M9312

BOOTSTRAP TERMINATOR 8K

CZM9BC0

AH-E058C-MC

FICHE 1 OF 1

JAN 1979

digital

MADE IN USA

```
.REM !
```

IDENTIFICATION

PRODUCT CODE: AC-E057C-MC

PRODUCT NAME: CZM9BC0 M9312 BOOT TERMR 8K

DATE: JUNE, 1978

MAINTAINER: DIAGNOSTIC GROUP

HISTORY SECTION

CZM9BA0 WAS RELEASED MARCH, 1978.
CZM9BB0 WAS RELEASED JUNE, 1978.
CZM9BC0 WAS RELEASED JANUARY, 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. ILLEGAL PLACEMENT OF CONTINUATION ROMS ARE REPORTED IN ERROR MESSAGES. A DUPLICATE DEVICE ROM IS ALSO REPORTED IN AN ERROR MESSAGE.

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

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.

.REM !

1.0 ABSTRACT

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

2.0 REQUIREMENTS

2.1 HARDWARE

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

2.2 SOFTWARE

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

3.0 LOADING AND STARTING PROCEDURES

3.1 LOAD THE PROGRAM BY ANY OF THE STANDARD PROCEDURES FOR ABSOLUTE PROGRAM FORMATS.

3.2 STARTING ADDRESS

200- DO CRC VERIFICATION AND SIZING

3.3 RESTART ADDRESS

204- RESTART WITHOUT SIZING

3.4 SPECIAL ENVIRONMENTS

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

4.0 OPERATING PROCEDURES

4.1 OPERATIONAL SWITCH SETTINGS

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

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

4.2 EXECUTION TIMES

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

4.3 APT PROCEDURES

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

$ENV:	DON'T CARE
$ENVM:	200 OR 240
$SWREG:	DON'T CARE
$USWR:	NOT USED
$CPUOP:	NOT USED
$MAMS1:	NOT USED
$MTYP1:	NOT USED
$MADR1:	NOT USED
$MAMS2:	NOT USED
$MTYP2:	NOT USED
$MADR2:	NOT USED
$MAMS3:	NOT USED
$MTYP3:	NOT USED
$MADR3:	NOT USED
$MAMS4:	NOT USED
$MAMS4:	NOT USED
$MTYP4:	NOT USED
$MADR4:	NOT USED
$VECT1:	NOT USED
$VECT2:	NOT USED
$BASE	PSEUDO POWER-FAIL VECTOR ADDRESS
$DEVM:	NOT USED
$CDW1:	CONTENTS OF ADDRESS IN $BASE
$CDW2:	NOT USED
$DDW0:	DEVICE CODES EXPECTED
	FIRST LETTER/SECOND LETTER
DDW15:	

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	AO	248F1
04160	BO	233F1
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 DEFERMENT.

6.0 RELIABILITY//AVAILABILITY/SERVICEABILITY

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

CRC ERROR IN ROM EXX

THE FOLLOWING ERROR MESSAGES WERE ADDED IN REV 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 WILL BE IF THE PSEUDO POWER-FAIL VECTOR ADDRESS IN THE ETABLE DOES NOT MATCH THE ADDRESS DETERMINED BY THE PROGRAM. TO BE IN THE BOARDS' SWITCHES. THE MESSAGE WILL BE:

   POWER-FAIL VECTOR
   EXPECTED RECEIVED
   -------- --------

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

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

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

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
                                .TITLE  CZM9BC0 M9312 BOOT TERMR 8K
                                ;*COPYRIGHT (C) 1978
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY BARRY G. IRRGANG
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
000001                          $TN=1
160000                          $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT 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 *** 1100 ***
001100                          STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

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

```
(1)  000040  PR1=  40  ;;PRIORITY LEVEL 1
(1)  000100  PR2=  100  ;;PRIORITY LEVEL 2
(1)  000140  PR3=  140  ;;PRIORITY LEVEL 3
(1)  000200  PR4=  200  ;;PRIORITY LEVEL 4
(1)  000240  PR5=  240  ;;PRIORITY LEVEL 5
(1)  000300  PR6=  300  ;;PRIORITY LEVEL 6
(1)  000340  PR7=  340  ;;PRIORITY LEVEL 7
(1)
(1)  ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
(1)  100000  SW15=  100000
(1)  040000  SW14=  40000
(1)  020000  SW13=  20000
(1)  010000  SW12=  10000
(1)  004000  SW11=  4000
(1)  002000  SW10=  2000
(1)  001000  SW09=  1000
(1)  000400  SW08=  400
(1)  000200  SW07=  200
(1)  000100  SW06=  100
(1)  000040  SW05=  40
(1)  000020  SW04=  20
(1)  000010  SW03=  10
(1)  000004  SW02=  4
(1)  000002  SW01=  2
(1)  000001  SW00=  1
(1)  .EQUIV  SW09,SW9
(1)  .EQUIV  SW08,SW8
(1)  .EQUIV  SW07,SW7
(1)  .EQUIV  SW06,SW6
(1)  .EQUIV  SW05,SW5
(1)  .EQUIV  SW04,SW4
(1)  .EQUIV  SW03,SW3
(1)  .EQUIV  SW02,SW2
(1)  .EQUIV  SW01,SW1
(1)  .EQUIV  SW00,SW0
(1)
(1)  ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)  100000  BIT15=  100000
(1)  040000  BIT14=  40000
(1)  020000  BIT13=  20000
(1)  010000  BIT12=  10000
(1)  004000  BIT11=  4000
(1)  002000  BIT10=  2000
(1)  001000  BIT09=  1000
(1)  000400  BIT08=  400
(1)  000200  BIT07=  200
(1)  000100  BIT06=  100
(1)  000040  BIT05=  40
(1)  000020  BIT04=  20
(1)  000010  BIT03=  10
(1)  000004  BIT02=  4
(1)  000002  BIT01=  2
(1)  000001  BIT00=  1
(1)  .EQUIV  BIT09,BIT9
(1)  .EQUIV  BIT08,BIT8
(1)  .EQUIV  BIT07,BIT7
```

```
(1)                             .EQUIV  BIT06,BIT6
(1)                             .EQUIV  BIT05,BIT5
(1)                             .EQUIV  BIT04,BIT4
(1)                             .EQUIV  BIT03,BIT3
(1)                             .EQUIV  BIT02,BIT2
(1)                             .EQUIV  BIT01,BIT1
(1)                             .EQUIV  BIT00,BIT0
(1)
(1)                             ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
(1)         000004              ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)         000010              RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)         000014              TBITVEC=14              ;;''T'' BIT
(1)         000014              TRTVEC= 14              ;;TRACE TRAP
(1)         000014              BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)         000020              IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)         000024              PWRVEC= 24              ;;POWER FAIL
(1)         000030              EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)         000034              TRAPVEC=34              ;;''TRAP'' TRAP
(1)         000060              TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)         000064              TPVEC=  64              ;;TTY PRINTER VECTOR
(1)         000240              PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
 5948                           ;;*************************************************************
 5949                           ;*                  PROGRAM EQUATES
 5950                           ;;*************************************************************
 5951         000001                     APTER1 =  1
 5952         000002                     APTER2 =  2
 5953         000004                     APTER3 =  4
 5954         000010                     APTER4 = 10
 5955         000020                     CRCERR = 20
 5956         000040                     PFERR  = 40
 5957         000100                     NOROME =100
 5958         000200                     SEQERR =200
 5959         000400                     CONONE =400             ;+
 5960         001000                     CONTWO=1000             ;+
 5961         002000                     DUPERR=2000             ;+
 5962
 5963         000000                     .=0
 5964                            .SBTTL  TRAP CATCHER
 (1)
 (1)         000000                     .=0
 (1)                            ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ''.+2,HALT''
 (1)                            ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
 (1)                            ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
 (1)         000174             .=174
 (1)  000174 000000             DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
 (1)  000176 000000             SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
 (1)                            .SBTTL  STARTING ADDRESS(ES)
 (1)  000200 000137  001400             JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
 5966        000204             .=204
 5967  000204 000167  001656             JMP     RSTART
 5968
 5969                           .SBTTL  ACT11 HOOKS
 (1)
 (2)                            ;;*************************************************************
 (1)                            ;HOOKS REQUIRED BY ACT11
 (1)         000210                     $SVPC=.                 ;SAVE PC
```

```
(1)             000046                          .=46
(1)  000046     002174                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)             000052                          .=52
(1)  000052     000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
(1)             000210                          .=$SVPC                 ;; RESTORE PC
5970
5971            001100                          .=1100
5972  001100       000                  $AUTOB: .BYTE   0
5973  001101       000                  $INTAG: .BYTE   0
5974  001102    000000                          .WORD   0
5975  001104    177570                  SWR:    .WORD   DSWR
5976  001106    177570                  DISPLAY: .WORD  DDISP
5977  001110    000000                  ROMERR: 0
5978  001112    000000                  MESSAG: 0
5979  001114    173000                  FIRSTB: 173000
5980  001116    000000                  ROMFIN: 0
5981  001120    000000                  ERRCNT: 0
5982
5983                                    .SBTTL  APT PARAMETER BLOCK
(1)
(2)                                     ;;******************************************************************
(1)                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                     ;;******************************************************************
(1)             001122                          .$X=.   ;;SAVE CURRENT LOCATION
(1)             000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1)  000024     000200                          200     ;;FOR APT START UP
(1)             000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1)  000044     001122                          $APTHDR ;;POINT TO APT HEADER BLOCK
(1)             001122                          .=.$X   ;;RESET LOCATION COUNTER
(2)                                     ;;******************************************************************
(1)                                     ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)                                     ;INTERFACE SPEC.
(1)
(1)  001122                             $APTHD:
(1)  001122     000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1)  001124     001136                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1)  001126     000002                  $TSTM:  .WORD   2.      ;;RUN TIM OF LONGEST TEST
(1)  001130     000002                  $PASTM: .WORD   2.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1)  001132     000000                  $UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1)  001134     000052                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
5984                                    .SBTTL  APT MAILBOX-ETABLE
(1)
(2)                                     ;;******************************************************************
(1)                                     .EVEN
(1)  001136                             $MAIL:                  ;;APT MAILBOX
(1)  001136     000000                  $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
(1)  001140     000000                  $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
(1)  001142     000000                  $TESTN: .WORD   ATESTN  ;;TEST NUMBER
(1)  001144     000000                  $PASS:  .WORD   APASS   ;;PASS COUNT
(1)  001146     000000                  $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
(1)  001150     000000                  $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
(1)  001152     000000                  $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(1)  001154     000000                  $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(1)  001156                             $ETABLE:                ;;APT ENVIRONMENT TABLE
(1)  001156        000                  $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(1)  001157        000                  $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
```

```
(1)  001160  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(1)  001162  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(1)  001164  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(1)                                ;*                      BITS 15-11=CPU TYPE
(1)                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(1)                                ;*                              11/70=06,PDQ=07,Q=10
(1)                                ;*                      BIT 10=REAL TIME CLOCK
(1)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
(1)                                ;*                      BIT  8=MEMORY MANAGEMENT
(1)  001166      000               $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(1)  001167      000               $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(1)                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(1)                                ;*                              900 NSEC CORE=001
(1)                                ;*                              300 NSEC BIPOLAR=002
(1)                                ;*                              500 NSEC MOS=003
(1)  001170  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(1)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
(1)  001172      000               $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(1)  001173      000               $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(1)  001174  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(1)  001176      000               $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(1)  001177      000               $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(1)  001200  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(1)  001202      000               $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(1)  001203      000               $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(1)  001204  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(1)  001206  000000                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(1)  001210  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(1)  001212  000000                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(1)  001214  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(1)  001216  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(1)  001220  000000                $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
(1)  001222  000000                $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
(1)  001224  000000                $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
(1)  001226  000000                $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
(1)  001230  000000                $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
(1)  001232  000000                $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
(1)  001234  000000                $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
(1)  001236  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
(1)  001240  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
(1)  001242  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
(1)  001244  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
(1)  001246  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
(1)  001250  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
(1)  001252  000000                $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
(1)  001254  000000                $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
(1)  001256  000000                $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
(1)  001260  000000                $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
(1)
(1)
(1)  001262                        $ETEND:
(1)
5985            001400                      .=1400
5986  001400                    START:
 (1)                              .SBTTL  INITIALIZE THE COMMON TAGS
 (1)  001400  012706  001100              MOV     #STACK,SP       ;;SETUP THE STACK POINTER
```

```
(1)                                     ;;INITIALIZE A FEW VECTORS
(1)  001404  012737  006442  000034             MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1)  001412  012737  000340  000036             MOV     #340,@#TRAPVEC+2;LEVEL 7
(1)  001420  005067  177520                     CLR     $PASS            ;;CLEAR THE PASS COUNT
(1)  001424  016767  000520  000510             MOV     $ENDCT,$EOPCT    ;;SETUP END-OF-PROGRAM COUNTER
(2)                                     ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)                                     ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2)  001432  013746  000004                     MOV     @#ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
(2)  001436  012737  001472  000004             MOV     #64$,@#ERRVEC    ;;SET UP ERROR VECTOR
(2)  001444  012767  177570  177432             MOV     #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
(2)  001452  012767  177570  177426             MOV     #DDISP,DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
(2)  001460  022777  177777  177416             CMP     #-1,@SWR         ;;TRY TO REFERENCE HARDWARE SWR
(2)  001466  001012                             BNE     66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                                                                      ;;AND  THE HARDWARE SWR IS NOT = -1
(2)  001470  000403                             BR      65$              ;;BRANCH IF NO TIMEOUT
(2)  001472  012716  001500          64$:       MOV     #65$,(SP)        ;;SET UP FOR TRAP RETURN
(2)  001476  000002                             RTI
(2)  001500  012767  000176  177376  65$:       MOV     #SWREG,SWR       ;;POINT TO SOFTWARE SWR
(2)  001506  012767  000174  177372             MOV     #DISPREG,DISPLAY
(2)  001514  012637  000004          66$:       MOV     (SP)+,@#ERRVEC   ;;RESTORE ERROR VECTOR
(1)
(2)  001520  005067  177420                     CLR     $PASS            ;;CLEAR PASS COUNT
(2)  001524  132767  000200  177425             BITB    #APTSIZE,$ENVM   ;;TEST USER SIZE UNDER APT
(2)  001532  001403                             BEQ     67$              ;;YES,USE NON-APT SWITCH
(2)  001534  012767  001160  177342             MOV     #$SWREG,SWR      ;;NO,USE APT SWITCH REGISTER
(2)  001542                             67$:
5987                                    .SBTTL  TYPE PROGRAM NAME
(1)                                     ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1)  001542  005227  177777                     INC     #-1              ;;FIRST TIME?
(1)  001546  001046                             BNE     68$              ;;BRANCH IF NO
(1)  001550  022737  002174  000042             CMP     #$ENDAD,@#42     ;;ACT-11?
(1)  001556  001442                             BEQ     68$              ;;BRANCH IF YES
(1)  001560  104401  001626                     TYPE    ,69$             ;;TYPE ASCIZ STRING
(2)                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
(2)  001564  005737  000042                     TST     @#42             ;;ARE WE RUNNING UNDER XXDP/ACT?
(2)  001570  001012                             BNE     70$              ;;BRANCH IF YES
(2)  001572  126727  177360  000001             CMPB    $ENV,#1          ;;ARE WE RUNNING UNDER APT?
(2)  001600  001406                             BEQ     70$              ;;BRANCH IF YES
(2)  001602  026727  177276  000176             CMP     SWR,#SWREG       ;;SOFTWARE SWITCH REG SELECTED?
(2)  001610  001005                             BNE     71$              ;;BRANCH IF NO
(2)  001612  104405                             GTSWR                    ;;GET SOFT-SWR SETTINGS
(2)  001614  000403                             BR      71$
(2)  001616  112767  000001  177254  70$:       MOVB    #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
(2)  001624                             71$:
(1)  001624  000417                             BR      68$              ;;GET OVER THE ASCIZ
(1)
(1)  001664                             ;;69$:  .ASCIZ  <CRLF>*CZM9BC0 M9312 BOOT TERMR 8K*<CRLF>
5988  001664  012700  007540          68$:
5989  001670  005020                             MOV     #BUF1,          R0              ;CLR SIZING BUFFERS
5990  001672  020027  007624          8$:        CLR     (R0)+
5991  001676  002774                             CMP     R0,             #OCTBUF         ;HAVE WE CLEARED THE WHOLE
5992  001700  005037  001116                     BLT     8$                              ;BUFFER,NO,GO BACK
5993  001704  005037  003766                     CLR     @#ROMFIN                        ;INITIALIZE ROM FOUND INDICATORS
5994  001710  004767  007020                     CLR     @#TIMES                         ;INITIALIZE ENTRY COUNTER FOR PUTMES SUB
5995  001714  013746  000004                     JSR     PC,     CLEAR                   ;+CLEAR SOME MORE LOCATIONS
5996  001720  012737  001762  000004             MOV     @#ERRVEC,       -(R6)           ;SAVE CONTENTS OF LOCATION 4
                                                MOV     #2$,            @#ERRVEC        ;+SET UP FOR POSSIBLE TRAP
```

```
 5997  001726  122737  177777  165000          CMPB    #-1,        @#165000    ;IF CONTENTS = -1, OR TRAP
 5998  001734  001414                          BEQ     3$                      ;THEN CPU ROM NOT PLUGGED IN
 5999  001736  012700  165000                  MOV     #165000,    R0          ;GET FIRST ADDRESS OF ROM SPACE
 6000  001742  005720                  1$:     TST     (R0)+                   ;IF THIS INSTRUCTION TRAPS CPU ROM
 6001                                                                          ;NOT PLUGGED IN
 6002  001744  020027  166000                  CMP     R0,         #166000     ;HAVE WE CHECKED ALL ADDRESSES
 6003  001750  001374                          BNE     1$                      ;IF NO THEN CONTINUE LOOPING
 6004  001752  012737  000001  001116          MOV     #1,         @#ROMFIN    ;IF NO TRAPS OR LOW BYTE OF FIRST
 6005                                                                          ;ADDRESS NOT EQUAL -1 ASSUME ROM PRESENT
 6006  001760  000402                          BR      3$                      ;NO TRAP JUST RESTORE ERRUEL
 6007  001762  062706  000004          2$:     ADD     #4,         R6          ;+IF TRAP CLEAN OFF PC, PSW
 6008  001766  012637  000004          3$:     MOV     (R6)+,      @#ERRVEC    ;RESTORE ERRVEC
 6009  001772  012700  000002                  MOV     #2,         R0          ;INITIALIZE ROM FOUND INDICATOR
 6010  001776  013737  001114  003322          MOV     @#FIRSTB,   @#TESTAD    ;GET FIRST BOOT
 6011  002004  132777  000200  001310  5$:     BITB    #200,       @TESTAD     ;IF LOW BYTE HAS BIT 7 SET
 6012  002012  001411                          BEQ     7$                      ;DO NOT SET ROM FOUND INDICATOR
 6013  002014  022777  177776  001300          CMP     #-2,        @TESTAD     ;UNLESS, CONTENTS OF ADDRESS
 6014  002022  001011                          BNE     4$                      ;EQUAL TO -2.(CONTINUATION ROM)
 6015  002024  050067  010100                  BIS     R0,     CONFIN          ;+SET CONTINUATION ROM FOUND INDICATOR
 6016  002030  050067  177062                  BIS     R0,     ROMFIN          ;+SET ROM FOUND INDICATOR
 6017  002034  000404                          BR      4$                      ;+CONTINUE
 6018  002036  050037  001116          7$:     BIS     R0,         @#ROMFIN    ;SET ROM FOUND INDICATOR
 6019  002042  050067  010064                  BIS     R0,     DEVFIN          ;+SET DEVICE ROM FOUND INDICATOR
 6020  002046  062737  000200  003322  4$:     ADD     #200,       @#TESTAD    ;UPDATE TEST ADDRESS TO NEXT ROM
 6021  002054  006100                          ROL     R0                      ;UPDATE ROM FOUND INDICATOR
 6022  002056  022737  174000  003322          CMP     #174000,    @#TESTAD    ;HAVE CHECKED ALL THE ROMS
 6023  002064  001347                          BNE     5$                      ;IF NO CONTINUE CHECKING
 6024  002066  005737  001116          RSTART: TST     @#ROMFIN                ;ARE THERE ANY ROMS AVAILABLE
 6025  002072  001003                          BNE     6$                      ;IF YES GO TO ROM TEST
 6026  002074  004767  000126                  JSR     PC,         NOROMS      ;IF NO GO TEST NO ROMS
 6027  002100  000411                          BR      $EOP                    ;GO TO END OF PASS
 6028  002102  004767  000206          6$:     JSR     PC,         CHECKS      ;GO CALCULATE CHECKSUMS
 6029  002106  005737  001144                  TST     @#$PASS                 ;ARE WE ON 1ST PASS
 6030  002112  001004                          BNE     $EOP                    ;IF NO SKIP PROCESS OF PARAMETERS
 6031  002114  004767  000510                  JSR     PC,         PROMP       ;GO PROCESS ROM PARAMETERS
 6032  002120  004767  007110                  JSR     PC,         SEQTST      ;GO CHECK SEQUENCE OF DEVICE CODES
 6033                                  .SBTTL  END OF PASS ROUTINE
  (1)
  (2)                                 ;;**********************************************************************
  (1)                                 ;*INCREMENT THE PASS NUMBER ($PASS)
  (1)                                 ;*TYPE 'END PASS'
  (1)                                 ;*IF THERES A MONITOR GO TO IT
  (1)                                 ;*IF THERE ISN'T JUMP TO RSTART
  (1)                                 ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE 'END OF PASS' LOCATION
  (1)                                 ;*$ENDMG CAN BE CHANGED TO 7.
  (1)
  (1)   002124                        $EOP:
  (2)   002124  000240                        NOP
  (1)   002126  005267  177012                INC     $PASS               ;;INCREMENT THE PASS NUMBER
  (1)   002132  042767  100000  177004        BIC     #100000,$PASS       ;;DON'T ALLOW A NEG. NUMBER
  (1)   002140  005327                        DEC     (PC)+               ;;LOOP?
  (1)   002142  000001                $EOPCT: .WORD   1
  (1)   002144  003017                        BGT     $DOAGN              ;;YES
  (1)   002146  012737                        MOV     (PC)+,@(PC)+        ;;RESTORE COUNTER
  (1)   002150  000001                $ENDCT: .WORD   1
  (1)   002152  002142                        $EOPCT
```

```
(1)  002154  104401  002213            TYPE    ,$ENDMG          ;;TYPE 'END PASS'
(1)  002160  104401  002210            TYPE    ,$ENULL          ;;TYPE A NULL CHARACTER
(1)  002164  013700  000042  $GET42:   MOV     @#42,R0          ;;GET MONITOR ADDRESS
(1)  002170  001405                    BEQ     $DOAGN           ;;BRANCH IF NO MONITOR
(1)  002172  000005                    RESET                    ;;CLEAR THE WORLD
(1)  002174  004710          $ENDAD:   JSR     PC,(R0)          ;;GO TO MONITOR
(1)  002176  000240                    NOP                      ;;SAVE ROOM
(1)  002200  000240                    NOP                      ;;FOR
(1)  002202  000240                    NOP                      ;;ACT11
(1)  002204             $DOAGN:
(1)  002204  000137                    JMP     @(PC)+           ;;RETURN
(1)  002206  002066          $RTNAD:   .WORD   RSTART
(1)  002210    377    377    000  $ENULL:  .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
(1)  002213    015  042412  042116  $ENDMG:  .ASCIZ  <15><12>/END PASS/
(1)  002220  050040  051501  000123
6034
6035
6036
6037                            ;NO ROMS TEST
6038                            ;THIS ROUTINE IS CALLED ONLY IF NO ROMS WERE FOUND
6039                            ;DURING SIZING. IT DOES A READ OF ALL BOOTSTRAP ROM
6040                            ;ADDRESSES AND COMPARES THE CONTENTS TO A KNOWN
6041                            ;EXPECTED VALUE.
6042
6043  002226  104401  006542  NOROMS:  TYPE , MNORM                    ;- TYPE MESSAGE THAT NOROMS FOUND
6044  002232  012703  172000          MOV #172000,        R3         ; GET FIRST ADDRESS TO BE READ
6045  002236  013704  002312  1$:     MOV     @#NODATA,   R4         ;GET ADDRESS OF EXPECTED VALUE
6046  002242  121304                  CMPB    (R3),       R4         ;+DOES RECIEVED VALUE EQUAL EXPECTED
6047  002244  001406                  BEQ     2$                     ;IF YES CONTINUE TESTING
6048  002246  011304                  MOV     (R3),   R4             ;+GET BAD VALUE
6049  002250  052737  000100  001110  BIS     #NOROME,    @#ROMERR   ;IF NO SET ERROR INDICATER
6050  002256  004767  001506          JSR     PC,         ERRHAN     ;REPORT ERROR
6051  002262  062703  000002  2$:     ADD     #2,     R3             ;+GET TO NEXT EVEN ADDRESS
6052  002266  022703  173024          CMP     #173024,    R3         ;+AT A VECTOR ADDRESS?
6053  002272  001773                  BEQ     2$                     ;+BRANCH IF YES
6054  002274  022703  173224          CMP     #173224,    R3         ;+AT A VECTOR ADDRESS?
6055  002300  001770                  BEQ     2$                     ;+BRANCH IF YES
6056  002302  022703  174000          CMP     #174000,    R3         ;HAVE ALL ADDRESSES BEEN TESTED
6057  002306  003353                  BGT     1$                     ;IF NO GO TEST THIS ADDRESS
6058  002310  000207                  RTS     PC
6059
6060  002312  000777  NODATA:  .WORD   777                      ;+A HOLE LOOKS LIKE XXX777
6061
6062
6063
6064
6065                            ;CHECKSUM ROMS MODULE
6066                            ;THIS ROUTINE DOES A CHECKSUM OF ALL ROMS PRESENT
6067
6068  002314  012737  000001  002530  CHECKS:  MOV     #1,         @#ROMCNT   ;INITIALIZE RON COUNTER
6069  002322  033737  002530  001116          BIT     @#ROMCNT,   @#ROMFIN   ;IS DIAGNOSTIC ROM PRESENT
6070  002330  001424                          BEQ     1$                     ;IF NO GO CHECKSUM BOOT ROMS
6071  002332  012737  165775  002626          MOV     #165775,    @#LASTAD   ;SET UP LAST ADDRESS TO BE SUMMED
6072  002340  012737  000000  002624          MOV     #0,         @#EXCADD   ;SET UP EXCEPTION ADDRESS
6073  002346  012737  165000  002622          MOV     #165000,    @#FIRSTA   ;SET UP FIRST ADDRESS TO BE SUMMED
6074  002354  004767  000152                  JSR     PC,         CALSUM     ;GO CALCULATE CHECKSUM
```

```
6075  002360  013703  165776          MOV   @#165776,   R3        ;GET EXPECTED CHECKSUM
6076  002364  020304                  CMP   R3,         R4        ;COMPARE CHECKSUMS
6077  002366  001405                  BEQ   1$                    ;IF CHECKSUMS COMPARE CONTINUE
6078  002370  052737  000020  001110  BIS   #CRCERR,    @#ROMERR  ;IF ERROR SET INDICATER
6079  002376  004767  001366          JSR   PC,         ERRHAN    ;REPORT ERROR
6080  002402  012737  173000  002622  1$:   MOV   #173000,   @#FIRSTA  ;SET UP FIRST ADDRESS TO BE SUMMED
6081  002410  012737  173024  002624        MOV   #173024,   @#EXCADD  ;SET UP EXCEPTION ADDRESS
6082  002416  012737  173175  002626        MOV   #173175,   @#LASTAD  ;SET UP LAST ADDRESS TO BE SUMMED
6083  002424  012702  173176              MOV   #173176,   R2        ;GET ADDRESS OF GOOD DATA
6084  002430  006137  002530        2$:   ROL   @#ROMCNT              ;UPDATE ROM COUNTER
6085  002434  033737  002530  001116      BIT   @#ROMCNT,  @#ROMFIN  ;IS ROM PRESENT
6086  002442  001412                  BEQ   3$                    ;IF NO, GO UPDATE TO NEXT ROM
6087  002444  004767  000062          JSR   PC,         CALSUM    ;IF YES GO CALCULATE CHECKSUM
6088  002450  011203                  MOV   (R2),       R3        ;GET EXPECTED CHECKSUM
6089  002452  020304                  CMP   R3,         R4        ;COMPARE CHECKSUMS
6090  002454  001405                  BEQ   3$                    ;IF CHECKSUMS EQUAL CONTINUE
6091  002456  052737  000020  001110  BIS   #CRCERR,    @#ROMERR  ;IF ERROR SET INDICATER
6092  002464  004767  001300          JSR   PC,         ERRHAN    ;REPORT ERROR
6093  002470  062737  000200  002622  3$:   ADD   #200,      @#FIRSTA  ;UPDATE FIRST ADDRESS
6094  002476  062737  000200  002624        ADD   #200,      @#EXCADD  ;UPDATE EXCEPTION ADDRESS
6095  002504  062737  000200  002626        ADD   #200,      @#LASTAD  ;UPDATE LAST ADDRESS
6096  002512  062702  000200              ADD   #200,      R2        ;UPDATE ADDRESS OF GOOD DATA
6097  002516  022737  174000  002622        CMP   #174000,   @#FIRSTA  ;HAVE ALL ROMS BEEN CHECKED
6098  002524  001341                  BNE   2$                    ;IF NO GO CHECKSUM NEXT ONE
6099  002526  000207                  RTS   PC                    ;IF YES RETURN
6100
6101  002530  000000          ROMCNT: 0
6102
6103
6104                          ;CALCULATE CHECKSUMS MODULE
6105                          ;THIS ROUTINE CALCULATES THE CRC16 CHECKSUM OF THE CONTENTS
6106                          ;OF EVERY LOCATION EXCEPT THE EXCEPTION ADDRESS AND LAST ADDRESS
6107                          ;OF EVERY ROM
6108
6109
6110  002532  010246          CALSUM: MOV   R2,         -(SP)     ;SAVE R2
6111  002534  013700  002622          MOV   @#FIRSTA,   R0        ;GET STARTING ADDRESS
6112  002540  005004                  CLR   R4                    ;INITIALIZE CRC WORD
6113  002542  012005          LOOP:   MOV   (R0)+,      R5        ;GET A BYTE
6114  002544  012702  000020          MOV   #16.,       R2        ;SET BYTE COUNT
6115  002550  000241          CRCLOP: CLC                         ;THE NEXT NINE LINES
6116  002552  006004                  ROR   R4                    ;DO THE MATH CALCULATIONS
6117  002554  006005                  ROR   R5
6118  002556  102006                  BVC   1$
6119  002560  012701  120001          MOV   #120001,    R1
6120  002564  040401                  BIC   R4,         R1
6121  002566  042704  120001          BIC   #120001,    R4
6122  002572  050104                  BIS   R1,         R4
6123  002574  005302          1$:     DEC   R2
6124  002576  003364                  BGT   CRCLOP
6125  002600  020037  002624          CMP   R0,         @#EXCADD  ;IS NEXT ADDRESS AN EXCEPTION ADDRESS
6126  002604  001001                  BNE   2$                    ;IF NO TEST IT
6127  002606  005720                  TST   (R0)+                 ;IF YES SKIP ADDRESS
6128  002610  020037  002626  2$:     CMP   R0,         @#LASTAD  ;HAVE ALL LOCATIONS BEEN SUMMED
6129  002614  101752                  BLOS  LOOP                  ;IF NO CONTINUE
6130  002616  012602                  MOV   (SP)+,      R2        ;RESTORE R2
```

```
 6131  002620  000207                    RTS     PC                          ;IF YES RETURN
 6132
 6133  002622  000000                FIRSTA: 0
 6134  002624  000000                EXCADD: 0
 6135  002626  000000                LASTAD: 0
 6136
 6137
 6138
 6139                                  ;PROCESS ROM PARAMETERS MODULE
 6140                                  ;THIS ROUTINE DETERMINES IF SIZING IS TO BE DONE.  IF IT IS THE
 6141                                  ;MODULE WILL GET THE PARAMETERS FROM THE DEVCOD AND PPFVAR
 6142                                  ;MODULES AND ASSEMBLE MESSAGES TO BE OUTPUT.  IF SIZING IS
 6143                                  ;NOT TO BE DONE THIS MODULE WILL TAKE THE PARAMETERS RECEIVED
 6144                                  ;AND COMPARE THEM TO THE VALUES SUPPLIED IN THE ETABLE.
 6145
 6146  002630  105737  001157    PROMP:  TSTB    @#$ENVM                 ;IF BIT IS CLEAR SIZE
 6147  002634  100424                    BMI     1$                      ;IF BIT IS SET DON'T SIZE.
 6148  002636  004767  000250            JSR     PC,     DEVCOD          ;CALL DEVICE CODE MODULE
 6149  002642  004767  000570            JSR     PC,     PUTMES          ;FORMAT PARAMETER MESSAGE
 6150  002646  007634                    MES1
 6151  002650  004767  000452            JSR     PC,     PPFVAR          ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
 6152  002654  032737  000040  001110    BIT     #PFERR, @#ROMERR        ;WAS THERE AN ERR
 6153  002662  001403                    BEQ     10$                     ;IF NO GO FORMAT MESSAGE
 6154  002664  004767  001100            JSR     PC,     ERRHAN          ;IF YES GO REPORT MESSAGE
 6155  002670  000507                    BR      9$
 6156  002672  004767  000540    10$:    JSR     PC,     PUTMES          ;FORMAT PSEUDO POWER-FAIL VECTOR
 6157  002676  010234                    MES2                            ;ADDRESS MESSAGE
 6158  002700  005237  001112            INC     @#MESSAG                ;SET MESSAGE INDICATOR
 6159  002704  000501                    BR      9$                      ;GO TO EXIT
 6160  002706  004767  000200    1$:     JSR     PC,     DEVCOD          ;CALL GET DEVICE CODE MODULE
 6161  002712  012703  007540            MOV     #BUF1,  R3              ;GET BOARD DEVICE CODES
 6162  002716  012704  001222    2$:     MOV     #$DDW0, R4              ;GET ETABLE DEVICE CODE PARAMETERS
 6163  002722  062703  000002            ADD     #2,     R3              ;GET TO BOARD DEVICE CODE
 6164  002726  012402    3$:     MOV     (R4)+,  R2              ;GET ETABLE DEVICE CODE
 6165  002730  020213                    CMP     R2,     (R3)            ;DO THE TWO DEVICE CODES COMPARE
 6166  002732  001407                    BEQ     4$                      ;IF YES GET NEXT BOARD DEVICE CODE
 6167  002734  005714                    TST     (R4)                    ;IF NO HAVE WE CHECKED THE WHOLE ETABLE
 6168  002736  001373                    BNE     3$                      ;IF WE HAVEN'T CHECK NEXT ETABLE ENTRY
 6169  002740  052737  000001  001110    BIS     #APTER1, @#ROMERR       ;IF WE HAVE THEN DEVICE CODE ON BOARD
 6170  002746  004767  001016            JSR     PC,     ERRHAN          ;DOES NOT  EXIST IN ETABLE
 6171  002752  062703  000002    4$:     ADD     #2,     R3              ;UPDATE TO NEXT ADDRESS
 6172  002756  005713                    TST     (R3)                    ;IF CONTENTS EQUALS ZERO WE'RE DONE
 6173  002760  001356                    BNE     2$                      ;IF CONTENTS NO EQUAL ZERO CONTINUE
 6174  002762  012703  001222            MOV     #$DDW0, R3              ;GET BOARD DEVICE CODES
 6175  002766  012704  007536    5$:     MOV     #BUF1-2, R4             ;GET ETABLE DEVICE CODES
 6176  002772  062704  000004    6$:     ADD     #4,     R4              ;GET BOARD DEVICE CODE
 6177  002776  021314                    CMP     (R3),   (R4)            ;DO THE TWO DEVICE CODES COMPARE
 6178  003000  001407                    BEQ     7$                      ;IF YES GET NEXT ETABLE DEVICE CODE
 6179  003002  005714                    TST     (R4)                    ;IF NO HAVE WE CHECKED ALL THE BOARD
 6180  003004  001372                    BNE     6$                      ;DEVICE CODES, IF NO CONTINUE
 6181  003006  052737  000002  001110    BIS     #APTER2, @#ROMERR       ;IF YES ETABLE DEVICE CODE NOT
 6182  003014  004767  000750            JSR     PC,     ERRHAN          ;NOT ON BOARD
 6183
 6184  003020  062703  000002    7$:     ADD     #2,     R3
 6185  003024  005713                    TST     (R3)                    ;IF CONTENTS EQUAL ZERO-DONE
 6186  003026  001357                    BNE     5$                      ;IF CONTENTS NO EQUAL ZERO-CONTINUE
```

```
003030 004767 000272                   JSR     PC,             PPFVAR          ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
003034 013704 001212                   MOV     @#$BASE,        R4              ;GET ETABLE PPFVA
003040 013703 007620                   MOV     @#BUF2,         R3              ;GET BOARD PPFVA
003044 020403                          CMP     R4,             R3              ;DO THE TWO PARAMETERS COMPARE
003046 001405                          BEQ     8$                              ;IF YES CONTINUE
003050 052737 000004 001110            BIS     #APTER3,        @#ROMERR        ;SET APT ERROR INDICATOR
003056 004767 000706                   JSR     PC,             ERRHAN          ;GO TO ERROR ROUTINE
003062 013704 001216           8$:     MOV     @#$CDW1,        R4              ;GET ETABLE DATA
003066 013703 007622                   MOV     @#BUF2+2,       R3              ;GET BOARD DATA
003072 020403                          CMP     R4,             R3              ;DO THE TWO PARAMETER COMPARE
003074 001405                          BEQ     9$                              ;IF YES THEN DONE
003076 052737 000010 001110            BIS     #APTER4,        @#ROMERR        ;SET APT ERROR INDICATOR
003104 004767 000660                   JSR     PC,             ERRHAN          ;GO TO ERROR ROUTINE
003110 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
003112 012700 007540           DEVCOD: MOV     #BUF1,          R0
003116 012701 000001                   MOV     #1,             R1              ;INITIALIZE ROM POINTER
003122 030137 001116                   BIT     R1,             @#ROMFIN        ;IS DIAGNOSTIC ROM PRESENT
003126 001411                          BEQ     1$                              ;IF NO GO TEST BOOT ROMS
003130 012737 165774 003322            MOV     #165774,        @#TESTAD        ;GET ADDRESS OF DIAG. ROM DEVICE CODE.
003136 013720 003322                   MOV     @#TESTAD,       (R0)+           ;STORE ADDRESS OF DEVICE CODE
003142 017720 000154                   MOV     @TESTAD,        (R0)+           ;STORE DEVICE CODE
003146 005037 003324                   CLR     @#DAFLAG                        ;CLR DATA TYPE FLAG
003152 013737 001114 003322    1$:     MOV     @#FIRSTB,       @#TESTAD        ;GET ADDRESS OF FIRST BOOT ROM
003160 006101                          ROL     R1                              ;UPDATE POINTER TO NEXT ROM
003162 030167 006744           3$:     BIT     R1,             DEVFIN          ;+IS DEVICE ROM PRESENT
003166 001007                          BNE     4$                              ;IF YES GO GET PARAMETERS
003170 004767 005556                   JSR     PC,     CON1                    ;+CHECK FOR CONTINUATION CHIP
003174 062737 000200 003322            ADD     #200,           @#TESTAD        ;IF NO UPDATE TEST ADDRESS
```

```
6243  003202  006101                    ROL     R1                      ;UPDATE POINTER TO NEXT ROM
6244  003204  000441                    BR      9$                      ;GO SEE IF ALL ROM CHECKED
6245  003206  005737  003324      4$:   TST     @#DAFLAG                ;IF DATAFLAG=0 THEN DATA IS DEVICE CODE
6246  003212  001010                    BNE     5$                      ;IF DATAFLAG=1 THE DATA IS OFFSET TO NEX
6247  003214  013720  003322            MOV     @#TESTAD,       (R0)+   ;STORE ADDRESS OF DEVICE CODE
6248  003220  017720  000076            MOV     @TESTAD,        (R0)+   ;STORE DEVICE CODE
6249  003224  062737  000002  003322    ADD     #2,             @#TESTAD ;UPDATE TEST ADDRESS
6250  003232  000424                    BR      8$                      ;GET OVER SOME CODE
6251  003234  013702  003322      5$:   MOV     @#TESTAD,       R2      ;SAVE OLD TEST ADDRESS
6252  003240  067737  000056  003322    ADD     @TESTAD,        @#TESTAD ;UPDATE TO NEW TEST ADDRESS
6253  003246  013703  003322            MOV     @#TESTAD,       R3      ;GET THE NEW ADDRESS
6254  003252  042702  170177            BIC     #170177,        R2      ;+SAVE ONLY BITS 7,8,9,10,11
6255  003256  042703  170177            BIC     #170177,        R3      ;+IN R3 ALSO
6256  003262  160203                    SUB     R2,             R3      ;CALCULATE DISTANCE BETWEEN  ADD
6257  003264  005703              6$:   TST     R3                      ;IS R3 EQUAL TO 0
6258  003266  001406                    BEQ     8$                      ;IF YES THEN DONE
6259  003270  162703  000200            SUB     #200,           R3      ;IF NO THEN MOVE POINTER
6260  003274  006101                    ROL     R1                      ;ONE BIT FOR EVERY 200
6261  003276  004767  005476            JSR     PC,     CON2            ;+CHECK FOR CONTINUATION CHIP
6262  003302  000770                    BR      6$
6263  003304  005137  003324      8$:   COM     @#DAFLAG                ;CHANGE DATA FLAG TO RIGHT DATA TYPE
6264  003310  023727  003322  174000  9$:   CMP     @#TESTAD,       #174000 ;HAVE WE CHECKED ALL THE ROM
6265  003316  002721                    BLT     3$                      ;IF NO CONTINUE
6266  003320  000207                    RTS     PC                      ;IF YES RETURN
6267
6268  003322  000000            TESTAD: 0
6269  003324  000000            DAFLAG: 0
6270
6271                            ;GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE
6272                            ;THIS SUBROUTINE TESTS LOCATIONS 173024 AND 173224 TO DETERMINE
6273                            ;WHICH VECTOR WILL BE USED IF POWER-FAIL OPTION ENABLED ON THE
6274                            ;BOARD.  THE OPTION MUST BE ENABLED AND AT LEAST ONE ADDRESS
6275                            ;SWITCH MUST BE "ON" OR AN ERROR WILL BE DETECTED.
6276                            ;THE DATA WILL BE RETURNED IN "BUF2" IN THE FORMAT.
6277                            ;
6278                            ;       BUF2:   PSEUDO POWER-FAIL VECTOR ADDRESS
6279                            ;               CONTENTS OF VECTOR ADDRESS
6280
6281  003326  022737  173000  173024  PPFVAR: CMP #173000,            @#173024 ;- TEST IF LOCATION 173024 SELECTED
6282  003334  001414                    BEQ     1$                      ;IF NOT THEN GO TEST LOCATION 173224
6283  003336  012767  003366  006560        MOV     #1$,    RETURN          ;+SET UP AN ALTERNATE RETURN
6284  003344  004767  005474            JSR     PC,     HOLCK1          ;+CHECK FOR A HOLE
6285  003350  012737  173024  007620    MOV     #173024,        @#BUF2  ;IF IT IS THEN STORE ADDRESS
6286  003356  013737  173024  007622    MOV     @#173024,       @#BUF2+2 ;STORE CONTENTS OF LOCATION 173024
6287  003364  000423                    BR      3$                      ;GO TO RETURN
6288  003366  022737  173000  173224  1$:   CMP     #173000,        @#173224 ;-TEST IF LOCATION 173224 SELECTED
6289  003374  001414                    BEQ     2$                      ;IF NOT THEN SET ERROR INDICATOR
6290  003376  012767  003426  006520        MOV     #2$,    RETURN          ;+SET UP AN ALTERNATE RETURN
6291  003404  004767  005446            JSR     PC,     HOLCK2          ;+CHECK FOR A HOLE
6292  003410  012737  173224  007620    MOV     #173224,        @#BUF2  ;IF IT IS THEN STORE ADDRESS
6293  003416  013737  173224  007622    MOV     @#173224,       @#BUF2+2 ;STORE CONTENTS OF VECTOR
6294  003424  000403                    BR      3$                      ;GET OVER ERROR
6295  003426  052737  000040  001110  2$:   BIS     #PFERR,         @#ROMERR ;SET ERROR INDICATOR
6296  003434  000207              3$:   RTS     PC
6297
6298
```

```
6299
6300
6301                                    ;PUT MESSAGE IN BUFFER ROUTINE
6302                                    ;THIS SUBROUTINE FORMATS THE PARAMETER AND POWER-FAIL MESSAGES.
6303
6304  003436  017604  000000          PUTMES: MOV     @(R6),          R4      ;GET MESSAGE BUFFER ADDRESS
6305  003442  005737  003766                  TST     @#TIMES                 ;IS THIS FIRST TIME THROUGH
6306  003446  001110                          BNE     3$                      ;IF NOT FIRST TIME THEN FORMAT POWER-
6307                                                                          ;FAIL MESSAGE
6308  003450  005237  003766                  INC     @#TIMES                 ;TOGGLE WATCHDOG
6309  003454  012703  007540                  MOV     #BUF1,          R3      ;GET DATA BUFFER ADDRESS
6310  003460  022713  165774                  CMP     #165774,        (R3)    ;IS DIAGNOSTIC ROM PRESENT
6311  003464  001012                          BNE     1$                      ;IF NOT DON'T FORMAT DIAG. ROM MESSAGE
6312  003466  012705  006616                  MOV     #DRHEAD,        R5      ;IF IT IS GET DIAG. ROM MESSAGE HEADER
6313  003472  004767  000714                  JSR     PC,             FILBUF  ;GO PUT HEADER IN MESSAGE BUFFER
6314  003476  000015                          CR
6315  003500  062703  000002                  ADD     #2,             R3      ;SKIP OVER ADDRESS OF ASCII
6316  003504  000313                          SWAB    (R3)                    ;FORMAT ASCII FOR MESSAGE
6317  003506  112324                          MOVB    (R3)+,          (R4)+   ;PUT ASCII IN MESSAGE BUFFER
6318  003510  112324                          MOVB    (R3)+,          (R4)+
6319  003512  012705  006637          1$:     MOV     #BRHEAD,        R5
6320  003516  004767  000670                  JSR     PC,             FILBUF  ;GO PUT BOOT ROM HEADER IN MESS. BUF.
6321  003522  000015                          CR
6322  003524  012701  000001                  MOV     #1,     R1              ;+POINT TO DIAGNOSTIC ROM
6323  003530  012702  012144                  MOV     #MESTAB,        R2      ;+POINT TO MSG TABLE
6324  003534  012767  003550  006362          MOV     #2$,    RETURN          ;+SET UP AN ALTERNATE RETURN
6325  003542  012767  003734  006372          MOV     #5$,    FINISH          ;+SET UP ANOTHER ALTERNATE RETURN
6326  003550  112724  000015          2$:     MOVB    #CR,            (R4)+   ;PUT A CR/LF HERE
6327  003554  112724  000012                  MOVB    #LF,            (R4)+
6328  003560  004767  005330                  JSR     PC,     ROMTYP          ;+FIND THE ROM TYPE
6329  003564  062713  000004                  ADD     #4,             (R3)    ;GET FIRST ENTRY POINT
6330  003570  004767  000764                  JSR     PC,             OCADD   ;GO CONVERT OCTAL TO ASCII
6331  003574  004767  000612                  JSR     PC,             FILBUF  ;GO PUT ENTRY POINT IN MESSAGE BUF
6332  003600  000011                          HT
6333  003602  112724  000040                  MOVB    #40,            (R4)+
6334  003606  112724  000040                  MOVB    #40,            (R4)+
6335  003612  112724  000040                  MOVB    #40,            (R4)+
6336  003616  062713  000002                  ADD     #2,             (R3)    ;GET SECOND ENTRY POINT
6337  003622  004767  000732                  JSR     PC,             OCADD   ;GO CONVERT OCTAL TO ASCII
6338  003626  004767  000560                  JSR     PC,             FILBUF  ;GO PUT ENTRY POINT IN MESSAGE BUF.
6339  003632  000011                          HT
6340  003634  112724  000040                  MOVB    #40,            (R4)+
6341  003640  112724  000040                  MOVB    #40,            (R4)+
6342  003644  112724  000040                  MOVB    #40,            (R4)+
6343  003650  112724  000011                  MOVB    #HT,            (R4)+   ;PUT TAB IN HERE
6344  003654  062703  000002                  ADD     #2,             R3      ;UPDATE DATA BUFFER ADDRESS
6345  003660  000313                          SWAB    (R3)                    ;FORMAT ASCII FOR MESSAGE
6346  003662  112324                          MOVB    (R3)+,          (R4)+   ;MOV ASCII TO MES1
6347  003664  112324                          MOVB    (R3)+,          (R4)+
6348  003666  000730                          BR      2$                      ;GO BACK TO START OF LOOP
6349  003670  012703  007620          3$:     MOV     #BUF2,          R3      ;GET DATA BUFFER ADDRESS
6350  003674  012705  006774                  MOV     #PFHEAD,        R5      ;GET POWER-FAIL HEADER ADDRESS
6351  003700  004767  000506                  JSR     PC,             FILBUF  ;GO PUT POWER-FAIL HEADER IN MESSAGE BUF
6352  003704  000015                          CR
6353  003706  022703  007624          4$:     CMP     #BUF2+4,        R3      ;ARE WE DONE
6354  003712  001410                          BEQ     5$                      ;IF YES THEN GO RETURN
```

```
003714  004767  000640                  JSR     PC,             OCADD           ;GO CONVERT OCTAL TO ASCII
003720  004767  000466                  JSR     PC,             FILBUF
003724  000011                          HT
003726  062703  000002                  ADD     #2,             R3              ;UPDATE DATA BUFFER ADDRESS
003732  000765                          BR      4$                              ;GO BACK TO START OF LOOP
003734  112724  000015          5$:     MOVB    #CR,            (R4)+           ;PUT A CR/LF AT END OF MESSAGE
003740  112724  000012                  MOVB    #LF,            (R4)+
003744  105024                          CLRB    (R4)+                           ;PUT ZERO TERMINATOR AT END OF MESSAGE
003746  017667  000000  000002          MOV     @(R6),          6$              ;GET MESSAGE BUFFER ADDRESS
003754  104401                          TYPE
003756  000000                  6$:     .WORD   0
003760  062716  000002                  ADD     #2,             (R6)            ;GET OVER MESSAGE BUFFER ADDRESS
003764  000207                          RTS     PC
003766  000000                  TIMFS:  0









                                ;ERROR HANDLER ROUTINE
                                ;THIS SUBROUTINE FORMATS THE ERROR MESSAGES THE TYPES THEM
                                ;OUT.
003770  104411                  ERRHAN: SAVREG
003772  012704  010334                  MOV     #ERRMSG,        R4
003776  005237  001120                  INC     @#ERRCNT                                        ;INCREMENT ERROR COUNTER
004002  032777  002000  175074          BIT     #BIT10,         @SWR            ;BELL ON ERROR
004010  001402                          BEQ     1$                              ;BRANCH IF NO
004012  104401                          TYPE
004014  006536                          BELL
004016  032777  020000  175060  1$:     BIT     #BIT13,         @SWR            ;INHIBIT ERROR TYPEOUT
004024  001402                          BEQ     OVER1                           ; NO GO THROUGH
004026  000167  000310                  JMP     OVER10
004032  013700  001110          OVER1:  MOV     @#ROMERR,       R0              ;GET ERROR CODE
004036  013767  001110  000264          MOV     @#ROMERR,       OVER9           ;SAVE ERROR CODE FOR APT
004044  012701  004364                  MOV     #EHEADT,        R1              ;GET ERROR HEADER TABLE
004050  000241                          CLC                                     ;C-BIT USED TO STOP STEPPING THROUGH TAB
004052  006000                  2$:     ROR     R0                              ;ROTATE ERROR CODES
004054  103403                          BCS     3$                              ;IF C-BIT SET STOP STEPPING
004056  062701  000002                  ADD     #2,             R1              ;IF C-BIT CLEAR STEP TO NEXT HEADER
004062  000773                          BR      2$                              ;GO STEP
004064  011105                  3$:     MOV     (R1),           R5              ;PUT HEADER ADDRESS IN REGISTER.
004066  004767  000320                  JSR     PC,             FILBUF          ;GO PUT HEADER IN ERROR MESSAGE BUF.
004072  000015                          CR
004074  032767  000040  175006          BIT     #PFERR, ROMERR                  ;+IS IT A ''PPFVAR'' ERROR?
004102  001063                          BNE     8$                              ;+BRANCH IF YES
004104  032767  003600  174776          BIT     #3600,  ROMERR                  ;+IS IT A ''SEQTST'' ERROR?
004112  001411                          BEQ     14$                             ;+BRANCH IF NO
004114  016767  006016  000010          MOV     ARG2,   13$                     ;+PUT #CR OR #HT AT 13$
004122  016705  006006                  MOV     ARG1,   R5                      ;+ARG1 CONTAINS ADR. OF MSG
004126  004767  000260                  JSR     PC,     FILBUF                  ;+PUT DATA INTO BUFFER
004132  000000                  13$:    .WORD   0                               ;+CONTAINS CR OR HT
004134  000446                          BR      8$                              ;+
```

```
6411  004136  032737  000023  001110  14$:    BIT     #23,        @#ROMERR    ;+IS ERROR A CRC, APTER1 OR APTER2
6412  004144  001424                          BEQ     7$                      ;IF NO GO FORMAT APTER3, APTER4
6413  004146  032737  000020  001110          BIT     #20,        @#ROMERR    ;IS ERROR CRC
6414  004154  001415                          BEQ     6$                      ;IF NO FORMAT APTER1, APTER2
6415  004156  013700  002530                  MOV     @#ROMCNT,   R0          ;GET ROM COUNTER
6416  004162  012701  006524                  MOV     #ROMNUM,    R1          ;GET ROM NUMBER TABLE
6417  004166  000241                          CLC                             ;C-BIT USED TO STOP STEPPING THROUGH TAB
6418  004170  006000                  4$:     ROR     R0                      ;ROTATE ROM NUMBER
6419  004172  103403                          BCS     5$                      ;IF C-BIT SET STOP STEPPING
6420  004174  062701  000002                  ADD     #2,         R1          ;IF C-BIT CLEAR STEP TO NEXT HEADER
6421  004200  000773                          BR      4$                      ;CONTINUE STEPPING
6422  004202  112124                  5$:     MOVB    (R1)+,      (R4)+       ;PUT ROM NUMBER IN ERROR MESSAGE BUF
6423  004204  112124                          MOVB    (R1)+,      (R4)+
6424  004206  000421                          BR      8$                      ;GO TO END OF ROUTINE
6425  004210  112324                  6$:     MOVB    (R3)+,      (R4)+       ;PUT BAD DEVICE CODE IN ERROR MESS. BUF.
6426  004212  112324                          MOVB    (R3)+,      (R4)+
6427  004214  000416                          BR      8$                      ;GO TO END OF ROUTINE
6428  004216  016603  000006                  MOV     6(R6),      R3          ;GET SAVED EXPECTED VALUE
6429  004222  004767  000226                  JSR     PC,         OCASC       ;GO COVERT OCTAL TO ASCII
6430  004226  004767  000160                  JSR     PC,         FILBUF      ;GO PUT DATA IN ERROR MESSAGE BUF.
6431  004232  000011                          HT
6432  004234  016603  000010                  MOV     10(R6),     R3          ;GET SAVED RECEIVED VALUE
6433  004240  004767  000210                  JSR     PC,         OCASC       ;GO COVERT OCTAL TO ASCII
6434  004244  004767  000142                  JSR     PC,         FILBUF      ;GO PUT DATA IN ERROR MESSAGE BUFFER.
6435  004250  000011                          HT
6436  004252  112724  000015          8$:     MOVB    #CR,        (R4)+       ;PUT CR/LF AT END OF MESSAGE
6437  004256  112724  000012                  MOVB    #LF,        (R4)+
6438  004262  112724  000000                  MOVB    #0,         (R4)+       ;PUT TERMINATOR AT END OF MESSAGE
6439  004266  005767  174664                  TST     $ENV                    ;TST IF ON APT
6440  004272  001002                          BNE     OVER2                   ;IF YES BRANCH
6441  004274  000167  000034                  JMP     OVER11                  ;IF NO BRANCH
6442  004300  022737  000001  001116  OVER2:  CMP #1,          @#ROMFIN    ;- ONLY A CPU ROM??????
6443  004306  001005                          BNE             AROUND          ; -IF NO BRANCH
6444  004310  032737  000040  001110          BIT     #PFERR  , @#ROMERR      ;- NOW IF YES ,IS POWER-
6445                                                                          ;- FAIL VECTOR THE ERROR????
6446  004316  001401                          BEQ     AROUND                  ;- YES ,WELL THEN IT'S
6447  004320  000405                          BR OVER11                               ;- NOT A REAL ERROR , CONTINUE
6448  004322  004767  000544          AROUND: JSR     PC,         $ATY1       ;GO REPORT ERROR
6449  004326  010334                          ERRMSG
6450  004330  000000                  OVER9:  .WORD   0
6451  004332  000000                          HALT                            ;THEN HALT
6452  004334  004767  000540          OVER11: JSR     PC,         $ATY3       ;IF NOT ON APT JUST REPORT ERROR
6453  004340  010334                          ERRMSG
6454  004342  032777  100000  174534  OVER10: BIT     #BIT15,     @SWR        ;IS HALT ON ERROR SET
6455  004350  001401                          BEQ     12$                     ;IF NO SKIP OVER HALT
6456  004352  000000                          HALT
6457  004354  005037  001110          12$:    CLR     @#ROMERR                ;CLEAR ERROR FLAGS
6458  004360  104412                          RESREG
6459  004362  000207                          RTS     PC
6460
6461
6462  004364  007075                  EHEADT: ER1MSG
6463  004366  007041                          ER2MSG
6464  004370  007133                          ER3MSG
6465  004372  007211                          ER4MSG
6466  004374  007265                          CRCMSG
```

```
                                        PFMSG
                                        NOROMM
                                        SEQMSG                          ;+
                                        ERMES1                          ;+
                                        ERMES2                          ;+
                                        ERMES3                          ;+


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

                FILBUF: CMP     #HT,            @(R6)           ;IS FIRST CHARACTER A TAB OR CR
                        BNE     1$                              ;IF CR THEN GO PUT CR/LF IN BUFFER
                        MOVB    #HT,            (R4)+           ;IF TAB THEN PUT TAB IN BUFFER
                        BR      2$                              ;GET OVER NEXT LINE
                1$:     MOVB    #CR,            (R4)+           ;MOV CR/LF TO BUFFER
                        MOVB    #LF,            (R4)+
                2$:     MOVB    (R5)+,          (R4)+           ;PUT A CHARACTER IN MESSAGE BUFFER
                        TSTB    (R5)                            ;IS NEXT CHARACTER ZERO
                        BNE     2$                              ;IF NOT PUT IT IN MESSAGE BUFFER AND GET
                        ADD     #2,             (R6)            ;UPDATE RETURN POINTER TO GET OVER CHARA
                        RTS     PC                              ;THEN RETURN


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

                OCASC:  MOV     #OCTBUF,        R0              ;GET BUFFER ADDRESS
                        CLR     (R0)+                           ;CLEAR BUFFER
                        CLR     (R0)+
                        CLR     (R0)+
                        CLR     (R0)+
                        MOV     #OCTBUF,        R0              ;GET BUFFER ADDRESS
                        MOV     R3,             @#TEMP          ;GET OCTAL NUMBER
                        CLC                                     ;CLEAR CARRY
                        ROL     @#TEMP                          ;ROTATE BIT INTO CARRY BIT
                        ROL     (R0)                            ;ROTATE CARRY BIT INTO BUFFER
                1$:     BISB    #60,            (R0)            ;MAKE IT ASCII
                        INC     R0                              ;UPDATE BUFFER ADDRESS
                        CMP     R0,             #OCTBUF+6       ;HAVE WE CONVERTED ALL THE NUMBER
                        BNE     2$                              ;IF NO CONTINUE
                        MOV     #OCTBUF,        R5              ;IF YES PUT BUFFER ADDRESS IN REGISTER
                        RTS     PC                              ;RETURN
                2$:     ROL     @#TEMP                          ;ROTATE BIT INTO CARRY BIT
                        ROLB    (R0)                            ;ROTATE CARRY BIT INTO BUFFER
                        ROL     @#TEMP                          ;
                        ROLB    (R0)                            ;
                        ROL     @#TEMP                          ;
                        ROLB    (R0)                            ;
                        BR      1$                              ;GO TO START OF LOOP
                TEMP:   0
```

```
 6523
 6524
 6525
 6526                                   ;THIS ROUTINE IS CALLED BY THE PUT MESSAGE ROUTINE TO GET THE RIGHT
 6527                                   ;VALUE IN R3 SO THE OCTAL TO ASCII ROUTINE GETS THE RIGHT NUMBER.
 6528
 6529  004560  010346                  OCADD:  MOV     R3,             -(R6)           ;SAVE THE VALUE OF R3
 6530  004562  011303                          MOV     (R3),           R3              ;PUT THE DATA TO BE CONVERTED IN R3
 6531  004564  004767  177664                  JSR     PC,             OCASC           ;GO CONVERT OCTAL TO ASCII
 6532  004570  012603                          MOV     (R6)+,          R3              ;RESTORE R3
 6533  004572  000207                          RTS     PC                              ;RETURN
 6534
 6535
 6536
 6537
 6538
 6539                                   .SBTTL  TYPE ROUTINE
  (1)
  (2)                                  ;;*****************************************************************
  (1)                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
  (1)                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
  (1)                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
  (1)                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
  (1)                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
  (1)                                  ;*
  (1)                                  ;*CALL:
  (1)                                  ;*1) USING A TRAP INSTRUCTION
  (1)                                  ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
  (1)                                  ;*OR
  (1)                                  ;*      TYPE
  (1)                                  ;*      MESADR
  (1)                                  ;*
  (1)
  (1)  004574  105767  000265          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
  (1)  004600  100002                          BPL     1$              ;;BR IF YES
  (1)  004602  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
  (1)  004604  000430                          BR      3$              ;;LEAVE
  (1)  004606  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
  (1)  004610  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
  (1)  004614  122767  000001  174334          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)  004622  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
  (1)  004624  132767  000100  174325          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
  (1)  004632  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
  (1)  004634  010067  000004                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
  (1)  004640  004767  000234                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
  (1)  004644  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
  (1)  004646  132767  000040  174303  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
  (1)  004654  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
  (1)  004656  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
  (1)  004660  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
  (1)  004662  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
  (1)  004664  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
  (1)  004666  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
  (1)  004672  000002                          RTI                     ;;RETURN
  (1)  004674  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
  (1)  004700  001430                          BEQ     8$
```

```
    (1)  004702  122716  000200            CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
    (1)  004706  001006                     BNE     5$
    (1)  004710  005726                     TST     (SP)+           ;;POP  <CR><LF> EQUIV
    (1)  004712  104401                     TYPE                    ;;TYPE A CR AND LF
    (1)  004714  005067                     $CRLF
    (1)  004716  105067  000130             CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
    (1)  004722  000755                     BR      2$              ;;GET NEXT CHARACTER
    (1)  004724  004767  000056      5$:    JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
    (1)  004730  126726  000130      6$:    CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
    (1)  004734  001350                     BNE     2$              ;;IF NO GO GET NEXT CHAR.
    (1)  004736  016746  000120             MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
    (1)                                                             ;;AND THE NULL CHAR.
    (1)  004742  105366  000001      7$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
    (1)  004746  002770                     BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
    (1)  004750  004767  000032             JSR     PC,$TYPEC       ;;GO TYPE A NULL
    (1)  004754  105367  000072             DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
    (1)  004760  000770                     BR      7$              ;;LOOP
    (1)
    (1)                              ;HORIZONTAL TAB PROCESSOR
    (1)
    (1)  004762  112716  000040      8$:    MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
    (1)  004766  004767  000014      9$:    JSR     PC,$TYPEC       ;;TYPE A SPACE
    (1)  004772  132767  000007  000052     BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
    (1)  005000  001372                     BNE     9$              ;;TAB STOP
    (1)  005002  005726                     TST     (SP)+           ;;POP SPACE OFF STACK
    (1)  005004  000724                     BR      2$              ;;GET NEXT CHARACTER
    (1)  005006  105777  000044      $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
    (1)  005012  100375                     BPL     $TYPEC
    (1)  005014  116677  000002  000036     MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
    (1)  005022  122766  000015  000002     CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
    (1)  005030  001003                     BNE     1$              ;;BRANCH IF NO
    (1)  005032  105067  000014             CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
    (1)  005036  000406                     BR      $TYPEX          ;;EXIT
    (1)  005040  122766  000012  000002 1$: CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
    (1)  005046  001402                     BEQ     $TYPEX          ;;BRANCH IF YES
    (1)  005050  105227                     INCB    (PC)+           ;;COUNT THE CHARACTER
    (1)  005052  000000              $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
    (1)  005054  000207              $TYPEX: RTS     PC
    (1)
    (1)  005056  177564              $TPS:   .WORD   177564          ;;TTY PRINTER STATUS REG. ADDRESS
    (1)  005060  177566              $TPB:   .WORD   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
    (1)  005062     000              $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
    (1)  005063     002              $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
    (1)  005064     012              $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
    (1)  005065     000              $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
    (1)  005066     077              $QUES:  .ASCII  "?"             ;;QUESTION MARK
    (1)  005067     015              $CRLF:  .ASCII  <15>            ;;CARRAIGE RETURN
    (1)  005070  000012              $LF:    .ASCIZ  <12>            ;;LINEFEED
 6540                                .SBTTL  APT COMMUNICATIONS ROUTINE
    (1)
    (2)                              ;;**************************************************************************
    (1)  005072  112767  000001  000236 $ATY1: MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
    (1)  005100  112767  000001  000226 $ATY3: MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
    (1)  005106  000403                     BR      $ATYC
    (1)  005110  112767  000001  000220 $ATY4: MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
    (1)  005116                      $ATYC:
```

```
    (3)  005116  010046                        MOV     R0,-(SP)          ;;PUSH R0 ON STACK
    (3)  005120  010146                        MOV     R1,-(SP)          ;;PUSH R1 ON STACK
    (1)  005122  105767  000206                TSTB    $MFLG             ;;SHOULD TYPE A MESSAGE?
    (1)  005126  001450                        BEQ     5$                ;;IF NOT:  BR
    (1)  005130  122767  000001  174020        CMPB    #APTENV,$ENV      ;;OPERATING UNDER APT?
    (1)  005136  001031                        BNE     3$                ;;IF NOT:  BR
    (1)  005140  132767  000100  174011        BITB    #APTSPOOL,$ENVM   ;;SHOULD SPOOL MESSAGES?
    (1)  005146  001425                        BEQ     3$                ;;IF NOT:  BR
    (1)  005150  017600  000004                MOV     @4(SP),R0         ;;GET MESSAGE ADDR.
    (1)  005154  062766  000002  000004        ADD     #2,4(SP)                  ;;BUMP RETURN ADDR.
    (1)  005162  005767  173750          1$:   TST     $MSGTYPE          ;;SEE IF DONE W/ LAST XMISSION?
    (1)  005166  001375                        BNE     1$                ;;IF NOT:  WAIT
    (1)  005170  010067  173756                MOV     R0,$MSGAD         ;;PUT ADDR IN MAILBOX
    (1)  005174  105720                  2$:   TSTB    (R0)+             ;;FIND END OF MESSAGE
    (1)  005176  001376                        BNE     2$
    (1)  005200  166700  173746                SUB     $MSGAD,R0         ;;SUB START OF MESSAGE
    (1)  005204  006200                        ASR     R0                ;;GET MESSAGE LNGTH IN WORDS
    (1)  005206  010067  173742                MOV     R0,$MSGLGT        ;;PUT LENGTH IN MAILBOX
    (1)  005212  012767  000004  173716        MOV     #4,$MSGTYPE       ;;TELL APT TO TAKE MSG.
    (1)  005220  000413                        BR      5$
    (1)  005222  017667  000004  000016  3$:   MOV     @4(SP),4$         ;;PUT MSG ADDR IN JSR LINKAGE
    (1)  005230  062766  000002  000004        ADD     #2,4(SP)                  ;;BUMP RETURN ADDRESS
    (3)  005236  016746  172534                MOV     177776,-(SP)      ;;PUSH 177776 ON STACK
    (1)  005242  004767  177326                JSR     PC,$TYPE          ;;CALL TYPE MACRO
    (1)  005246  000000                  4$:   .WORD   0
    (1)  005250                          5$:
    (1)  005250  105767  000062          10$:  TSTB    $FFLG             ;;SHOULD REPORT FATAL ERROR?
    (1)  005254  001416                        BEQ     12$               ;;IF NOT:  BR
    (1)  005256  005767  173674                TST     $ENV              ;;RUNNING UNDER APT?
    (1)  005262  001413                        BEQ     12$               ;;IF NOT:  BR
    (1)  005264  005767  173646          11$:  TST     $MSGTYPE          ;;FINISHED LAST MESSAGE?
    (1)  005270  001375                        BNE     11$               ;;IF NOT:  WAIT
    (1)  005272  017667  000004  173640        MOV     @4(SP),$FATAL     ;;GET ERROR #
    (1)  005300  062766  000002  000004        ADD     #2,4(SP)                  ;;BUMP RETURN ADDR.
    (1)  005306  005267  173624                INC     $MSGTYPE          ;;TELL APT TO TAKE ERROR
    (1)  005312  105067  000020          12$:  CLRB    $FFLG             ;;CLEAR FATAL FLAG
    (1)  005316  105067  000013                CLRB    $LFLG             ;;CLEAR LOG FLAG
    (1)  005322  105067  000006                CLRB    $MFLG             ;;CLEAR MESSAGE FLAG
    (3)  005326  012601                        MOV     (SP)+,R1          ;;POP STACK INTO R1
    (3)  005330  012600                        MOV     (SP)+,R0          ;;POP STACK INTO R0
    (1)  005332  000207                        RTS     PC                ;;RETURN
    (1)  005334     000                $MFLG:  .BYTE   0                 ;;MESSG. FLAG
    (1)  005335     000                $LFLG:  .BYTE   0                 ;;LOG FLAG
    (1)  005336     000                $FFLG:  .BYTE   0                 ;;FATAL FLAG
    (1)          005340                        .EVEN
    (1)          000200                APTSIZE=200
    (1)          000001                APTENV=001
    (1)          000100                APTSPOOL=100
    (1)          000040                APTCSUP=040
  6541                                  .SBTTL  TTY INPUT ROUTINE
    (1)
    (2)                                 ;;**********************************************************************
    (1)  005340  177560                $TKS:   .WORD   177560            ;;TTY KBD STATUS
    (1)  005342  177562                $TKB:   .WORD   177562            ;;TTY KBD BUFFER
    (1)                                 .ENABL  LSB
    (1)
```

```
(2)                                     ;;**********************************************************************
(1)                                     ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)                                     ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1)                                     ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1)                                     ;*WHEN OPERATING IN TTY FLAG MODE.
(1)  005344  022767  000176  173532  $CKSWR: CMP   #SWREG,SWR        ;;IS THE SOFT-SWR SELECTED?
(1)  005352  001074                         BNE   15$               ;;BRANCH IF NO
(1)  005354  105777  177760                 TSTB  @$TKS             ;;CHAR THERE?
(1)  005360  100071                         BPL   15$               ;;IF NO, DON'T WAIT AROUND
(1)  005362  117746  177754                 MOVB  @$TKB,-(SP)       ;;SAVE THE CHAR
(1)  005366  042716  177600                 BIC   #^C177,(SP)       ;;STRIP-OFF THE ASCII
(1)  005372  022726  000007                 CMP   #7,(SP)+          ;;IS IT A CONTROL G?
(1)  005376  001062                         BNE   15$               ;;NO, RETURN TO USER
(1)  005400  126727  173474  000001         CMPB  $AUTOB,#1         ;;ARE WE RUNNING IN AUTO-MODE?
(1)  005406  001456                         BEQ   15$               ;;BRANCH IF YES
(1)
(1)  005410  104401  006071                 TYPE  ,$CNTLG           ;;ECHO THE CONTROL-G (^G)
(1)  005414  104401  006076  $GTSWR: TYPE   ,$MSWR              ;;TYPE CURRENT CONTENTS
(2)  005420  016746  172552                 MOV   SWREG,-(SP)       ;;SAVE SWREG FOR TYPEOUT
(2)  005424  104402                         TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  005426  104401  006107                 TYPE  ,$MNEW            ;;PROMPT FOR NEW SWR
(1)  005432  005046                  19$:   CLR   -(SP)             ;;CLEAR COUNTER
(1)  005434  005046                         CLR   -(SP)             ;;THE NEW SWR
(1)  005436  105777  177676          7$:    TSTB  @$TKS             ;;CHAR THERE?
(1)  005442  100375                         BPL   7$                ;;IF NOT TRY AGAIN
(1)
(1)  005444  117746  177672                 MOVB  @$TKB,-(SP)       ;;PICK UP CHAR
(1)  005450  042716  177600                 BIC   #^C177,(SP)       ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1)  005454  021627  000025          9$:    CMP   (SP),#25          ;;IS IT A CONTROL-U?
(1)  005460  001005                         BNE   10$               ;;BRANCH IF NOT
(1)  005462  104401  006064                 TYPE  ,$CNTLU           ;;YES, ECHO CONTROL-U (^U)
(1)  005466  062706  000006          20$:   ADD   #6,SP             ;;IGNORE PREVIOUS INPUT
(1)  005472  000757                         BR    19$               ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  005474  021627  000015          10$:   CMP   (SP),#15          ;;IS IT A <CR>?
(1)  005500  001023                         BNE   16$               ;;BRANCH IF NO
(1)  005502  005766  000004                 TST   4(SP)             ;;YES, IS IT THE FIRST CHAR?
(1)  005506  001403                         BEQ   11$               ;;BRANCH IF YES
(1)  005510  016677  000002  173366         MOV   2(SP),@SWR        ;;SAVE NEW SWR
(1)  005516  062706  000006          11$:   ADD   #6,SP             ;;CLEAR UP STACK
(1)  005522  104401  005067          14$:   TYPE  ,$CRLF            ;;ECHO <CR> AND <LF>
(1)  005526  126727  173347  000001         CMPB  $INTAG,#1         ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  005534  001003                         BNE   15$               ;;BRANCH IF NOT
(1)  005536  012777  000100  177574         MOV   #100,@$TKS        ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  005544  000002                  15$:   RTI                     ;;RETURN
(1)  005546  004767  177234          16$:   JSR   PC,$TYPEC         ;;ECHO CHAR
(1)  005552  021627  000060                 CMP   (SP),#60          ;;CHAR < 0?
(1)  005556  002420                         BLT   18$               ;;BRANCH IF YES
(1)  005560  021627  000067                 CMP   (SP),#67          ;;CHAR > 7?
(1)  005564  003015                         BGT   18$               ;;BRANCH IF YES
(1)  005566  042726  000060                 BIC   #60,(SP)+         ;;STRIP-OFF ASCII
(1)  005572  005766  000002                 TST   2(SP)             ;;IS THIS THE FIRST CHAR
```

```
(1)  005576  001403                          BEQ     17$             ;;BRANCH IF YES
(1)  005600  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  005602  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
(1)  005604  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
(1)  005606  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  005612  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  005616  000707                          BR      7$              ;;GET THE NEXT ONE
(1)  005620  104401  005066          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  005624  000720                          BR      20$             ;;SIMULATE CONTROL-U
(1)                                  .DSABL  LSB
(1)
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                  ;*CALL:
(1)                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                  ;
(1)
(1)  005626  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  005630  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  005636  105777  177476          1$:     TSTB    @$TKS           ;;WAIT FOR
(1)  005642  100375                          BPL     1$              ;A CHARACTER
(1)  005644  117766  177472  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  005652  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  005660  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  005666  001013                          BNE     3$              ;;BRANCH IF NO
(1)  005670  105777  177444          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  005674  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  005676  117746  177440                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  005702  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  005706  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  005712  001366                          BNE     2$              ;;IF NOT DISCARD IT
(1)  005714  000750                          BR      1$              ;;YES, RESUME
(1)  005716  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  005724  002407                          BLT     4$              ;;BRANCH IF YES
(1)  005726  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  005734  003003                          BGT     4$              ;;BRANCH IF YES
(1)  005736  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
(1)  005744  000002                  4$:     RTI                     ;;GO BACK TO USER
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                  ;*CALL:
(1)                                  ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                  ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                  ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  005746  010346          $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  005750  012703  006054          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  005754  022703  006064          2$:     CMP     #$TTYIN+80.,R3  ;;BUFFER FULL?
(1)  005760  101405                          BLOS    4$              ;;BR IF YES
(1)  005762  104407                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  005764  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  005766  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  005772  001003                          BNE     3$              ;;SKIP IF NOT
```

```
(1)  005774  104401  005066          4$:   TYPE   ,$QUES          ;;TYPE A '?'
(1)  006000  000763                        BR     1$              ;;CLEAR THE BUFFER AND LOOP
(1)  006002  111367  000044          3$:   MOVB   (R3),9$          ;;ECHO THE CHARACTER
(1)  006006  104401  006052                TYPE   ,9$
(1)  006012  122723  000015                CMPB   #15,(R3)+        ;;CHECK FOR RETURN
(1)  006016  001356                        BNE    2$              ;;LOOP IF NOT RETURN
(1)  006020  105063  177777                CLRB   -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  006024  104401  005070                TYPE   ,$LF            ;;TYPE A LINE FEED
(1)  006030  012603                        MOV    (SP)+,R3        ;;RESTORE R3
(1)  006032  011646                        MOV    (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  006034  016666  000004  000002        MOV    4(SP),2(SP)     ;;       FIRST ASCII CHARACTER ON IT
(1)  006042  012766  006054  000004        MOV    #$TTYIN,4(SP)
(1)  006050  000002                        RTI                    ;;RETURN
(1)  006052     000                9$:    .BYTE  0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  006053     000                       .BYTE  0               ;;TERMINATOR
(1)  006054  000010                $TTYIN: .BLKB  8.              ;;RESERVE 8 BYTES FOR TTY INPUT
(1)  006064  052536  005015  000   $CNTLU: .ASCIZ  /^U/<15><12>   ;;CONTROL 'U'
(1)  006071     136  006507  000012 $CNTLG: .ASCIZ  /^G/<15><12>   ;;CONTROL 'G'
(1)  006076  005015  053523  020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  006104  020075     000
(1)  006107     040  047040  053505 $MNEW:  .ASCIZ  /  NEW = /
(1)  006114  036440  000040
6542
(1)                                 .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(2)
(1)                                 ;;**********************************************************************
(1)                                 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                                 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                                 ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                 ;*      TYPOS                   ;;CALL FOR TYPEOUT
(1)                                 ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                                 ;*      .BYTE   M               ;;M=1 OR 0
(1)                                 ;*                                      ;;1=TYPE LEADING ZEROS
(1)                                 ;*                                      ;;0=SUPPRESS LEADING ZEROS
(1)                                 ;*
(1)                                 ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                                 ;*$TYPOS OR $TYPOC
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                 ;*      TYPON                   ;;CALL FOR TYPEOUT
(1)                                 ;*
(1)                                 ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                 ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)
(1)  006120  017646  000000        $TYPOS: MOV    @(SP),-(SP)     ;;PICKUP THE MODE
(1)  006124  116667  000001  000211        MOVB   1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  006132  112667  000207                MOVB   (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  006136  062716  000002                ADD    #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  006142  000406                        BR     $TYPON
(1)  006144  112767  000001  000171 $TYPOC: MOVB   #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  006152  112767  000006  000165        MOVB   #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  006160  112767  000005  000154 $TYPON: MOVB   #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  006166  010346                        MOV    R3,-(SP)        ;;SAVE R3
```

```
(1)  006170  010446                        MOV     R4,-(SP)        ;;SAVE R4
(1)  006172  010546                        MOV     R5,-(SP)        ;;SAVE R5
(1)  006174  116704  000145                MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  006200  005404                        NEG     R4
(1)  006202  062704  000006                ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  006206  110467  000132                MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  006212  116704  000125                MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  006216  016605  000012                MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  006222  005003                        CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  006224  006105                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)  006226  000404                        BR      3$              ;;GO DO MSB
(1)  006230  006105                2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  006232  006105                        ROL     R5
(1)  006234  006105                        ROL     R5
(1)  006236  010503                        MOV     R5,R3
(1)  006240  006103                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  006242  105367  000076                DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  006246  100016                        BPL     7$              ;;BR IF NO
(1)  006250  042703  177770                BIC     #177770,R3      ;;GET RID OF JUNK
(1)  006254  001002                        BNE     4$              ;;TEST FOR 0
(1)  006256  005704                        TST     R4              ;;SUPPRESS THIS 0?
(1)  006260  001403                        BEQ     5$              ;;BR IF YES
(1)  006262  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  006264  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  006270  052703  000040        5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  006274  110367  000040                MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  006300  104401  006340                TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  006304  105367  000032        7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  006310  003347                        BGT     2$              ;;BR IF MORE TO DO
(1)  006312  002402                        BLT     6$              ;;BR IF DONE
(1)  006314  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  006316  000744                        BR      2$              ;;GO DO THE LAST DIGIT
(1)  006320  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  006322  012604                        MOV     (SP)+,R4        ;;RESTORE R4
(1)  006324  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  006326  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  006334  012616                        MOV     (SP)+,(SP)
(1)  006336  000002                        RTI                     ;;RETURN
(1)  006340     000                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  006341     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  006342     000                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  006343     000                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  006344  000000                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
6543
 (1)                               .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
 (2)
 (1)                               ;;**********************************************************************
 (1)                               ;*SAVE R0-R5
 (1)                               ;*CALL:
 (1)                               ;*      SAVREG
 (1)                               ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
 (1)                               ;*
 (1)                               ;*TOP---(+16)
 (1)                               ;* +2---(+18)
 (1)                               ;* +4---R5
 (1)                               ;* +6---R4
 (1)                               ;* +8---R3
```

```
                                        ;*+10---R2
                                        ;*+12---R1
                                        ;*+14---R0
                                        ;
006346                                  $SAVREG:
006346  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006350  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006352  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006354  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006356  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
006360  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
006362  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
006366  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
006372  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
006376  016646  000022                         MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
006402  000002                                  RTI
                                        ;
                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
006404                                  $RESREG:
006404  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
006410  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
006414  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
006420  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
006424  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
006426  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
006430  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
006432  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
006434  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
006436  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
006440  000002                                  RTI
6546                                    .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.
                                        ;
006442  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
006444  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
006450  005740                                  TST     -(R0)           ;;BACKUP BY 2
006452  111000                                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
006454  006300                                  ASL     R0              ;;POSITION FOR INDEXING
006456  016000  006476                          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
006462  000200                                  RTS     R0              ;;GO TO ROUTINE
                                        ;
                                        ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
                                        ;
006464  011646                          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
006466  016666  000004  000002                  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
006474  000002                                  RTI                     ;;RESTORE THE PSW
                                        ;
                                        .SBTTL  TRAP TABLE
```

```
   (3)
   (3)                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
   (3)                                  ;*BY THE "TRAP" INSTRUCTION.
   (3)
   (3)                                  ;       ROUTINE
   (3)                                  ;       -------
   (3)  006476  006464                  $TRPAD: .WORD   $TRAP2
   (3)  006500  004574                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
   (3)  006502  006144                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
   (3)  006504  006120                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
   (3)  006506  006160                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
   (1)
   (3)  006510  005414                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING
   (1)
   (3)  006512  005344                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
   (3)  006514  005626                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
   (3)  006516  005746                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
   (3)  006520  006346                          $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
   (3)  006522  006404                          $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
 6547
 6548
 6549                                   ;MESSAGES
 6550
 6551
 6552  006524  030062                  ROMNUM: .ASCII  /20/
 6553  006526  032463                          .ASCII  /35/
 6554  006530  031463                          .ASCII  /33/
 6555  006532  032063                          .ASCII  /34/
 6556  006534  031063                          .ASCII  /32/
 6557  006536  177607  000377          BELL:   .ASCIZ  <207><377><377>
 6558  006542  006412  047516  051040  MNORM:.ASCIZ  <12><15>/NO ROMS RESPONDED TO THE SIZING ROUTINE/<12><15>
       006550  046517  020123  042522
       006556  050123  047117  042504
       006564  020104  047524  052040
       006572  042510  051440  055111
       006600  047111  020107  047522
       006606  052125  047111  005105
       006614  000015
 6559
 6560
 6561  006616  044504  043501  020056  DRHEAD: .ASCIZ  /DIAG. ROM (E20) /
       006624  047522  020115  042450
       006632  030062  020051     000
 6562  006637     012  047502  052117  BRHEAD: .ASCII  <12>/BOOTSTRAP ROM ENTRY POINTS AND DEVICE CODES/<15><12>
       006644  052123  040522  020120
       006652  047522  020115  047105
       006660  051124  020131  047520
       006666  047111  051524  040440
       006674  042116  042040  053105
       006702  041511  020105  047503
       006710  042504  006523     012
 6563  006715     114  041517  020056          .ASCIZ  /LOC.            NO DIAG.       RUN DIAG.       DEVICE CODE/
       006722  020040  020040  020040
       006730  020040  004440  047040
       006736  020117  044504  043501
       006744  004456  052522  020116
```

```
                006752  044504  043501  004456
                006760  042504  044526  042503
                006766  041440  042117  000105
 6564           006774  051520  052505  047504  PFHEAD: .ASCIZ  @PSEUDO POWER-FAIL VECTOR ADR./NEW PC@
                007002  050040  053517  051105
                007010  043055  044501  020114
                007016  042526  052103  051117
                007024  040440  051104  027456
                007032  042516  020127  041520
                007040     000
 6565           007041     103  052517  042114  ER2MSG: .ASCIZ  /COULD NOT FIND DEVICE CODE /
                007046  047040  052117  043040
                007054  047111  020104  042504
                007062  044526  042503  041440
                007070  042117  020105     000
 6566           007075     106  052517  042116  ER1MSG: .ASCIZ  /FOUND UNEXPECTED DEVICE CODE /
                007102  052440  042516  050130
                007110  041505  042524  020104
                007116  042504  044526  042503
                007124  041440  042117  020105
                007132     000
 6567           007133     120  053517  051105  ER3MSG: .ASCII  /POWER-FAIL VECTOR ERROR/<15><12>
                007140  043055  044501  020114
                007146  042526  052103  051117
                007154  042440  051122  051117
                007162  005015
 6568           007164  042411  050130  041505          .ASCIZ  /       EXPECTED        RECIEVED/<15><12>
                007172  042524  004504  042522
                007200  044503  053105  042105
                007206  005015     000
 6569           007211     120  053517  051105  ER4MSG: .ASCII  /POWER-FAIL DATA ERROR/<15><12>
                007216  043055  044501  020114
                007224  040504  040524  042440
                007232  051122  051117  005015
 6570           007240  042411  050130  041505          .ASCIZ  /       EXPECTED        RECIEVED/<15><12>
                007246  042524  004504  042522
                007254  044503  053105  042105
                007262  005015     000
 6571           007265     103  041522  042440  CRCMSG: .ASCIZ  /CRC ERROR IN ROM E-/
                007272  051122  051117  044440
                007300  020116  047522  020115
                007306  026505     000
 6572           007311     103  052517  042114  PFMSG:  .ASCIZ  /COULD NOT DETERMINE POWER-FAIL VECTOR ADDRESS/
                007316  047040  052117  042040
                007324  052105  051105  044515
                007332  042516  050040  053517
                007340  051105  043055  044501
                007346  020114  042526  052103
                007354  051117  040440  042104
                007362  042522  051523     000
 6573           007367     116  020117  047522  NOROMM: .ASCII  /NO ROMS TEST ERROR/<15><12>
                007374  051515  052040  051505
                007402  020124  051105  047522
                007410  006522     012
 6574           007413     040  020040  020040          .ASCIZ  /       VALUE   ADDRESS/<15><12>
                007420  020040  053040  046101
```

```
                007426  042525  020040  040440
                007434  042104  042522  051523
                007442  005015  000
 6575           007445  015     051012  046517  SEQMSG: .ASCII  <15><12>/ROM SEQUENCE IS INCORRECT AS PER /
                007452  051440  050505  042525
                007460  041516  020105  051511
                007466  044440  041516  051117
                007474  042522  052103  040440
                007502  020123  042520  020122
 6576           007510  047111  052123  046101          .ASCIZ  /INSTALLATION PROCEDURE./
                007516  040514  044524  047117
                007524  050040  047522  042503
                007532  052504  042522  000056
 6577                                            .EVEN
 6578
 6579
 6580                                            ;BUFFERS
 6581
 6582           007540  000030                  BUF1:
 6584           007540  000000                          .WORD   0
  (1)           007542  000000                          .WORD   0
  (1)           007544  000000                          .WORD   0
  (1)           007546  000000                          .WORD   0
  (1)           007550  000000                          .WORD   0
  (1)           007552  000000                          .WORD   0
  (1)           007554  000000                          .WORD   0
  (1)           007556  000000                          .WORD   0
  (1)           007560  000000                          .WORD   0
  (1)           007562  000000                          .WORD   0
  (1)           007564  000000                          .WORD   0
  (1)           007566  000000                          .WORD   0
  (1)           007570  000000                          .WORD   0
  (1)           007572  000000                          .WORD   0
  (1)           007574  000000                          .WORD   0
  (1)           007576  000000                          .WORD   0
  (1)           007600  000000                          .WORD   0
  (1)           007602  000000                          .WORD   0
  (1)           007604  000000                          .WORD   0
  (1)           007606  000000                          .WORD   0
  (1)           007610  000000                          .WORD   0
  (1)           007612  000000                          .WORD   0
  (1)           007614  000000                          .WORD   0
  (1)           007616  000000                          .WORD   0
 6585           007620  000000                  BUF2:   .WORD   0
 6586           007622  000000                          .WORD   0
 6587           007624  000010                  OCTBUF: .BLKB   10
 6588           007634  000400                  MES1:   .BLKB   400
 6589           010234  000100                  MES2:   .BLKB   100
 6590           010334  000400                  ERRMSG: .BLKB   400                     ;+
 6591
 6592                                            ;+
 6593                                            ;+THE FOLLOWING CODE WAS ADDED IN REV B0:
 6594                                            ;+
 6595                                            ;+''CLEAR'' CLEARS THE NEW LABEL AND BUFFER LOCATIONS
 6596                                            ;+
 6597                                            ;+''CON1'' AND ''CON2'' FILL 'BUF1'' WITH THE CONTINUATION ROM DATA AND
```

```
                                        ;+SET ERROR FLAGS WHEN NECESSARY
                                        ;+
                                        ;+'HOLCK1' AND 'HOLCK2' CHECK FOR HOLES TO VERIFY THE POWER FAIL
                                        ;+VECTOR ADDRESS; ALSO SET 11/60 INDICATOR WHEN APPLICABLE
                                        ;+
                                        ;+'ROMTYP' AND 'ROMNM' FILL THE MESSAGE BUFFER WITH A CONTINUATION ROM
                                        ;+MESSAGE AND WITH A ROM IDENTIFICATION MESSAGE
                                        ;+

010734  012700  012100          CLEAR:  MOV     #SEQBUF,        R0      ;+SET UP TO CLEAR LOCATIONS
010740  005020                  1$:     CLR     (R0)+                   ;+
010742  020027  012142                  CMP     R0,     #FINISH         ;+ALL CLEARED?
010746  101774                          BLOS    1$                      ;+BRANCH IF NO
010750  000207                          RTS     PC                      ;+
010752  030167  001152          CON1:   BIT     R1,     CONFIN          ;+CONTINUATION CHIP?
010756  001001                          BNE     1$                      ;+BRANCH IF YES
010760  000207                          RTS     PC                      ;+
010762  050167  001132          1$:     BIS     R1,     CONER1          ;+CHIP DOES NOT BELONG HERE-SET ERROR FLAG
010766  012720  000001                  MOV     #1,     (R0)+           ;+SET CONTINUATION ROM INDICATOR
010772  012720  177777                  MOV     #-1,    (R0)+           ;+ILLEGAL ROM'S DEVICE CODE =-1
010776  000207                          RTS     PC                      ;+
011000  005703          CON2:   TST     R3                              ;+POINTING OVER THIS CHIP?
011002  001001                          BNE     1$                      ;+BRANCH IF YES
011004  000207                          RTS     PC                      ;+
011006  030167  001116          1$:     BIT     R1,     CONFIN          ;+CONTINUATION CHIP?
011012  001005                          BNE     2$                      ;+BRANCH IF YES
011014  014067  001102                  MOV     -(R0),  CONER2          ;+CHIP IS MISSING-SAVE DEVICE CODE
011020  062700  000002                  ADD     #2,     R0              ;+
011024  000207                          RTS     PC                      ;+
011026  012710  000001          2$:     MOV     #1,     (R0)            ;+SET CONTINUATION CHIP INDICATOR
011032  014002                          MOV     -(R0),  R2              ;+GET PREVIOUS DEVICE CODE
011034  062700  000004                  ADD     #4,     R0              ;+POINT TO CORRECT LOCATION
011040  010220                          MOV     R2,     (R0)+           ;+PUT IN DEVICE CODE
011042  000207                          RTS     PC                      ;+
011044  032767  000002  170044  HOLCK1: BIT     #2,     ROMFIN          ;+A CHIP IN SOCKET #1?
011052  001415                          BEQ     END                     ;+BRANCH IF NO
011054  000207                          RTS     PC                      ;+
011056  032767  000004  170032  HOLCK2: BIT     #4,     ROMFIN          ;+A CHIP IN SOCKET #2?
011064  001401                          BEQ     1$                      ;+BRANCH IF NO
011066  000207                          RTS     PC                      ;+
011070  122737  000777  173224  1$:     CMPB    #777,   @#173224        ;+ARE SWITCHES SET?
011076  001403                          BEQ     END                     ;+BRANCH IF NO
011100  052767  000002  001020          BIS     #2,     FLAG            ;+SET 11/60 INDICATOR
011106  016716  001012          END:    MOV     RETURN, (R6)            ;+SET UP ALTERNATE RETURN
011112  000207                          RTS     PC                      ;+
011114  005713          ROMTYP: TST     (R3)                            ;+ARE WE FINISHED?
011116  001003                          BNE     1$                      ;+BRANCH IF NO
011120  016716  001016                  MOV     FINISH, (R6)            ;+SET UP ALTERNATE RETURN
011124  000207                          RTS     PC                      ;+
011126  022713  000001          1$:     CMP     #1,     (R3)            ;+CONTINUATION CHIP?
011132  001401                          BEQ     2$                      ;+BRANCH IF YES
011134  000414                          BR      ROMNM                   ;+ITS A DEVICE CHIP-FIND SOCKET #
011136  004767  000024          2$:     JSR     PC,     ROMNM           ;+FIND SOCKET #
011142  012705  012154                  MOV     #CONMES,        R5      ;+LOAD MSG INTO BUFFER
011146  004767  173240                  JSR     PC,     FILBUF          ;+
011152  000011                          HT                              ;+
```

```
6657  011154  062703  000004            ADD     #4,     R3              ;+POP OVER CONTINUATION CHIP DATA
6658  011160  016716  000740            MOV     RETURN, (R6)            ;+SET UP ALTERNATE RETURN
6659  011164  000207                    RTS     PC                      ;+
6660  011166  032713  000170  ROMNM:   BIT     #170,   (R3)            ;+BEGINNING A NEW ROM?
6661  011172  001015                    BNE     3$                      ;+BRANCH IF NOT
6662  011174  000241            1$:     CLC                             ;+
6663  011176  006101                    ROL     R1                      ;+POINT TO NEXT SOCKET
6664  011200  030167  167712            BIT     R1,     ROMFIN          ;+IS A ROM THERE?
6665  011204  001003                    BNE     2$                      ;+BRANCH IF YES
6666  011206  062702  000002            ADD     #2,     R2              ;+POINT TO NEXT MSG
6667  011212  000770                    BR      1$                      ;+CONTINUE
6668  011214  012205            2$:     MOV     (R2)+,  R5              ;+LOAD ROM ID
6669  011216  004767  173170            JSR     PC,     FILBUF          ;+INTO THE MSG BUFFER
6670  011222  000015                    CR                              ;+
6671  011224  000402                    BR      4$                      ;+
6672  011226  112724  000011  3$:     MOVB    #HT,    (R4)+           ;+LOAD A TAB
6673  011232  000207            4$:     RTS     PC                      ;+
6674
6675                                    ;+SEQUENCE TEST
6676                                    ;+THIS TEST VERIFIES THAT CONTINUATION CHIPS ARE LOCATED WHERE
6677                                    ;+THEY BELONG, THAT THERE ARE NO DUPLICATE DEVICE ROMS,
6678                                    ;+THAT THE ROMS ARE IN ALPHABETICAL ORDER, THAT THERE ARE NO
6679                                    ;+HOLES, AND THAT SOCKET #2 HAS A CHIP IF THE PROCESSOR IS AN 11/60.
6680                                    ;+IF THE ONLY PROBLEM IS ALPHABETICAL ORDER AND/OR HOLES,
6681                                    ;+THEN A CORRECT SEQUENCE IS DETERMINED AND PRINTED OUT.
6682
6683                                    ;+FIRST, CHECK FOR ILLEGAL CONTINUATION ROM ERRORS
6684                                    ;+IF ANY ARE FOUND, PRINT ERROR MSG AND EXIT
6685
6686  011234  012767  000011  000674  SEQTST: MOV     #HT,    ARG2            ;+SET UP FOR 'ERRHAN'
6687  011242  005767  000652            TST     CONER1                  ;+WAS THERE AN ERROR?
6688  011246  001417                    BEQ     2$                      ;+BRANCH IF NO
6689  011250  016703  000644            MOV     CONER1, R3              ;+THE BIT IDENTIFIES THE CHIP
6690  011254  012702  012144            MOV     #MESTAB,        R2      ;+POINT TO MSG TABLE
6691  011260  012767  000400  167622    MOV     #CONONE,        ROMERR  ;+SET ERROR FLAG
6692  011266  032703  000002  1$:     BIT     #2,     R3              ;+IDENTIFY THE ROM FOR PRINTOUT
6693  011272  001026                    BNE     3$                      ;+BRANCH IF FOUND IT
6694  011274  062702  000002            ADD     #2,     R2              ;+SET UP FOR NEXT ROM MSG
6695  011300  000241                    CLC                             ;+
6696  011302  006003                    ROR     R3                      ;+SET UP TO IDENTIFY NEXT ROM
6697  011304  000770                    BR      1$                      ;+
6698  011306  005767  000610  2$:     TST     CONER2                  ;+WAS THERE AN ERROR?
6699  011312  001423                    BEQ     FILTAB                  ;+BRANCH IF NO
6700  011314  012767  001000  167566    MOV     #CONTWO,        ROMERR  ;+SET ERROR FLAG
6701  011322  012767  007634  000604    MOV     #MES1,  ARG1            ;+PASS THIS ADDRESS TO ERRHAN
6702  011330  000367  000566            SWAB    CONER2                  ;+BUILD THE MSG
6703  011334  016767  000562  176272    MOV     CONER2, MES1            ;+
6704  011342  105067  176270            CLRB    MES1+2                  ;+
6705  011346  000402                    BR      4$                      ;+REPORT THE ERROR
6706  011350  011267  000560  3$:     MOV     (R2),   ARG1            ;+PASS MSG ADDRESS TO 'ERRHAN'
6707  011354  004767  172410  4$:     JSR     PC,     ERRHAN          ;+REPORT ERROR
6708  011360  000207                    RTS     PC                      ;+
6709
6710                                    ;+IF THERE WERE NO CONTINUATION ROM ERRORS, BUILD THE FOLLOWING TABLE:
6711                                    ;+
6712                                    ;+      SEQBUF:         FIRST DEVICE CODE
```

```
6713                                    ;+                  # OF CONTINUATION ROMS FOR FIRST DEVICE
6714                                    ;+                  SECOND DEVICE CODE
6715                                    ;+                  # OF CONTINUATION ROMS FOR SECOND DEVICE
6716                                    ;+                        .
6717                                    ;+                        .
6718                                    ;+      ENDSEQ:     FOURTH (LAST) DEVICE CODE
6719                                    ;+
6720                                    ;+FOR A 'HOLE', THE DEVICE CODE WILL BE A 0.
6721
6722  011362  012703  012100           FILTAB: MOV     #SEQBUF,        R3      ;+SET UP TO FILL A TABLE CALLED 'SEQBUF'
6723  011366  012702  173000                   MOV     #173000,        R2      ;+POINT TO FIRST ROM
6724  011372  012700  000001                   MOV     #1,     R0              ;+
6725  011376  000241                   1$:     CLC                             ;+
6726  011400  006100                           ROL     R0                      ;+POINT TO NEXT ROM
6727  011402  030067  167510                   BIT     R0,     ROMFIN          ;+IS IT A HOLE?
6728  011406  001404                           BEQ     3$                      ;+BRANCH IF YES
6729  011410  030067  000514           2$:     BIT     R0,     CONFIN          ;+IS IT A CONTINUATION CHIP?
6730  011414  001011                           BNE     4$                      ;+BRANCH IF YES
6731  011416  011213                           MOV     (R2),   (R3)            ;+ITS A DEVICE ROM-PUT DEVICE CODE INTO TABLE
6732  011420  062703  000004           3$:     ADD     #4,     R3              ;+POINT TO NEXT DEVICE CODE LOCATION
6733  011424  062702  000200           5$:     ADD     #200,   R2              ;+POINT TO NEXT ROM
6734  011430  030027  000020                   BIT     R0,     #20             ;+ALL DONE?
6735  011434  001760                           BEQ     1$                      ;+NO-BRANCH
6736  011436  000404                           BR      ALPHCK                  ;+TABLE COMPLETELY BUILT
6737  011440  005243                   4$:     INC     -(R3)                   ;+COUNT ONE CONTINUATION CHIP IN LOCATION
6738  011442  062703  000002                   ADD     #2,     R3              ;+POINT TO NEXT DEVICE CODE LOCATION
6739  011446  000766                           BR      5$                      ;+
6740
6741                                    ;+NOW CHECK DEVICE CODES IN 'SEQBUF' FOR ALPHABETICAL ORDER AND
6742                                    ;+CHECK FOR 'HOLES'.  IF THERE IS A DUPLICATE DEVICE CODE, PRINT
6743                                    ;+AN ERROR MESSAGE AND EXIT.  IF DEVICE CODES NEED SORTING
6744                                    ;+AND/OR HOLES FILLING, DO IT AND SET THE SEQUENCE ERROR
6745                                    ;+FLAG.  ALSO CHECK FOR THE 11/60 SPECIAL CASE.
6746
6747  011450  012703  012100           ALPHCK: MOV     #SEQBUF,        R3      ;+SET UP TO CHECK ALPHABETIC SEQUENCE IN 'SEQBUF
6748  011454  010304                   1$:     MOV     R3,     R4              ;+WILL BE COMPARING (R3) TO (R4)
6749  011456  062704  000004           2$:     ADD     #4,     R4              ;+WHERE R3 AND R4 POINT TO DEVICE CODE LOCATIONS
6750  011462  020427  012114                   CMP     R4,     #ENDSEQ         ;+PAST THE LAST DEVICE CODE?
6751  011466  101010                           BHI     3$                      ;+BRANCH IF YES
6752  011470  005713                           TST     (R3)                    ;+IS THERE A DEVICE CODE HERE?
6753  011472  001446                           BEQ     7$                      ;+BRANCH IF NO
6754  011474  005714                           TST     (R4)                    ;+IS THERE A DEVICE CODE HERE?
6755  011476  001767                           BEQ     2$                      ;+BRANCH IF NO
6756  011500  021314                           CMP     (R3),   (R4)            ;+IS THE SEQUENCE CORRECT?
6757  011502  001410                           BEQ     4$                      ;+BRANCH IF NO-FOUND A DUPLICATE
6758  011504  003025                           BGT     5$                      ;+BRANCH IF NO-NEED TO DO A SHIFT
6759  011506  000763                           BR      2$                      ;+YES-CONTINUE
6760  011510  062703  000004           3$:     ADD     #4,     R3              ;+POINT TO NEXT DEVICE CODE
6761  011514  020327  012114                   CMP     R3,     #ENDSEQ         ;+ALL DONE?
6762  011520  103056                           BHIS    9$                      ;+BRANCH IF YES
6763  011522  000754                           BR      1$                      ;+NOT YET
6764  011524  012767  002000  167356   4$:     MOV     #DUPERR,        ROMERR  ;+SET ERROR FLAG
6765  011532  012767  007634  000374           MOV     #MES1,  ARG1            ;+PASS THIS ADDRESS TO 'ERRHAN'
6766  011540  000314                           SWAB    (R4)                    ;+BUILD THE MSG
6767  011542  011467  176066                   MOV     (R4),   MES1            ;+
6768  011546  105067  176064                   CLRB    MES1+2                  ;+
```

```
6769  011552  004767  172212            JSR     PC,     ERRHAN          ;+REPORT THE ERROR
6770  011556  000207                    RTS     PC                      ;+
6771  011560  010301            5$:     MOV     R3,     R1              ;+SET UP FOR THE SHUFFLE
6772  011562  010402                    MOV     R4,     R2              ;+
6773  011564  011200            6$:     MOV     (R2),   R0              ;+SHIFT THE VALUES
6774  011566  011122                    MOV     (R1),   (R2)+           ;+
6775  011570  010021                    MOV     R0,     (R1)+           ;+
6776  011572  011200                    MOV     (R2),   R0              ;+
6777  011574  011112                    MOV     (R1),   (R2)            ;+
6778  011576  010011                    MOV     R0,     (R1)            ;+
6779  011600  012767  000200  167302    MOV     #SEQERR,        ROMERR  ;+SET THE SEQUENCE ERROR FLAG
6780  011606  000723                    BR      2$                      ;+CONTINUE SORTING
6781  011610  010305            7$:     MOV     R3,     R5              ;+SET UP TO SEE IF WE HAVE A 'HOLE'
6782  011612  020527  012114    8$:     CMP     R5,     #ENDSEQ         ;+END OF TABLE?
6783  011616  103017                    BHIS    9$                      ;+YES-THIS WAS NOT A 'HOLE'
6784  011620  062705  000004            ADD     #4,     R5              ;+NO-POINT TO NEXT DEVICE CODE LOCATION
6785  011624  005715                    TST     (R5)                    ;+IS THERE A ROM HERE?
6786  011626  001771                    BEQ     8$                      ;+BRANCH IF NO
6787  011630  032767  000032  167260    BIT     #32,    ROMFIN          ;+A ROM IN SOCKET #2 ONLY?
6788  011636  001004                    BNE     11$                     ;+BRANCH IF NO
6789  011640  105737  173224            TSTB    @#173224                ;+11/60 SPECIAL CASE?
6790  011644  001401                    BEQ     11$                     ;+BRANCH IF NO
6791  011646  000422                    BR      10$                     ;+THE ONLY ROM MUST STAY IN SOCKET #2
6792  011650  010301           11$:      MOV     R3,     R1              ;+YES-THERE IS A HOLE THAT CAN BE FILLED
6793  011652  010502                    MOV     R5,     R2              ;+GET READY TO SHIFT THE VALUES
6794  011654  000743                    BR      6$                      ;+FILL IN THE HOLE
6795  011656  032767  000002  000242    BIT     #2,     FLAG            ;+AN 11/60?
6796  011664  001413                    BEQ     10$                     ;+BRANCH IF NO
6797  011666  005767  000212            TST     SEQBUF+4                ;+IS SOCKET #2 EMPTY?
6798  011672  001366                    BNE     11$                     ;+BRANCH IF NO
6799  011674  005767  000200            TST     SEQBUF                  ;+IS SOCKET #1 EMPTY?
6800  011700  001405                    BEQ     10$                     ;+BRANCH IF YES
6801  011702  012701  012100            MOV     #SEQBUF,        R1      ;+SET UP TO SHIFT
6802  011706  012702  012104            MOV     #SEQBUF+4,      R2      ;+CHIPS #1 AND #2
6803  011712  000724                    BR      6$                      ;+DO THE SHIFT
6804  011714  032767  000200  167166    BIT     #SEQERR,        ROMERR  ;+WAS THERE A SEQUENCE ERROR?
6805  011722  001001                    BNE     OUTTAB                  ;+BRANCH IF YES
6806  011724  000207                    RTS     PC                      ;+
6807
6808                                    ;+THE FOLLOWING ROUTINE BUILDS THE CORRECT SEQUENCE MSG TO BE
6809                                    ;+PRINTED AS AN ERROR MSG BY 'ERRHAN'
6810
6811  011726  012767  007634  000200  OUTTAB: MOV     #MES1,  ARG1            ;+PASS MSG ADR. TO 'ERRHAN'
6812  011734  012767  000015  000174          MOV     #CR,    ARG2            ;+PASS THIS TOO
6813  011742  012700  012100                  MOV     #SEQBUF,        R0      ;+
6814  011746  012702  012144                  MOV     #MESTAB,        R2      ;+
6815  011752  012704  007634                  MOV     #MES1,  R4              ;+START BUILDING THE ERROR MSG
6816  011756  012705  012415                  MOV     #ERMES4,        R5      ;+PUT IN THE FIRST PART
6817  011762  004767  172424                  JSR     PC,     FILBUF          ;+
6818  011766  000015                          CR                              ;+
6819  011770  005710                          TST     (R0)                    ;+IS A ROM IN SOCKET #1?
6820  011772  001004                          BNE     1$                      ;+BRANCH IF YES
6821  011774  062700  000004                  ADD     #4,     R0              ;+DO SOCKET #2 ONLY
6822  012000  062702  000002                  ADD     #2,     R2              ;+
6823  012004  012767  010234  000126  1$:     MOV     #MES2,  BUILD           ;+SET UP THE DEVICE CODE PART
6824  012012  000310                          SWAB    (R0)                    ;+
```

```
6825  012014  012067  176214                  MOV     (R0)+,  MES2            ;+
6826  012020  105067  176212                  CLRB    MES2+2                  ;+
6827  012024  012205                  2$:     MOV     (R2)+,  R5              ;+IDENTIFY THE ROM #
6828  012026  004767  17236C                  JSR     PC,     FILBUF          ;+
6829  012032  000015                          CR                              ;+
6830  012034  016705  000100                  MOV     BUILD,  R5              ;+PUT IN EITHER DEVICE CODE OR CONTINUATION MSG
6831  012040  004767  172346                  JSR     PC,     FILBUF          ;+
6832  012044  000011                          HT                              ;+
6833  012046  005720                          TST     (R0)+                   ;+ANY CONTINUATION ROMS?
6834  012050  001405                          BEQ     3$                      ;+BRANCH IF NO
6835  012052  005340                          DEC     -(R0)                   ;+COUNT DOWN ONE ROM
6836  012054  012767  012154  000056          MOV     #CONMES,        BUILD   ;+SET UP FOR CONTINUATION MSG
6837  012062  000760                          BR      2$                      ;+CONTINUE
6838  012064  005710                  3$:     TST     (R0)                    ;+ANY MORE DEVICE ROMS?
6839  012066  001346                          BNE     1$                      ;+BRANCH IF YES
6840  012070  105014                          CLRB    (R4)                    ;+MUST END WITH 0 BYTE
6841  012072  004767  171672                  JSR     PC,     ERRHAN          ;+REPORT THE ERROR
6842  012076  000207                          RTS     PC                      ;+
6843
6844
6845  012100  000000                  SEQBUF: .WORD   0                       ;+FIRST DEVICE CODE
6846  012102  000000                          .WORD   0                       ;+# OF CONTINUATION CHIPS FOR FIRST DEVICE
6847  012104  000000                          .WORD   0                       ;+SECOND DEVICE CODE, ETC
6848  012106  000000                          .WORD   0                       ;+
6849  012110  000000                          .WORD   0                       ;+
6850  012112  000000                          .WORD   0                       ;+
6851  012114  000000                  ENDSEQ: .WORD   0                       ;+FOURTH (LAST) DEVICE CODE
6852  012116  000000                          .WORD   0                       ;+
6853  012120  000000                  CONER1: .WORD   0                       ;+EXTRA CONTINUATION CHIP ERROR FLAG
6854  012122  000000                  CONER2: .WORD   0                       ;+MISSING CONTINUATION CHIP FOR THIS DEVICE
6855  012124  000000                  RETURN: .WORD   0                       ;+ALTERNATE RETURN ADDRESS
6856  012126  000000                  FLAG:   .WORD   0                       ;+BIT 1:11/60
6857  012130  000000                  CONFIN: .WORD   0                       ;+CONTINUATION ROM FOUND INDICATOR
6858  012132  000000                  DEVFIN: .WORD   0                       ;+DEVICE ROM FOUND INDICATOR
6859  012134  000000                  ARG1:   .WORD   0                       ;+PASSES MSG ADR. TO 'ERRHAN'
6860  012136  000000                  ARG2:   .WORD   0                       ;+PASSES CR OR HT TO 'ERRHAN'
6861  012140  000000                  BUILD:  .WORD   0                       ;+
6862  012142  000000                  FINISH: .WORD   0                       ;+ALTERNATE RETURN ADDRESS
6863  012144  012442                  MESTAB: ROM1                            ;+ASCII MSG TABLE
6864  012146  012456                          ROM2                            ;+
6865  012150  012472                          ROM3                            ;+
6866  012152  012506                          ROM4                            ;+
6867  012154  051511  040440  041440  CONMES: .ASCIZ /IS A CONTINUATION ROM/
      012162  047117  044524  052516
      012170  052101  047511  020116
      012176  047522  000115
6868  012202  005015  020101  047503  ERMES1: .ASCIZ <15><12>/A CONTINUATION ROM IS INCORRECTLY LOCATED IN/
      012210  052116  047111  040525
      012216  044524  047117  051040
      012224  046517  044440  020123
      012232  047111  047503  051122
      012240  041505  046124  020131
      012246  047514  040503  042524
      012254  020104  047111    000
6869  012261     015  040412  041440  ERMES2: .ASCIZ <15><12>/A CONTINUATION ROM IS MISSING FOR DEVICE CODE/
      012266  047117  044524  052516
```

```
                012274  052101  047511  020116
                012302  047522  020115  051511
                012310  046440  051511  044523
                012316  043516  043040  051117
                012324  042040  053105  041511
                012332  020105  047503  042504
                012340     000
 6870           012341     015  052012  042510  ERMES3: .ASCIZ <15><12>/THERE IS A DUPLICATE ROM WITH DEVICE CODE/
                012346  042522  044440  020123
                012354  020101  052504  046120
                012362  041511  052101  020105
                012370  047522  020115  044527
                012376  044124  042040  053105
                012404  041511  020105  047503
                012412  042504     000
 6871           012415     123  050505  042525  ERMES4: .ASCIZ /SEQUENCE SHOULD BE:/<12>
                012422  041516  020105  044123
                012430  052517  042114  041040
                012436  035105  000012
 6872           012442  047522  020115  024061  ROM1:   .ASCIZ /ROM 1(E35) /
                012450  031505  024465  000040
 6873           012456  047522  020115  024062  ROM2:   .ASCIZ /ROM 2(E33) /
                012464  031505  024463  000040
 6874           012472  047522  020115  024063  ROM3:   .ASCIZ /ROM 3(E34) /
                012500  031505  024464  000040
 6875           012506  047522  020115  024064  ROM4:   .ASCIZ /ROM 4(E32) /
                012514  031505  024462  000040
 6876
 6877
 6878                   000001                   .END
```

```
ABASE = 000000          5984
ACDW1 = 000000          5984
ACDW2 = 000000          5984
ACPUOP= 000000          5984
ADDW0 = 000000          5984
ADDW1 = 000000          5984
ADDW10= 000000          5984
ADDW11= 000000          5984
ADDW12= 000000          5984
ADDW13= 000000          5984
ADDW14= 000000          5984
ADDW15= 000000          5984
ADDW2 = 000000          5984
ADDW3 = 000000          5984
ADDW4 = 000000          5984
ADDW5 = 000000          5984
ADDW6 = 000000          5984
ADDW7 = 000000          5984
ADDW8 = 000000          5984
ADDW9 = 000000          5984
ADEVCT= 000000          5984
ADEVM = 000000          5984
AENV  = 000000          5984
AENVM = 000000          5984
AFATAL= 000000          5984
ALPHCK  011450          6736    6747#
AMADR1= 000000          5984
AMADR2= 000000          5984
AMADR3= 000000          5984
AMADR4= 000000          5984
AMAMS1= 000000          5984
AMAMS2= 000000          5984
AMAMS3= 000000          5984
AMAMS4= 000000          5984
AMSGAD= 000000          5984
AMSGLG= 000000          5984
AMSGTY= 000000          5984
AMTYP1= 000000          5984
AMTYP2= 000000          5984
AMTYP3= 000000          5984
AMTYP4= 000000          5984
APASS = 000000          5984
APRIOR= 000000          5984
APTCSU= 000040          6539    6540#
APTENV= 000001          6539    6540#
APTER1= 000001          5951#   6169
APTER2= 000002          5952#   6181
APTER3= 000004          5953#   6192
APTER4= 000010          5954#   6198
APTSIZ= 000200          5986    6540#
APTSPO= 000100          6539    6540#
ARG1    012134          6407    6701*   6706*   6765*   6811*   6859#
ARG2    012136          6406    6686*   6812*   6860#
AROUND  004322          6443    6446    6448#
ASWREG= 000000          5984
ATESTN= 000000          5984
```

```
AUNIT = 000000          5984
AUSWR = 000000          5984
AVECT1= 000000          5984
AVECT2= 000000          5984
BELL    006536          6387    6557#
BIT0  = 000001          5947#
BIT00 = 000001          5947#
BIT01 = 000002          5947#
BIT02 = 000004          5947#
BIT03 = 000010          5947#
BIT04 = 000020          5947#
BIT05 = 000040          5947#
BIT06 = 000100          5947#
BIT07 = 000200          5947#
BIT08 = 000400          5947#
BIT09 = 001000          5947#
BIT1  = 000002          5947#
BIT10 = 002000          5947#   6384
BIT11 = 004000          5947#
BIT12 = 010000          5947#
BIT13 = 020000          5947#   6388
BIT14 = 040000          5947#
BIT15 = 100000          5947#   6454
BIT2  = 000004          5947#
BIT3  = 000010          5947#
BIT4  = 000020          5947#
BIT5  = 000040          5947#
BIT6  = 000100          5947#
BIT7  = 000200          5947#
BIT8  = 000400          5947#
BIT9  = 001000          5947#
BPTVEC= 000014          5947#
BRHEAD  006637          6319    6562#
BUF1    007540          5988    6161    6175    6229    6309    6582#
BUF2    007620          6189    6195    6285*   6286*   6292*   6293*   6349    6353    6585#
BUILD   012140          6823*   6830    6836*   6861#
CALSUM  002532          6074    6087    6110#
CHECKS  002314          6028    6068#
CKSWR = 104406          6546#
CLEAR   010734          5994    6610#
CONER1  012120          6618*   6687    6689    6853#
CONER2  012122          6627*   6698    6702*   6703    6854#
CONFIN  012130          6015*   6615    6625    6729    6857#
CONMES  012154          6654    6836    6867#
CONONE= 000400          5959#   6691
CONTWO= 001000          5960#   6700
CON1    010752          6241    6615#
CON2    011000          6261    6622#
CR    = 000015          5947#   6314    6321    6326    6352    6360    6401    6436    6484    6539    6670    6812    6818
                        6829
CRCERR= 000020          5955#   6078    6091
CRCLOP  002550          6115#   6124
CRCMSG  007265          6466    6571#
CRLF  = 000200          5947#   5987    6539
DAFLAG  003324          6236*   6245    6263*   6269#
DDISP = 177570          5947#   5976    5986
```

```
DEVCOD  003112          6148    6160    6229#
DEVFIN  012132          6019*   6239    6858#
DISPLA  001106          5976#   5986*
DISPRE  000174          5964#   5986
DRHEAD  006616          6312    6561#
DSWR    = 177570        5947#   5975    5986
DUPERR= 002000          5961#   6764
EHEADT  004364          6393    6462#
EMTVEC= 000030          5947#
END     011106          6636    6642    6644#
ENDSEQ  012114          6750    6761    6782    6851#
ERMES1  012202          6470    6868#
ERMES2  012261          6471    6869#
ERMES3  012341          6472    6870#
ERMES4  012415          6816    6871#
ERRCNT  001120          5981#   6383*
ERRHAN  003770          6050    6079    6092    6154    6170    6182    6193    6199    6381#   6707    6769    6841
ERRMSG  010334          6382    6449    6453    6590#
ERRVEC= 000004          5947#   5986*   5995    5996*   6008*
ER1MSG  007075          6462    6566#
ER2MSG  007041          6463    6565#
ER3MSG  007133          6464    6567#
ER4MSG  007211          6465    6569#
EXCADD  002624          6072*   6081*   6094*   6125    6134#
FILBUF  004412          6313    6320    6331    6338    6351    6356    6400    6408    6430    6434    6480#   6655    6669
                        6817    6828    6831
FILTAB  011362          6699    6722#
FINISH  012142          6325*   6612    6648    6862#
FIRSTA  002622          6073*   6080*   6093*   6097    6111    6133#
FIRSTB  001114          5979#   6010    6237
FLAG    012126          6643*   6795    6856#
GNS     = ****** U      5964    5987    6546
GTSWR   = 104405        5987    6546#
HOLCK1  011044          6284    6635#
HOLCK2  011056          6291    6638#
HT      = 000011        5947#   6332    6339    6343    6357    6431    6435    6480    6482    6539    6656    6672    6686
                        6832
IOTVEC= 000020          5947#
LASTAD  002626          6071*   6082*   6095*   6128    6135#
LF      = 000012        5947#   6327    6361    6437    6485    6539
LOOP    002542          6113#   6129
MESSAG  001112          5978#   6158*
MESTAB  012144          6323    6690    6814    6863#
MES1    007634          6150    6588#   6701    6703*   6704*   6765    6767*   6768*   6811    6815
MES2    010234          6157    6589#   6823    6825*   6826*
MNORM   006542          6043    6558#
NODATA  002312          6045    6060#
NOROME= 000100          5957#   6049
NOROMM  007367          6468    6573#
NOROMS  002226          6026    6043#
OCADD   004560          6330    6337    6355    6529#
OCASC   004454          6429    6433    6499#   6531
OCTBUF  007624          5990    6499    6504    6511    6513    6587#
OUTTAB  011726          6805    6811#
OVER1   004032          6389    6391#
OVER10  004342          6390    6454#
```

```
OVER11  004334            6441    6447    6452#
OVER2   004300            6440    6442#
OVER9   004330            6392*   6450#
PFERR = 000040            5956#   6152    6295    6402    6444
PFHEAD  006774            6350    6564#
PFMSG   007311            6467    6572#
PIRQ  = 177772            5947#
PIRQVE= 000240            5947#
PPFVAR  003326            6151    6187    6281#
PROMP   002630            6031    6146#
PR0   = 000000            5947#
PR1   = 000040            5947#
PR2   = 000100            5947#
PR3   = 000140            5947#
PR4   = 000200            5947#
PR5   = 000240            5947#
PR6   = 000300            5947#
PR7   = 000340            5947#
PS    = 177776            5947#
PSW   = 177776            5947#
PUTMES  003436            6149    6156    6304#
PWRVEC= 000024            5947#
RDCHR = 104407            6541    6546#
RDLIN = 104410            6546#
RESREG= 104412            6458    6546#
RESVEC= 000010            5947#
RETURN  012124            6283*   6290*   6324*   6644    6658    6855#
ROMCNT  002530            6068*   6069    6084*   6085    6101#   6415
ROMERR  001110            5977#   6049*   6078*   6091*   6152    6169*   6181*   6192*   6198*   6295*   6391    6392    6402
                          6404    6411    6413    6444    6457*   6691*   6700*   6764*   6779*   6804
ROMFIN  001116            5980#   5992*   6004*   6016*   6018*   6024    6069    6085    6231    6442    6635    6638    6664
                          6727    6787
ROMNM   011166            6652    6653    6660#
ROMNUM  006524            6416    6552#
ROMTYP  011114            6328    6646#
ROM1    012442            6863    6872#
ROM2    012456            6864    6873#
ROM3    012472            6865    6874#
ROM4    012506            6866    6875#
RSTART  002066            5967    6024#   6033
SAVREG= 104411            6381    6546#
SEQBUF  012100            6610    6722    6747    6797    6799    6801    6802    6813    6845#
SEQERR= 000200            5958#   6779    6804
SEQMSG  007445            6469    6575#
SEQTST  011234            6032    6686#
STACK = 001100            5947#   5986
START   001400            5964    5986#
STKLMT= 177774            5947#
SWR     001104            5975#   5986*   5987    6384    6388    6454    6541*
SWREG   000176            5964#   5986    5987    6541
SW0   = 000001            5947#
SW00  = 000001            5947#
SW01  = 000002            5947#
SW02  = 000004            5947#
SW03  = 000010            5947#
SW04  = 000020            5947#
```

```
SW05  = 000040          5947#
SW06  = 000100          5947#
SW07  = 000200          5947#
SW08  = 000400          5947#
SW09  = 001000          5947#
SW1   = 000002          5947#
SW10  = 002000          5947#
SW11  = 004000          5947#
SW12  = 010000          5947#
SW13  = 020000          5947#
SW14  = 040000          5947#
SW15  = 100000          5947#
SW2   = 000004          5947#
SW3   = 000010          5947#
SW4   = 000020          5947#
SW5   = 000040          5947#
SW6   = 000100          5947#
SW7   = 000200          5947#
SW8   = 000400          5947#
SW9   = 001000          5947#
TBITVE= 000014          5947#
TEMP    004556          6505*  6507*  6515*  6517*  6519*  6522#
TESTAD  003322          6010*  6011   6013   6020*  6022   6233*  6234  6235  6237*  6242*  6247  6248  6249*
                        6251   6252*  6253   6264   6268#
TIMES   003766          5993*  6305   6308*  6368#
TKVEC = 000060          5947#
TPVEC = 000064          5947#
TRAPVE= 000034          5947#  5986*
TRTVEC= 000014          5947#
TYPE  = 104401          5987   6033   6043   6364   6386   6539   6541  6542  6546#
TYPOC = 104402          6541   6546#
TYPON = 104404          6546#
TYPOS = 104403          6546#
$APTHD  001122          5983#
$ASTAT= ****** U        6540
$ATYC   005116          6540#
$ATY1   005072          6448   6540#
$ATY3   005100          6452   6539   6540#
$ATY4   005110          6540#
$AUTOB  001100          5972#  5987*  6541
$BASE   001212          5984#  6188
$CDW1   001216          5984#  6194
$CDW2   001220          5984#
$CHARC  005052          6539#*
$CKSWR  005344          6541#  6546
$CMTAG= ****** U        5986
$CNTLG  006071          6541#
$CNTLU  006064          6541#
$CPUOP  001164          5984#
$CRLF   005067          6539#  6541
$DDW0   001222          5984#  6162   6174
$DDW1   001224          5984#
$DDW10  001246          5984#
$DDW11  001250          5984#
$DDW12  001252          5984#
$DDW13  001254          5984#
```

```
$DDW14  001256            5984#
$DDW15  001260            5984#
$DDW2   001226            5984#
$DDW3   001230            5984#
$DDW4   001232            5984#
$DDW5   001234            5984#
$DDW6   001236            5984#
$DDW7   001240            5984#
$DDW8   001242            5984#
$DDW9   001244            5984#
$DEVCT  001146            5984#
$DEVM   001214            5984#
$DOAGN  002204            6033#
$ENDAD  002174            5969    5987    6033#
$ENDCT  002150            5986    6033#
$ENDMG  002213            6033#
$ENULL  002210            6033#
$ENV    001156            5984#   5987    6439    6539    6540
$ENVM   001157            5984#   5986    6146    6539    6540
$EOP    002124            6027    6030    6033#
$EOPCT  002142            5986*   6033#
$ETABL  001156            5984#
$ETEND  001262            5983    5984#
$FATAL  001140            5984#   6540*
$FFLG   005336            6540#*
$FILLC  005064            6539#
$FILLS  005063            6539#
$GET42  002164            6033#
$GTSWR  005414            6541#   6546
$HD   = 000003            5945
$HIBTS  001122            5983#
$INTAG  001101            5973#   6541
$LF     005070            6539#   6541
$LFLG   005335            6540#*
$MADR1  001170            5984#
$MADR2  001174            5984#
$MADR3  001200            5984#
$MADR4  001204            5984#
$MAIL   001136            5983    5984#   5986    5987    6539
$MAMS1  001166            5984#
$MAMS2  001172            5984#
$MAMS3  001176            5984#
$MAMS4  001202            5984#
$MBADR  001124            5983#
$MFLG   005334            6540#*
$MNEW   006107            6541#
$MSGAD  001152            5984#   6540*
$MSGLG  001154            5984#   6540*
$MSGTY  001136            5984#   6540*
$MSWR   006076            6541#
$MTYP1  001167            5984#
$MTYP2  001173            5984#
$MTYP3  001177            5984#
$MTYP4  001203            5984#
$NULL   005062            6539#
$OCNT   006342            6542#*
```

```
$OMODE  006344          6542#*
$PASS   001144          5984#   5986*   6029    6033*
$PASTM  001130          5983#
$QUES   005066          6539#   6541
$RDCHR  005626          6541#   6546
$RDDEC= ****** U        6546
$RDLIN  005746          6541#   6546
$RDOCT= ****** U        6546
$RDSZ = 000010          6541#
$RESRE  006404          6543#   6546
$RTNAD  002206          6033#
$R2A  = ****** U        6546
$SAVRE  006346          6543#   6546
$SETUP= 000124          5965#   5986    5987    6033    6541
$STUP = 177777          5965#
$SVPC = 000210          5969#
$SWR  = 160000          5945#   5946    5986    6033
$SWREG  001160          5984#   5986
$TESTN  001142          5984#
$TKB    005342          6541#
$TKS    005340          6541#*
$TN   = 000001          5945#
$TPB    005060          6539#*
$TPFLG  005065          6539#
$TPS    005056          6539#
$TRAP   006442          5986    6546#
$TRAP2  006464          6546#
$TRP  = 000013          6546#
$TRPAD  006476          6546#
$TSTM   001126          5983#
$TTYIN  006054          6541#
$TYPBN= ****** U        6546
$TYPDS= ****** U        6546
$TYPE   004574          6539#   6540    6546
$TYPEC  005006          6539#   6541
$TYPEX  005054          6539#
$TYPOC  006144          6542#   6546
$TYPON  006160          6542#   6546
$TYPOS  006120          6542#   6546
$UNIT   001150          5984#
$UNITM  001132          5983#
$USWR   001162          5984#
$VECT1  001206          5984#
$VECT2  001210          5984#
$$GET4= 000000          6033#
$OFILL  006343          6542#*
.     = 012522          5963#   5964#   5966#   5969#   5971#   5983#   5985#   6033    6539    6540#   6541#   6587#   6588#
                        6589#   6590#
.$ASTA= ****** U        6540
.$X   = 001122          5983#
```

```
COMMEN   1526#  5947#
ENDCOM   1538#  5947#
ERROR    5947#
ESCAPE   1654#  5947#
GETPRI   1278#  5947#
GETSWR   1725#  5947#  5987#
MULT     4393#  5947#
NEWTST   1585#  5947#
POP      2103#  5943#  5947#  6540  6543  6544
PUSH     2095#  5943#  5947#  6540  6543  6545
REPORT   5352#  5943#  5947#
SCOPE    5947#
SETPRI   1246#  5947#
SETTRA   6546#
SETUP    1302#  5942#  5947#  5986
SKIP     1688#  5947#
SLASH    1478#  5947#
SOFTSW   5944#
SPACE    5947#
STARS    1447#  5947#  5948  5950  5969  5983  5984  6033  6539  6540  6541  6542  6543  6546
SWRSU    1416#  5947#  5986#
TRMTRP   6546#
TYPBIN   2039#  5947#
TYPDEC   2009#  5947#
TYPNAM   1779#  5943#  5947#  5987
TYPNUM   1976#  5947#
TYPOCS   1929#  5947#
TYPOCT   1892#  5947#  6541
TYPTXT   1846#  5947#
$$ESCA   1667#  5947#
$$NEWT   1621#  5947#
$$SET    6546#
$$SETM   5986#
$$SKIP   1701#  5947#
.EQUAT    176#  5942#  5947
.HEADE     50#  5943#  5945
.KT11     319#
.SETUP   1180#  5941#  5965
.STARS   5942#
.SWRHI     92#  5943#  5946
.SWRLO   5946#
.$ACT1   4961#  5943#  5969
.$APTB   5005#  5942#  5984
.$APTH   5261#  5942#  5983
.$APTY   5436#  5942#  6540
.$ASTA   5307#  5941#
.$CATC    905#  5941#  5964
.$CMTA   1016#  5941#
.$DB2D   4591#
.$DB2O   4714#
.$DIV    4494#
.$EOP    2162#  5941#  6033
.$ERRO   2643#
.$ERRT   2838#
.$MULT   4431#
.$POWE   4143#
```

```
.$RAND   4218#
.$RDDE   3814#
.$RDOC   3723#
.$READ   3328#   5944#   6541
.$R2AZ   4858#
.$SAVE   3889#   5941#   6543
.$SB2D   4675#
.$SB2O   4776#
.$SCOP   2397#
.$SIZE   4271#
.$SUPR   4814#
.$TRAP   3991#   5943#   6546
.$TYPB   3221#
.$TYPD   3144#
.$TYPE   2925#   5941#   6539
.$TYPO   3048#   5944#   6542
.$40CA    944#
.1170     498#


. ABS.  012522     000


 ERRORS DETECTED:  0

 CZM9BC.BIN,CZM9BC.SEQ/CRF/NL:TOC=CZM9BC.SML,CZM9BC.P11
 RUN-TIME: 10 13 .8 SECONDS
 RUN-TIME RATIO: 122/24=4.8
 CORE USED:  32K  (63 PAGES)
```