016                  BIT     R1,     ROMFIN          ,+IS A ROM THERE?
1900  011100  001003                          BNE     2$                      ,+BRANCH IF YES
1901  011102  062702  000002                  ADD     #2,     R2              ,+POINT TO NEXT MSG
1902  011106  000770                          BR      1$                      ,+CONTINUE
1903  011110  012205                  2$      MOV     (R2)+,  R5              ,+LOAD ROM ID
1904  011112  004767  173244                  JSR     PC,     FILBUF          ,+INTO THE MSG BUFFER
1905  011116  000015                          CR                              ,+
```

```
1906  011120  000402                          BR      4$                      ,+
1907  011122  112724  000011          3$      MOVB    #HT,    (R4)+           ,+LOAD A TAB
1908  011126  000207                  4$      RTS     PC                      ,+
1909
1910                                  ,+SEQUENCE TEST
1911                                  ,+THIS TEST VERIFIES THAT CONTINUATION CHIPS ARE LOCATED WHERE
1912                                  ,+THEY BELONG, THAT THERE ARE NO DUPLICATE DEVICE ROMS,
1913                                  ,+THAT THE ROMS ARE IN ALPHABETICAL ORDER, THAT THERE ARE NO
1914                                  ;+HOLES, AND THAT SOCKET #2 HAS A CHIP IF THE PROCESSOR IS AN 11/60
1915                                  ,+IF THE ONLY PROBLEM IS ALPHABETICAL ORDER AND/OR HOLES,
1916                                  ;+THEN A CORRECT SEQUENCE IS DETERMINED AND PRINTED OUT
1917
1918                                  ;+FIRST, CHECK FOR ILLEGAL CONTINUATION ROM ERRORS
1919                                  ;+IF ANY ARE FOUND, PRINT ERROR MSG AND EXIT
1920
1921  011130  012767  000011  000674  SEQTST: MOV     #HT,    ARG2            ,+SET UP FOR "ERRHAN"
1922  011136  005767  000652                  TST     CONER1                  ,+WAS THERE AN ERROR?
1923  011142  001417                          BEQ     2$                      ,+BRANCH IF NO
1924  011144  016703  000644                  MOV     CONER1, R3              ,+THE BIT IDENTIFIES THE CHIP
1925  011150  012702  012040                  MOV     #MESTAB,        R2      ,+POINT TO MSG TABLE
1926  011154  012767  000400  167726          MOV     #CONONE,        ROMERR  ,+SET ERROR FLAG
1927  011162  032703  000002          1$:     BIT     #2,     R3              ,+IDENTIFY THE ROM FOR PRINTOUT
1928  011166  001026                          BNE     3$                      ,+BRANCH IF FOUND IT
1929  011170  062702  000002                  ADD     #2,     R2              ,+SET UP FOR NEXT ROM MSG
1930  011174  000241                          CLC                             ,+
1931  011176  006003                          ROR     R3                      ,+SET UP TO IDENTIFY NEXT ROM
1932  011200  000770                          BR      1$                      ,+
1933  011202  005767  000610          2$      TST     CONER2                  ,+WAS THERE AN ERROR?
1934  011206  001423                          BEQ     FILTAB                  ,+BRANCH IF NO
1935  011210  012767  001000  167672          MOV     #CONTWO,        ROMERR  ,+SET ERROR FLAG
1936  011216  012767  007530  000604          MOV     #MES1,  ARG1            ,+PASS THIS ADDRESS TO ERRHAN
1937  011224  000367  000566                  SWAB    CONER2                  ,+BUILD THE MSG
1938  011230  016767  000562  176272          MOV     CONER2, MES1            ,+
1939  011236  105067  176270                  CLRB    MES1+2                  ,+
1940  011242  000402                          BR      4$                      ,+REPORT THE ERROR
1941  011244  011267  000560          3$      MOV     (R2),   ARG1            ,+PASS MSG ADDRESS TO "ERRHAN"
1942  011250  004767  172516          4$      JSR     PC,     ERRHAN          ,+REPORT ERROR
1943  011254  000207                          RTS     PC                      ,+
1944
1945                                  ,+IF THERE WERE NO CONTINUATION ROM ERRORS, BUILD THE FOLLOWING TABLE
1946                                  ,+
1947                                  ,+      SEQBUF          FIRST DEVICE CODE
1948                                  ,+                      # OF CONTINUATION ROMS FOR FIRST DEVICE
1949                                  ,+                      SECOND DEVICE CODE
1950                                  ,+                      # OF CONTINUATION ROMS FOR SECOND DEVICE
1951                                  ,+
1952                                  ,+
1953                                  ,+      ENDSEQ:         FOURTH (LAST) DEVICE CODE
1954                                  ,+
1955                                  ,+FOR A "HOLE", THE DEVICE CODE WILL BE A 0
1956
1957  011256  012703  011774          FILTAB  MOV     #SEQBUF,        R3      ,+SET UP TO FILL A TABLE CALLED "SEQBUF"
1958  011262  012702  173000                  MOV     #173000,        R2      ,+POINT TO FIRST ROM
1959  011266  012700  000001                  MOV     #1,     R0              ,+
1960  011272  000241                  1$      CLC                             ,+
1961  011274  006100                          ROL     R0                      ,+POINT TO NEXT ROM
```

```
1962  011276  030067  167614          BIT     R0,     ROMFIN          ,+IS IT A HOLE?
1963  011302  001404                  BEQ     3$                      ,+BRANCH IF YES
1964  011304  030067  000514  2$      BIT     R0,     CONFIN          ,+IS IT A CONTINUATION CHIP?
1965  011310  001011                  BNE     4$                      ,+BRANCH IF YES
1966  011312  011213                  MOV     (R2),   (R3)            ;+ITS A DEVICE ROM-PUT DEVICE CODE INTO TABLE
1967  011314  062703  000004  3$      ADD     #4,     R3              ;+POINT TO NEXT DEVICE CODE LOCATION
1968  011320  062702  000200  5$      ADD     #200,   R2              ;+POINT TO NEXT ROM
1969  011324  030027  000020          BIT     R0,     #20             ,+ALL DONE?
1970  011330  001760                  BEQ     1$                      ,+NO-BRANCH
1971  011332  000404                  BR      ALPHCK                  ,+TABLE COMPLETELY BUILT
1972  011334  005243  4$      INC     -(R3)                           ,+COUNT ONE CONTINUATION CHIP IN LOCATION
1973  011336  062703  000002          ADD     #2,     R3              ,+POINT TO NEXT DEVICE CODE LOCATION
1974  011342  000766                  BR      5$                      ;+
1975
1976                          ,+NOW CHECK DEVICE CODES IN "SEQBUF" FOR ALPHABETICAL ORDER AND
1977                          ,+CHECK FOR "HOLES".  IF THERE IS A DUPLICATE DEVICE CODE, PRINT
1978                          ,+AN ERROR MESSAGE AND EXIT.  IF DEVICE CODES NEED SORTING
1979                          ,+AND/OR HOLES FILLING, DO IT AND SET THE SEQUENCE ERROR
1980                          ,+FLAG.  ALSO CHECK FOR THE 11/60 SPECIAL CASE.
1981
1982  011344  012703  011774  ALPHCK: MOV     #SEQBUF,        R3      ;+SET UP TO CHECK ALPHABETIC SEQUENCE IN "SEQBUF
1983  011350  010304  1$      MOV     R3,     R4                      ;+WILL BE COMPARING (R3) TO (R4)
1984  011352  062704  000004  2$      ADD     #4,     R4              ;+WHERE R3 AND R4 POINT TO DEVICE CODE LOCATIONS
1985  011356  020427  012010          CMP     R4,     #ENDSEQ         ;+PAST THE LAST DEVICE CODE?
1986  011362  101010                  BHI     3$                      ;+BRANCH IF YES
1987  011364  005713                  TST     (R3)                    ,+IS THERE A DEVICE CODE HERE?
1988  011366  001446                  BEQ     7$                      ,+BRANCH IF NO
1989  011370  005714                  TST     (R4)                    ,+IS THERE A DEVICE CODE HERE?
1990  011372  001767                  BEQ     2$                      ;+BRANCH IF NO
1991  011374  021314                  CMP     (R3),   (R4)            ;+IS THE SEQUENCE CORRECT?
1992  011376  001410                  BEQ     4$                      ;+BRANCH IF NO-FOUND A DUPLICATE
1993  011400  003025                  BGT     5$                      ;+BRANCH IF NO-NEED TO DO A SHIFT
1994  011402  000763                  BR      2$                      ,+YES-CONTINUE
1995  011404  062703  000004  3$.     ADD     #4,     R3              ;+POINT TO NEXT DEVICE CODE
1996  011410  020327  012010          CMP     R3,     #ENDSEQ         ;+ALL DONE?
1997  011414  103056                  BHIS    9$                      ;+BRANCH IF YES
1998  011416  000754                  BR      1$                      ;+NOT YET
1999  011420  012767  002000  167462  4$      MOV     #DUPERR,        ROMERR  ,+SET ERROR FLAG
2000  011426  012767  007530  000374          MOV     #MES1,  ARG1            ,+PASS THIS ADDRESS TO "ERRHAN"
2001  011434  000314                  SWAB    (R4)                    ,+BUILD THE MSG
2002  011436  011467  176066          MOV     (R4),   MES1            ,+
2003  011442  105067  176064          CLRB    MES1+2                  ,+
2004  011446  004767  172320          JSR     PC,     ERRHAN          ,+REPORT THE ERROR
2005  011452  000207                  RTS     PC                      ,+
2006  011454  010301  5$      MOV     R3,     R1                      ,+SET UP FOR THE SHUFFLE
2007  011456  010402                  MOV     R4,     R2              ,+
2008  011460  011200  6$      MOV     (R2),   R0                      ,+SHIFT THE VALUES
2009  011462  011122                  MOV     (R1),   (R2)+           ,+
2010  011464  010021                  MOV     R0,     (R1)+           ,+
2011  011466  011200                  MOV     (R2),   R0              ,+
2012  011470  011112                  MOV     (R1),   (R2)            ,+
2013  011472  010011                  MOV     R0,     (R1)            ;+
2014  011474  012767  000200  167406          MOV     #SEQERR,        ROMERR  ,+SET THE SEQUENCE ERROR FLAG
2015  011502  000723                  BR      2$                      ,+CONTINUE SORTING
2016  011504  010305  7$      MOV     R3,     R5                      ,+SET UP TO SEE IF WE HAVE A "HOLE"
2017  011506  020527  012010  8$      CMP     R5,     #ENDSEQ         ,+END OF TABLE?
```

```
2018  011512  103017                          BHIS    9$                      ;+YES-THIS WAS NOT A "HOLE"
2019  011514  062705  000004                  ADD     #4,     R5              ;+NO-POINT TO NEXT DEVICE CODE LOCATION
2020  011520  005715                          TST     (R5)                    ;+IS THERE A ROM HERE?
2021  011522  001771                          BEQ     8$                      ;+BRANCH IF NO
2022  011524  032767  000032  167364          BIT     #32,    ROMFIN          ;+A ROM IN SOCKET #2 ONLY?
2023  011532  001004                          BNE     11$                     ;+BRANCH IF NO
2024  011534  105737  173224                  TSTB    @#173224                ;+11/60 SPECIAL CASE?
2025  011540  001401                          BEQ     11$                     ;+BRANCH IF NO
2026  011542  000422                          BR      10$                     ;+THE ONLY ROM MUST STAY IN SOCKET #2
2027  011544  010301                  11$     MOV     R3,     R1              ;+YES-THERE IS A HOLE THAT CAN BE FILLED
2028  011546  010502                          MOV     R5,     R2              ;+GET READY TO SHIFT THE VALUES
2029  011550  000743                          BR      6$                      ;+FILL IN THE HOLE
2030  011552  032767  000002  000242  9$      BIT     #2,     FLAG            ;+AN 11/60?
2031  011560  001413                          BEQ     10$                     ;+BRANCH IF NO
2032  011562  005767  000212                  TST     SEQBUF+4                ;+IS SOCKET #2 EMPTY?
2033  011566  001010                          BNE     10$                     ;+BRANCH IF NO
2034  011570  005767  000200                  TST     SEQBUF                  ;+IS SOCKET #1 EMPTY?
2035  011574  001405                          BEQ     10$                     ;+BRANCH IF YES
2036  011576  012701  011774                  MOV     #SEQBUF,        R1      ;+SET UP TO SHIFT
2037  011602  012702  012000                  MOV     #SEQBUF+4,      R2      ;+CHIPS #1 AND #2
2038  011606  000724                          BR      6$                      ;+DO THE SHIFT
2039  011610  032767  000200  167272  10$.    BIT     #SEQERR,        ROMERR  ;+WAS THERE A SEQUENCE ERROR?
2040  011616  001001                          BNE     OUTTAB                  ;+BRANCH IF YES
2041  011620  000207                          RTS     PC                      ;+
2042
2043                                  ;+THE FOLLOWING ROUTINE BUILDS THE CORRECT SEQUENCE MSG TO BE
2044                                  ;+PRINTED AS AN ERROR MSG BY "ERRHAN"
2045
2046  011622  012767  007530  000200  OUTTAB: MOV     #MES1,  ARG1            ;+PASS MSG ADR  TO "ERRHAN"
2047  011630  012767  000015  000174          MOV     #CR,    ARG2            ;+PASS THIS TOO
2048  011636  012700  011774                  MOV     #SEQBUF,        R0      ;+
2049  011642  012702  012040                  MOV     #MESTAB,        R2      ;+
2050  011646  012704  007530                  MOV     #MES1,  R4              ;+START BUILDING THE ERROR MSG
2051  011652  012705  012311                  MOV     #ERMES4,        R5      ;+PUT IN THE FIRST PART
2052  011656  004767  172500                  JSR     PC,     FILBUF          ;+
2053  011662  000015                          CR                              ;+
2054  011664  005710                          TST     (R0)                    ;+IS A ROM IN SOCKET #1?
2055  011666  001004                          BNE     1$                      ;+BRANCH IF YES
2056  011670  062700  000004                  ADD     #4,     R0              ;+DO SOCKET #2 ONLY
2057  011674  062702  000002                  ADD     #2,     R2              ;+
2058  011700  012767  010130  000126  1$      MOV     #MES2,  BUILD           ;+SET UP THE DEVICE CODE PART
2059  011706  000310                          SWAB    (R0)                    ;+
2060  011710  012067  176214                  MOV     (R0)+,  MES2            ;+
2061  011714  105067  176212                  CLRB    MES2+2                  ;+
2062  011720  012205                  2$      MOV     (R2)+,  R5              ;+IDENTIFY THE ROM #
2063  011722  004767  172434                  JSR     PC,     FILBUF          ;+
2064  011726  000015                          CR                              ;+
2065  011730  016705  000100                  MOV     BUILD,  R5              ;+PUT IN EITHER DEVICE CODE OR CONTINUATION MSG
2066  011734  004767  172422                  JSR     PC,     FILBUF          ;+
2067  011740  000011                          HT                              ;+
2068  011742  005720                          TST     (R0)+                   ;+ANY CONTINUATION ROMS?
2069  011744  001405                          BEQ     3$                      ;+BRANCH IF NO
2070  011746  006340                          DEC     -(R0)                   ;+COUNT DOWN ONE ROM
2071  011750  012767  012050  000056          MOV     #CONMES,        BUILD   ;+SET UP FOR CONTINUATION MSG
2072  011756  000760                          BR      2$                      ;+CONTINUE
2073  011760  005710                  3$      TST     (R0)                    ;+ANY MORE DEVICE ROMS?
```

```
2074  011762  001346                    BNE     1$                      ;+BRANCH IF YES
2075  011764  105014                    CLRB    (R4)                    ;+MUST END WITH 0 BYTE
2076  011766  004767  172000            JSR     PC,     ERRHAN          ;+REPORT THE ERROR
2077  011772  000207                    RTS     PC                      ,+
2078
2079
2080  011774  000000            SEQBUF  WORD    0                       ,+FIRST DEVICE CODE
2081  011776  000000                    WORD    0                       ,+# OF CONTINUATION CHIPS FOR FIRST DEVICE
2082  012000  000000                    .WORD   0                       ,+SECOND DEVICE CODE, ETC
2083  012002  000000                    .WORD   0                       ,+
2084  012004  000000                    WORD    0                       ,+
2085  012006  000000                    WORD    0                       ,+
2086  012010  000000            ENDSEQ  WORD    0                       ;+FOURTH (LAST) DEVICE CODE
2087  012012  000000                    WORD    0                       ;+
2088  012014  000000            CONER1  WORD    0                       ;+EXTRA CONTINUATION CHIP ERROR FLAG
2089  012016  000000            CONER2  WORD    0                       ,+MISSING CONTINUATION CHIP FOR THIS DEVICE
2090  012020  000000            RETURN  WORD    0                       ;+ALTERNATE RETURN ADDRESS
2091  012022  000000            FLAG    WORD    0                       ;+BIT 1: 11/60
2092  012024  000000            CONFIN  WORD    0                       ;+CONTINUATION ROM FOUND INDICATOR
2093  012026  000000            DEVFIN. .WORD   0                       ;+DEVICE ROM FOUND INDICATOR
2094  012030  000000            ARG1    .WORD   0                       ,+PASSES MSG ADR.  TO "ERRHAN"
2095  012032  000000            ARG2.   WORD    0                       ,+PASSES CR OR HT TO "ERRHAN"
2096  012034  000000            BUILD   .WORD   0                       ,+
2097  012036  000000            FINISH. .WORD   0                       ,+ALTERNATE RETURN ADDRESS
2098  012040  012336            MESTAB  ROM1                            ,+ASCII MSG TABLE
2099  012042  012352                    ROM2                            ,+
2100  012044  012366                    ROM3                            ,+
2101  012046  012402                    ROM4                            ,+
2102  012050  051511  040440  041440  CONMES:  ASCIZ /IS A CONTINUATION ROM/
2103  012056  047117  044524  052516
2104  012064  052101  047511  020116
2105  012072  047522  000115
2106  012076  006015  020101  047503  ERMES1  .ASCIZ <15><12>/A CONTINUATION ROM IS INCORRECTLY LOCATED IN/
2107  012104  052116  047111  040525
2108  012112  044524  047117  051040
2109  012120  046517  044440  020123
2110  012126  047111  047503  051122
2111  012134  041505  046124  020131
2112  012142  047514  040503  042524
2113  012150  020104  047111     000
2114  012156     015  040412  041440  ERMES2   ASCIZ <15><12>/A CONTINUATION ROM IS MISSING FOR DEVICE CODE/
2115  012162  047117  044524  052516
2116  012170  052101  047511  020116
2117  012176  047522  020115  051511
2118  012204  046440  051511  044523
2119  012212  043516  043040  051117
2120  012220  042040  053105  041511
2121  012226  020105  047503  042504
2122  012234     000
2123  012236     015  052012  042510  ERMES3   ASCIZ <15><12>/THERE IS A DUPLICATE ROM WITH DEVICE CODE/
2124  012242  042522  044440  020123
2125  012250  020101  052604  046120
2126  012256  041511  052101  020105
2127  012264  047522  020115  044527
2128  012272  044124  042040  053105
2129  012300  041511  020105  047503
```

```
 2130  012306  042504     000
 2131  012311     123  050505  042525  ERMES4: .ASCIZ /SEQUENCE SHOULD BE /<12>
 2132  012316  041516  020105  044123
 2133  012324  052517  042114  041040
 2134  012332  035105  000012
 2135  012336  047522  020115  024061  ROM1·    ASCIZ /ROM 1(E35) /
 2136  012344  031505  024465  000040
 2137  012352  047522  020115  024062  ROM2:   .ASCIZ /ROM 2(E33) /
 2138  012360  031505  024463  000040
 2139  012366  047522  020115  024063  ROM3    .ASCIZ /ROM 3(E34) /
 2140  012374  031505  024464  000040
 2141  012402  047522  020115  024064  ROM4     ASCIZ /ROM 4(E32) /
 2142  012410  031505  024462  000040
 2143
 2144
 2145          000001                    END
```

```
ABASE = 000000            549     590
ACDW1 = 000000            549     592
ACDW2 = 000000            549     593
ACPUOP= 000000            549     564
ADDW0 = 000000            549     594
ADDW1 = 000000            549     595
ADDW10= 000000            549     604
ADDW11= 000000            549     605
ADDW12= 000000            549     606
ADDW13= 000000            549     607
ADDW14= 000000            549     608
ADDW15= 000000            549     609
ADDW2 = 000000            549     596
ADDW3 = 000000            549     597
ADDW4 = 000000            549     598
ADDW5 = 000000            549     599
ADDW6 = 000000            549     600
ADDW7 = 000000            549     601
ADDW8 = 000000            549     602
ADDW9 = 000000            549     603
ADEVCT= 000000            549     555
ADEVM1 = 000000           549     591
AENV  = 000000            549     560
AENVM1 = 000000           549     561
AFATAL= 000000            549     552
ALPHCK  011344           1971    1982#
AMADR1= 000000            549     577
AMADR2= 000000            549     581
AMADR3= 000000            549     584
AMADR4= 000000            549     587
AMAMS1= 000000            549     571
AMAMS2= 000000            549     579
AMAMS3= 000000            549     582
AMAMS4= 000000            549     585
AMSGRD= 000000            549     557
AMSGLG= 000000            549     558
AMSGTY= 000000            549     551
AMTYP1= 000000            549     572
AMTYP2= 000000            549     580
AMTYP3= 000000            549     583
AMTYP4= 000000            549     586
APASS = 000000            549     554
APRIOR= 000000            549
APTCSU= 000040           1274    1388#
APTENV= 000001           1267    1344    1386#
APTER1= 000001            474#    880
APTER2= 000002            475#    894
APTER3= 000004            476#    905
APTER4= 000010            477#    911
APTSIZ= 000200            640    1385#
APTSP0= 000100           1269    1346    1387#
ARG1    012030           1119    1936*   1941*   2000*   2046*   2094#
ARG2    012032           1118    1921*   2047*   2095#
ASWREG= 000000            549     562
ATESTN= 000000            549     553
AUNIT = 000000            549     556
```

```
AUSWR = 000000   549   563
AVECT1= 000000   549   588
AVECT2= 000000   549   589
BELL    006506  1100  1705#
BIT0  = 000001   456#
BIT00 = 000001   446#   456
BIT01 = 000002   445#   455
BIT02 = 000004   444#   454
BIT03 = 000010   443#   453
BIT04 = 000020   442#   452
BIT05 = 000040   441#   451
BIT06 = 000100   440#   450
BIT07 = 000200   439#   449
BIT08 = 000400   438#   448
BIT09 = 001000   437#   447
BIT1  = 000002   455#
BIT10 = 002000   436#  1097
BIT11 = 004000   435#
BIT12 = 010000   434#
BIT13 = 020000   433#  1101
BIT14 = 040000   432#
BIT15 = 100000   431#  1159
BIT2  = 000004   454#
BIT3  = 000010   453#
BIT4  = 000020   452#
BIT5  = 000040   451#
BIT6  = 000100   450#
BIT7  = 000200   449#
BIT8  = 000400   448#
BIT9  = 001000   447#
BPTVEC= 000014   463#
BRHEAD  006533  1032  1711#
BUF1    007434   665   871   886   942  1022  1798#
BUF2    007514   902   908   998*  999*  1005*  1006*  1062  1066  1823#
BUILD   012034  2058*  2065  2071*  2096#
CALSUM  002526   784   797   820#
CHECKS  002310   705   778#
CKSWR = 104406  1690#
CLEAR   010630   671  1845#
CONER1  012014  1853*  1922  1924  2088#
CONER2  012016  1862*  1933  1937*  1938  2089#
CONFIN  012024   692*  1850  1860  1964  2092#
CONMES  012050  1889  2071  2102#
CONONE= 000400   482#  1926
CONTWO= 001000   483#  1935
CON1    010646   954  1850#
CON2    010674   974  1857#
CR    = 000015   371#  1027  1034  1039  1065  1073  1113  1148  1189  1313  1323  1905  2047
                2053  2064
CRCERR= 000020   478#   788   801
CRCLOP  002544   825#   834
CRCMSG  007161  1171  1762#
CRLF  = 000200   372#   664  1284  1323
DAFLAG  003326   949*   958   976*   982#
DDISP = 177570   378#   517   628
DEVCOD  003114   858   870   942#
```

```
DEVFIN  012026            696*   952   2093#
DISPLA  001106            517#   628*   636*
DISPRE  000174            494#   636
DRHEAD  006512           1025   1708#
DSWR   = 177570            377#   516    627
DUPERR= 002000            484#  1999
EHEADT  004334           1105   1167#
EMTVEC= 000030            466#
END     011002           1871   1877   1879#
ENDSEQ  012010           1985   1996   2017   2086#
ERMES1  012076           1175   2106#
ERMES2  012155           1176   2114#
ERMES3  012235           1177   2123#
ERMES4  012311           2051   2131#
ERRCNT  001120            522#  1096*
ERRHAN  003772            760    789    802    864    881    895    906    912   1094#  1942  2004  2076
ERRMSG  010230           1095   1154   1158   1828#
ERRVEC= 000004            459#   625    626*   637*   672    673*   685*
ER1MSG  006771           1167   1739#
ER2MSG  006735           1168   1734#
ER3MSG  007027           1169   1745#
ER4MSG  007105           1170   1754#
EXCADD  002620            782*   791*   804*   835    844#
FILBUF  004362           1026   1033   1044   1051   1064   1069   1112   1120   1142  1146  1185#  1890  1904
                         2052   2063   2066
FILTAB  011256           1934   1957#
FINISH  012036           1038*  1847   1883   2097#
FIRSTA  002616            783*   790*   803*   807    821    843#
FIRSTB  001114            520#   687    950
FLAG    012022           1878*  2030   2091#
GWS    = ******  U        493    663   1683   1684   1685   1686   1688   1690  1691  1692  1693  1694
GTSWR  = 104405            658   1688#
HOLCK1  010740            997   1870#
HOLCK2  010752           1004   1873#
HT     = 000011            369#  1045   1052   1056   1070   1143   1147   1185  1187  1282  1323  1891  1907
                         1921   2067
IOTVEC= 000020            464#
LASTAD  002622            781*   792*   805*   838    845#
LF     = 000012            370#  1040   1074   1149   1190   1317   1323
LOOP    002536            823#   839
MESSAG  001112            519#   868*
MESTAB  012040           1036   1925   2049   2098#
MES1    007530            860   1826#  1936   1938*  1939*  2000   2002*  2003*  2046  2050
MES2    010130            867   1827#  2058   2060*  2061*
NODATA  002306            755    770#
NOROME= 000100            480#   759
NOROMM  007263           1173   1774#
NOROMS  002226            703    754#
OCADD   004530           1043   1050   1068   1234#
OCASC   004424           1141   1145   1204#  1236
OCTBUF  007520            667   1204   1209   1216   1218   1825#
OUTTAB  011622           2040   2046#
PFERR  = 000040            479#   862   1008   1114
PFHEAD  006670           1063   1727#
PFMSG   007205           1172   1766#
PIRQ   = 177772            376#
```

```
PIRQVE= 000240  470#
PPFVAR  003330  861  900  994#
PROMP   002624  708  856#
PR0   = 000000  393#
PR1   = 000040  394#
PR2   = 000100  395#
PR3   = 000140  396#
PR4   = 000200  397#
PR5   = 000240  398#
PR6   = 000300  399#
PR7   = 000340  400#
PS    = 177776  373#  374
PSW   = 177776  374#
PUTMES  003440  859  866  1017#
PWRVEC= 000024  465#
RDCHR = 104407  1504  1691#
RDLIN = 104410  1692#
RESREG= 104412  1163  1694#
RESVEC= 000010  460#
RETURN  012020  996*  1003*  1037*  1879  1893  2090#
ROMCNT  002524  778*  779  794*  795  811#  1127
ROMERR  001110  518#  759*  788*  801*  862  880*  894*  905*  911*  1008*  1103  1104  1114
                1116  1123  1125  1162*  1926*  1935*  1999*  2014*  2039
ROMFIN  001116  521#  669*  681*  693*  695*  701  779  795  944  1870  1873  1899  1962
                2022
ROMNM   011062  1887  1888  1895#
ROMNUM  006474  1128  1700#
ROMTYP  011010  1041  1881#
ROM1    012336  2098  2135#
ROM2    012352  2099  2137#
ROM3    012366  2100  2139#
ROM4    012402  2101  2141#
RSTART  002066  499  701#  741
SAVREG= 104411  1094  1693#
SEQBUF  011774  1845  1957  1982  2032  2034  2036  2037  2048  2080#
SEQERR= 000200  481#  2014  2039
SEQMSG  007341  1174  1783#
SEQTST  011130  709  1921#
STACK = 001100  364#  617
START   001400  497  615#
STKLMT= 177774  375#
SWR     001104  516#  627*  629  635*  642*  656  1097  1101  1159  1401  1438*
SWREG   000176  495#  635  656  1401  1414
SW0   = 000001  428#
SW00  = 000001  418#  428
SW01  = 000002  417#  427
SW02  = 000004  416#  426
SW03  = 000010  415#  425
SW04  = 000020  414#  424
SW05  = 000040  413#  423
SW06  = 000100  412#  422
SW07  = 000200  411#  421
SW08  = 000400  410#  420
SW09  = 001000  409#  419
SW1   = 000002  427#
SW10  = 002000  408#
```

```
SW11   = 004000            407#
SW12   = 010000            406#
SW13   = 020000            405#
SW14   = 040000            404#
SW15   = 100000            403#
SW2    = 000004            426#
SW3    = 000010            425#
SW4    = 000020            424#
SW5    = 000040            423#
SW6    = 000100            422#
SW7    = 000200            421#
SW8    = 000400            420#
SW9    = 001000            419#
TBITVE= 000014             461#
TEMP    004526            1210#  1212#  1220#  1222#  1224#  1227#
TESTAD  003324             687#   688    690    697#   699    946#   947    948    950#   955#   960    961    962#
                           964    965#   966    977    981#
TIMES   003770             670#  1018   1021#  1081#
TKVEC  = 000060            468#
TPVEC  = 000064            469#
TRAPVE= 000034             467#   619#   620#
TRTVEC= 000014             462#
TYPE   = 104401            650    730    731   1077   1099   1287   1412   1413   1416   1429   1440   1459   1508
                          1511   1515   1590   1683#
TYPOC  = 104402           1415   1684#
TYPON  = 104404           1686#
TYPOS  = 104403           1685#
$APTHD  001122             533    539#
$ASTAT= ******  U         1366   1381
$ATYC   005066            1337   1339#
$ATY1   005042            1153   1335#
$ATY3   005050            1157   1272   1336#
$ATY4   005060            1338#
$AUTOB  001100             513#   660#  1409   1530
$BASE   001212             590#   901
$CDW1   001216             592#   907
$CDW2   001220             593#
$CHARC  005022            1289#  1299#  1306   1315#  1320#
$CKSUM  005314            1401#  1690
$CMTAG= ******  U          617    621
$CNTLG  006041            1412   1525#
$CNTLU  006034            1429   1524#
$CPUOP  001164             564#
$CRLF   005037            1288   1330#  1440   1524
$DDW0   001222             594#   872    885
$DDW1   001224             595#
$DDW10  001246             604#
$DDW11  001250             605#
$DDW12  001252             606#
$DDW13  001254             607#
$DDW14  001256             608#
$DDW15  001260             609#
$DDW2   001226             596#
$DDW3   001230             597#
$DDW4   001232             598#
$DDW5   001234             599#
```

```
$DONE   001236            600#
$DONT   001240            601#
$DONTB  001242            602#
$DONTS  001244            603#
$DEVCT  001146            555#
$DEVM   001214            591#
$DORGN  002204            726     733     739#
$ENDRD  002174            507     648     735#
$ENDCT  002150            622     728#
$ENDMG  002213            730     743#
$ENULL  002210            731     742#
$ENV    001156            560#    654     1151    1267    1344    1368
$ENVM   001157            561#    640     856     1269    1274    1346
$EOP    002124            704     707     720#
$EOPCT  002142            622*    725#    729
$ETABL  001156            559#
$ETEND  001262            545     612#
$FATAL  001140            552#    1372*
$FFLG   005306            1335*   1338*   1366    1375*   1383#
$FILLC  005034            1292    1327#
$FILLS  005033            1326#
$GET42  002164            732#
$GTSWR  005364            1413#   1688
$HD   = 000003             352     353
$HIBTS  001122            540#
$INTAG  001101            514#    1441    1530
$LF     005040            1331#   1515    1524
$LFLG   005305            1376*   1382#
$MADR1  001170            577#
$MADR2  001174            581#
$MADR3  001200            584#
$MADR4  001204            587#
$MAIL   001136            541     545     550#    639     654     1267
$MAMS1  001166            571#
$MAMS2  001172            579#
$MAMS3  001176            582#
$MAMS4  001202            585#
$MBADR  001124            541#
$MFLG   005304            1336*   1342    1377*   1381#
$MNEW   006057            1416    1528#
$MSGRD  001152            557#    1352*   1355
$MSGLG  001154            558#    1357*
$MSGTY  001136            551#    1350    1358*   1370    1374*
$MSWR   006046            1413    1526#
$MTYP1  001167            572#
$MTYP2  001173            580#
$MTYP3  001177            583#
$MTYP4  001203            586#
$NULL   005032            1294    1325#
$OCNT   006312            1562*   1591*   1604#
$OMODE  006314            1557*   1561*   1566    1569*   1580*   1606#
$PASS   001144             554#    621*    639*    706     722*    723*    742
$PASTM  001130            543#
$QUES   005036            1329#   1459    1508    1524
$RDCHR  005576            1472#   1691
$RDDEC= ******  U         1693
```

```
$RD.IN  005716          1500#   1692
$RDXCT= ****** U        1693
$RDXZ = 000010          1493#
$RESRE  006354          1640#   1694
$RTIRD  002206           741#
$R2A  = ****** U        1695
$SAVRE  006316          1624#   1693
$SETUP= 000124           498#    618     619     621     623     648     651     722    1396    1530
$STUP = 177777           498#
$SVPC = 000210           505#    510
$SWR  = 160000           352     353#    358     359     360     361     623     715     722     734     740     742
$SWREG  001160           562#    642
$TESTN  001142           553#
$TKB    005312          1393#   1405    1422    1476    1482
$TKS    005310          1392#   1403    1419    1443*   1474    1480
$TN   = 000001           352#
$TPB    005030          1312*   1324#
$TPFLG  005035          1261    1328#
$TPS    005026          1310    1323#
$TRAP   006412           619    1660#
$TRAP2  006434          1671#   1682
$TRP  = 000013          1675#   1684#   1685#   1686#   1687#   1688    1689#   1690    1691#   1692#   1693#   1694#   1695#
$TRPRD  006446          1665    1682#
$TSTN   001126           542#
$TTYIN  006024          1501    1502    1519    1523#
$TYPBN= ****** U        1687
$TYPDS= ****** U        1687
$TYPE   004544          1261#   1363    1675    1683
$TYPEC  004756          1291    1298    1305    1310#   1311    1445
$TYPEX  005024          1316    1318    1321#
$TYPOC  006114          1560#   1684
$TYPON  006130          1559    1562#   1686
$TYPOS  006070          1555#   1685
$UNIT   001150           556#
$UNITM  001132           544#
$USWR   001162           563#
$VECT1  001206           588#
$VECT2  001210           589#
$$GET4= 000000           734#
$OFILL  006313          1556*   1560*   1570    1605#
      = 012416           486#    489#    493#    498#    505     506#    508#    510#    512#    529     530#    532#    534#
                         614#    742     745    1323    1324    1325    1326    1327    1328    1329    1330    1331    1332
                        1384#   1392    1393    1523#   1524    1530    1825#   1826#   1827#   1828#
. $ASTA= ****** U        1336    1339
 $X   = 001122           529#    534
```

```
COMMEN      1#    471#
ENDCOM      1#    471#
ERROR     365#
ESCAPE      1#    471#
GETPRI      1#    471#
GETSWR      1#    471#    651#
MULT        1#    471#
NEWTST      1#    471#
POP         1#    342#    471#   1378    1379    1645    1652
PUSH        1#    342#    471#   1339    1341    1362    1625    1652
REPORT      1#    342#    471#
SCOPE     366#
SETPRI      1#    471#
SETTRA   1675#   1684     1685    1686    1688    1690    1691    1692    1693    1694
SETUP       1#    342#    471#    615
SKIP        1#    471#
SLASH       1#    471#
SOFTSW    342#
SPACE     471#
STARS       1#    471#    473     503     526     528     535     548     712    1246    1334    1391    1396    1464    1493
         1532   1609    1654
SWRSU       1#    471#    623#
TRMTRP   1675#
TYPBIN      1#    471#
TYPDEC      1#    471#
TYPNAM      1#    342#    471#    644
TYPNUM      1#    471#
TYPOCS      1#    471#
TYPOCT      1#    471#   1414
TYPTXT      1#    471#
$$ESCA      1#    471#
$$NEWT      1#    471#
$$SET    1675#   1684     1685    1686    1688    1690    1691    1692    1693    1694
$$SETM    639#
$$SKIP      1#    471#
.EQUAT      1#    342#     361
.HEADE      1#    342#
.KT11       1#
.SETUP      1#    342#     498
.STARS    342#
.SWRHI      1#    342#     354
.SWRLO    361#
.SYCT1      1#    342#     501
.SYPTB      1#    342#     546
.SYPTH      1#    342#     524
.SYPTY      1#    342#    1332
.SYSTA      1#    342#
.SYCTC      1#    342#     487
.SYMTA      1#    342#
.SYB2D      1#
.SYB2O      1#
.SYIV       1#
.SYOP       1#    342#     710
.SYRRO      1#
.SYRRT      1#
.SYULT      1#
```

```
 $POWE       1#
 $RAND       1#
 $RDDE       1#
 $RDOC       1#
 $READ       1#     342#    1389
 $R2AZ       1#
 $SAVE       1#     342#    1607
 $SB2D       1#
 $SB2O       1#
 $SCOP       1#
 $SIZE       1#
 $SUPR       1#
 $TRAP       1#     342#    1652
 $TYPB       1#
 $TYPD       1#
 $TYPE       1#     342#    1244
 $TYPO       1#     342#    1530
 $4OCA       1#
 1170        1#

  ABS   012416      000

 ERRORS DETECTED   0

 CZM9BB.BIN,CZM9BB.LST/CRF/SOL/NL TOC=DSKZ CZM9BB SML,DSKM CZM9BB P11
 RUN-TIME  11 13  8 SECONDS
 RUN-TIME RATIO  144/25=5 6
 CORE USED   32K  (63 PAGES)
```