M9312

M9312/1144 UBI BOOT

CZM9BD0

AH-E058D-MC

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

```
.HE M
```

IDENTIFICATION

PRODUCT CODE: AC-E057D-MC

PRODUCT NAME: CZM9BD0 M9312/1144 JBI BOOT

DATE: OCT 1979

MAINTAINER: DIAGNOSTIC ENGINEERING

HISTORY SECTION

PBRDA0 WAS RELEASED MARCH, 1978.
PBRDB0 WAS RELEASED JUNE, 1978.
PBRD440 WAS RELEASED JANUARY, 1979
[illegible]ZM9BD0 WAS RELEASED OCTOBER 1979

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. ILLECAL 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.

REVISION C WAS CREATED TO PROVIDE THE FOLLOWING ENHANCEMENTS:

1. TO UPDATE THE DIAGNOSTIC BASED ON FAULT INSERTION OF THE M9312 MODULE.

2. TO ADD A ' NO ROMS FOUND ' MESSAGE.

3. TO SIZE FOR A FALSE ERROR CAUSED BY A REAL FAILURE TO FIND THE POWER FAIL VECTOR WHEN ONLY THE CPU ROM IS PRESENT. THIS SHOULD NOT BE TREATED AS A FATAL ERROR (WHICH IT WAS IN REV. B) PREVIOULY , APT WOULD NOT RUN IN PRINTING MODE DISABLED DUE TO THIS ERROR.

COMMENT FIELDS BEGINNING WITH THE CHARATERS ";-" IDENTIFY ALL LINES ADDED OR MODIFIED BY REV. C.

REVISION D WAS CREATED TO ACCOMODATE TESTING OF 11/44 UBI MODULE.
THE UBI MODULE HAS LOGIC WHICH IS FUNCTIONALLY EQUIVALENT TO THE M9312.

1. SINCE THE DIAGNOSTIC CALLS OUT E NUMBERS REFERENCING THE 4 BOOTSTRAP & 1 BOOT DIAGNOSTIC ROM IC'S, USE MFPT INSTRUCTION TO DETERMINE IF CPU IS 11/44. IF IT IS THEN USE SET OF E NUMBERS CONSISTENT WITH UBI MODULE.

COMMENT FIELDS BEGINNING WITH THE CHARACTERS ";**" IDENTIFY ALL LINES ADDED OR MODIFIED BY REV. D.

.REM

1.0 ABSTRACT

THIS PROGRAM VERIFIES THE ROM INFORMATION FOR THE M9312 BOOTSTRAP TERMINATOR OR 11/44 UBI 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

NOTE: IF 11/44 (P'), M9312 BOOTSTRAP TERMINATOR IS NOT NEEDED

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:	

NOTE: THE ORDER FOR LOADING $DDW0 IS IMPORTANT. THE FIRST

VALUE SHOULD BE FOR THE DIAGNOSTIC / CPU ROM, FOLLOWED
BY THE APPROPRIATE OCTAL VALUES FOR THE BOOT ROMS PRESENT.
SOME OCTAL VALUES USED ARE:

OCTAL DATA	MNEMONIC	PIN LABELING
040460	A0	248F1
04160	B0	233F1
041460	C0	
042113	DK	756A9
042114	DL	751A9
042130	DX	753A9
046524	MT	757A9

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 DEFEPMENT.

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 B0:

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 W'LL 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
--------	--------

7.0 NOTE: DIAGNOSTIC COVERAGE

THIS DIAGNOSTIC DOESNOT PERFORM ANY EXAMINATION OF THE BOOT CIRCUITRY ON THE M9312. IF YOU WISH TO CHECK THIS CIRCUITRY YOU MUST FIRST, DETERMINE WHICH DEVICE YOU WILL BOOT AND SECOND, YOU MUST SET THE SWITCHES ON THE M9312 TO THE APPROPRIATE SETTINGS. YOU MAY NOW ATTEMPT TO BOOT

THE M9312 . IF THE DEVICE DOES BOOT THIS SHOULD BE
EVIDENT FROM THE CONSOLE TERMINAL, IF PRESENT, OR FROM THE RUN
LIGHT ON THE FRONT PANEL, WHICH SHOULD BE LIT.

```
                                .ENABLE ABS
                                .LIST   ME
                                .NLIST  MC,MD,CND
                                .MCALL  .$ASTAT,.$SAVE,.$CATCH,.$CMTAG,.SETUP,.$EOP,.$TYPE
                                .MCALL  STARS,.EQUATE,.$APTHDR,.$APTBLS,.$APTYPE
                                .MCALL  .$ACT11,.HEADER,.SWRHI,REPORT,TYPNAM,PUSH,POP,.$TRAP
                                .MCALL  .$READ,.$TYPOCT
                                .MCALL  .EQUIV
                                .TITLE  CZM9BD0 M9312/1144 UBI BOOT
                                ;*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 ERROR 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 *** 1 00 ***
001100                          STACK=  1100
104000                                  ERROR=EMT
000004                                  SCOPE=IOT

                                ;*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
177776                                  PSW=PS
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 DEFINITIONS
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
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7

                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                          SW9=SW09
000400                          SW8=SW08
000200                          SW7=SW07
000100                          SW6=SW06
000040                          SW5=SW05
000020                          SW4=SW04
000010                          SW3=SW03
000004                          SW2=SW02
000002                          SW1=SW01
000001                          SW0=SW00

                        ;*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
                001000                  BIT9=BIT09
                000400                  BIT8=BIT08
                000200                  BIT7=BIT07
                000100                  BIT6=BIT06
                000040                  BIT5=BIT05
                000020                  BIT4=BIT04
                000010                  BIT3=BIT03
                000004                  BIT2=BIT02
                000002                  BIT1=BIT01
                000001                  BIT0=BIT00

                                ;*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
    433                         ;;*************************************************************
    434                         ;*               PROGRAM EQUATES
    435                         ;;*************************************************************
    436         000001                  APTER1 =  1
    437         000002                  APTER2 =  2
    438         000004                  APTER3 =  4
    439         000010                  APTER4 = 10
    440         000020                  CRCERR = 20
    441         000040                  PFERR  = 40
    442         000100                  NOROME =100
    443         000200                  SEQERR =200
    444         000400                  CONONE =400             ;*
    445         001000                  CONTWO=1000             ;*
    446         002000                  DUPERR=2000             ;*
    447
    448         000000                  .=0
    449                         .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
451          000204                            .=204
452  000204  000167  001730                    JMP     RSTART
453
454                                  .SBTTL  ACT11 HOOKS

                                     ;;*****************************************************************
                                     ;HOOKS REQUIRED BY ACT11
                 000210                      $SVPC=.                 ;SAVE PC
                 000046                      .=46
     000046      002246                      $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
455
456              001100                      .=1100
457  001100         000     $AUTOB: .BYTE   0
458  001101         000     $INTAG: .BYTE   0
459  001102      000000             .WORD   0
460  001104      177570     SWR:    .WORD   DSWR
461  001106      177570     DISPLAY: .WORD  DDISP
462  001110      000000     ROMERR: 0
463  001112      000000     MESSAG: 0
464  001114      173000     FIRSTB: 173000
465  001116      000000     ROMFIN: 0
466  001120      000000     ERRCNT: 0
467
468                                  .SBTTL  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)
469                                  .SBTTL  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
001160  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001162  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001164  000000                  $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 10=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:

470          000007                         MFPT=000007
471          001400                         .=1400
472 001400                          START:
                                    .SBTTL  INITIALIZE THE COMMON TAGS
     001400  012706  001100                 MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                    ;;INITIALIZE A FEW VECTORS
     001404  012737  006564  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  000572  000562         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     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
     001542                          67$:
473                                 .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  002246  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
     001614  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$:  .ASCIZ  <CRLF>*CZM98DO M9312/1144 UBI BOOT*<CRLF>
     001664                          68$:
474 001664  013746  000010                 MOV     @#10,-(SP)              ;** SAVE VECTOR ON STACK
475 001670  012737  001724  000010         MOV     #10$,@#10               ;** SET ERROR VECTOR
```

```
                                        MFPT                            ;** WHAT CPU IS IT
                                        CMPB    #1,R0                   ;** 11/44?
                                        BNE     12$                     ;** NO USE M9312 ROM NUMBERS
                                        MOV     #MESS44,@#MESSPT        ;** YES USE UBI ROM NUMBERS
                                        MOV     #AROM,@#ROMPTR          ;** UBI ROM NUMBERS
                                        BR      12$                     ;** BR IF NO TRAP
                                10$:    MOV     #12$,(SP)               ;SET UP RETURN VECTOR
                                        RTI                             ;RETURN
                                12$:    MOV     (SP)+,@#10              ;RESTORE ERROR VECTOR
                                        MOV     #BUF1,          R0              ;CLR SIZING BUFFERS
                                8$:     CLR     (R0)+
                                        CMP     R0,             #OCTBUF         ;HAVE WE CLEARED THE WHOLE
                                        BLT     8$                              ;BUFFER,NO,GO BACK
                                        CLR     @#ROMFIN                        ;INITIALIZE ROM FOUND INDICATORS
                                        CLR     @#TIMES                         ;INITIALIZE ENTRY COUNTER FOR PUTMES SUB.
                                        JSR     PC,     CLEAR                   ;*CLEAR SOME MORE LOCATIONS
                                        MOV     @#ERRVEC,       -(R6)           ;SAVE CONTENTS OF LOCATION 4
                                        MOV     #2$,            @#ERRVEC        ;*SET UP FOR POSSIBLE TRAP
                                        CMPB    #-1,            @#165000        ;IF CONTENTS = -1, OR TRAP
                                        BEQ     3$                              ;THEN CPU ROM NOT PLUGGED IN
                                        MOV     #165000,        R0              ;GET FIRST ADDRESS OF ROM SPACE
                                1$:     TST     (R0)+                           ;IF THIS INSTRUCTION TRAPS CPU ROM
                                                                                ;NOT PLUGGED IN
                                        CMP     R0,             #166000         ;HAVE WE CHECKED ALL ADDRESSES
                                        BNE     1$                              ;IF NO THEN CONTINUE LOOPING
                                        MOV     #1,             @#ROMFIN        ;IF NO TRAPS OR LOW BYTE OF FIRST
                                                                                ;ADDRESS NOT EQUAL -1 ASSUME ROM PRESENT
                                        BR      3$                              ;NO TRAP JUST RESTORE ERRVEC
                                2$:     ADD     #4,             R6              ;*IF TRAP CLEAN OFF PC, PSW
                                3$:     MOV     (R6)+,          @#ERRVEC        ;RESTORE ERRVEC
                                        MOV     #2,             R0              ;INITIALIZE ROM FOUND INDICATOR
                                        MOV     @#FIRSTB,       @#TESTAD        ;GET FIRST BOOT
                                5$:     BITB    #200,           @TESTAD         ;IF LOW BYTE HAS BIT 7 SET
                                        BEQ     7$                              ;DO NOT SET ROM FOUND INDICATOR
                                        CMP     #-2,            @TESTAD         ;UNLESS, CONTENTS OF ADDRESS
                                        BNE     4$                              ;EQUAL TO -2.(CONTINUATION ROM)
                                        BIS     R0,     CONFIN                  ;*SET CONTINUATION ROM FOUND INDICATOR
                                        BIS     R0,     ROMFIN                  ;*SET ROM FOUND INDICATOR
                                        BR      4$                              ;*CONTINUE
                                7$:     BIS     R0,             @#ROMFIN        ;SET ROM FOUND INDICATOR
                                        BIS     R0,     DEVFIN                  ;*SET DEVICE ROM FOUND INDICATOR
                                4$:     ADD     #200,           @#TESTAD        ;UPDATE TEST ADDRESS TO NEXT ROM
                                        ROL     R0                              ;UPDATE ROM FOUND INDICATOR
                                        CMP     #174000,        @#TESTAD        ;HAVE CHECKED ALL THE ROMS
                                        BNE     5$                              ;IF NO CONTINUE CHECKING
                                RSTART: TST     @#ROMFIN                        ;ARE THERE ANY ROMS AVAILABLE
                                        BNE     6$                              ;IF YES GO TO ROM TEST
                                        JSR     PC,             NOROMS          ;IF NO GO TEST NO ROMS
                                        BR      $EOP                            ;GO TO END OF PASS
                                6$:     JSR     PC,             CHECKS          ;GO CALCULATE CHECKSUMS
                                        TST     @#$PASS                 ;ARE WE ON 1ST PASS
                                        BNE     $EOP                            ;IF NO SKIP PROCESS OF PARAMETERS
                                        JSR     PC,             PROMP           ;GO PROCESS ROM PARAMETERS
                                        JSR     PC,             SEQTST          ;GO CHECK SEQUENCE OF DEVICE CODES
                                .SBTTL  END OF PASS ROUTINE

                                ;;*****************************************************************
```

```
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*TYPE 'END PASS'
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO RSTART
                                ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE 'END OF PASS' LOCATION
                                ;*$ENDMG CAN BE CHANGED TO 7.

002176                          $EOP:
002176  000240                         NOP
002200  005267  176740                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
002204  042767  100000  176732          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
002212  005327                          DEC     (PC)+           ;;LOOP?
002214  000001                  $EOPCT: .WORD   1
002216  003017                          BGT     $DOAGN          ;;YES
002220  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
002222  000001                  $ENDCT: .WORD   1
002224  002214                          $EOPCT
002226  104401  002265                  TYPE    ,$ENDMG         ;;TYPE 'END PASS'
002232  104401  002262                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
002236  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
002242  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
002244  000005                          RESET                   ;;CLEAR THE WORLD
002246  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
002250  000240                          NOP                     ;;SAVE ROOM
002252  000240                          NOP                     ;;FOR
002254  000240                          NOP                     ;;ACT11
002256                          $DOAGN:
002256  000137                          JMP     @(PC)+          ;;RETURN
002260  002140                  $RTNAD: .WORD   RSTART
002262     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
002265     015     012     105  $ENDMG: .ASCIZ  <15><12>/END PASS/
002270     116     104     040
002273     120     101     123
002276     123     000


                                ;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.

002300  104401  006700          NOROMS:  TYPE   ,MNORM                          ;- TYPE MESSAGE THAT NOROMS FOUND
002304  012703  172000                   MOV #172000,           R3              ; GET FIRST ADDRESS TO BE READ
002310  013704  002364          1$:      MOV    @#NODATA,       R4              ;GET ADDRESS OF EXPECTED VALUE
002314  121304                           CMPB   (R3),           R4              ;+DOES RECIEVED VALUE EQUAL EXPECTED
002316  001406                           BEQ    2$                              ;IF YES CONTINUE TESTING
002320  011304                           MOV    (R3),   R4                      ;+GET BAD VALUE
002322  052737  000100  001110          BIS    #NOROME,        @#ROMERR        ;IF NO SET ERROR INDICATER
002330  004767  001506                   JSR    PC,             ERRHAN          ;REPORT ERROR
002334  062703  000002          2$:      ADD    #2,     R3                      ;+GET TO NEXT EVEN ADDRESS
002340  022703  173024                   CMP    #173024,        R3              ;+AT A VECTOR ADDRESS?
002344  001773                           BEQ    2$                              ;+BRANCH IF YES
002346  022703  173224                   CMP    #173224,        R3              ;+AT A VECTOR ADDRESS?
002352  001770                           BEQ    2$                              ;+BRANCH IF YES
002354  022703  174000                   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
        CMP     R3,     R4              ;COMPARE CHECKSUMS
        BEQ     1$                      ;IF CHECKSUMS COMPARE CONTINUE
        BIS     #CRCERR, @#ROMERR       ;IF ERROR SET INDICATER
        JSR     PC,     ERRHAN          ;REPORT ERROR
1$:     MOV     #173000, @#FIRSTA       ;SET UP FIRST ADDRESS TO BE SUMMED
        MOV     #173024, @#EXCADD       ;SET UP EXCEPTION ADDRESS
        MOV     #173175, @#LASTAD       ;SET UP LAST ADDRESS TO BE SUMMED
        MOV     #173176, R2             ;GET ADDRESS OF GOOD DATA
2$:     ROL     @#ROMCNT                ;UPDATE ROM COUNTER
        BIT     @#ROMCNT, @#ROMFIN      ;IS ROM PRESENT
        BEQ     3$                      ;IF NO, GO UPDATE TO NEXT ROM
        JSR     PC,     CALSUM          ;IF YES GO CALCULATE CHECKSUM
        MOV     (R2),   R3              ;GET EXPECTED CHECKSUM
        CMP     R3,     R4              ;COMPARE CHECKSUMS
        BEQ     3$                      ;IF CHECKSUMS EQUAL CONTINUE
        BIS     #CRCERR, @#ROMERR       ;IF ERROR SET INDICATER
        JSR     PC,     ERRHAN          ;REPORT ERROR
3$:     ADD     #200,   @#FIRSTA        ;UPDATE FIRST ADDRESS
        ADD     #200,   @#EXCADD        ;UPDATE EXCEPTION ADDRESS
        ADD     #200,   @#LASTAD        ;UPDATE LAST ADDRESS
        ADD     #200,   R2              ;UPDATE ADDRESS OF GOOD DATA
        CMP     #174000, @#FIRSTA       ;HAVE ALL ROMS BEEN CHECKED
        BNE     2$                      ;IF NO GO CHECKSUM NEXT ONE
        RTS     PC                      ;IF YES RETURN

ROMCNT: 0


                ;CALCULATE CHECKSUMS MODULE
                ;THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF THE CONTENTS
                ;OF EVERY LOCATION EXCEPT THE EXCEPTION ADDRESS AND LAST ADDRESS
                ;OF EVERY ROM


CALSUM: MOV     R2,     -(SP)           ;SAVE R2
        MOV     @#FIRSTA, R0            ;GET STARTING ADDRESS
        CLR     R4                      ;INITIALIZE CRC WORD
LOOP:   MOV     (R0)+,  R5              ;GET A BYTE
```

```
611 002616  01c702  000020                  MOV     #16.,           R2              ;SET BYTE COUNT
612 002622  000241                  CRCLOP: CLC                                     ;THE NEXT NINE LINES
613 002624  006004                          ROR     R4                              ;DO THE MATH CALCULATIONS
614 002626  006005                          ROR     R5
615 002630  102006                          BvC     1$
616 002632  012701  120001                  MOV     #120001,        R1
617 002636  040401                          BIC     R4,             R1
618 002640  042704  120001                  BIC     #120001,        R4
619 002644  050104                          BIS     R1,             R4
620 002646  005302                  1$:     DEC     R2
621 002650  003364                          BGT     CRCLOP
622 002652  020037  002676                  CMP     R0,             @#EXCADD        ;IS NEXT ADDRESS AN EXCEPTION ADDRESS
623 002656  001001                          BNE     2$                              ;IF NO TEST IT
624 002660  005720                          TST     (R0)+                           ;IF YES SKIP ADDRESS
625 002662  020037  002700          2$:     CMP     R0,             @#LASTAD        ;HAVE ALL LOCATIONS BEEN SUMMED
626 002666  101752                          BLOS    LOOP                            ;IF NO CONTINUE
627 002670  012602                          MOV     (SP)+,          R2              ;RESTORE R2
628 002672  000207                          RTS     PC                              ;IF YES RETURN
629
630 002674  000000                  FIRSTA: 0
631 002676  000000                  EXCADD: 0
632 002700  000000                  LASTAD: 0
633
634
635
636                                 ;PROCESS ROM PARAMETERS MODULE
637                                 :THIS ROUTINE DETERMINES IF SIZING IS TO BE DONE.  IF IT IS THE
638                                 ;MODULE WILL GET THE PARAMETERS FROM THE DEVCOD AND PPFVAR
639                                 ;MODULES AND ASSEMBLE MESSAGES TO BE OUTPUT.  IF SIZING IS
640                                 ;NOT TO BE DONE THIS MODULE WILL TAKE THE PARAMETERS RECEIVED
641                                 ;AND COMPARE THEM TO THE VALUES SUPPLIED IN THE ETABLE.
642
643 002702  105737  001157          PROMP:  TSTB    @#$ENVM                 ;IF BIT IS CLEAR SIZE
644 002706  100424                          BMI     1$                              ;IF BIT IS SET DON'T SIZE.
645 002710  004767  000250                  JSR     PC,             DEVCOD          ;CALL DEVICE CODE MODULE
646 002714  004767  000570                  JSR     PC,             PUTMES          ;FORMAT PARAMETER MESSAGE
647 002720  010016                          MES1
648 002722  004767  000452                  JSR     PC,             PPFVAR          ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
649 002726  032737  000040  001110          BIT     #PFERR,         @#ROMERR        ;WAS THERE AN ERR
650 002734  001403                          BEQ     10$                             ;IF NO GO FORMAT MESSAGE
651 002736  004767  001100                  JSR     PC,             ERRHAN          ;IF YES GO REPORT MESSAGE
652 002742  000507                          BR      9$
653 002744  004767  000540          10$:    JSR     PC,             PUTMES          ;FORMAT PSEUDO POWER-FAIL VECTOR
654 002750  010416                          MES2                                    ;ADDRESS MESSAGE
655 002752  005237  001112                  INC     @#MESSAG                        ;SET MESSAGE INDICATOR
656 002756  000501                          BR      9$                              ;GO TO EXIT
657 002760  004767  000200          '$:     JSR     PC,             DEVCOD          ;CALL GET DEVICE CODE MODULE
658 002764  012703  007722                  MOV     #BUF1,          R3              ;GET BOARD DEVICE CODES
659 002770  012704  001222          2$:     MOV     #$DDW0,         R4              ;GET ETABLE DEVICE CODE PARAMETERS
660 002774  062703  000002                  ADD     #2,             R3              ;GET TO BOARD DEVICE CODE
661 003000  012402                  3$:     MOV     (R4)+,          R2              ;GET ETABLE DEVICE CODE
662 003002  020213                          CMP     R2,             (R3)            ;DO THE TWO DEVICE CODES COMPARE
663 003004  001407                          BEQ     4$                              ;IF YES GET NEXT BOARD DEVICE CODE
664 003006  005714                          TST     (R4)                            ;IF NO HAVE WE CHECKED THE WHOLE ETABLE
665 003010  001373                          BNE     3$                              ;IF WE HAVEN'T CHECK NEXT ETABLE ENTRY
666 003012  052737  000001  001110          BIS     #APTER1,        @#ROMERR        ;IF WE HAVE THEN DEVICE CODE ON BOARD
667 003020  004767  001016                  JSR     PC,             ERRHAN          ;DOES NOT  EXIST IN ETABLE
```

```
003024  062703  000002          4$:     ADD     #2,             R3              ;UPDATE TO NEXT ADDRESS
003030  005713                          TST     (R3)                            ;IF CONTENTS EQUALS ZERO WE'RE DONE
003032  001356                          BNE     2$                              ;IF CONTENTS NO EQUAL ZERO CONTINUE
003034  012703  001222                  MOV     #$DDW0,         R3              ;GET BOARD DEVICE CODES
003040  012704  007720          5$:     MOV     #BUF1-2,        R4              ;GET ETABLE DEVICE CODES
003044  062704  000004          6$:     ADD     #4,             R4              ;GET BOARD DEVICE CODE
003050  021314                          CMP     (R3),           (R4)            ;DO THE TWO DEVICE CODES COMPARE
003052  001407                          BEQ     7$                              ;IF YES GET NEXT ETABLE DEVICE CODE
003054  005714                          TST     (R4)                            ;IF NO HAVE WE CHECKED ALL THE BOARD
003056  001372                          BNE     6$                              ;DEVICE CODES, IF NO CONTINUE
003060  052737  000002  001110          BIS     #APTER2,        @#ROMERR        ;IF YES ETABLE DEVICE CODE NOT
003066  004767  000750                  JSR     PC,             ERRHAN          ;NOT ON BOARD

003072  062703  000002          7$:     ADD     #2,             R3
003076  005713                          TST     (R3)                            ;IF CONTENTS EQUAL ZERO-DONE
003100  001357                          BNE     5$                              ;IF CONTENTS NO EQUAL ZERO-CONTINUE
003102  004767  000272                  JSR     PC,             PPFVAR          ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
003106  013704  001212                  MOV     @#$BASE,        R4              ;GET ETABLE PPFVA
003112  013703  010002                  MOV     @#BUF2,         R3              ;GET BOARD PPFVA
003116  020403                          CMP     R4,             R3              ;DO THE TWO PARAMETERS COMPARE
003120  001405                          BEQ     8$                              ;IF YES CONTINUE
003122  052737  000004  001110          BIS     #APTER3,        @#ROMERR        ;SET APT ERROR INDICATOR
003130  004767  000706                  JSR     PC,             ERRHAN          ;GO TO ERROR ROUTINE
003134  013704  001216          8$:     MOV     @#$CDW1,        R4              ;GET ETABLE DATA
003140  013703  010004                  MOV     @#BUF2+2,       R3              ;GET BOARD DATA
003144  020403                          CMP     R4,             R3              ;DO THE TWO PARAMETER COMPARE
003146  001405                          BEQ     9$                              ;IF YES THEN DONE
003150  052737  000010  001110          BIS     #APTER4,        @#ROMERR        ;SET APT ERROR INDICATOR
003156  004767  000660                  JSR     PC,             ERRHAN          ;GO TO ERROR ROUTINE
003162  000207                  9$:     RTS     PC

                                ;GET DEVICE CODES MODULE
                                ;THIS SUBROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE
                                ;ADDRESS IN WHICH IT WAS FOUND BACK TO THE CALLING ROUTINE.
                                ;DATA IS STORED IN BUFFER 'BUF1' IN THIS FORMAT:
                                ;
                                ;       BUF1:   ADDRESS OF FIRST DEVICE CODE
                                ;               DEVICE CODE
                                ;               ADDRESS OF SECOND DEVICE CODE
                                ;               DEVICE CODE
                                ;                 .
                                ;                 .
                                ;                 .
                                ;                 .
                                ;               ADDRESS OF NTH DEVICE CODE
                                ;               DEVICE CODE
                                ;
                                ;+CONTINUATION ROM DATA IS STORED IN 'BUF1' AS FOLLOWS:
                                ;+
                                ;+                .
                                ;+                .
                                ;+              CONTINUATION CHIP IDENTIFIER (ALWAYS A +1)
                                ;+              DEVICE CODE (PREVIOUS ROM'S DEVICE CODE,OR -1 IF ILLEGAL ROM)
                                ;+                .
                                ;+                .
                                ;+
                                ;IF DIAGNOSTIC ROM PRESENT IT WILL BE THE FIRST DEVICE CODE.
```

```
                                        ;THERE IS ROOM FOR UP TO 12 DEVICE CODES IN TEMP
003164 012700 007722        DEVGOD: MOV     #BUF1,          R0
003170 012701 000001                MOV     #1,             R1              ;INITIALIZE ROM POINTER
003174 030137 001116                BIT     R1,             @#ROMFIN        ;IS DIAGNOSTIC ROM PRESENT
003200 001411                       BEQ     1$                              ;IF NO GO TEST BOOT ROMS
003202 012737 165774 003374         MOV     #165774,        @#TESTAD        ;GET ADDRESS OF DIAG. ROM DEVICE CODE.
003210 013720 003374                MOV     @#TESTAD,       (R0)+           ;STORE ADDRESS OF DEVICE CODE
003214 017720 000154                MOV     @TESTAD,        (R0)+           ;STORE DEVICE CODE
003220 005037 003376                CLR     @#DAFLAG                        ;CLR DATA TYPE FLAG
003224 013737 001114 003374 1$:     MOV     @#FIRSTB,       @#TESTAD        ;GET ADDRESS OF FIRST BOOT ROM
003232 006101                       ROL     R1                              ,UPDATE POINTER TO NEXT ROM
003234 030167 007054                3$:     BIT     R1,             DEVFIN  ;+IS DEVICE ROM PRESENT
003240 001007                       BNE     4$                              ;IF YES GO GET PARAMETERS
003242 004767 005666                JSR     PC,     CON                     ;+CHECK FOR CONTINUATION CHIP
003246 062737 000200 003374         ADD     #200,           @#TESTAD        ;IF NO UPDATE TEST ADDRESS
003254 006101                       ROL     R1                              ;UPDATE POINTER TO NEXT ROM
003256 000441                       BR      9$                              ;GO SEE IF ALL ROM CHECKED
003260 005737 003376                4$:     TST     @#DAFLAG                ;IF DATAFLAG=0 THEN DATA IS DEVICE CODE
003264 001010                       BNE     5$                              ;IF DATAFLAG=1 THE DATA IS OFFSET TO NEXT
003266 013720 003374                MOV     @#TESTAD,       (R0)+           ;STORE ADDRESS OF DEVICE CODE
003272 017720 000076                MOV     @TESTAD,        (R0)+           ;STORE DEVICE CODE
003276 062737 000002 003374         ADD     #2,             @#TESTAD        ;UPDATE TEST ADDRESS
003304 000424                       BR      8$                              ;GET OVER SOME CODE
003306 013702 003374                5$:     MOV     @#TESTAD,       R2      ;SAVE OLD TEST ADDRESS
003312 067737 000056 003374         ADD     @TESTAD,        @#TESTAD        ;UPDATE TO NEW TEST ADDRESS
003320 013703 003374                MOV     @#TESTAD,       R3              ;GET THE NEW ADDRESS
003324 042702 170177                BIC     #170177,        R2              ;+SAVE ONLY BITS 7,8,9,10,11
003330 042703 170177                BIC     #170177,        R3              ;+IN R3 ALSO
003334 160203                       SUB     R2,             R3              ;CALCULATE DISTANCE BETWEEN  ADD
003336 005703                       6$:     TST     R3                      ;IS R3 EQUAL TO 0
003340 001406                       BEQ     8$                              ;IF YES THEN DONE
003342 162703 000200                SUB     #200,           R3              ;IF NO THEN MOVE POINTER
003346 006101                       ROL     R1                              ;ONE BIT FOR EVERY 200.
003350 004767 005606                JSR     PC,     CON2                    ;+CHECK FOR CONTINUATION CHIP
003354 000770                       BR      6$
003356 005137 003376        8$:     COM     @#DAFLAG                        ;CHANGE DATA FLAG TO RIGHT DATA TYPE
003362 023727 003374 174000 9$:     CMP     @#TESTAD,       #174000         ;HAVE WE CHECKED ALL THE ROM
003370 002721                       BLT     3$                              ;IF NO CONTINUE
003372 000207                       RTS     PC                              ;IF YES PETURN

003374 000000               TESTAD: 0
003376 000000               DAFLAG: 0

                            ;GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE
                            ;THIS SUBROUTINE TESTS LOCATIONS 173024 AND 173224 TO DETERMINE
                            ;WHICH VECTOR WILL BE USED IF POWER-FAIL OPTION ENABLED ON THE
                            ;BOARD.  THE OPTION MUST BE ENABLED AND AT LEAST ONE ADDRESS
                            ;SWITCH MUST BE "ON" OR AN ERROR WILL BE DETECTED.
                            ;THE DATA WILL BE RETURNED IN "BUF2" IN THE FORMAT.
                            ;
                            ;       BUF2:   PSEUDO POWER-FAIL VECTOR ADDRESS
                            ;               CONTENTS OF VECTOR ADDRESS

003400 022737 173000 173024 PPFVAR: CMP #173000,            @#173024        ;- TEST IF LOCATION 173024 SELECTED
003406 001414                       BEQ     1$                              ;IF NOT THEN GO TEST LOCATION 173224
003410 012767 003440 006670         MOV     #1$,    RETURN                  ;+SET UP AN ALTERNATE RETURN
003416 004767 005604                JSR     PC,     HOLCK1                  ;+CHECK FOR A HOLE
```

```
 782 003422  012737  173024  010002          MOV     #173024,        @#BUF2          ;IF IT IS THEN STORE ADDRESS
 783 003430  013737  173024  010004          MOV     @#173024,       @#BUF2+2        ;STORE CONTENTS OF LOCATION 173024
 784 003436  000423                          BR      3$                              ;GO TO RETURN
 785 003440  022737  173000  173224  1$:     CMP     #173000,        @#173224        ;-TEST IF LOCATION 173224 SELECTED
 786 003446  001414                          BEQ     2$                              ;IF NOT THEN SET ERROR INDICATOR
 787 003450  012767  003500  006630          MOV     #2$,    RETURN                  ;+SET UP AN ALTERNATE RETURN
 788 003456  004767  005556                  JSR     PC,     HOLCK2                  ;+CHECK FOR A HOLE
 789 003462  012737  173224  010002          MOV     #173224,        @#BUF2          ;IF IT IS THEN STORE ADDRESS
 790 003470  013737  173224  010004          MOV     @#173224,       @#BUF2+2        ;STORE CONTENTS OF VECTOR
 791 003476  000403                          BR      3$                              ;GET OVER ERROR
 792 003500  052737  000040  001110  2$:     BIS     #PFERR,         @#ROMERR        ;SET ERROR INDICATOR
 793 003506  000207          3$:     RTS     PC
 794
 795
 796
 797
 798                                 ;PUT MESSAGE IN BUFFER ROUTINE
 799                                 ;THIS SUBROUTINE FORMATS THE PARAMETER AND POWER-FAIL MESSAGES.
 800
 801 003510  017604  000000          PUTMES: MOV     @(R6),          R4              ;GET MESSAGE BUFFER ADDRESS
 802 003514  005737  004040                  TST     @#TIMES                         ;IS THIS FIRST TIME THROUGH
 803 003520  001110                          BNE     3$                              ;IF NOT FIRST TIME THEN FORMAT POWER-
 804                                                                                 ;FAIL MESSAGE
 805 003522  005237  004040                  INC     @#TIMES                         ;TOGGLE WATCHDOG
 806 003526  012703  007722                  MOV     #BUF1,          R3              ;GET DATA BUFFER ADDRESS
 807 003532  022713  165774                  CMP     #165774,        (R3)            ;IS DIAGNOSTIC ROM PRESENT
 808 003536  001012                          BNE     1$                              ;IF NOT DON'T FORMAT DIAG. ROM MESSAGE
 809 003540  012705  006754                  MOV     #DRHEAD,        R5              ;IF IT IS GET DIAG. ROM MESSAGE HEADER
 810 003544  004767  000714                  JSR     PC,             FILBUF          ;GO PUT HEADER IN MESSAGE BUFFER
 811 003550  000015                          CR
 812 003552  062703  000002                  ADD     #2,             R3              ;SKIP OVER ADDRESS OF ASCII
 813 003556  000313                          SWAB    (R3)                            ;FORMAT ASCII FOR MESSAGE
 814 003560  112324                          MOVB    (R3)+,          (R4)+           ;PUT ASCII IN MESSAGE BUFFER
 815 003562  112324                          MOVB    (R3)+,          (R4)+
 816 003564  012705  007021          1$:     MOV     #BRHEAD,        R5
 817 003570  004767  000670                  JSR     PC,             FILBUF          ;GO PUT BOOT ROM HEADER IN MESS. BUF.
 818 003574  000015                          CR
 819 003576  012701  000001                  MOV     #1,     R1                      ;+POINT TO DIAGNOSTIC ROM
 820 003602  013702  012326                  MOV     @#MESSPT,       R2              ;+POINT TO MSG TABLE
 821 003606  012767  003622  006472          MOV     #2$,    RETURN                  ;+SET UP AN ALTERNATE RETURN
 822 003614  012767  004006  006502          MOV     #5$,    FINISH                  ;+SET UP ANOTHER ALTERNATE RETURN
 823 003622  112724  000015          2$:     MOVB    #CR,            (R4)+           ;PUT A CR/LF HERE
 824 003626  112724  000012                  MOVB    #LF,            (R4)+
 825 003632  004767  005440                  JSR     PC,     ROMTYP                  ;+FIND THE ROM TYPE
 826 003636  062713  000004                  ADD     #4,             (R3)            ;GET FIRST ENTRY POINT
 827 003642  004767  000764                  JSR     PC,             OCADD           ;GO CONVERT OCTAL TO ASCII
 828 003646  004767  000612                  JSR     PC,             FILBUF          ;GO PUT ENTRY POINT IN MESSAGE BUF
 829 003652  000011                          HT
 830 003654  112724  000040                  MOVB    #40,            (R4)+
 831 003660  112724  000040                  MOVB    #40,            (R4)+
 832 003664  112724  000040                  MOVB    #40,            (R4)+
 833 003670  062713  000002                  ADD     #2,             (R3)            ;GET SECOND ENTRY POINT
 834 003674  004767  000732                  JSR     PC,             OCADD           ;GO CONVERT OCTAL TO ASCII
 835 003700  004767  000560                  JSR     PC,             FILBUF          ;GO PUT ENTRY POINT IN MESSAGE BUF.
 836 003704  000011                          HT
 837 003706  112724  000040                  MOVB    #40,            (R4)+
 838 003712  112724  000040                  MOVB    #40,            (R4)+
```

```
 839 003716  112724  000040                    MOVB    #40,            (R4)+
 840 003722  112724  000011                    MOVB    #HT,            (R4)+       ;PUT TAB IN HERE
 841 003726  062703  000002                    ADD     #2,             R3          ;UPDATE DATA BUFFER ADDRESS
 842 003732  000313                            SWAB    (R3)                        ;FORMAT ASCII FOR MESSAGE
 843 003734  112324                            MOVB    (R3)+,          (R4)+       ;MOV ASCII TO MES1
 844 003736  112324                            MOVB    (R3)+,          (R4)+
 845 003740  000730                            BR      2$                          ;GO BACK TO START OF LOOP
 846 003742  012703  010002            3$:     MOV     #BUF2,          R3          ;GET DATA BUFFER ADDRESS
 847 003746  012705  007156                    MOV     #PFHEAD,        R5          ;GET POWER-FAIL HEADER ADDRESS
 848 003752  004767  000506                    JSR     PC,             FILBUF      ;GO PUT POWER-FAIL HEADER IN MESSAGE BUF.
 849 003756  000015                            CR
 850 003760  022703  010006            4$:     CMP     #BUF2+4,        R3          ;ARE WE DONE
 851 003764  001410                            BEQ     5$                          ;IF YES THEN GO RETURN
 852 003766  004767  000640                    JSR     PC,             OCADD       ;GO CONVERT OCTAL TO ASCII
 853 003772  004767  000466                    JSR     PC,             FILBUF
 854 003776  000011                            HT
 855 004000  062703  000002                    ADD     #2,             R3          ;UPDATE DATA BUFFER ADDRESS
 856 004004  000765                            BR      4$                          ;GO BACK TO START OF LOOP
 857 004006  112724  000015            5$:     MOVB    #CR,            (R4)+       ;PUT A CR/LF AT END OF MESSAGE
 858 004012  112724  000012                    MOVB    #LF,            (R4)+
 859 004016  105024                            CLRB    (R4)+                       ;PUT ZERO TERMINATOR AT END OF MESSAGE
 860 004020  017667  000000  000002            MOV     @(R6),          6$          ;GET MESSAGE BUFFER ADDRESS
 861 004026  104401                            TYPE
 862 004030  000000            6$:     .WORD   0
 863 004032  062716  000002                    ADD     #2,             (R6)        ;GET OVER MESSAGE BUFFER ADDRESS
 864 004036  000207                            RTS     PC
 865 004040  000000            TIMES:  0
 866
 867
 868
 869
 870
 871
 872
 873
 874
 875                                    ;ERROR HANDLER ROUTINE
 876                                    ;THIS SUBROUTINE FORMATS THE ERROR MESSAGES THE TYPES THEM
 877                                    ;OUT.
 878 004042  104411            ERRHAN: SAVREG
 879 004044  012704  010516                    MOV     #ERRMSG,        R4
 880 004050  005237  001120                    INC     @#ERRCNT                    ;INCREMENT ERROR COUNTER
 881 004054  032777  002000  175022            BIT     #BIT10,         @SWR        ;BELL ON ERROR
 882 004062  001402                            BEQ     1$                          ;BRANCH IF NO
 883 004064  104401                            TYPE
 884 004066  006674                            BELL
 885 004070  032777  020000  175006    1$:     BIT     #BIT13,         @SWR        ;INHIBIT ERROR TYPEOUT
 886 004076  001402                            BEQ     OVER1                       ; NO GO THROUGH
 887 004100  000167  000310                    JMP     OVER10
 888 004104  013700  001110            OVER1:  MOV     @#ROMERR,       R0          ;GET ERROR CODE
 889 004110  013767  001110  000264            MOV     @#ROMERR,       OVER9       ;SAVE ERROR CODE FOR APT
 890 004116  012701  004436                    MOV     #EHEADT,        R1          ;GET ERROR HEADER TABLE
 891 004122  000241                            CLC                                 ;C-BIT USED TO STOP STEPPING THROUGH TABLE
 892 004124  006000            2$:     ROR     R0                                  ;ROTATE ERROR CODES
 893 004126  103403                            BCS     3$                          ;IF C-BIT SET STOP STEPPING
 894 004130  062701  000002                    ADD     #2,             R1          ;IF C-BIT CLEAR STEP TO NEXT HEADER
 895 004134  000773                            BR      2$                          ;GO STEP
```

```
896 004136 011105                3$:   MOV     (R1),          R5          ;PUT HEADER ADDRESS IN REGISTER.
897 004140 004767 000320               JSR     PC,            FILBUF      ;GO PUT HEADER IN ERROR MESSAGE BUF.
898 004144 000015                      CR
899 004146 032767 000040 174734        BIT     #PFERR, ROMERR             ;+IS IT A ''PPFVAR'' ERROR?
900 004154 001063                      BNE     8$                         ;+BRANCH IF YES
901 004156 032767 003600 174724        BIT     #3600,  ROMERR             ;+IS IT A ''SEQTST'' ERROR?
902 004164 001411                      BEQ     14$                        ;+BRANCH IF NO
903 004166 016767 006126 000010        MOV     ARG2,   13$                ;+PUT #CR OR #HT AT 13$
904 004174 016705 006116               MOV     ARG1,   R5                 ;+ARG1 CONTAINS ADR. OF MSG
905 004200 004767 000260               JSR     PC,     FILBUF             ;+PUT DATA INTO BUFFER
906 004204 000000                13$:  .WORD   0                          ;+CONTAINS CR OR HT
907 004206 000446                      BR      8$                         ;+
908 004210 032737 000023 001110  14$:  BIT     #23,           @#ROMERR    ;+IS ERROR A CRC, APTER1 OR APTER2
909 004216 001424                      BEQ     7$                         ;IF NO GO FORMAT APTER3, APTER4
910 004220 032737 000020 001110        BIT     #20,           @#ROMERR    ;IS ERROR CRC
911 004226 001415                      BEQ     6$                         ;IF NO FORMAT APTER1, APTER2
912 004230 013700 002602               MOV     @#ROMCNT,      R0          ;GET ROM COUNTER
913 004234 013701 006646               MOV     @#ROMPTR,      R1          ;GET ROM NUMBER TABLE
914 004240 000241                      CLC                                ;C-BIT USED TO STOP STEPPING THROUGH TABLE
915 004242 006000                4$:   ROR     R0                         ;ROTATE ROM NUMBER
916 004244 103403                      BCS     5$                         ;IF C-BIT SET STOP STEPPING
917 004246 062701 000002               ADD     #2,            R1          ;IF C-BIT CLEAR STEP TO NEXT HEADER
918 004252 000773                      BR      4$                         ;CONTINUE STEPPING
919 004254 112124                5$:   MOVB    (R1)+,         (R4)+       ;PUT ROM NUMBER IN ERROR MESSAGE BUF
920 004256 112124                      MOVB    (R1)+,         (R4)+
921 004260 000421                      BR      8$                         ;GO TO END OF ROUTINE
922 004262 112324                6$:   MOVB    (R3)+,         (R4)+       ;PUT BAD DEVICE CODE IN ERROR MESS. BUF.
923 004264 112324                      MOVB    (R3)+,         (R4)+
924 004266 000416                      BR      8$                         ;GO TO END OF ROUTINE
925 004270 016603 000006        7$:   MOV     6(R6),         R3          ;GET SAVED EXPECTED VALUE
926 004274 004767 000226               JSR     PC,            OCASC       ;GO COVERT OCTAL TO ASCII
927 004300 004767 000160               JSR     PC,            FILBUF      ;GO PUT DATA IN ERROR MESSAGE BUF.
928 004304 000011                      HT
929 004306 016603 000010               MOV     10(R6),        R3          ;GET SAVED RECEIVED VALUE
930 004312 004767 000210               JSR     PC,            OCASC       ;GO COVERT OCTAL TO ASCII
931 004316 004767 000142               JSR     PC,            FILBUF      ;GO PUT DATA IN ERROR MESSAGE BUFFER.
932 004322 000011                      HT
933 004324 112724 000015        8$:   MOVB    #CR,           (R4)+       ;PUT CR/LF AT END OF MESSAGE
934 004330 112724 000012               MOVB    #LF,           (R4)+
935 004334 112724 000000               MOVB    #0,            (R4)+       ;PUT TERMINATOR AT END OF MESSAGE
936 004340 005767 174612               TST     $ENV                       ;TST IF ON APT
937 004344 001002                      BNE     OVER2                      ;IF YES BRANCH
938 004346 000167 000034               JMP     OVER11                     ;IF NO BRANCH
939 004352 022737 000001 001116 OVER2: CMP #1,         @#ROMFIN           ;- ONLY A CPU ROM??????
940 004360 001005                      BNE     AROUND                     ; -IF NO BRANCH
941 004362 032737 000040 001110        BIT     #PFERR , @#ROMERR          ;- NOW IF YES ,IS POWER-
942                                                                       ;- FAIL VECTOR THE ERROR????
943 004370 001401                      BEQ     AROUND                     ;- YES ,WELL THEN IT'S
944 004372 000405                      BR OVER11                                ;- NOT A REAL ERROR , CONTINUE
945 004374 004767 000614        AROUND: JSR     PC,            $ATY1       ;GO REPORT ERROR
946 004400 010516                      ERRMSG
947 004402 000000                OVER9: .WORD   0
948 004404 000000                      HALT                               ;THEN HALT
949 004406 004767 000610        OVER11: JSR     PC,            $ATY3       ;IF NOT ON APT JUST REPORT ERROR
950 004412 010516                      ERRMSG
951 004414 032777 100000 174462 OVER10: BIT     #BIT15,        @SWR        ;IS HALT ON ERROR SET
952 004422 001401                      BEQ     12$                        ;IF NO SKIP OVER HALT
```

```
                                        HALT
004424  000000
004426  005037  001110          12$:  CLR     @#ROMERR                ;CLEAR ERROR FLAGS
004432  104412                        RESREG
004434  000207                        RTS     PC

004436  007257                EHEADT: ER1MSG
004440  007223                        ER2MSG
004442  007315                        ER3MSG
004444  007373                        ER4MSG
004446  007447                        CRCMSG
004450  007473                        PFMSG
004452  007551                        NOROMM
004454  007627                        SEQMSG                          ;+
004456  012376                        ERMES1                          ;+
004460  012455                        ERMES2                          ;+
004462  012535                        ERMES3                          ;+


                                ;FILL BUFFER ROUTINE
                                ;THIS SUBROUTINE FILLS THE MESSAGE BUFFER WILL ASCII
                                ;CHARACTERS.

004464  022776  000011  000000  FILBUF: CMP     #HT,    @(R6)           ;IS FIRST CHARACTER A TAB OR CR
004472  001003                        BNE     1$                      ;IF CR THEN GO PUT CR/LF IN BUFFER
004474  112724  000011                MOVB    #HT,    (R4)+           ;IF TAB THEN PUT TAB IN BUFFER
004500  000404                        BR      2$                      ;GET OVER NEXT LINE
004502  112724  000015          1$:   MOVB    #CR,    (R4)+           ;MOV CR/LF TO BUFFER
004506  112724  000012                MOVB    #LF,    (R4)+
004512  112524              2$:   MOVB    (R5)+,  (R4)+           ;PUT A CHARACTER IN MESSAGE BUFFER
004514  105715                        TSTB    (R5)                    ;IS NEXT CHARACTER ZERO
004516  001375                        BNE     2$                      ;IF NOT PUT IT IN MESSAGE BUFFER AND GET NEX
004520  062716  000002                ADD     #2,     (R6)            ;UPDATE RETURN POINTER TO GET OVER CHARACTER
004524  000207                        RTS     PC                      ;THEN RETURN


                                ;OCTAL TO ASCII CONVERSION ROUTINE
                                ;THIS SUBROUTINE TAKES A SIXTEEN BIT OCTAL NUMBER AND
                                ;CONVERTS IT TO 6 ASCII CHARACTERS

004526  012700  010006          OCASC:  MOV     #OCTBUF,        R0      ;GET BUFFER ADDRESS
004532  005020                        CLR     (R0)+                   ;CLEAR BUFFER
004534  005020                        CLR     (R0)+
004536  005020                        CLR     (R0)+
004540  005020                        CLR     (R0)+
004542  012700  010006                MOV     #OCTBUF,        R0      ;GET BUFFER ADDRESS
004546  010337  004630                MOV     R3,     @#TEMP          ;GET OCTAL NUMBER
004552  000241                        CLC                             ;CLEAR CARRY
004554  006137  004630                ROL     @#TEMP                  ;ROTATE BIT INTO CARRY BIT
004560  006110                        ROL     (R0)                    ;ROTATE CARRY BIT INTO BUFFER
004562  152710  000060          1$:   BISB    #60,    (R0)            ;MAKE IT ASCII
004566  005200                        INC     R0                      ;UPDATE BUFFER ADDRESS
004570  020027  010014                CMP     R0,     #OCTBUF+6       ;HAVE WE CONVERTED ALL THE NUMBER
004574  001003                        BNE     2$                      ;IF NO CONTINUE
```

```
1010 004576  012705  010006          MOV     #OCTBUF,        R5              ;IF YES PUT BUFFER ADDRESS IN REGISTER
1011 004602  000207                  RTS     PC                              ;RETURN
1012 004604  006137  004630    2$:   ROL     @#TEMP                          ;ROTATE BIT INTO CARRY BIT
1013 004610  106110                  ROLB    (R0)                            ;ROTATE CARRY BIT INTO BUFFER
1014 004612  006137  004630          ROL     @#TEMP                          ;
1015 004616  106110                  ROLB    (R0)                            ;
1016 004620  006137  004630          ROL     @#TEMP                          ;
1017 004624  106110                  ROLB    (R0)                            ;
1018 004626  000755                  BR      1$                              ;GO TO START OF LOOP
1019 004630  000000          TEMP:   0
1020
1021
1022                                 ;THIS ROUTINE IS CALLED BY THE PUT MESSAGE ROUTINE TO GET THE RIGHT
1023                                 ;VALUE IN R3 SO THE OCTAL TO ASCII ROUTINE GETS THE RIGHT NUMBER.
1024
1025
1026 004632  010346          OCADD:  MOV     R3,             -(R6)           ;SAVE THE VALUE OF R3
1027 004634  011303                  MOV     (R3),           R3              ;PUT THE DATA TO BE CONVERTED IN R3
1028 004636  004767  177664          JSR     PC,             OCASC           ;GO CONVERT OCTAL TO ASCII
1029 004642  012603                  MOV     (R6)+,          R3              ;RESTORE R3
1030 004644  000207                  RTS     PC                              ;RETURN
1031
1032
1033
1034
1035
1036                                 .SBTTL  TYPE ROUTINE

                                     ;;**********************************************************************
                                     ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                     ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                     ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                     ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                     ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                     ;*
                                     ;*CALL:
                                     ;*1) USING A TRAP INSTRUCTION
                                     ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                     ;*OR
                                     ;*      TYPE
                                     ;*      MESADR
                                     ;*

004646  105767  000335       $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
004652  100002                  BPL     1$              ;;BR IF YES
004654  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
004656  000430                  BR      3$              ;;LEAVE
004660  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
004662  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
004666  122767  000001  174262  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
004674  001011                  BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
004676  132767  000100  174253  BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
004704  001405                  BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
004706  010067  000004          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
004712  004767  000304          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
004716  000000          61$:    .WORD   0               ;;MESSAGE ADDRESS
004720  132767  000040  174231  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
```

```
004726  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
004730  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
004732  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
004734  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
004736  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
004740  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
004744  000002                          RTI                     ;;RETURN
004746  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
004752  001430                          BEQ     8$
004754  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
004760  001006                          BNE     5$
004762  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
004764  104401                          TYPE                    ;;TYPE A CR AND LF
004766  005211                          $CRLF
004770  105067  000200                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
004774  000755                          BR      2$              ;;GET NEXT CHARACTER
004776  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
005002  126726  000200          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
005006  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
005010  016746  000170                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
005014  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
005020  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
005022  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
005026  105367  000142                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
005032  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

005034  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
005040  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
005044  132767  000007  000122          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
005052  001372                          BNE     9$              ;;TAB STOP
005054  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
005056  000724                          BR      2$              ;;GET NEXT CHARACTER
005060  105777  000114          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
005064  100375                          BPL     $TYPEC
005066  116677  000002  000106          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
005074  105777  000362                  TSTB    @$TKS           ;;SEE IF KEYBOARD IS TALKING.
005100  100021                          BPL     2$              ;;BRANCH IF IT ISN'T.
005102  017746  000356                  MOV     @$TKB,-(SP)     ;;PUSH CHARACTER ONTO STACK.
005106  042716  177600                  BIC     #177600,(SP)    ;;BIT CLEAR TOP BYTE AND PARITY BIT.
005112  022726  000023                  CMP     #23,(SP)+       ;;SEE IF THIS IS A ^S.
005116  001012                          BNE     2$              ;;BRANCH TO CONTINUE IF IT ISN'T.
005120  105777  000336          3$:     TSTB    @$TKS           ;;WAIT FOR ANOTHER INPUT.
005124  100375                          BPL     3$              ;;BRANCH BACK IF NOT READY.
005126  017746  000332                  MOV     @$TKB,-(SP)     ;;PUSH NEXT CHARACTER ON STACK.
005132  042716  177600                  BIC     #177600,(SP)    ;;BIT CLEAR TOP BYTE AND PARITY BIT.
005136  022726  000021                  CMP     #21,(SP)+       ;;SEE IF THIS IS A ^Q.
005142  001366                          BNE     3$              ;;BRANCH BACK FOR MORE WAIT IF NOT.
005144  122766  000015  000002  2$:     CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
005152  001003                          BNE     1$              ;;BRANCH IF NO
005154  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
005160  000406                          BR      $TYPEX          ;;EXIT
005162  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
005170  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
005172  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
```

```
005174  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
005176  000207                  $TYPEX: RTS     PC

005200  177564                  $TPS:   .WORD   177564          ;;TTY PRINTER STATUS REG. ADDRESS
005202  177566                  $TPB:   .WORD   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
005204     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
005205     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
005206     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
005207     000                  $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
005210     077                  $QUES:  .ASCII  "?"             ;;QUESTION MARK
005211     015                  $CRLF:  .ASCII  <15>            ;;CARRAIGE RETURN
005212     012     000          $LF:    .ASCIZ  <12>            ;;LINEFEED
1037                            .SBTTL  APT COMMUNICATIONS ROUTINE

                                ;;*************************************************************
005214  112767  000001  000236  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
005222  112767  000001  000226  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
005230  000403                          BR      $ATYC
005232  112767  000001  00022C  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
005240                          $ATYC:
005240  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
005242  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
005244  105767  000206                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
005250  001450                          BEQ     5$              ;;IF NOT:  BR
005252  122767  000001  173676          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
005260  001031                          BNE     3$              ;;IF NOT:  BR
005262  132767  000100  173667          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
005270  001425                          BEQ     3$              ;;IF NOT:  BR
005272  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
005276  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
005304  005767  173626          1$.     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
005310  001375                          BNE     1$              ;;IF NOT:  WAIT
005312  010067  173634                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
005316  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
005320  001376                          BNE     2$
005322  166700  173624                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
005326  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
005330  010067  173620                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
005334  012767  000004  173574          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
005342  000413                          BR      5$
005344  017667  000004  000016  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
005352  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
005360  016746  172412                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
005364  004767  177256                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
005370  000000                  4$:     .WORD   0
005372                          5$:
005372  105767  000062          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
005376  001416                          BEQ     12$             ;;IF NOT:  BR
005400  005767  173552                  TST     $ENV            ;;RUNNING UNDER APT?
005404  001413                          BEQ     12$             ;;IF NOT:  BR
005406  005767  173524          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
005412  001375                          BNE     11$             ;;IF NOT:  WAIT
005414  017667  000004  173516          MOV     @4(SP),$FATAL   ;;GET ERROR #
005422  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
005430  005267  173502                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
005434  105067  000020          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
005440  105067  000013                  CLRB    $LFLG           ;;CLEAR LOG FLAG
```

```
005444  105067  000006                  CLRB    $MFLG           ;,CLEAR MESSAGE FLAG
005450  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
005452  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
005454  000207                          RTS     PC              ;;RETURN
005456     000                  $MFLG:  .BYTE   0               ;;MESSG. FLAG
005457     000                  $LFLG:  .BYTE   0               ;;LOG FLAG
005460     000                  $FFLG:  .BYTE   0               ;;FATAL FLAG
                                        .EVEN
            000200              APTSIZE=200
            000001              APTENV=001
            000100              APTSPOOL=100
            000040              APTCSUP=040
1038                            .SBTTL  TTY INPUT RO  INE

                                ;;**************.*************************************************
005462  177560                  $TKS:   .WORD   177560          ;;TTY KBD STATUS
005464   77562                  $TKB:   .WORD   177562          ;;TTY KBD BUFFER
                                .ENABL  LSB

                                ;;**************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*WHEN OPERATING IN TTY FLAG MODE.
005466  022767  000176  173410  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
005474  001074                          BNE     15$             ;;BRANCH IF NO
005476  105777  177760                  TSTB    @$TKS           ;;CHAR THERE?
005502  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
005504  117746  177754                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
005510  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
005514  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
005520  001062                          BNE     15$             ;;NO, RETURN TO USER
005522  126727  173352  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
005530  001456                          BEQ     15$             ;;BRANCH IF YES

005532  104401  006213                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
005536  104401  006220          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
005542  016746  172430                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
005546  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
005550  104401  006231                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
005554  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
005556  005046                          CLR     -(SP)           ;;THE NEW SWR
005560  105777  177676          7$:     TSTB    @$TKS           ;;CHAR THERE?
005564  100375                          BPL     7$              ;;IF NOT TRY AGAIN

005566  117746  177672                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
005572  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII


005576  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
005602  001005                          BNE     10$             ;;BRANCH IF NOT
005604  104401  006206                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
005610  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
005614  000757                          BR      19$             ;;LET'S TRY IT AGAIN
```

```
005616  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
005622  001022                          BNE     16$             ;;BRANCH IF NO
005624  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
005630  001403                          BEQ     11$             ;;BRANCH IF YES
005632  016677  000002  173244          MOV     2(SP),@SWR      ;;SAVE NEW SWR
005640  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
005644  104401  005211          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
005650  126727  173225  000001          CMPB    @$INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
005656  001003                          BNE     15$             ;;BRANCH IF NOT
005660  012777  000100  177574          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
005666  000002          15$:    RTI                             ;;RETURN
005670  004767  177164          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
005674  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
005700  002420                          BLT     18$             ;;BRANCH IF YES
005702  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
005706  003015                          BGT     18$             ;;BRANCH IF YES
005710  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
005714  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
005720  001403                          BEQ     17$             ;;BRANCH IF YES
005722  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
005724  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
005726  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
005730  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
005734  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
005740  000707                          BR      7$              ;;GET THE NEXT ONE
005742  104401  005210          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
005746  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB

                        ;;**************************************************************************
                        ;;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                        ;;*CALL:
                        ;;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                        ;;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                        ;;*                              ;;WITH PARITY BIT STRIPPED OFF
                        ;;

005750  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
005752  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
005760  105777  177476          1$:     TSTB    @$TKS           ;;WAIT FOR
005764  100375                          BPL     1$              ;;A CHARACTER
005766  117766  177472  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
005774  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
006002  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
006010  001013                          BNE     3$              ;;BRANCH IF NO
006012  105777  177444          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
006016  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
006020  117746  177440                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
006024  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
006030  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
006034  001366                          BNE     2$              ;;IF NOT DISCARD IT
006036  000750                          BR      1$              ;;YES, RESUME
006040  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
006046  002407                          BLT     4$              ;;BRANCH IF YES
006050  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
006056  003003                          BGT     4$              ;;BRANCH IF YES
```

```
006060  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
006066  000002                  4$:     RTI                     ;;GO BACK TO USER
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE T Y
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

006070  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
006072  012703  006176          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
006076  022703  006206          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
006102  101405                          BLOS    4$              ;;BR IF YES
006104  104407                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
006106  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
006110  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
006114  001003                          BNE     3$              ;;SKIP IF NOT
006116  104401  005210          4$:     TYPE    ,$QUES          ;;TYPE A '?'
006122  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
006124  111367  000044          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
006130  104401  006174                  TYPE    ,9$
006134  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
006140  001356                          BNE     2$              ;;LOOP IF NOT RETURN
006142  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
006146  104401  005212                  TYPE    ,$LF            ;;TYPE A LINE FEED
006152  012603                          MOV     (SP)+,R3        ;;RESTORE R3
006154  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
006156  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
006164  012766  006176  000004          MOV     #$TTYIN,4(SP)
006172  000002                          RTI                     ;;RETURN
006174     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
006175     000                          .BYTE   0               ;;TERMINATOR
006176                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
006206     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
006211     012     000
006213     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
006216     012     000
006220     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
006223     127     122     040
006226     075     040     000
006231     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
006234     105     127     040
006237     075     040     000
1039                            .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
```

```
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

006242  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
006246  116667  000001  000211          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
006254  112667  000207                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
006260  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
006264  000406                          BR      $TYPON
006266  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
006274  112767  000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
006302  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
006310  010346                          MOV     R3,-(SP)        ;;SAVE R3
006312  010446                          MOV     R4,-(SP)        ;;SAVE R4
006314  010546                          MOV     R5,-(SP)        ;;SAVE R5
006316  116704  000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
006322  005404                          NEG     R4
006324  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
006330  110467  000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
006334  116704  000125                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
006340  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
006344  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
006346  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
006350  000404                          BR      3$              ;;GO DO MSB
006352  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
006354  006105                          ROL     R5
006356  006105                          ROL     R5
006360  010503                          MOV     R5,R3
006362  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
006364  105367  000076                  DECB    $OMODE          ;;TYPE THIS DIGIT?
006370  100016                          BPL     7$              ;;BR IF NO
006372  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
006376  001002                          BNE     4$              ;;TEST FOR 0
006400  005704                          TST     R4              ;;SUPPRESS THIS 0?
006402  001403                          BEQ     5$              ;;BR IF YES
006404  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
006406  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
006412  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
006416  110367  000040                  MOVB    R3,8$           ;;SAVE FOR TYPING
006422  104401  006462                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
006426  105367  000032          7$:     DECB    $OCNT           ;;COUNT BY 1
006432  003347                          BGT     2$              ;;BR IF MORE TO DO
006434  002402                          BLT     6$              ;;BR IF DONE
006436  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
006440  000744                          BR      2$              ;;GO DO THE LAST DIGIT
006442  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
006444  012604                          MOV     (SP)+,R4        ;;RESTORE R4
006446  012603                          MOV     (SP)+,R3        ;;RESTORE R3
006450  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
006456  012616                          MOV     (SP)+,(SP)
```

```
006460  000002                  RTI                     ;;RETURN
006462     000          8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
006463     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
006464     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
006465     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
006466  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
1040                    .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES

                        ;;**********************************************************************
                        ;*SAVE R0-R5
                        ;*CALL:
                        ;*      SAVREG
                        ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                        ;*
                        ;*TOP---(+16)
                        ;* +2---(+18)
                        ;* +4---R5
                        ;* +6---R4
                        ;* +8---R3
                        ;*+10---R2
                        ;*+12---R1
                        ;*+14---R0

006470                  $SAVREG:
006470  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006472  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006474  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006476  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006500  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
006502  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
006504  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
006510  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
006514  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
006520  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
006524  000002                  RTI

                        ;*RESTORE R0-R5
                        ;*CALL:
                        ;*      RESREG
006526                  $RESREG:
006526  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
006532  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
006536  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
006542  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
006546  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
006550  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
006552  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
006554  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
006556  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
006560  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
006562  000002                  RTI
1043                    .SBTTL  TRAP DECODER

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

```
                                ;*GO TO THAT ROUTINE.

006564  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
006566  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
006572  005740                          TST     -(R0)           ;;BACKUP BY 2
006574  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
006576  006300                          ASL     R0              ;;POSITION FOR INDEXING
006600  016000  006620                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
006604  000200                          RTS     R0              ;;GO TO ROUTINE


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

006606  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
006610  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
006616  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
006620  006606                  $TRPAD: .WORD   $TRAP2
006622  004646                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
006624  006266                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
006626  006242                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
006630  006302                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

006632  005536                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

006634  005466                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
006636  005750                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
006640  006070                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
006642  006470                          $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
006644  006526                          $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
1044
1045
1046                                    ;MESSAGES
1047
1048
1049 006646  006662                    ROMPTR: .WORD   ROMNUM
1050 006650     065     070            AROM:   .ASCII  /58/                    ;**
1051 006652     064     070                    .ASCII  /48/                    ;**
1052 006654     064     071                    .ASCII  /49/                    ;**
1053 006656     065     060                    .ASCII  /50/                    ;**
1054 006660     065     071                    .ASCII  /59/                    ;**
1055 006662     062     060            ROMNUM: .ASCII  /20/
1056 006664     063     065                    .ASCII  /35/
1057 006666     063     063                    .ASCII  /33/
1058 006670     063     064                    .ASCII  /34/
1059 006672     063     062                    .ASCII  /32/
1060 006674     207     377     377    BELL:   .ASCIZ  <207><377><377>
     006677     000
1061 006700     012     015     116    MNORM:.ASCIZ  <12><15>/NO ROMS RESPONDED TO THE SIZING ROUTINE/<12><15>
     006703     117     040     122
```

```
        006706      117     115     123
        006711      040     122     105
        006714      123     120     117
        006717      116     104     105
        006722      104     040     124
        006725      117     040     124
        006730      110     105     040
        006733      123     111     132
        006736      111     116     107
        006741      040     122     117
        006744      125     124     111
        006747      116     105     012
        006752      015     000
  1062
```

```
 1064
 1065 006754  104  111  101  DRHEAD: .ASCIZ  /DIAG. ROM (E20) (FOR 11-44 UBI: E58)/  ;**
      006757  107  056  040
      006762  122  117  115
      006765  040  050  105
      006770  062  060  051
      006773  040  050  106
      006776  117  122  040
      007001  061  061  055
      007004  064  064  040
      007007  125  102  111
      007012  072  040  105
      007015  065  070  051
      007020  000
 1066 007021  012  102  117  BRHEAD: .ASCII  <12>/BOOTSTRAP ROM ENTRY POINTS AND DEVICE CODES/<15><12>
      007024  117  124  123
      007027  124  122  101
      007032  120  040  122
      007035  117  115  040
      007040  105  116  124
      007043  122  131  040
      007046  120  117  111
      007051  116  124  123
      007054  040  101  116
      007057  104  040  104
      007062  105  126  111
      007065  103  105  040
      007070  103  117  104
      007073  105  123  015
      007076  012
 1067 007077  114  117  103          .ASCIZ  /LOC.          NO DIAG.       RUN DIAG.       DEVICE CODE/
      007102  056  040  040
      007105  040  040  040
      007110  040  040  040
      007113  040  040  011
      007116  040  116  117
      007121  040  104  111
      007124  101  107  056
      007127  011  122  125
      007132  116  040  104
      007135  111  101  107
      007140  056  011  104
      007143  105  126  111
      007146  103  105  040
      007151  103  117  104
      007154  105  000
```

```
1069 007156  120  123  105  PFHEAD: .ASCIZ  @PSEUDO POWER-FAIL VECTOR ADR./NEW PC@
     007161  125  104  117
     007164  040  120  117
     007167  127  105  122
     007172  055  106  101
     007175  111  114  040
     007200  126  105  103
     007203  124  117  122
     007206  040  101  104
     007211  122  056  057
     007214  116  105  127
     007217  040  120  103
     007222  000
1070 007223  103  117  125  ER2MSG: .ASCIZ  /COULD NOT FIND DEVICE CODE /
     007226  114  104  040
     007231  116  117  124
     007234  040  106  111
     007237  116  104  040
     007242  104  105  126
     007245  111  103  105
     007250  040  103  117
     007253  104  105  040
     007256  000
1071 007257  106  117  125  ER1MSG: .ASCIZ  /FOUND UNEXPECTED DEVICE CODE /
     007262  116  104  040
     007265  125  116  105
     007270  130  120  105
     007273  103  124  105
     007276  104  040  104
     007301  105  126  111
     007304  103  105  040
     007307  103  117  104
     007312  105  040  000
1072 007315  120  117  127  ER3MSG: .ASCII  /POWER-FAIL VECTOR ERROR/<15><12>
     007320  105  122  055
     007323  106  101  111
     007326  114  040  126
     007331  105  103  124
     007334  117  122  040
     007337  105  122  122
     007342  117  122  015
     007345  012
1073 007346  011  105  130          .ASCIZ  /       EXPECTED        RECIEVED/<15><12>
     007351  120  105  103
     007354  124  105  104
     007357  011  122  105
     007362  103  111  105
     007365  126  105  104
     007370  015  012  000
1074 007373  120  117  127  ER4MSG: .ASCII  /POWER-FAIL DATA ERROR/<15><12>
     007376  105  122  055
     007401  106  101  111
     007404  114  040  104
     007407  101  124  101
     007412  040  105  122
     007415  122  117  122
     007420  015  012
```

```
 1075 007422  011  105  130          .ASCIZ  /       EXPE^TED        RECIEVED/<15><12>
      007425  120  105  103
      007430  124  105  104
      007433  011  122  105
      007436  103  111  105
      007441  126  105  104
      007444  015  012  000
 1076 007447  103  122  103  CRCMSG: .ASCIZ  /CRC ERROR IN ROM E-/
      007452  040  105  122
      007455  122  117  122
      007460  040  111  116
      007463  040  122  117
      007466  115  040  105
      007471  055  000
 1077 007473  103  117  125  PFMSG:  .ASCIZ  /COULD NOT DETERMINE POWER-FAIL VECTOR ADDRESS/
      007476  114  104  040
      007501  116  117  124
      007504  040  104  105
      007507  124  105  122
      007512  115  111  116
      007515  105  040  120
      007520  117  127  105
      007523  122  055  106
      007526  101  111  114
      007531  040  126  105
      007534  103  124  117
      007537  122  040  101
      007542  104  104  122
      007545  105  123  123
      007550  000
```

```
 1079 007551  116  117  040  NOROMM: .ASCII  /NO ROMS TEST ERROR/<15><12>
      007554  122  117  115
      007557  123  040  124
      007562  105  123  124
      007565  040  105  122
      007570  122  117  122
      007573  015  012
 1080 007575  040  040  040          .ASCIZ  /        VALUE   ADDRESS/<15><12>
      007600  040  040  040
      007603  040  040  126
      007606  101  114  125
      007611  105  040  040
      007614  040  101  104
      007617  104  122  105
      007622  123  123  015
      007625  012  000
 1081 007627  015  012  122  SEQMSG: .ASCII  <15><12>/ROM SEQUENCE IS INCORRECT AS PER /
      007632  117  115  040
      007635  123  105  121
      007640  125  105  116
      007643  103  105  040
      007646  111  123  040
      007651  111  116  103
      007654  117  122  122
      007657  105  103  124
      007662  040  101  123
      007665  040  120  105
      007670  122  040
 1082 007672  111  116  123          .ASCIZ  /INSTALLATION PROCEDURE./
      007675  124  101  114
      007700  114  101  124
      007703  111  117  116
      007706  040  120  122
      007711  117  103  105
      007714  104  125  122
      007717  105  056  000
 1083                               .EVEN
 1084
 1085
 1086                               ;BUFFERS
 1087
 1088 007722  000030                BUF1:   .REPT 30
 1090 007722  000000                        .WORD   0
      007724  000000                        .WORD   0
      007726  000000                        .WORD   0
      007730  000000                        .WORD   0
      007732  000000                        .WORD   0
      007734  000000                        .WORD   0
      007736  000000                        .WORD   0
      007740  000000                        .WORD   0
      007742  000000                        .WORD   0
      007744  000000                        .WORD   0
      007746  000000                        .WORD   0
      007750  000000                        .WORD   0
      007752  000000                        .WORD   0
      007754  000000                        .WORD   0
      007756  000000                        .WORD   0
```

```
                007760  000000                          .WORD   0
                007762  000000                          .WORD   0
                007764  000000                          .WORD   0
                007766  000000                          .WORD   0
                007770  000000                          .WORD   0
                007772  000000                          .WORD   0
                007774  000000                          .WORD   0
                007776  000000                          .WORD   0
                010000  000000                          .WORD   0
     1091       010002  000000              BUF2:       .WORD   0
     1092       010004  000000                          .WORD   0
     1093       010006                      OCTBUF:     .BLKB   10
     1094       010016                      MES1:       .BLKB   400
     1095       010416                      MES2:       .BLKB   100
     1096       010516                      ERRMSG:     .BLKB   40C                 ;+
     1097
     1098                                               ;+
     1099                                               ;+THE FOLLOWING CODE WAS ADDED IN RFV B0:
     1100                                               ;+
     1101                                               ;+''CLEAR'' CLEARS THE NEW LABEL AND BUFFER LOCATIONS
     1102                                               ;+
     1103                                               ;+''CON1'' AND ''CON2'' FILL 'BUF1'' WITH THE CONTINUATION ROM DATA AND
     1104                                               ;+SET ERROR FLAGS WHEN NECESSARY
     1105                                               ;+
     1106                                               ;+'HOLCK1'' AND 'HOLCK2'' CHECK FOR HOLES TO VERIFY THE POWER FAIL
     1107                                               ;+VECTOR ADDRESS; ALSO SET 11/60 INDICATOR WHEN APPLICABLE
     1108                                               ;+
     1109                                               ;+'ROMTYP'' AND 'ROMNM'' FILL THE MESSAGE BUFFER WITH A CONTINUATION ROM
     1110                                               ;+MESSAGE AND WITH A ROM IDENTIFICATION MESSAGE
     1111                                               ;+
     1112
```

```
1116 011116  012700  012262          CLEAR:  MOV   #SEQBUF,      R0      ;+SET UP TO CLEAR LOCATIONS
1117 011122  005020                  1$:     CLR   (R0)+                 ;+
1118 011124  020027  012324                  CMP   R0,     #FINISH       ;+ALL CLEARED?
1119 011130  101774                          BLOS  1$                    ;+BRANCH IF NO
1120 011132  000207                          RTS   PC                    ;+
1121 011134  030167  001152          CON1:   BIT   R1,     CONFIN        ;+CONTINUATION CHIP?
1122 011140  001001                          BNE   1$                    ;+BRANCH IF YES
1123 011142  000207                          RTS   PC                    ;+
1124 011144  050167  001132          1$:     BIS   R1,     CONERI        ;+CHIP DOES NOT BELONG HERE-SET ERROR FLAG
1125 011150  012720  000001                  MOV   #1,     (R0)+         ;+SET CONTINUATION ROM INDICATOR
1126 011154  012720  177777                  MOV   #-1,    (R0)+         ;+ILLEGAL ROM'S DEVICE CODE --1
1127 011160  000207                          RTS   PC                    ;+
1128 011162  005703                  CON2:   TST   R3                    ;+POINTING OVER THIS CHIP?
1129 011164  001001                          BNE   1$                    ;+BRANCH IF YES
1130 011166  000207                          RTS   PC                    ;+
1131 011170  030167  001116          1$:     BIT   R1,     CONFIN        ;+CONTINUATION CHIP?
1132 011174  001005                          BNE   2$                    ;+BRANCH IF YES
1133 011176  014067  001102                  MOV   -(R0),  CONER2        ;+CHIP IS MISSING-SAVE DEVICE CODE
1134 011202  062700  000002                  ADD   #2,     R0            ;+
1135 011206  000207                          RTS   PC                    ;+
1136 011210  012710  000001          2$:     MOV   #1,     (R0)          ;+SET CONTINUATION CHIP INDICATOR
1137 011214  014002                          MOV   -(R0),  R2            ;+GET PREVIOUS DEVICE CODE
1138 011216  062700  000004                  ADD   #4,     R0            ;+POINT TO CORRECT LOCATION
1139 011222  010220                          MOV   R2,     (R0)+         ;+PUT IN DEVICE CODE
1140 011224  000207                          RTS   PC                    ;+
1141 011226  032767  000002  167662  HOLCK1: BIT   #2,     ROMFIN        ;+A CHIP IN SOCKET #1?
1142 011234  001415                          BEQ   END                   ;+BRANCH IF NO
1143 011236  000207                          RTS   PC                    ;+
1144 011240  032767  000004  167650  HOLCK2: BIT   #4,     ROMFIN        ;+A CHIP IN SOCKET #2?
1145 011246  001401                          BEQ   1$                    ;+BRANCH IF NO
1146 011250  000207                          RTS   PC                    ;+
1147 011252  122737  000777  173224  1$:     CMPB  #777,   @#173224      ;+ARE SWITCHES SET?
1148 011260  001403                          BEQ   END                   ;+BRANCH IF NO
1149 011262  052767  000002  001020          BIS   #2,     FLAG          ;+SET 11/60 INDICATOR
1150 011270  016716  001012          END:    MOV   RETURN, (R6)          ;+SET UP ALTERNATE RETURN
1151 011274  000207                          RTS   PC                    ;+
1152 011276  005713                  ROMTYP: TST   (R3)                  ;+ARE WE FINISHED?
1153 011300  001003                          BNE   1$                    ;+BRANCH IF NO
1154 011302  016716  001016                  MOV   FINISH, (R6)          ;+SET UP ALTERNATE RETURN
1155 011306  000207                          RTS   PC                    ;+
1156 011310  022713  000001          1$:     CMP   #1,     (R3)          ;+CONTINUATION CHIP?
1157 011314  001401                          BEQ   2$                    ;+BRANCH IF YES
1158 011316  000414                          BR    ROMNM                 ;+ITS A DEVICE CHIP-FIND SOCKET #
1159 011320  004767  000024          2$:     JSR   PC,     ROMNM         ;+FIND SOCKET #
1160 011324  012705  012350                  MOV   #CONMES,      R5      ;+LOAD MSG INTO BUFFER
1161 011330  004767  173130                  JSR   PC,     FILBUF        ;+
1162 011334  000011                          HT                          ;+
1163 011336  062703  000004                  ADD   #4,     R3            ;+POP OVER CONTINUATION CHIP DATA
1164 011342  016716  000740                  MOV   RETURN, (R6)          ;+SET UP ALTERNATE RETURN
1165 011346  000207                          RTS   PC                    ;+
1166 011350  032713  000170          ROMNM:  BIT   #170,   (R3)          ;+BEGINNING A NEW ROM?
1167 011354  001015                          BNE   3$                    ;+BRANCH IF NOT
1168 011356  000241                  1$:     CLC                         ;+
1169 011360  006101                          ROL   R1                    ;+POINT TO NEXT SOCKET
1170 011362  030167  167530                  BIT   R1,     RCMFIN        ;+IS A ROM THERE?
1171 011366  001003                          BNE   2$                    ;+BRANCH IF YES
1172 011370  062702  000002                  ADD   #2,     R2            ;+POINT TO NEXT MSG
```

```
1173 011374  000770                    BR      1$                      ;+CONTINUE
1174 011376  012205              2$:   MOV     (R2)+,  R5              ;+LOAD ROM ID
1175 011400  004767  173060            JSR     PC,     FILBUF          ;+INTO THE MSG BUFFER
1176 011404  000015                    CR                              ;+
1177 011406  000402                    BR      4$                      ;+
1178 011410  112724  000011        3$:   MOVB    #HT,    (R4)+           ;+LOAD A TAB
1179 011414  000207              4$:   RTS     PC                      ;+
1180
1181                                   ;+SEQUENCE TEST
1182                                   ;+THIS TEST VERIFIES THAT CONTINUATION CHIPS ARE LOCATED WHERE
1183                                   ;+THEY BELONG, THAT THERE ARE NO DUPLICATE DEVICE ROMS,
1184                                   ;+THAT THE ROMS ARE IN ALPHABETICAL ORDER, THAT THERE ARE NO
1185                                   ;+HOLES, AND THAT SOCKET #2 HAS A CHIP IF THE PROCESSOR IS AN 11/60.
1186                                   ;+IF THE ONLY PROBLEM IS ALPHABETICAL ORDER AND/OR HOLES,
1187                                   ;+THEN A CORRECT SEQUENCE IS DETERMINED AND PRINTED OUT.
1188
1189                                   ;+FIRST, CHECK FOR ILLEGAL CONTINUATION ROM ERRORS
1190                                   ;+IF ANY ARE FOUND, PRINT ERROR MSG AND EXIT
1191
1192 011416  012767  000011  000674  SEQTST: MOV     #H',    ARG2            ;+SET UP FOR 'ERRHAN'
1193 011424  005767  000652                  TST     CONER1                  ;+WAS THERE AN ERROR?
1194 011430  001417                          BEQ     2$                      ;+BRANCH IF NO
1195 011432  016703  000644                  MOV     CONER1, R3              ;+THE BIT IDENTIFIES THE CHIP
1196 011436  013702  012326                  MOV     @#MESSPT,       R2      ;+POINT TO MSG TABLE
1197 011442  012767  000400  167440          MOV     #CONONE,        ROMERR  ;+SET ERROR FLAG
1198 011450  032703  000002        1$:       BIT     #2,     R3              ;+IDENTIFY THE ROM FOR PRINTOUT
1199 011454  001026                          BNE     3$                      ;+BRANCH IF FOUND IT
1200 011456  062702  000002                  ADD     #2,     R2              ;+SET UP FOR NEXT ROM MSG
1201 011462  000241                          CLC                             ;+
1202 011464  006003                          ROR     R3                      ;+SET UP TO IDENTIFY NEXT ROM
1203 011466  000770                          BR      1$                      ;+
1204 011470  005767  000610        2$:       TST     CONER2                  ;+WAS THERE AN ERROR?
1205 011474  001423                          BEQ     FILTAB                  ;+BRANCH IF NO
1206 011476  012767  001000  167404          MOV     #CONTWO,        ROMERR  ;+SET ERROR FLAG
1207 011504  012767  010016  000604          MOV     #MES1,  ARG1            ;+PASS THIS ADDRESS TO ERRHAN
1208 011512  000367  000566                  SWAB    CONER2                  ;+BUILD THE MSG
1209 011516  016767  000562  176272          MOV     CONER2, MES1            ;+
1210 011524  105067  176270                  CLRB    MES1+2                  ;+
1211 011530  000402                          BR      4$                      ;+REPORT THE ERROR
1212 011532  011267  000560        3$:       MOV     (R2),   ARG1            ;+PASS MSG ADDRESS TO 'ERRHAN'
1213 011536  004767  172300        4$:       JSR     PC      ERRHAN          ;+REPORT ERROR
1214 011542  000207                          RTS     PC                      ;+
1215
1216                                   ;+IF THERE WERE NO CONTINUATION ROM ERRORS, BUILD THE FOLLOWING TABLE:
1217                                   ;+
1218                                   ;+      SEQBUF:         FIRST DEVICE CODE
1219                                   ;+                      # OF CONTINUATION ROMS FOR FIRST DEVICE
1220                                   ;+                      SECOND DEVICE CODE
1221                                   ;+                      # OF CONTINUATION ROMS FOR SECOND DEVICE
1222                                   ;+                              .
1223                                   ;+                              .
1224                                   ;+      ENDSEQ:         FOURTH (LAST) DEVICE CODE
1225                                   ;+
1226                                   ;+FOR A 'HOLE', THE DEVICE CODE WILL BE A 0.
1227
1228 011544  012703  012262  FILTAB: MOV     #SEQBUF,        R3      ;+SET UP TO FILL A TABLE CALLED 'SEQBUF'
1229 011550  012702  173000          MOV     #173000,        R2      ;+POINT TO FIRST ROM
```

```
1230 011554  012700  000001            MOV     #1,     R0              ;+
1231 011560  000241              1$:   CLC                             ;+
1232 011562  006100                    ROL     R0                      ;+POINT TO NEXT ROM
1233 011564  030067  167326            BIT     R0,     ROMFIN          ;+IS IT A HOLE?
1234 011570  001404                    BEQ     3$                      ;+BRANCH IF YES
1235 011572  030067  000514      2$:   BIT     R0,     CONFIN          ;+IS IT A CONTINUATION CHIP?
1236 011576  001011                    BNE     4$                      ;+BRANCH IF YES
1237 011600  011213                    MOV     (R2),   (R3)            ;+ITS A DEVICE ROM-PUT DEVICE CODE INTO TABLE
1238 011602  062703  000004      3$:   ADD     #4,     R3              ;+POINT TO NEXT DEVICE CODE LOCATION
1239 011606  062702  000200      5$:   ADD     #200,   R2              ;+POINT TO NEXT ROM
1240 011612  030027  000020            BIT     R0,     #20             ;+ALL DONE?
1241 011616  001760                    BEQ     1$                      ;+NO-BRANCH
1242 011620  000404                    BR      ALPHCK                  ;+TABLE COMPLETELY BUILT
1243 011622  005243              4$:   INC     -(R3)                   ;+COUNT ONE CONTINUATION CHIP IN LOCATION
1244 011624  062703  000002            ADD     #2,     R3              ;+POINT TO NEXT DEVICE CODE LOCATION
1245 011630  000766                    BR      5$                      ;+
1246
1247                                ;+NOW CHECK DEVICE CODES IN "SEQBUF" FOR ALPHABETICAL ORDER AND
1248                                ;+CHECK FOR "HOLES".  IF THERE IS A DUPLICATE DEVICE CODE, PRINT
1249                                ;+AN ERROR MESSAGE AND EXIT.  IF DEVICE CODES NEED SORTING
1250                                ;+AND/OR HOLES FILLING, DO IT AND SET THE SEQUENCE ERROR
1251                                ;+FLAG.  ALSO CHECK FOR THE 11/60 SPECIAL CASE.
1252
1253 011632  012703  012262  ALPHCK: MOV     #SEQBUF,        R3      ;+SET UP TO CHECK ALPHABETIC SEQUENCE IN "SEQBUF"
1254 011636  010304        1$:   MOV     R3,     R4                      ;+WILL BE COMPARING (R3) TO (R4)
1255 011640  062704  000004      2$:   ADD     #4,     R4              ;+WHERE R3 AND R4 POINT TO DEVICE CODE LOCATIONS
1256 011644  020427  012276            CMP     R4,     #ENDSEQ         ;+PAST THE LAST DEVICE CODE?
1257 011650  101010                    BHI     3$                      ;+BRANCH IF YES
1258 011652  005713                    TST     (R3)                    ;+IS THERE A DEVICE CODE HERE?
1259 011654  001446                    BEQ     7$                      ;+BRANCH IF NO
1260 011656  005714                    TST     (R4)                    ;+IS THERE A DEVICE CODE HERE?
1261 011660  001767                    BEQ     2$                      ;+BRANCH IF NO
1262 011662  021314                    CMP     (R3),   (R4)            ;+IS THE SEQUENCE CORRECT?
1263 011664  001410                    BEQ     4$                      ;+BRANCH IF NO-FOUND A DUPLICATE
1264 011666  003025                    BGT     5$                      ;+BRANCH IF NO-NEED TO DO A SHIFT
1265 011670  000763                    BR      2$                      ;+YES-CONTINUE
1266 011672  062703  000004      3$:   ADD     #4,     R3              ;+POINT TO NEXT DEVICE CODE
1267 011676  020327  012276            CMP     R3,     #ENDSEQ         ;+ALL DONE?
1268 011702  103056                    BHIS    9$                      ;+BRANCH IF YES
1269 011704  000754                    BR      1$                      ;+NOT YET
1270 011706  012767  002000  167174  4$:   MOV     #DUPERR,        ROMERR  ;+SET ERROR FLAG
1271 011714  012767  010016  000374        MOV     #MES1,  ARG1            ;+PASS THIS ADDRESS TO "ERRHAN"
1272 011722  000314                    SWAB    (R4)                    ;+BUILD THE MSG
1273 011724  011467  176066            MOV     (R4),   MES1            ;+
1274 011730  105067  176064            CLRB    MES1+2                  ;+
1275 011734  004767  172102            JSR     PC,     ERRHAN          ;+REPORT THE ERROR
1276 011740  000207                    RTS     PC                      ;+
1277 011742  010301        5$:   MOV     R3,     R1                      ;+SET UP FOR THE SHUFFLE
1278 011744  010402                    MOV     R4,     R2              ;+
1279 011746  011200        6$:   MOV     (R2),   R0                      ;+SHIFT THE VALUES
1280 011750  011122                    MOV     (R1),   (R2)+           ;+
1281 011752  010021                    MOV     R0,     (R1)+           ;+
1282 011754  011200                    MOV     (R2),   R0              ;+
1283 011756  011112                    MOV     (R1),   (R2)            ;+
1284 011760  010011                    MOV     R0,     (R1)            ;+
1285 011762  012767  000200  167120    MOV     #SEQERR,        ROMERR  ;+SET THE SEQUENCE ERROR FLAG
1286 011770  000723                    BR      2$                      ;+CONTINUE SORTING
```

```
1287 011772  010305                    7$:   MOV   R3,     R5              ;+SET UP TO SEE IF WE HAVE A 'HOLE'
1288 011774  020527  012276              8$:   CMP   R5,     #ENDSEQ         ;+END OF TABLE?
1289 012000  103017                          BHIS  9$                      ;+YES-THIS WAS NOT A 'HOLE'
1290 012002  062705  000004                  ADD   #4,     R5              ;+NO-POINT TO NEXT DEVICE CODE LOCATION
1291 012006  005715                          TST   (R5)                    ;+IS THERE A ROM HERE?
1292 012010  001771                          BEQ   8$                      ;+BRANCH IF NO
1293 012012  032767  000032  167076          BIT   #32,    ROMFIN          ;+A ROM IN SOCKET #2 ONLY?
1294 012020  001004                          BNE   11$                     ;+BRANCH IF NO
1295 012022  105737  173224                  TSTB  @#173224                ;+11/60 SPECIAL CASE?
1296 012026  001401                          BEQ   11$                     ;+BRANCH IF NO
1297 012030  000422                          BR    10$                     ;+THE ONLY ROM MUST STAY IN SOCKET #2
1298 012032  010301              11$:   MOV   R3,     R1              ;+YES-THERE IS A HOLE THAT CAN BE FILLED
1299 012034  010502                          MOV   R5,     R2              ;+GET READY TO SHIFT THE VALUES
1300 012036  000743                          BR    6$                      ;+FILL IN THE HOLE
1301 012040  032767  000002  000242  9$:     BIT   #2,     FLAG            ;+AN 11/60?
1302 012046  001413                          BEQ   10$                     ;+BRANCH IF NO
1303 012050  005767  000212                  TST   SEQBUF+4                ;+IS SOCKET #2 EMPTY?
1304 012054  001366                          BNE   11$                     ;+BRANCH IF NO
1305 012056  005767  000200                  TST   SEQBUF                  ;+IS SOCKET #1 EMPTY?
1306 012062  001405                          BEQ   10$                     ;+BRANCH IF YES
1307 012064  012701  012262                  MOV   #SEQBUF,        R1      ;+SET UP TO SHIFT
1308 012070  012702  012266                  MOV   #SEQBUF+4,      R2      ;+CHIPS #1 AND #2
1309 012074  000724                          BR    6$                      ;+DO THE SHIFT
1310 012076  032767  000200  167004  10$:    BIT   #SEQERR,        ROMERR  ;+WAS THERE A SEQUENCE ERROR?
1311 012104  001001                          BNE   OUTTAB                  ;+BRANCH IF YES
1312 012106  000207                          RTS   PC                      ;+
1313
1314                                  ;+THE FOLLOWING ROUTINE BUILDS THE CORRECT SEQUENCE MSG TO BE
1315                                  ;+PRINTED AS AN ERROR MSG BY 'ERRHAN'
1316
1317 012110  012767  010016  000200  OUTTAB: MOV   #MES1,  ARG1            ;+PASS MSG ADR. TO 'ERRHAN'
1318 012116  012767  000015  000174          MOV   #CR,    ARG2            ;+PASS THIS TOO
1319 012124  012700  012262                  MOV   #SEQBUF,        R0      ;+
1320 012130  013702  012326                  MOV   @#MESSPT,       R2      ;+
1321 012134  012704  010016                  MOV   #MES1,  R4              ;+START BUILDING THE ERROR MSG
1322 012140  012705  012611                  MOV   #ERMES4,        R5      ;+PUT IN THE FIRST PART
1323 012144  004767  172314                  JSR   PC,     FILBUF          ;+
1324 012150  000015                          CR                            ;+
1325 012152  005710                          TST   (R0)                    ;+IS A ROM IN SOCKET #1?
1326 012154  001004                          BNE   1$                      ;+BRANCH IF YES
1327 012156  062700  000004                  ADD   #4,     R0              ;+DO SOCKET #2 ONLY
1328 012162  062702  000002                  ADD   #2,     R2              ;+
1329 012166  012767  010416  000126  1$:     MOV   #MES2,  BUILD           ;+SET UP THE DEVICE CODE PART
1330 012174  000310                          SWAB  (R0)                    ;+
1331 012176  012067  176214                  MOV   (R0)+,  MES2            ;+
1332 012202  105067  176212                  CLRB  MES2+2                  ;+
1333 012206  012205              2$:    MOV   (R2)+,  R5              ;+IDENTIFY THE ROM #
1334 012210  004767  172250                  JSR   PC,     FILBUF          ;+
1335 012214  000015                          CR                            ;+
1336 012216  016705  000100                  MOV   BUILD,  R5              ;+PUT IN EITHER DEVICE CODE OR CONTINUATION MSG
1337 012222  004767  172236                  JSR   PC,     FILBUF          ;+
1338 012226  000011                          WAIT                          ;+
1339 012230  005720                          TST   (R0)+                   ;+ANY CONTINUATION ROMS?
1340 012232  001405                          BEQ   3$                      ;+BRANCH IF NO
1341 012234  005340                          DEC   -(R0)                   ;+COUNT DOWN ONE ROM
1342 012236  012767  012350  000056          MOV   #CONMES,        BUILD   ;+SET UP FOR CONTINUATION MSG
1343 012244  000760                          BR    2$                      ;+CONTINUE
```

```
1344 012246  005710                3$:     TST     (R0)                    ;+ANY MORE DEVICE ROMS?
1345 012250  001346                        BNE     1$                      ;+BRANCH IF YES
1346 012252  105014                        CLRB    (R4)                    ;+MUST END WITH 0 BYTE
1347 012254  004767  171562               JSR     PC,     ERRHAN          ;+REPORT THE ERROR
1348 012260  000207                        RTS     PC                      ;+
1349
1350
1351 012262  000000                SEQBUF: .WORD   0                       ;+FIRST DEVICE CODE
1352 012264  000000                        .WORD   0                       ;+# OF CONTINUATION CHIPS FOR FIRST DEVICE
1353 012266  000000                        .WORD   0                       ;+SECOND DEVICE CODE, ETC
1354 012270  000000                        .WORD   0                       ;+
1355 012272  000000                        .WORD   0                       ;+
1356 012274  000000                        .WORD   0                       ;+
1357 012276  000000                ENDSEQ: .WORD   0                       ;+FOURTH (LAST) DEVICE CODE
1358 012300  000000                        .WORD   0                       ;+
1359 012302  000000                CONER1: .WORD   0                       ;+EXTRA CONTINUATION CHIP ERROR FLAG
1360 012304  000000                CONER2: .WORD   0                       ;+MISSING CONTINUATION CHIP FOR THIS DEVICE
1361 012306  000000                RETURN: .WORD   0                       ;+ALTERNATE RETURN ADDRESS
1362 012310  000000                FLAG:   .WORD   0                       ;+BIT 1:11/60
1363 012312  000000                CONFIN: .WORD   0                       ;+CONTINUATION ROM FOUND INDICATOR
1364 012314  000000                DEVFIN: .WORD   0                       ;+DEVICE ROM FOUND INDICATOR
1365 012316  000000                ARG1:   .WORD   0                       ;+PASSES MSG ADR. TO 'ERRHAN'
1366 012320  000000                ARG2:   .WORD   0                       ;+PASSES CR OR HT TO 'ERRHAN'
1367 012322  000000                BUILD:  .WORD   0                       ;+
1368 012324  000000                FINISH: .WORD   0                       ;+ALTERNATE RETURN ADDRESS
1369
1370 012326  012340                MESSPT: .WORD   MESTAB
1371 012330  012636                MESS44: AROM1                           ;**
1372 012332  012652                        AROM2                           ;**
1373 012334  012666                        AROM3                           ;**
1374 012336  012702                        AROM4                           ;**
1375 012340  012716                MESTAB: ROM1                            ;+ASCII MSG TABLE
1376 012342  012732                        ROM2                            ;+
1377 012344  012746                        ROM3                            ;+
1378 012346  012762                        ROM4                            ;+
1379 012350      111     123     040 CONMES: .ASCIZ /IS A CONTINUATION ROM/
     012353      101     040     103
     012356      117     116     124
     012361      111     116     125
     012364      101     124     111
     012367      117     116     040
     012372      122     117     115
     012375      000
1380 012376      015     012     101 ERMES1: .ASCIZ <15><12>/A CONTINUATION ROM IS INCORRECTLY LOCATED IN/
     012401      040     103     117
     012404      116     124     111
     012407      116     125     101
     012412      124     111     117
     012415      116     040     122
     012420      117     115     040
     012423      111     123     040
     012426      111     116     103
     012431      117     122     122
     012434      105     103     124
     012437      114     131     040
     012442      114     117     103
     012445      101     124     105
```

```
        012450  104  040  111
        012453  116  000
 1381   012455  015  012  101  ERMES2: .ASCIZ <15><12>/A CONTINUATION ROM IS MISSING FOR DEVICE CODE/
        012460  040  103  117
        012463  116  124  111
        012466  116  125  101
        012471  124  111  117
        012474  116  040  122
        012477  117  115  040
        012502  111  123  040
        012505  115  111  123
        012510  123  111  116
        012513  107  040  106
        012516  117  122  040
        012521  104  105  126
        012524  111  103  105
        012527  040  103  117
        012532  104  105  000
 1382   012535  015  012  124  ERMES3: .ASCIZ <15><12>/THERE IS A DUPLICATE ROM WITH DEVICE CODE/
        012540  110  105  122
        012543  105  040  111
        012546  123  040  101
        012551  040  104  125
        012554  120  114  111
        012557  103  101  124
        012562  105  040  122
        012565  117  115  040
        012570  127  111  124
        012573  110  040  104
        012576  105  126  111
        012601  103  105  040
        012604  103  117  104
        012607  105  000
 1383   012611  123  105  121  ERMES4: .ASCIZ /SEQUENCE SHOULD BE:/<12>
        012614  125  105  116
        012617  103  105  040
        012622  123  110  117
        012625  125   14  104
        012630  040  102  105
        012633  072  012  000
 1384   012636  122  117  115  AROM1:  .ASCIZ  /ROM 1(E48) /          ;**
        012641  040  061  050
        012644  105  064  070
        012647  051  040  000
 1385   012652  122  117  115  AROM2:  .ASCIZ  /ROM 2(E49) /          ;**
        012655  040  062  050
        012660  105  064  071
        012663  051  040  000
 1386   012666  122  117  115  AROM3:  .ASCIZ  /ROM 3(E50) /          ;**
        012671  040  063  050
        012674  105  065  060
        012677  051  040  000
 1387   012702  122  117  115  AROM4:  .ASCIZ  /ROM 4(E59) /          ;**
        012705  040  064  050
        012710  105  065  071
        012713  051  040  000
 1388   012716  122  117  115  ROM1:   .ASCIZ /ROM 1(E35) /
```

```
        012721      040      06'      050
        012724      105      063      065
        012727      051      040      000
  1389  012732      122      117      115  ROM2:   .ASCIZ /ROM 2(E33) /
        012735      040      062      050
        012740      105      063      063
        012743      051      040      000
  1390  012746      122      117      115  ROM3:   .ASCIZ /ROM 3(E34) /
        012751      040      063      050
        012754      105      063      064
        012757      051      040      000
  1391  012762      122      117      115  ROM4:   .ASCIZ /ROM 4(E32) /
        012765      040      064      050
        012770      105      063      062
        012773      051      040      000
  1392
  1393
  1394            000001                   .END
```

SYMBOL TABLE

```
ABASE = 000000   AROM4   012702   DAFLAG  003376   OVER1   004104   SW02  = 000004
ACDW1 = 000000   AROUND  004374   DDISP = 177570   OVER10  004414   SW03  = 000010
ACDW2 = 000000   ASWREG= 000000   DEVCOD  003164   OVER11  004406   SW04  = 000020
ACPUOP= 000000   ATESTN= 000000   DEVFIN  012314   OVER2   004352   SW05  = 000040
ADDW0 = 000000   AUNIT = 000000   DISPLA  001106   OVER9   004402   SW06  = 000100
ADDW1 = 000000   AUSWR - 000000   DISPRE  000174   PFERR = 000040   SW07  - 000200
ADDW10= 000000   AVECT1= 000000   DRHEAD  006754   PFHEAD  007156   SW08  = 000400
ADDW11- 000000   AVECT2= 000000   DSWR  = 177570   PFMSG   007473   SW09  = 001000
ADDW12= 000000   BELL    006674   DUPERR= 002000   PIRQ  = 177772   SW1   = 000002
ADDW13= 000000   BIT0  = 000001   EHEADT  004436   PIRQVE= 000240   SW10  = 002000
ADDW14= 000000   BIT00 = 000001   EMTVEC= 000030   PPFVAR  003400   SW11  = 004000
ADDW15= 000000   BIT01 = 000002   END     011270   PROMP   002702   SW12  = 010000
ADDW2 = 000000   BIT02 = 000004   ENDSEQ  012276   PR0   = 000000   SW13  = 020000
ADDW3 = 000000   BIT03 = 000010   ERMES1  012376   PR1   = 000040   SW14  = 040000
ADDW4 = 000000   BIT04 = 000020   ERMES2  012455   PR2   = 000100   SW15  = 100000
ADDW5 = 000000   BIT05 = 000040   ERMES3  012535   PR3   - 000140   SW2   = 000004
ADDW6 = 000000   BIT06 = 000100   ERMES4  012611   PR4   = 000200   SW3   = 000010
ADDW7 = 000000   BIT07 = 000200   ERRCNT  001120   PR5   = 000240   SW4   = 000020
ADDW8 = 000000   BIT08 = 000400   ERRHAN  004042   PR6   = 000300   SW5   = 000040
ADDW9 = 000000   BIT09 = 001000   ERRMSG  010516   PR7   = 000340   SW6   = 000100
ADEVCT= 000000   BIT'  = 000002   ERROR = 104000   PS    = 177776   SW7   = 000200
ADEVM = 000000   BIT10 = 002000   ERRVEC= 000004   PSW   = 177776   SW8   = 000400
AENV  = 000000   BIT11 = 004000   ER1MSG  007257   PUTMES  003510   SW9   = 001000
AENVM = 000000   BIT12 = 010000   ER2MSG  007223   PWRVEC= 000024   TBITVE= 000014
AFATAL= 000000   BIT13 = 020000   ER3MSG  007315   RDCHR = 104407   TEMP    004630
ALPHCK  011632   BIT14 = 040000   ER4MSG  007373   RDLIN = 104410   TESTAD  003374
AMADR1= 000000   BIT15 - 100000   EXCADD  002676   RESREG= 104412   TIMES   004040
AMADR2= 000000   BIT2  = 000004   FILBUF  004464   RESVEC= 000010   TKVEC = 000060
AMADR3= 000000   BIT3  = 000010   FILTAB  011544   RETURN  012306   TPVEC = 000064
AMADR4= 000000   BIT4  = 000020   FINISH  012324   ROMCNT  002602   TRAPVE= 000034
AMAMS1= 000000   BIT5  = 000040   FIRSTA  002674   ROMERR  001110   TRTVEC= 000014
AMAMS2= 000000   BIT6  = 000100   FIRSTB  001114   ROMFIN  001116   TYPE  = 104401
AMAMS3= 000000   BIT7  = 000200   FLAG    012310   ROMNM   011350   TYPOC = 104402
AMAMS4= 000000   BIT8  - 000400   GTSWR = 104405   ROMNUM  006662   TYPON = 104404
AMSGAD= 000000   BIT9  = 001000   HOLCK1  011226   ROMPTR  006646   TYPOS = 104403
AMSGLG= 000000   BPTVEC= 000014   HOLCK2  011240   ROMTYP  011276   $APTHD  001122
AMSGTY= 000000   BRHEAD  007021   HT    = 000011   ROM1    012716   $ATYC   005240
AMTYP1= 000000   BUF1    007722   IOTVEC= 000020   ROM2    012732   $ATY1   005214
AMTYP2- 000000   BUF2    010002   LASTAD  002700   ROM3    012746   $ATY3   005222
AMTYP3- 000000   BUILD   012322   LF    = 000012   ROM4    012762   $ATY4   005232
AMTYP4= 000000   CALSUM  002604   LOOP    002614   RSTART  002140   $AUTOB  001100
APASS = 000000   CHECKS  002366   MESSAG  001112   R6    =%000006   $BASE   001212
APRIOR= 000000   CKSWR = 104406   MESSPT  012326   R7    -%000007   $CDW1   001216
APTCSU= 000040   CLEAR   011116   MESS44  012330   SAVREG= 104411   $CDW2   001220
APTENV= 000001   CONER1  012302   MESTAB  012340   SCOPE = 000004   $CHARC  005174
APTER1= 000001   CONER2  012304   MFS1    010016   SEQBUF  012262   $CKSWR  005466
APTER2= 000002   CONFIN  012312   MFS2    010416   SEQERR= 000200   $CNTLG  006213
APTER3= 000004   CONMES  012350   MFPT  = 000007   SEQMSG  007627   $CNTLU  006206
APTER4= 000010   CONONE= 000400   MNORM   006700   SEQTST  011416   $CPUOP  001164
APTSIZ= 000200   CONTWO= 001000   NODATA  002364   STACK = 001100   $CRLF   005211
APTSPO= 000100   CON1    011134   NOROME= 000100   START   001400   $DDW0   001222
ARG1    012316   CON2    011162   NOROMM  007551   STKLMT= 177774   $DDW1   001224
ARG2    012320   CR    = 000015   NOROMS  002300   SWR     001104   $DDW10  001246
AROM    006650   CRCERR= 000020   OCADD   004632   SWREG   000176   $DDW11  001250
AROM1   012636   CRCLOP  002622   OCASC   004526   SW0   = 000001   $DDW12  001252
AROM2   012652   CRCMSG  007447   OCTBUF  010006   SW00  = 000001   $DDW13  001254
AROM3   012666   CRLF  = 000200   OUTTAB  012110   SW01  = 000002   $DDW14  001256
```

```
$DDW15  001260        $ETABL  001156        $MAMS3  001176        $RDCHR  005750        $TRAP2  006606
$DDW2   001226        $ETEND  001262        $MAMS4  001202        $RDLIN  006070        $TRP    -000013
$DDW3   001230        $FATAL  001140        $MBADR  001124        $RDSZ   =000010        $TRPAD  006620
$DDW4   001232        $FFLG   005460        $MFLG   005456        $RESRE  006526        $TSTM   001126
$DDW5   001234        $FILLC  005206        $MNEW   006231        $RTNAD  002260        $TTYIN  006176
$DDW6   001236        $FILLS  005205        $MSGAD  001152        $SAVRE  006470        $TYPE   004646
$DDW7   001240        $GET42  002236        $MSGLG  001154        $SETUP= 000124        $TYPEC  005060
$DDW8   001242        $GTSWR  005536        $MSGTY  001136        $STUP   =177777       $TYPEX  005176
$DDW9   001244        $HD     =000003       $MSWR   006220        $SVPC   =000210       $TYPOC  006266
$DEVCT  001146        $HIB    001122        $MTYP1  001167        $SWR    =160000       $TYPON  006302
$DEVM   001214        $INTAG  001101        $MTYP2  001173        $SWREG  001160        $TYPOS  006242
$DOAGN  002256        $LF     005212        $MTYP3  001177        $TESTN  001142        $UNIT   001150
$ENDAD  002246        $LFLG   005457        $MTYP4  001203        $TKB    005464        $UNITM  001132
$ENDCT  002222        $MADR1  001170        $NULL   005204        $TKS    005462        $USWR   001162
$ENDMG  002265        $MADR2  001174        $OCNT   006464        $TN     =000001       $VECT1  001206
$ENULL  002262        $MADR3  001200        $OMODE  006466        $TPB    005202        $VECT2  001210
$ENV    001156        $MADR4  001204        $PASS   001144        $TPFLG  005207        $$GET4= 000000
$ENVM   001157        $MAIL   001136        $PASTM  001130        $TPS    005200        $OFILL  006465
$EOP    002176        $MAMS1  001166        $QUES   005210        $TRAP   006564        .$X     -001122
$EOPCT  002214        $MAMS2  001172

. ABS.  012776      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  58280 WORDS  ( 228 PAGES)
DYNAMIC MEMORY:  20434 WORDS  ( 78 PAGES)
ELAPSED TIME:  00:05:20
CZM9BDO,CZM9BDO.SEQ/NL:TOC/CRF/-SP=CZM9BDO.SML,CZM9BDO.P11
```

CZM9BDO CREATED BY MACRO ON 24-SEP-79 AT 16:40 PAGE 1
SYMBOL CROSS REFERENCE CREF SEQ 0049

SYMBOL	VALUE	REFERENCES					
ABASE	= 000000	61-469	61-469				
ACDW1	= 000000	61-469	61-469				
ACDW2	= 000000	61-469	61-469				
ACPUOP	= 000000	61-469	61-469				
ADDW0	= 000000	61-469	61-469				
ADDW1	= 000000	61-469	61-469				
ADDW10	= 000000	61-469	61-469				
ADDW11	= 000000	61-469	61-469				
ADDW12	= 000000	61-469	61-469				
ADDW13	= 000000	61-469	61-469				
ADDW14	= 000000	61-469	61-469				
ADDW15	= 000000	61-469	61-469				
ADDW2	= 000000	61-469	61-469				
ADDW3	= 000000	61-469	61-469				
ADDW4	= 000000	61-469	61-469				
ADDW5	= 000000	61-469	61-469				
ADDW6	- 000000	61-469	61-469				
ADDW7	- 000000	61-469	61-469				
ADDW8	- 000000	61-469	61-469				
ADDW9	- 000000	61-469	61-469				
ADEVCT	= 000000	61-469	61-469				
ADEVM	- 000000	61-469	61-469				
AENV	- 000000	61-469	61-469				
AENVM	- 000000	61-469	61-469				
AFATAL	= 000000	61-469	61-469				
ALPHCK	011632	65-1242	#65-1253				
AMADR1	= 000000	61-469	61-469				
AMADR2	= 000000	61-469	61-469				
AMADR3	= 000000	61-469	61-469				
AMADR4	000000	61-469	61-469				
AMAMS1	- 000000	61-469	61-469				
AMAMS2	= 000000	61-469	61-469				
AMAMS3	= 000000	61-469	61-469				
AMAMS4	= 000000	61-469	61-469				
AMSGAD	= 000000	61-469	61-469				
AMSGLG	- 000000	61-469	61-469				
AMSGTY	000000	61-469	61-469				
AMTYP1	- 000000	61-469	61-469				
AMTYP2	= 000000	61-469	61-469				
AMTYP3	= 000000	61-469	61-469				
AMTYP4	- 000000	61-469	61-469				
APASS	- 000000	61-469	61-469				
APRIOR	= 000000	61-469					
APTCSU	= 000040	61-1036	#61-1037				
APTENV	= 000001	61-1036	61-1037	#61-1037			
APTER1	000001	#61-436	61-666				
APTER2	000002	#61-437	61-678				
APTER3	= 000004	#61-438	61-689				
APTER4	- 000010	#61-439	61-695				
APTSIZ	000200	61-472	#61-1037				
APTSPO	- 000100	61-1036	61-1037	#61-1037			
ARG1	012316	61-904	*65-1207	*65-1212	*65-1271	*65-1317	#65-1365

SYMBOL	VALUE	REFERENCES								
ARG2	012320	61-903	*65-1192	*65-1318	#65-1366					
AROM	006650	61-480	#61-1050							
AROM1	012636	65-1371	#65-1384							
AROM2	012652	65-1372	#65-1385							
AROM3	012666	65-1373	#65-1386							
AROM4	012702	65-1374	#65-1387							
AROUND	004374	61-940	61-943	#61-945						
ASWREG	= 000000	61-469	61-469							
ATESTN	= 000000	61-469	61-469							
AUNIT	= 000000	1-469	61-469							
AUSWR	= 000000	61-469	61-469							
AVECT1	= 000000	61-469	61-469							
AVECT2	= 000000	61-469	61-469							
BELL	006674	61-884	#61-1060							
BIT0	= 000001	#61-432								
BIT00	= 000001	#61-432	61-432							
BIT01	= 000002	#61-432	61-432							
BIT02	= 000004	#61-432	61-432							
BIT03	= 000010	#61-432	61-432							
BIT04	= 000020	#61-432	61-432							
BIT05	= 000040	#61-432	61-432							
BIT06	= 000100	#61-432	61-432							
BIT07	= 000200	#61-432	61-432							
BIT08	= 000400	#61-432	61-432							
BIT09	= 001000	#61-432	61-432							
BIT1	= 000002	#61-432								
BIT10	= 002000	#61-432	61-881							
BIT11	= 004000	#61-432								
BIT12	= 010000	#61-432								
BIT13	= 020000	#61-432	61-885							
BIT14	= 040000	#61-432								
BIT15	= 100000	#61-432	61-951							
BIT2	= 000004	#61-432								
BIT3	= 000010	#61-432								
BIT4	= 000020	#61-432								
BIT5	= 000040	#61-432								
BIT6	= 000100	#61-432								
BIT7	= 000200	#61-432								
BIT8	= 000400	#61-432								
BIT9	= 001000	#61-432								
BPTVEC	= 000014	#61-432								
BRHEAD	007021	61-816	#62-1066							
BUF1	007722	61-485	61-658	61-672	61-726	61-806	#64-1088			
BUF2	010002	61-686	61-692	61-782	61-783	61-789	61-790	61-846	61-850	#64-1091
BUILD	012322	*65-1329	65-1336	*65-1342	#65-1367					
CALSUM	002604	61-571	61-584	#61-607						
CHECKS	002366	61-525	#61-565							
CKSWR	- 104406	#61-1043								
CLEAR	011116	61-491	#65-1116							
CONER1	012302	*65-1124	65-1193	65-1195	#65-1359					
CONER2	012304	*65-1133	65-1204	*65-1208	65-1209	#65-1360				
CONFIN	012312	*61-512	65-1121	65-1131	65-1235	#65-1363				

SYMBOL	VALUE	REFERENCES								
CONMES	012350	65-1160	65-1342	#65-1379						
CONONE	= 000400	#61-444	65-1197							
CONTWO	= 001000	#61-445	65-1206							
CON1	011134	61-738	#65-1121							
CON2	011162	61-758	#65-1128							
CR	= 000015	#61-432	61-811	61-818	61-823	61-849	61-857	61-898	61-955	61-981
		61-1036	61-1036	65-1176	65-1318	65-1324	65-1335			
CRCERR	= 000020	#61-440	61-575	61-588						
CRCLOP	002622	#61-612	61-621							
CRCMSG	007447	61-963	#63-1076							
CRLF	= 000200	#61-432	61-473	61-473	61-1036	61-1036				
DAFLAG	003376	61-733	61-742	61-760	#61-766					
DDISP	= 177570	#61-432	61-461	61-472						
DEVCOD	003164	61-645	61-657	#61-726						
DEVFIN	012314	*61-516	61-736	#65-1364						
DISPLA	001106	#61-461	*61-472	*61-472						
DISPRE	000174	#61-449	61-472							
DRHEAD	006754	61-809	#62-1065							
DSWR	= 177570	#61-432	61-460	61-472						
DUPERR	= 002000	#61-446	65-1270							
EHEADT	004436	61-890	#61-959							
EMTVEC	= 000030	#61-432								
END	011270	65-1142	65-1148	#65-1150						
ENDSEQ	012276	65-1256	65-1267	65-1288	#65-1357					
ERMES1	012376	61-967	#65-1380							
ERMES2	012455	61-968	#65-1381							
ERMES3	012535	61-969	#65-1382							
ERMES4	012611	65-1322	#65-1383							
ERRCNT	001120	#61-466	61-880							
ERRHAN	004042	61-547	61-576	61-589	61-651	61-667	61-679	61-690	61-696	#61-878
		65-1213	65-1275	65-1347						
ERRMSG	010516	61-879	61-946	61-950	#64-1096					
ERROR	= 104000	#61-432								
ERRVEC	= 000004	#61-432	61-472	61-472	61-472	61-492	61-493	61-505		
ER1MSG	007257	61-959	#63-1071							
ER2MSG	007223	61-960	#63-1070							
ER3MSG	007315	61-961	#63-1072							
ER4MSG	007373	61-962	#63-1074							
EXCADD	002676	61-569	61-578	61-591	61-622	#61-631				
FILBUF	004464	61-810	61-817	61-828	61-835	61-848	61-853	61-897	61-905	61-927
		61-931	#61-977	65-1161	65-1175	65-1323	65-1334	65-1337		
FILTAB	011544	65-1205	#65-1228							
FINISH	012324	*61-822	65-1118	65-1154	#65-1368					
FIRSTA	002674	61-570	61-577	61-590	61-594	61-608	#61-630			
FIRSTB	001114	#61-464	61-507	61-734						
FLAG	012310	*65-1149	65-1301	#65-1362						
GNS	******	61-449	61-449	61-473	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043
		61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043
		61-1043	61-1043	61-1043	61-1043	61-1043				
GTSWR	= 104405	61-473	#61-1043							
HOLCK1	011226	61-781	#65-1141							
HOLCK2	011240	61-788	#65-1144							

SYMBOL	VALUE	REFERENCES								
HT	= 000011	#61-432	61-829	61-836	61-840	61-854	61-928	61-932	61-977	61-979
		61-1036	61-1036	65-1162	65-1178	65-1192	65-1338			
IOTVEC	= 000020	#61-432								
LASTAD	002700	61-568	61-579	61-592	61-625	#61-632				
LF	= 000012	#61-432	61-824	61-858	61-934	61-982	61-1036	61-1036		
LOOP	002614	#61-610	61-626							
MESSAG	001112	#61-463	61-655							
MESSPT	012326	61-479	61-820	65-1196	65-1320	#65-1370				
MESS44	012330	61-479	#65-1371							
MESTAB	012340	65-1370	#65-1375							
MES1	010016	61-647	#64-1094	65-1207	*65-1209	*65-1210	65-1271	*65-1273	*65-1274	65-1317
		65-1321								
MES2	010416	61-654	#64-1095	65-1329	*65-1331	*65-1332				
M PT	= 000007	#61-470	61-476							
MNORM	006700	61-540	#61-1061							
NODATA	002364	61-542	#61-557							
NOROME	= 000100	#61-442	61-546							
NOROMM	007551	61-965	#64-1079							
NOROMS	002300	61-523	#61-540							
OCADD	004632	61-827	61-834	61-852	#61-1026					
OCASC	004526	61-926	61-930	#61-996	61-1028					
OCTBUF	010006	61-487	61-996	61-1001	61-1008	61-1010	#64-1093			
OUTTAB	012110	65-1311	#65-1317							
OVER1	004104	61-886	#61-888							
OVER10	004414	61-887	#61-951							
OVER11	004406	61-938	61-944	#61-949						
OVER2	004352	61-937	#61-939							
OVER9	004402	*61-889	#61-947							
PFERR	- 000040	#61-441	61-649	61-792	61-899	61-941				
PFHEAD	007156	61-847	#63-1069							
PFMSG	007473	61-964	#63-1077							
PIRQ	= 177772	#61-432								
PIRQVE	000240	#61-432								
PPFVAR	003400	61-648	61-684	#61-778						
PROMP	002702	61-528	#61-643							
PR0	= 000000	#61-432								
PR1	- 000040	#61-432								
PR2	- 000100	#61-432								
PR3	- 000140	#61-432								
PR4	= 000200	#61-432								
PR5	- 000240	#61-432								
PR6	= 000300	#61-432								
PR7	= 000340	#61-432								
PS	= 177776	#61-432	61-432							
PSW	= 177776	#61-432								
PUTMES	003510	61-646	61-653	#61-801						
PWRVEC	= 000024	#61-432								
RDCHR	= 104407	61-1038	#61-1043							
RDLIN	- 104410	#61-1043								
RESREG	= 104412	61-955	#61-1043							
RESVEC	000010	#61-432								
RETURN	012306	*61-780	*61-787	*61-821	65-1150	65-1164	#65-1361			

SYMBOL CROSS REFERENCE CREF

SYMBOL	VALUE	REFERENCES
ROMCNT	002602	61-565 61-566 61-581 61-582 #61-598 61-912
ROMERR	001110	#61-462 61-546 61-575 61-588 61-649 61-666 61-678 61-689 61-695
		61-792 61-888 61-889 61-899 61-901 61-908 61-910 61-941 61-954
		*65-1197 *65-1206 *65-1270 *65-1285 65-1310
ROMFIN	001116	#61-465 61-489 61-501 *61-513 61-515 61-521 61-566 61-582 61-728
		61-939 65-1141 65-1144 65-1170 65-1233 65-1293
ROMNM	011350	65-1158 65-1159 #65-1166
ROMNUM	006662	61-1049 #61-1055
ROMPTR	006646	61-480 61-913 #61-1049
ROMTYP	011276	61-825 #65-1152
ROM1	012716	65-1375 #65-1388
ROM2	012732	65-1376 #65-1389
ROM3	012746	65-1377 #65-1390
ROM4	012762	65-1378 #65-1391
RSTART	002140	61-452 #61-521 61-530
R6	-%000006	#61-432 *61-492 *61-504 *61-505 61-801 61-860 61-863 61-925 61-929
		61-977 61-986 *61-1026 *61-1029 65-1150 65-1154 65-1164
R7	%000007	#61-432
SAVREG	= 104411	61-878 #61-1043
SCOPE	= 000004	#61-432
SEQBUF	012262	65-1116 65-1228 65-1253 65-1303 65-1305 65-1307 65-1308 65-1319 #65-1351
SEQERR	- 000200	#61-443 65-1285 65-1310
SEQMSG	007627	61-966 #64-1081
SEQTST	011416	61-529 #65-1192
STACK	- 001100	#61-432 61-472
START	001400	61-449 00400 #61-432 61-432
SW09	- 001000	#61-432 61-432
SW1	- 000002	#61-432
SW10	= 002000	#61-432
SW11	- 004000	#61-432
SW12	010000	#61-432
SW13	= 020000	#61-432
SW14	- 040000	#61-432
SW15	= 100000	#61-432
SW2	= 000004	#61-432
SW3	- 000010	#61-432
SW4	000020	#61-432
SW5	000040	#61-432

SYMBOL	VALUE	REFERENCES
SW6	= 000100	#61-432
SW7	= 000200	#61-432
SW8	= 000400	#61-432
SW9	= 001000	#61-432
TBITVE	= 000014	#61-432
TEMP	004630	61-1002 61-1004 61-1012 61-1014 61-1016 #61-1019
TESTAD	003374	61-507 61-508 61-510 61-517 61-519 61-730 61-731 61-732 61-734
		61-739 61-744 61-745 61-746 61-748 61-749 61-749 61-750 61-761
		#61-765
TIMES	004040	61-490 61-802 61-805 #61-865
TKVEC	= 000060	#61-432
TPVEC	= 000064	#61-432
TRAPVE	= 000034	#61-432 61-472 61-472
TRTVEC	= 000014	#61-432
TYPE	- 104401	61-473 61-530 61-530 61-540 61-861 61-883 61-1036 61-1038 61-1038
		61-1038 61-1038 61-1038 61-1038 61-1038 61-1038 61-1038 61-1039 #61-1043
TYPOC	= 104402	61-1038 #61-1043
TYPON	= 104404	#61-1043
TYPOS	= 104403	#61-1043
$APTHD	001122	61-468 #61-468
$ASTAT	= ******	61-1037 61-1037
$ATYC	005240	61-1037 #61-1037
$ATY1	005214	61-945 #61-1037
$ATY3	005222	61-949 61-1036 #61-1037
$ATY4	005232	#61-1037
$AUTOB	001100	#61-457 *61-473 61-1038 61-1038 61-1038
$BASE	001212	#61-469 61-685
$CDW1	001216	#61-469 61-691
$CDW2	001220	#61-469
$CHARC	005174	*61-1036 *61-1036 61-1036 *61-1036 #61-1036
$CKSWR	005466	#61-1038 61-1043 61-1043
$CMTAG	- ******	61-472 61-472
$CNTLG	006213	61-1038 #61-1038
$CNTLU	006206	61-1038 #61-1038
$CPUOP	001164	#61-469
$CRLF	005211	61-1036 61-1036 61-1036 #61-1036 61-1038 61-1038 61-1038
$DDW0	001222	#61-469 61-659 61-671
$DDW1	001224	#61-469
$DDW10	001246	#61-469
$DDW11	001250	#61-469
$DDW12	001252	#61-469
$DDW13	001254	#61-469
$DDW14	001256	#61-469
$DDW15	001260	#61-469
$DDW2	001226	#61-469
$DDW3	001230	#61-469
$DDW4	001232	#61-469
$DDW5	001234	#61-469
$DDW6	001236	#61-469
$DDW7	001240	#61-469
$DDW8	001242	#61-469
$DDW9	001244	#61-469

SYMBOL	VALUE	REFERENCES							
$DEVCT	001146	#61-469							
$DEVM	001214	#61-469							
$DOAGN	002256	61-530	61-530	#61-530					
$ENDAD	002246	61-454	61-473	#61-530					
$ENDCT	002222	61-472	#61-530						
$ENDMG	002265	61-530	#61-530						
$ENULL	002262	61-530	#61-530						
$ENV	001156	#61-469	61-473	61-936	61-1036	61-1037	61-1037		
$ENVM	001157	#61-469	61-472	61-643	61-1036	61-1036	61-1037		
$EOP	002176	61-524	61-527	#61-530					
$EOPCT	002214	*61-472	#61-530	61-530					
$ETABL	001156	#61-469							
$ETEND	001262	61-468	#61-469						
$FATAL	001140	#61-469	*61-1037						
$FFLG	005460	*61-1037	*61-1037	61-1037	*61-1037	#61-1037			
$FILLC	005206	61-1036	61-1036	61-1036	#61-1036				
$FILLS	005205	61-1036	61-1036	#61-1036					
$GET42	002236	#61-530							
$GTSWR	005536	#61-1038	61-1043	61-1043					
$HD	= 000003	61-430	61-430	61-430					
$HIBTS	001122	#61-468							
$INTAG	001101	#61-458	61-1038	61-1038	61-1038				
$LF	005212	61-1036	61-1036	#61-1036	61-1038	61-1038	61-1038		
$LFLG	005457	*61-1037	#61-1037						
$MADR1	001170	#61-469							
$MADR2	001174	#61-469							
$MADR3	001200	#61-469							
$MADR4	001204	#61-469							
$MAIL	001136	61-468	61-468	#61-469	61-472	61-473	61-1036		
$MAMS1	001166	#61-469							
$MAMS2	001172	#61-469							
$MAMS3	001176	#61-469							
$MAMS4	001202	#61-469							
$MBADR	001124	#61-468							
$MFLG	005456	*61-1037	61-1037	*61-1037	#61-1037				
$MNEW	006231	61-1038	#61-1038						
$MSGAD	001152	#61-469	*61-1037	61-1037					
$MSGLG	001154	#61-469	*61-1037						
$MSGTY	001136	#61-469	61-1037	*61-1037	61-1037	*61-1037			
$MSWR	006220	61-1038	#61-1038						
$MTYP1	001167	#61-469							
$MTYP2	001173	#61-469							
$MTYP3	001177	#61-469							
$MTYP4	001203	#61-469							
$NULL	005204	61-1036	61-1036	61-1036	#61-1036				
$OCNT	006464	*61-1039	*61-1039	#61-1039					
$OMODE	006466	*61-1039	*61-1039	61-1039	*61-1039	*61-1039	#61-1039		
$PASS	001144	#61-469	*61-472	*61-472	61-526	*61-530	*61-530	61-530	61-530
$PASTM	001130	#61-468							
$QUES	005210	61-1036	61-1036	#61-1036	61-1038	61-1038	61-1038	61-1038	
$RDCHR	005750	#61-1038	61-1043	61-1043					
$RDDEC	******	61-1043							

SYMBOL		VALUE	REFERENCES								
$RDLIN		006070	#61-1038	61-1043	61-1043						
$RDOCT	=	******	61-1043								
$RDSZ	=	000010	#61-1038	61-1038							
$RESRE		006526	#61-1040	61-1043							
$RTNAD		002260	#61-530								
$R2A	=	******	61-1043								
$SAVRE		006470	#61-1040	61-1043	61-1043						
$SETUP	=	000124	#61-450	61-450	#61-450	61-450	#61-450	61-450	#61-450	61-472	61-472
			61-472	61-472	61-472	61-472	61-472	61-472	61-472	61-472	61-472
			61-472	61-473	61-473	61-473	61-530	61-530	61-1038	61-1038	
$STUP	-	177777	#61-450	#61-450	61-450	#61-450	#61-450	61-450	#61-450	#61-450	61-450
$SVPC	=	000210	#61-454	61-454							
$SWR	=	160000	61-430	#61-430	61-431	61-431	61-431	61-431	61-431	61-431	61-431
			61-431	61-472	61-530	61-530	61-530	61-530	61-530		
$SWREG		001160	#61-469	61-472							
$TESTN		001142	#61-469								
$TKB		005464	61-1036	61-1036	61-1038	61-1038	#61-1038	61-1038	61-1038	61-1038	61-1038
$TKS		005462	61-1036	61-1036	61-1038	61-1038	#61-1038	61-1038	61-1038	61-1038	61-1038
			61-1038								
$TN	=	000001	61-430	#61-430							
$TPB		005202	61-1036	61-1036	61-1036	#61-1036					
$TPFLG		005207	61-1036	61-1036	61-1036	#61-1036					
$TPS		005200	61-1036	61-1036	61-1036	#61-1036					
$TRAP		006564	61-472	#61-1043							
$TRAP2		006606	#61-1043	61-1043							
$TRP	-	000013	#61-1043	61-1043	61-1043	61-1043	61-1043	#61-1043	61-1043	61-1043	61-1043
			61-1043	#61-1043	61-1043	61-1043	61-1043	61-1043	#61-1043	61-1043	61-1043
			61-1043	61-1043	#61-1043	61-1043	61-1043	61-1043	61-1043	#61-1043	61-1043
			61-1043	61-1043	61-1043	#61-1043	61-1043	61-1043	61-1043	61-1043	#61-1043
			61-1043	61-1043	61-1043	61-1043	#61-1043	61-1043	61-1043	61-1043	61-1043
			#61-1043	61-1043	61-1043	61-1043	61-1043	#61-1043			
$TRPAD		006620	61-1043	#61-1043							
$TSTM		001126	#61-468								
$TTYIN		006176	61-1038	61-1038	61-1038	#61-1038					
$TYPBN	=	******	61-1043								
$TYPDS	-	******	61-1043								
$TYPE		004646	#61-1036	61-1037	61-1043	61-1043					
$TYPEC		005060	61-1036	61-1036	61-1036	#61-1036	61-1036	61-1038			
$TYPEX		005176	61-1036	61-1036	#61-1036						
$TYPOC		006266	#61-1039	61-1043	61-1043						
$TYPON		006302	61-1039	#61-1039	61-1043						
$TYPOS		006242	#61-1039	61-1043							
$UNIT		001150	#61-469								
$UNITM		001132	#61-468								
$USWR		001162	#61-469								
$VECT1		001206	#61-469								
$VECT2		001210	#61-469								
$$GET4	-	000000	#61-530	61-530							
$OFILL		006465	*61-1039	*61-1039	61-1039	#61-1039					
.$ASTA		******	61-1037	61-1037							
.$X		001122	#61-468	61-468							

MACRO NAME	REFERENCES									
COMMEN	#18-1530	#61-432								
ENDCOM	#18-1542	#61-432								
ESCAPE	#20-1658	#61-432								
GETPRI	#13-1282	#61-432								
GETSWR	#22-1729	#61-432	#61-473	#61-473						
MULT	#43-4409	#61-432								
NEWTST	#19-1589	#61-432								
POP	#26-2107	#61-427	#61-432	61-1037	61-1037	61-1040	61-1041			
PUSH	#26-2099	#61-427	#61-432	61-1037	61-1037	61-1037	61-1040	61-1042		
REPORT	#56-5368	#61-427	#61-432							
SETPRI	#12-1250	#61-432								
SETTRA	#61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043
	61-1043									
SETUP	#14-1306	#61-432	#61-472							
SKIP	#21-1692	#61-432								
SLASH	#17-1482	#61-432								
SPACE	#61-420	#61-432								
STARS	#16-1451	#61-426	#61-432	61-433	61-435	61-454	61-468	61-468	61-468	61-469
	61-530	61-1036	61-1037	61-1038	61-1038	61-1038	61-1038	61-1039	61-1040	61-1043
SWRSU	#15-1420	#61-432	#61-472	#61-472						
TRMTRP	#61-1043									
TYPBIN	#25-2043	#61-432								
TYPDEC	#25-2013	#61-432								
TYPNAM	#23-1783	#61-427	#61-432	#61-473						
TYPNUM	#25-1980	#61-432								
TYPOCS	#25-1933	#61-432								
TYPOCT	#25-1896	#61-432	#61-1038							
TYPTXT	#24-1850	#61-432								
$$ESCA	#20-1671	#61-432								
$$NEWT	#19-1625	#61-432								
$$SET	#61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043	61-1043
	61-1043									
$$SETM	#61-472	61-472								
$$SKIP	#21-1705	#61-432								
.EQUAT	#4-180	#61-426	61-432							
.EQUIV	#61-416	#61-429	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432
	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432	#61-432
	#61-432	#61-432	#61-432	#61-432	#61-432					
.HEADE	#2-54	#61-427	61-430							
.KT11	#5-323									
.SETUP	#10-1184	#61-425	#61-450							
.SWRHI	#3-96	#61-427	61-431							
.SWRLO	#61-431									
.$ACT1	#52-4977	#61-427	#61-454							
.$APTB	#53-5021	#61-426	61-469							
.$APTH	#54-5277	#61-426	61-468							
.$APTY	#57-5452	#61-426	#61-1037							
.$ASTA	#55-5323	#61-425								
.$CATC	#7-909	#61-425	61-449							
.$CMTA	#9-1020	#61-425								
.$DB2D	#46-4607									
.$DB2O	#48-4730									

MACRO NAME	REFERENCES		
.$DIV	#45-4510		
.$EOP	#27-2166	#61-425	#61-530
.$ERRO	#29-2647		
.$ERRT	#30-2842		
.$MULT	#44-4447		
.$POWE	#40-4159		
.$RAND	#41-4234		
.$RDDE	#37-3830		
.$RDOC	#36-3739		
.$READ	#35-3344	#61-428	61-1038
.$R2AZ	#51-4874		
.$SAVE	#38-3905	#61-425	61-1040
.$SB2D	#47-4691		
.$SB2O	#49-4792		
.$SCOP	#28-2401		
.$SIZE	#42-4287		
.$SUPR	#50-4830		
.$TRAP	#39-4007	#61-427	#61-1043
.$TYPB	#34-3237		
.$TYPD	#33-3160		
.$TYPE	#31-2929	#61-425	61-1036
.$TYPO	#32-3064	#61-428	#61-1039
.$40CA	#8-948		
.117C	#6-502		