M9312

BOOTSTRAP TERMINATOR 8K

CZM9BA0

AH-E058A-MC

COPYRIGHT © 1978

FICHE 1 OF 1

APR 1978

digital

MADE IN USA



.REM !

IDENTIFICATION

PRODUCT CODE: AC-E057A-MC

PRODUCT NAME: CZM9BA0 M9312.BOOT.TERMR.8K

DATE: MARCH, 1978

MAINTAINER: DIAGNOSTIC GROUP

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DITITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT. THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE SUED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1978 BY DIGITAL EQUIPEMNT CORPORATION

1.0 ABSTRACT

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

2.0 REQUIREMENTS

2.1 HARDWARE

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

2.2 SOFTWARE

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

3.0 LOADING AND STARTING PROCEDURES

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

3.2 STARTING ADDRESS

200- DO CRC VERIFICATION AND SIZING

3.3 RESTART ADDRESS

204- RESTART WITHOUT SIZING

3.4 SPECIAL ENVIRONMENTS

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

4.0 OPERATING PROCEDURES

4.1 OPERATIONAL SWITCH SETTINGS

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

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

4.2 EXECUTION TIMES

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

4.3 APT PROCEDURES

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

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

5.0 SUBROUTINE ABSTRACTS

5.1 NOROMS

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

5.2 CHECKS

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

5.3 CALSUM

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

5.4 PROMP

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

5.5 DEVCOD

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

5.6 PPFVAR

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

5.7 PUTMES

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

5.8 ERRHAN

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

5.9 FILBUF

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

5.10 OCASC

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

5.11 OCADD

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

6.0 RELIABILITY//AVAILABILITY/SERVICEABILITY

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

CRC ERROR IN ROM EXX

WHEN RUNNING IN THE APT ENVIRONMENT THREE OTHER ERRORS MAY OCCUR.

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

   1. COULD NOT FIND DEVICE CODE XX.

   2. FOUND UNEXPECTED DEVICE CODE XX.

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

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

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

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

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

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

!

```
                        .ENABLE ABS
                        .LIST   ME
                        .NLIST  MC,MD,CND
                        .TITLE  CZM9BA0 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          ;;HARDWARE DISPLAY REGISTER

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

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

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

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

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

000000                          .=0
                        .SBTTL  TRAP CATCHER

000000                          .=0
                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
000174                  .=174
000174  000000  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                        .SBTTL  STARTING ADDRESS(ES)
000200  000137  001400          JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204
000204  000167  001634          JMP     RSTART

                        .SBTTL  ACT11 HOOKS

                        ;;*****************************************************************
                        ;HOOKS REQUIRED BY ACT11
000210                          $SVPC=.                 ;SAVE PC
000046                          .=46
000046  002152                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052                          .=52
```

```
                        .WORD   0               ;;2)SET LOC.52 TO ZERO
                        .=$SVPC                 ;; RESTORE PC

                        .=1100
$AUTOB: .BYTE   0
$INTAG: .BYTE   0
        .WORD   0
SWR:    .WORD   DSWR
DISPLAY: .WORD  DDISP
ROMERR: 0
MESSAG: 0
FIRSTB: 173000
ROMFIN: 0
ERRCNT: 0

.SBTTL  APT PARAMETER BLOCK

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

$APTHD:
$HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD   2.      ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD   2.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
.SBTTL  APT MAILBOX-ETABLE

;;*****************************************************************
.EVEN
$MAIL:                  ;;APT MAILBOX
$MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
$FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
$TESTN: .WORD   ATESTN  ;;TEST NUMBER
$PASS:  .WORD   APASS   ;;PASS COUNT
$DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
$UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
$MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
$MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
$ETABLE:                ;;APT ENVIRONMENT TABLE
$ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
$ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
$SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
$USWR:  .WORD   AUSWR   ;;USER SWITCHES
$CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
```

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


001262                  $SETEND:

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

                        ;;INITIALIZE A FEW VECTORS
001404  012737  006430  000034  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001412  012737  000340  000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
```

```
                                        CLR     $PASS           ;;CLEAR THE PASS COUNT
                                        MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
                                        MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
                                        MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
                                        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
                                        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
                                        CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
                                        BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
                                        BR      65$             ;;BRANCH IF NO TIMEOUT
                                64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
                                        RTI
                                65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
                                        MOV     #DISPREG,DISPLAY
                                66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                                        CLR     $PASS           ;;CLEAR PASS COUNT
                                        BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
                                        BEQ     67$             ;;YES,USE NON-APT SWITCH
                                        MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
                                67$:
                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
                                        INC     #-1             ;;FIRST TIME?
                                        BNE     68$             ;;BRANCH IF NO
                                        CMP     #$ENDAD,@#42    ;;ACT-11?
                                        BEQ     68$             ;;BRANCH IF YES
                                        TYPE    ,69$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
                                        TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
                                        BNE     70$             ;;BRANCH IF YES
                                        CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
                                        BEQ     70$             ;;BRANCH IF YES
                                        CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
                                        BNE     71$             ;;BRANCH IF NO
                                        GTSWR                   ;;GET SOFT-SWR SETTINGS
                                        BR      71$
                                70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
                                71$:
                                        BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  <CRLF>*CZM9BA0 M9312 BOOT TERMR 8K*<CRLF>
                                68$:
                                        MOV     #BUF1,          R0              ;CLR SIZING BUFFERS
                                8$:     CLR     (R0)+
                                        CMP     R0,             #OCTBUF         ;HAVE WE CLEARED THE WHOLE
                                        BLT     8$                              ;BUFFER,NO,GO BACK
                                        CLR     @#ROMFIN                        ;INITIALIZE ROM FOUND INDICATORS
                                        CLR     @#TIMES                         ;INITIALIZE ENTRY COUNTER FOR PUTMES SUB
                                        MOV     @#ERRVEC,       -(R6)           ;SAVE CONTENTS OF LOCATION 4
                                        MOV     2$,             @#ERRVEC        ;SET UP FOR POSSIBLE TRAP
                                        CMPB    #-1,            @#165000        ;IF CONTENTS = -1, OR TRAP
                                        BEQ     3$                              ;THEN CPU ROM NOT PLUGGED IN
                                        MOV     #165000,        R0              ;GET FIRST ADDRESS OF ROM SPACE
                                1$:     TST     (R0)+                           ;IF THIS INSTRUCTION TRAPS CPU ROM
```

```
                                                                ;NOT PLUGGED IN
001740  020027  166000          CMP     R0,     #166000       ;HAVE WE CHECKED ALL ADDRESSES
001744  001374                  BNE     1$                    ;IF NO THEN CONTINUE LOOPING
001746  012737  000001  001116  MOV     #1,     @#ROMFIN      ;IF NO TRAPS OR LOW BYTE OF FIRST
                                                                ;ADDRESS NOT EQUAL -1 ASSUME ROM PRESENT
001754  000402                  BR      3$                    ;NO TRAP JUST RESTORE ERRUEL
001756  062716  000004  2$:     ADD     #4,     (R6)          ;IF TRAP CLEAN OFF PC, PSW
001762  012637  000004  3$:     MOV     (R6)+,  @#ERRVEC      ;RESTORE ERRVEC
001766  012700  000002          MOV     #2,     R0            ;INITIALIZE ROM FOUND INDICATOR
001772  013737  001114  003250  MOV     @#FIRSTB, @#TESTAD    ;GET FIRST BOOT
002000  132777  000200  001242  5$:     BITB    #200,   @TESTAD ;IF LOW BYTE HAS BIT 7 SET
002006  001404                  BEQ     7$                    ;DO NOT SET ROM FOUND INDICATOR
002010  022777  177776  001232  CMP     #-2,    @TESTAD       ;UNLESS, CONTENTS OF ADDRESS
002016  001002                  BNE     4$                    ;EQUAL TO -2.(CONTINUATION ROM)
002020  050037  001116  7$:     BIS     R0,     @#ROMFIN      ;SET ROM FOUND INDICATOR
002024  062737  000200  003250  4$:     ADD     #200,   @#TESTAD ;UPDATE TEST ADDRESS TO NEXT ROM
002032  006100                  ROL     R0                    ;UPDATE ROM FOUND INDICATOR
002034  022737  174000  003250  CMP     #174000, @#TESTAD     ;HAVE CHECKED ALL THE ROMS
002042  001356                  BNE     5$                    ;IF NO CONTINUE CHECKING
002044  005737  001116  RSTART: TST     @#ROMFIN              ;ARE THERE ANY ROMS AVAILABLE
002050  001003                  BNE     6$                    ;IF YES GO TO ROM TEST
002052  004767  000126          JSR     PC,     NOROMS        ;IF NO GO TEST NO ROMS
002056  000411                  BR      $EOP                  ;GO TO END OF PASS
002060  004767  000160  6$:     JSR     PC,     CHECKS        ;GO CALCULATE CHECKSUMS
002064  005737  001144          TST     @#$PASS               ;ARE WE ON 1ST PASS
002070  001004                  BNE     $EOP                  ;IF NO SKIP PROCESS OF PARAMETERS
002072  004767  000462          JSR     PC,     PROMP         ;GO PROCESS ROM PARAMETERS
002076  004767  001544          JSR     PC,     SEQTST        ;GO CHECK SEQUENCE OF DEVICE CODES
                        .SBTTL  END OF PASS ROUTINE

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

002102                  $EOP:
002102  000240                  NOP
002104  005267  177034          INC     $PASS           ;;INCREMENT THE PASS NUMBER
002110  042767  100000  177026  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
002116  005327                  DEC     (PC)+           ;;LOOP?
002120  000001          $EOPCT: .WORD   1
002122  003017                  BGT     $DOAGN          ;;YES
002124  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
002126  000001          $ENDCT: .WORD   1
002130  002120                  $EOPCT
002132  104401  002171          TYPE    ,$ENDMG         ;;TYPE "END PASS"
002136  104401  002166          TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
002142  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
002146  001405                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
002150  000005                  RESET                   ;;CLEAR THE WORLD
002152  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
002154  000240                  NOP                     ;;SAVE ROOM
002156  000240                  NOP                     ;;FOR
```

```
                                        NOP                             ;;ACT11
                                $DOAGN:
                                        JMP     @(PC)+                  ;;RETURN
                                $RTNAD: .WORD   RSTART
                                $ENULL: .BYTE   -1,-1,0                 ;;NULL CHARACTER STRING
                                $ENDMG: .ASCIZ  <15><12>/END PASS/


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

002204  012703  173000        NOROMS: MOV     #173000,        R3              ;GET FIRST ADDRESS TO BE READ
002210  013704  002242                MOV     @#NODATA,       R4              ;GET ADDRESS OF EXPECTED VALUE
002214  022304                1$:     CMP     (R3)+,          R4              ;DOES RECIEVED VALUE EQUAL EXPECTED
002216  001405                        BEQ     2$                              ;IF YES CONTINUE TESTING
002220  052737  000100  001110        BIS     #NOROME,        @#ROMERR        ;IF NO SET ERROR INDICATER
002226  004767  001630                JSR     PC,             ERRHAN          ;REPORT ERROR
002232  022703  174000        2$:     CMP     #174000,        R3              ;HAVE ALL ADDRESSES BEEN TESTED
002236  003366                        BGT     1$                              ;IF NO GO TEST THIS ADDRESS
002240  000207                        RTS     PC

002242  161777                NODATA: .WORD   161777


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

002244  012737  000001  002460  CHECKS: MOV     #1,             @#ROMCNT        ;INITIALIZE RON COUNTER
002252  033737  002460  001116        BIT     @#ROMCNT,       @#ROMFIN        ;IS DIAGNOSTIC ROM PRESENT
002260  001424                        BEQ     1$                              ;IF NO GO CHECKSUM BOOT ROMS
002262  012737  165775  002556        MOV     #165775,        @#LASTAD        ;SET UP LAST ADDRESS TO BE SUMMED
002270  012737  000000  002554        MOV     #0,             @#EXCADD        ;SET UP EXCEPTION ADDRESS
002276  012737  165000  002552        MOV     #165000,        @#FIRSTA        ;SET UP FIRST ADDRESS TO BE SUMMED
002304  004767  000152                JSR     PC              CALSUM          ;GO CALCULATE CHECKSUM
002310  013703  165776                MOV     @#165775,       R3              ;GET EXPECTED CHECKSUM
002314  020304                        CMP     R3,             R4              ;COMPARE CHECKSUMS
002316  001405                        BEQ     1$                              ;IF CHECKSUMS COMPARE CONTINUE
002320  052737  000020  001110        BIS     #CRCERR,        @#ROMERR        ;IF ERROR SET INDICATER
002326  004767  001530                JSR     PC,             ERRHAN          ;REPORT ERROR
002332  012737  173000  002552  1$:   MOV     #173000,        @#FIRSTA        ;SET UP FIRST ADDRESS TO BE SUMMED
002340  012737  173024  002554        MOV     #173024,        @#EXCADD        ;SET UP EXCEPTION ADDRESS
002346  012737  173175  002556        MOV     #173175,        @#LASTAD        ;SET UP LAST ADDRESS TO BE SUMMED
002354  012702  173176                MOV     #173176,        R2              ;GET ADDRESS OF GOOD DATA
002360  006137  002460        2$:     ROL     @#ROMCNT                        ;UPDATE ROM COUNTER
002364  033737  002460  001116        BIT     @#ROMCNT,       @#ROMFIN        ;IS ROM PRESENT
002372  001412                        BEQ     3$                              ;IF NO, GO UPDATE TO NEXT ROM
002374  004767  000062                JSR     PC,             CALSUM          ;IF YES GO CALCULATE CHECKSUM
002400  011203                        MOV     (R2),           R3              ;GET EXPECTED CHECKSUM
002402  020304                        CMP     R3,             R4              ;COMPARE CHECKSUMS
```

```
                                        BEQ     3$                      ;IF CHECKSUMS EQUAL CONTINUE
                                        BIS     #CRCERR,        @#ROMERR;IF ERROR SET INDICATER
                                        JSR     PC,             ERRHAN  ;REPORT ERROR
                                3$:     ADD     #200,           @#FIRSTA;UPDATE FIRST ADDRESS
                                        ADD     #200,           @#EXCADD;UPDATE EXCEPTION ADDRESS
                                        ADD     #200,           @#LASTAD;UPDATE LAST ADDRESS
                                        ADD     #200,           R2      ;UPDATE ADDRESS OF GOOD DATA
                                        CMP     #174000,        @#FIRSTA;HAVE ALL ROMS BEEN CHECKED
                                        BNE     2$                      ;IF NO GO CHECKSUM NEXT ONE
                                        RTS     PC                      ;IF YES RETURN

                                ROMCNT: 0

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

                                CALSUM: MOV     R2,             -(SP)   ;SAVE R2
                                        MOV     @#FIRSTA,       R0      ;GET STARTING ADDRESS
                                        CLR     R4                      ;INITIALIZE CRC WORD
                                LOOP:   MOV     (R0)+,          R5      ;GET A BYTE
                                        MOV     #16.,           R2      ;SET BYTE COUNT
                                CRCLOP: CLC                             ;THE NEXT NINE LINES
                                        ROR     R4                      ;DO THE MATH CALCULATIONS
                                        ROR     R5
                                        BVC     1$
                                        MOV     #120001,        R1
                                        BIC     R4,             R1
                                        BIC     #120001,        R4
                                        BIS     R1,             R4
                                1$:     DEC     R2
                                        BGT     CRCLOP
                                        CMP     R0,             @#EXCADD;IS NEXT ADDRESS AN EXCEPTION ADDRESS
                                        BNE     2$                      ;IF NO TEST IT
                                        TST     (R0)+                   ;IF YES SKIP ADDRESS
                                2$:     CMP     R0,             @#LASTAD;HAVE ALL LOCATIONS BEEN SUMMED
                                        BLOS    LOOP                    ;IF NO CONTINUE
                                        MOV     (SP)+,          R2      ;RESTORE R2
                                        RTS     PC                      ;IF YES RETURN

                                FIRSTA: 0
                                EXCADD: 0
                                LASTAD: 0


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

```
                                PROMP:  TSTB    @#$ENVM                 ;IF BIT IS CLEAR SIZE
                                        BMI     1$                      ;IF BIT IS SET DON'T SIZE.
                                        JSR     PC,     DEVCOD          ;CALL DEVICE CODE MODULE
                                        JSR     PC,     PUTMES          ;FORMAT PARAMETER MESSAGE
                                        MES1
                                        JSR     PC,     PPFVAR          ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
                                        BIT     #PFERR, @#ROMERR        ;WAS THERE AN ERR
                                        BEQ     10$                     ;IF NO GO FORMAT MESSAGE
                                        JSR     PC,     ERRHAN          ;IF YES GO REPORT MESSAGE
                                        BR      9$
                                10$:    JSR     PC,     PUTMES          ;FORMAT PSEUDO POWER-FAIL VECTOR
                                        MES2                            ;ADDRESS MESSAGE
                                        INC     @#MESSAG                ;SET MESSAGE INDICATOR
                                        BR      9$                      ;GO TO EXIT
                                1$:     JSR     PC,     DEVCOD          ;CALL GET DEVICE CODE MODULE
                                        MOV     #BUF1,  R3              ;GET BOARD DEVICE CODES
                                2$:     MOV     #$DDW0, R4              ;GET ETABLE DEVICE CODE PARAMETERS
                                        ADD     #2,     R3              ;GET TO BOARD DEVICE CODE
                                3$:     MOV     (R4)+,  R2              ;GET ETABLE DEVICE CODE
                                        SWAB    R2                      ;REFORMAT ASCII
                                        CMP     R2,     (R3)            ;DO THE TWO DEVICE CODES COMPARE
                                        BEQ     4$                      ;IF YES GET NEXT BOARD DEVICE CODE
                                        TST     (R4)                    ;IF NO HAVE WE CHECKED THE WHOLE ETABLE
                                        BNE     3$                      ;IF WE HAVEN'T CHECK NEXT ETABLE ENTRY
                                        BIS     #APTER1, @#ROMERR       ;IF WE HAVE THEN DEVICE CODE ON BOARD
                                        JSR     PC,     ERRHAN          ;DOES NOT  EXIST IN ETABLE
                                4$:     ADD     #2,     R3              ;UPDATE TO NEXT ADDRESS
                                        TST     (R3)                    ;IF CONTENTS EQUALS ZERO WE'RE DONE
                                        BNE     2$                      ;IF CONTENTS NO EQUAL ZERO CONTINUE
                                        MOV     #$DDW0, R3              ;GET BOARD DEVICE CODES
                                5$:     MOV     #BUF1-2, R4             ;GET ETABLE DEVICE CODES
                                        SWAB    (R3)                    ;FORMAT ASCII
                                6$:     ADD     #4,     R4              ;GET BOARD DEVICE CODE
                                        CMP     (R3),   (R4)            ;DO THE TWO DEVICE CODES COMPARE
                                        BEQ     7$                      ;IF YES GET NEXT ETABLE DEVICE CODE
                                        TST     (R4)                    ;IF NO HAVE WE CHECKED ALL THE BOARD
                                        BNE     6$                      ;DEVICE CODES, IF NO CONTINUE
                                        SWAB    (R3)                    ;PUT ASCII IN RIGHT ORDER FOR OUTPUT
                                        BIS     #APTER2, @#ROMERR       ;IF YES ETABLE DEVICE CODE NOT
                                        JSR     PC,     ERRHAN          ;NOT ON BOARD

                                7$:     ADD     #2,     R3
                                        TST     (R3)                    ;IF CONTENTS EQUAL ZERO-DONE
                                        BNE     5$                      ;IF CONTENTS NO EQUAL ZERO-CONTINUE
                                        JSR     PC,     PPFVAR          ;GO GET PSEUDO POWER-FAIL VECTOR ADDRESS
                                        MOV     @#$BASE, R4             ;GET ETABLE PPFVA
                                        MOV     @#BUF2, R3              ;GET BOARD PPFVA
                                        CMP     R4,     R3              ;DO THE TWO PARAMETERS COMPARE
                                        BEQ     8$                      ;IF YES CONTINUE
                                        BIS     #APTER3, @#ROMERR       ;SET APT ERROR INDICATOR
                                        JSR     PC,     ERRHAN          ;GO TO ERROR ROUTINE
                                8$:     MOV     @#$CDW1, R4             ;GET ETABLE DATA
                                        MOV     @#BUF2+2, R3            ;GET BOARD DATA
                                        CMP     R4,     R3              ;DO THE TWO PARAMETER COMPARE
                                        BEQ     9$                      ;IF YES THEN DONE
                                        BIS     #APTER4, @#ROMERR       ;SET APT ERROR INDICATOR
```

```
                                        JSR     PC,             ERRHAN          ;GO TO ERROR ROUTINE
                                9$:     RTS     PC
                                ;GET DEVICE CODES MODULE
                                ;THIS SUBROUTINE LOCATES EACH DEVICE CODE AND PASSES IT AND THE
                                ;ADDRESS IN WHICH IT WAS FOUND BACK TO THE CALLING ROUTINE.
                                ;DATA IS STORED IN BUFFER "BUF1" IN THIS FORMAT:
                                ;
                                ;       BUF1:   ADDRESS OF FIRST DEVICE CODE
                                ;               DEVICE CODE
                                ;               ADDRESS OF SECOND DEVICE CODE
                                ;               DEVICE CODE
                                ;                 .
                                ;                 .
                                ;                 .
                                ;               ADDRESS OF NTH DEVICE CODE
                                ;               DEVICE CODE  .
                                ;
                                ;IF DIAGNOSTIC ROM PRESENT IT WILL BE THE FIRST DEVICE CODE.
                                ;THERE IS ROOM FOR UP TO 12 DEVICE CODES IN TEMP
003050  012700  007420          DEVCOD: MOV     #BUF1,          R0              ;INITIALIZE ROM POINTER
003054  012701  000001                  MOV     #1,             R1              ;IS DIAGNOSTIC ROM PRESENT
003060  030137  001116                  BIT     R1,             @#ROMFIN        ;IF NO GO TEST BOOT ROMS
003064  001411                          BEQ     1$                              ;GET ADDRESS OF DIAG. ROM DEVICE CODE.
003066  012737  165774  003250          MOV     #165774,        @#TESTAD        ;STORE ADDRESS OF DEVICE CODE
003074  013720  003250                  MOV     @#TESTAD,       (R0)+           ;STORE DEVICE CODE
003100  017720  000144                  MOV     @TESTAD,        (R0)+           ;CLR DATA TYPE FLAG
003104  005037  003252                  CLR     @#DAFLAG                        ;GET ADDRESS OF FIRST BOOT ROM
003110  013737  001114  003250  1$:     MOV     @#FIRSTB,       @#TESTAD        ;UPDATE POINTER TO NEXT ROM
003116  006101                          ROL     R1                              ;IS ROM PRESENT
003120  030137  001116          3$:     BIT     R1,             @#ROMFIN        ;IF YES GO GET PARAMETERS
003124  001005                          BNE     4$                              ;IF NO UPDATE TEST ADDRESS
003126  062737  000200  003250          ADD     #200,           @#TESTAD        ;UPDATE POINTER TO NEXT ROM
003134  006101                          ROL     R1                              ;GO SEE IF ALL ROM CHECKED
003136  000437                          BR      9$                              ;IF DATAFLAG=0 THEN DATA IS DEVICE CODE
003140  005737  003252          4$:     TST     @#DAFLAG                        ;IF DATAFLAG=1 THE DATA IS OFFSET TO NEX
003144  001010                          BNE     5$                              ;STORE ADDRESS OF DEVICE CODE
003146  013720  003250                  MOV     @#TESTAD,       (R0)+           ;STORE DEVICE CODE
003152  017720  000072                  MOV     @TESTAD,        (R0)+           ;UPDATE TEST ADDRESS
003156  062737  000002  003250          ADD     #2,             @#TESTAD        ;GET OVER SOME CODE
003164  000422                          BR      8$                              ;SAVE OLD TEST ADDRESS
003166  013702  003250          5$:     MOV     @#TESTAD,       R2              ;UPDATE TO NEW TEST ADDRESS
003172  067737  000052  003250          ADD     @TESTAD,        @#TESTAD        ;GET THE NEW ADDRESS
003200  013703  003250                  MOV     @#TESTAD,       R3              ;SAVE ONLY BITS 7,8
003204  042702  177177                  BIC     #177177,        R2              ;IN R3 ALSO
003210  042703  177177                  BIC     #177177,        R3              ;CALCULATE DISTANCE BETWEEN  ADD
003214  160203                          SUB     R2,             R3              ;IS R3 EQUAL TO 0
003216  005703          6$:     TST     R3                              ;IF YES THEN DONE
003220  001404                          BEQ     8$                              ;IF NO THEN MOVE POINTER
003222  162703  000200                  SUB     #200,           R3              ;ONE BIT FOR EVERY 200
003226  006101                          ROL     R1
003230  000772                          BR      6$
003232  005137  003252          8$:     COM     @#DAFLAG                        ;CHANGE DATA FLAG TO RIGHT DATA TYPE
003236  023727  003250  174000  9$:     CMP     @#TESTAD,       #174000         ;HAVE WE CHECKED ALL THE ROM
003244  002725                          BLT     3$                              ;IF NO CONTINUE
```

```
                                        RTS     PC                              ;IF YES RETURN

                                TESTAD: 0
                                DAFLAG: 0

                                ;GET PSEUDO POWER-FAIL VECTOR ADDRESS ROUTINE
                                ;THIS SUBROUTINE TESTS LOCATIONS 173024 AND 173224 TO DETERMINE
                                ;WHICH VECTOR WILL BE USED IF POWER-FAIL OPTION ENABLED ON THE
                                ;BOARD.  THE OPTION MUST BE ENABLED AND AT LEAST ONE ADDRESS
                                ;SWITCH MUST BE "ON" OR AN ERROR WILL BE DETECTED.
                                ;THE DATA WILL BE RETURNED IN "BUF2" IN THE FORMAT.
                                ;
                                ;       BUF2:   PSEUDO POWER-FAIL VECTOR ADDRESS
                                ;               CONTENTS OF VECTOR ADDRESS
                                ;
003254  032737  000777  173024  PPFVAR: BIT     #777,           @#173024        ;TEST IF _OCATION 173024 SELECTED
003262  001407                          BEQ     1$                              ;IF NOT THEN GO TEST LOCATION 173224
003264  012737  173024  007500          MOV     #173024,        @#BUF2          ;IF IT IS THEN STORE ADDRESS
003272  013737  173024  007502          MOV     @#173024,       @#BUF2+2        ;STORE CONTENTS OF LOCATION 173024
003300  000416                          BR      3$                              ;GO TO RETURN
003302  032737  000777  173224  1$:     BIT     #777,           @#173224        ;TEST IF LOCATION 173224 SELECTED
003310  001407                          BEQ     2$                              ;IF NOT THEN SET ERROR INDICATOR
003312  012737  173224  007500          MOV     #173224,        @#BUF2          ;IF IT IS THEN STORE ADDRESS
003320  013737  173224  007502          MOV     @#173224,       @#BUF2+2        ;STORE CONTENTS OF VECTOR
003326  000403                          BR      3$                              ;GET OVER ERROR
003330  052737  000040  001110  2$:     BIS     #PFERR,         @#ROMERR        ;SET ERROR INDICATOR
003336  000207          3$:     RTS     PC

                                ;PUT MESSAGE IN BUFFER ROUTINE
                                ;THIS SUBROUTINE FORMATS THE PARAMETER AND POWER-FAIL MESSAGES.

003340  017604  000000  PUTMES: MOV     @(R6),          R4              ;GET MESSAGE BUFFER ADDRESS
003344  005737  003644          TST     @#TIMES                         ;IS THIS FIRST TIME THROUGH
003350  001076                  BNE     3$                              ;IF NOT FIRST TIME THEN FORMAT POWER-
                                                                        ;FAIL MESSAGE
003352  005237  003644          INC     @#TIMES                         ;TOGGLE WATCHDOG
003356  012703  007420          MOV     #BUF1,          R3              ;GET DATA BUFFER ADDRESS
003362  022713  165774          CMP     #165774,        (R3)            ;IS DIAGNOSTIC ROM PRESENT
003366  001012                  BNE     1$                              ;IF NOT DON'T FORMAT DIAG. ROM MESSAGE
003370  012705  006530          MOV     #DRHEAD,        R5              ;IF IT IS GET DIAG. ROM MESSAGE HEADER
003374  004767  001000          JSR     PC,             FILBUF          ;GO PUT HEADER IN MESSAGE BUFFER
003400  000015                  CR
003402  062703  000002          ADD     #2,             R3              ;SKIP OVER ADDRESS OF ASCII
003406  000313                  SWAB    (R3)                            ;FORMAT ASCII FOR MESSAGE
003410  112324                  MOVB    (R3)+,          (R4)+           ;PUT ASCII IN MESSAGE BUFFER
003412  112324                  MOVB    (R3)+,          (R4)+
003414  012705  006543  1$:     MOV     #BRHEAD,        R5
003420  004767  000754          JSR     PC,             FILBUF          ;GO PUT BOOT ROM HEADER IN MESS. BUF.
003424  000015                  CR
003426  112724  000015  2$:     MOVB    #CR,            (R4)+           ;PUT A CR/LF HERE
003432  112724  000012          MOVB    #LF,            (R4)+
003436  005713                  TST     (R3)                            ;IS DATA EQUAL TO ZERO
003440  001464                  BEQ     5$
```

```
 970  003442  062713  000004                 ADD     #4,          (R3)          ;GET FIRST ENTRY POINT
 971  003446  004767  001074                 JSR     PC,          OCADD         ;GO CONVERT OCTAL TO ASCII
 972  003452  004767  000722                 JSR     PC,          FILBUF        ;GO PUT ENTRY POINT IN MESSAGE BUF
 973  003456  000011                         HT
 974  003460  112724  000040                 MOVB    #40,         (R4)+
 975  003464  112724  000040                 MOVB    #40,         (R4)+
 976  003470  112724  000040                 MOVB    #40,         (R4)+
 977  003474  062713  000002                 ADD     #2,          (R3)          ;GET SECOND ENTRY POINT
 978  003500  004767  001042                 JSR     PC,          OCADD         ;GO CONVERT OCTAL TO ASCII
 979  003504  004767  000670                 JSR     PC,          FILBUF        ;GO PUT ENTRY POINT IN MESSAGE BUF.
 980  003510  000011                         HT
 981  003512  112724  000040                 MOVB    #40,         (R4)+
 982  003516  112724  000040                 MOVB    #40,         (R4)+
 983  003522  112724  000040                 MOVB    #40,         (R4)+
 984  003526  112724  000011                 MOVB    #HT,         (R4)+         ;PUT TAB IN HERE
 985  003532  062703  000002                 ADD     #2,          R3            ;UPDATE DATA BUFFER ADDRESS
 986  003536  000313                         SWAB    (R3)                       ;FORMAT ASCII FOR MESSAGE
 987  003540  112324                         MOVB    (R3)+,       (R4)+         ;MOV ASCII TO MES1
 988  003542  112324                         MOVB    (R3)+,       (R4)+
 989  003544  000730                         BR      2$                         ;GO BACK TO START OF LOOP
 990  003546  012703  007500         3$:     MOV     #BUF2,       R3            ;GET DATA BUFFER ADDRESS
 991  003552  012705  006660                 MOV     #PFHEAD,     R5            ;GET POWER-FAIL HEADER ADDRESS
 992  003556  004767  000616                 JSR     PC,          FILBUF        ;GO PUT POWER-FAIL HEADER IN MESSAGE BUF
 993  003562  000015                         CR
 994  003564  022703  007504         4$:     CMP     #BUF2+4,     R3            ;ARE WE DONE
 995  003570  001410                         BEQ     5$                         ;IF YES THEN GO RETURN
 996  003572  004767  000750                 JSR     PC,          OCADD         ;GO CONVERT OCTAL TO ASCII
 997  003576  004767  000576                 JSR     PC,          FILBUF
 998  003602  000011                         HT
 999  003604  062703  000002                 ADD     #2,          R3            ;UPDATE DATA BUFFER ADDRESS
1000  003610  000765                         BR      4$                         ;GO BACK TO START OF LOOP
1001  003612  112724  000015         5$:     MOVB    #CR,         (R4)+         ;PUT A CR/LF AT END OF MESSAGE
1002  003616  112724  000012                 MOVB    #LF,         (R4)+
1003  003622  105024                         CLRB    (R4)+                      ;PUT ZERO TERMINATOR AT END OF MESSAGE
1004  003624  017667  000000  000002         MOV     @(R6),       6$            ;GET MESSAGE BUFFER ADDRESS
1005  003632  104401                         TYPE
1006  003634  000000                 6$:     .WORD   0
1007  003636  062716  000002                 ADD     #2,          (R6)          ;GET OVER MESSAGE BUFFER ADDRESS
1008  003642  000207                         RTS     PC
1009  003644  000000                 TIMES:  0
1010
1011
1012
1013
1014                                 ;SEQUENCE TEST
1015                                 ;THIS TEST VERIFIES THE ORDER OF THE ROMS ADHERES TO THAT SPECIFIED
1016                                 ;IN THE INSTALLATION PROCEDURE. THAT IS THE THEY HAVE BEEN INSTALLED
1017                                 ;IN ALPHABETICAL ORDER ACCORDING TO THE FIRST DEVICE CODE OF THE ROM.
1018                                 ;THE TYPEOUT FROM THIS SECTION DOES NOT INDICATE AN ERROR BUT IS JUST
1019                                 ;A WARNING TO THE OPERATOR THAT THE SPECIFICATION HAS NOT BEEN FULLY
1020                                 ;ADHERED TO.
1021
1022  003646  013700  001116         SEQTST: MOV     @#ROMFIN,    R0            ;GET ROM LIST
1023  003652  006000                         ROR     R0                         ;GET RID OF DIAG ROM INDICATOR
1024  003654  000241                         CLC                                ;OK START, WE MUST FIND MORE THAN
1025  003656  005037  003252                 CLR     @#DAFLAG                   ;ONE BOOT ROM TO DO THIS TEST
```

```
1026  003662  006000                  1$:     ROR     R0                      ;IF ROM PRESENT C-BIT SETS
1027  003664  103406                          BCS     2$                      ;WHEN C-BITS SETS THATS "ONE"
1028  003666  005237  003252                  INC     @#DAFLAC                ;INCREMENT COUNTER-IF WE DO THIS
1029  003672  022737  000004  003252          CMP     #4,     @#DAFLAG        ;LOOP 4 TIMES, NO BOOT ROM PRESENT AT AL
1030  003700  003033                          BGT     SEQEX                   ;SO EXIT TEST
1031  003702  005700                  2$:     TST     R0                      ;WE FOUND ONE, BUT IF R0 IS EQUAL TO
1032  003704  001431                          BEQ     SEQEX                   ;ZERO THERE ONLY THE ONE SO EXIT
1033  003706  005037  003252                  CLR     @#DAFLAG
1034  003712  012700  007420                  MOV     #BUF1,  R0              ;AT LEAST TWO START TEST, GET ROM DATA
1035  003716  032737  000001  001116          BIT     #1,     @#ROMFIN        ;IS DIAG ROM PRESENT
1036  003724  001402                          BEQ     3$                      ;IF NO DON'T UPDATE DATA TABLE ADDRESS
1037  003726  062700  000004                  ADD     #4,     R0              ;IF YES THAN UPDATE TABLE ADDRESS
1038  003732  004767  000040          3$:     JSR     PC,     GETCOD          ;GO GET A DEVICE CODE
1039  003736  005704                          TST     R4                      ;UPON RETURN FROM GET COD R4 INDICATES
1040  003740  001013                          BNE     SEQEX                   ;IF VALID DEVICE CODE WAS FOUND
1041  003742  004767  000030                  JSR     PC,     GETCOD          ;CODES SO EXIT, IF NOT THEN GET NEXT COD
1042  003746  026767  000020  000020          CMP     FRSCOD, LASCOD          ;IS 1ST CODE ALPHABETICALLY LESS THE 2ND
1043  003754  003003                          BGT     4$                      ;IF NO SEQUENCE ERROR
1044  003756  005704                          TST     R4                      ;IF R4 NOT EQUAL TO ZERO GET CON
1045  003760  001003                          BNE     SEQEX           ;COULD NOT FIND VALID DEVICE CODE, SO EXIT
1046  003762  000763                          BR      3$                      ;IF NOT THEN GO GET NEXT DEVICE CODE
1047  003764  104401                  4$:     TYPE                            ;TYPE OUT WARNING MESSAGE
1048  003766  007317                          SEQMSG
1049  003770  000207                  SEQEX:  RTS     PC                      ;RETURN
1050
1051  003772  000000                  FRSCOD: .WORD   0
1052  003774  000000                  LASCOD: .WORD   0
1053
1054
1055
1056                                  ;GET DEVICE CODES SUBROUTINE
1057                                  ;THIS SUBROUTINE IS CALLED BY THE SEQUENCE TEST ROUTINE TO GET THE
1058                                  ;DEVICE CODES FROM THE SIZING BUFFER. THE ROUTINE VALIDATES EACH DEVICE
1059                                  ;CODE,THAT IS IT MAKES SURE THE CODE IS THE FIRST CODE IN A ROM. IF
1060                                  ;IT CAN NOT FIND A VALID DEVICE CODE IT MAKES R4 NONZERO BEFORE RETURNING.
1061
1062  003776  005004                  GETCOD: CLR     R4
1063  004000  005737  003252                  TST     @#DAFLAG                ;IF DAFLAG IS ZERO PUT DEVICE CODE
1064  004004  001405                          BEQ     1$                      ;IN ERSCOD
1065  004006  012703  003774                  MOV     #LASCOD,        R3      ;IF DAFLAG IS ONE PUT DEVICE CODE
1066  004012  005037  003252                  CLR     @#DAFLAG                ;IN LASCOD AND CLEAR DAFLAG
1067  004016  000404                          BR      2$                      ;GET OVER DAFLAG=ZERO SETUP
1068  004020  012703  003772          1$:     MOV     #FRSCOD,        R3      ;PUT DEVICE CODE IN FRS COD
1069  004024  005237  003252                  INC     @#DAFLAG                ;MAKE DAFLAG NOT ZERO
1070  004030  005710                  2$:     TST     (R0)                    ;DOES DEVICE CODE EXIST
1071  004032  001002                          BNE     3$                      ;IF YES CHECK IF VALID
1072  004034  005104                          COM     R4                      ;IF NO MAKE R4 NON-ZERO
1073  004036  000410                          BR      GETEND                  ;AND EXIT
1074  004040  012001                  3$:     MOV     (R0)+,          R1      ;GET ADDRESS OF DEVICE CODE
1075  004042  032701  000170                  BIT     #170,           R1      ;IF BIT 3-6 CLEAR THIS IS IST CODE
1076  004046  001403                          BEQ     4$                      ;OF A ROM SO GET IT
1077  004050  062700  000002                  ADD     #2,             R0      ;IF BITS SET UPDATE TO NEXT CODE
1078  004054  000765                          BR      2$                      ;AND VALIDATE IT
1079  004056  012013                  4$:     MOV     (R0)+,          (R3)    ;GET DEVICE CODE AND STORE IT
1080  004060  000207                  GETEND: RTS     PC                      ;RETURN
1081
```

```
                                        ;ERROR HANDLER ROUTINE
                                        ;THIS SUBROUTINE FORMATS THE ERROR MESSAGES THE TYPES THEM
                                        ;OUT.
004062  104411          ERRHAN: SAVREG
004064  012704  010214            MOV     #ERRMSG,        R4
004070  005237  001120            INC     @#ERRCNT                ;INCREMENT ERROR COUNTER
004074  032777  002000  175002    BIT     #BIT10,         @SWR    ;BELL ON ERROR
004102  001402                    BEQ     1$                      ;BRANCH IF NO
004104  104401                    TYPE
004106  006524                    BELL
004110  032777  020000  174766  1$:       BIT     #BIT13,         @SWR    ;INHIBIT ERROR TYPEOUT
004116  001110                    BNE     10$
004120  013700  001110            MOV     @#ROMERR,       R0      ;GET ERROR CODE
004124  013767  001110  000174    MOV     @#ROMERR,       9$      ;SAVE ERROR CODE FOR APT
004132  012701  004362            MOV     #EHEADT,        R1      ;GET ERROR HEADER TABLE
004136  000241                    CLC                             ;C-BIT USED TO STOP STEPPING THROUGH TAB
004140  006000            2$:     ROR     R0                      ;ROTATE ERROR CODES
004142  103403                    BCS     3$                      ;IF C-BIT SET STOP STEPPING
004144  062701  000002            ADD     #2,             R1      ;IF C-BIT CLEAR STEP TO NEXT HEADER
004150  000773                    BR      2$                      ;GO STEP
004152  011105            3$:     MOV     (R1),           R5      ;PUT HEADER ADDRESS IN REGISTER.
004154  004767  000220            JSR     PC,             FILBUF  ;GO PUT HEADER IN ERROR MESSAGE BUF.
004160  000015                    CR
004162  032737  000023  001110    BIT     #23,            @#ROMERR        ;IS ERROR A CRC, APTER1 OR APTER2
004170  001424                    BEQ     7$                      ;IF NO GO FORMAT APTER3, APTER4
004172  032737  000020  001110    BIT     #20,            @#ROMERR        ;IS ERROR CRC
004200  001415                    BEQ     6$                      ;IF NO FORMAT APTER1, APTER2
004202  013700  002460            MOV     @#ROMCNT,       R0      ;GET ROM COUNTER
004206  012701  006512            MOV     #ROMNUM,        R1      ;GET ROM NUMBER TABLE
004212  000241                    CLC                             ;C-BIT USED TO STOP STEPPING THROUGH TAB
004214  006000            4$:     ROR     R0                      ;ROTATE ROM NUMBER
004216  103403                    BCS     5$                      ;IF C-BIT SET STOP STEPPING
004220  062701  000002            ADD     #2,             R1      ;IF C-BIT CLEAR STEP TO NEXT HEADER
004224  000773                    BR      4$                      ;CONTINUE STEPPING
004226  112124            5$:     MOVB    (R1)+,          (R4)+   ;PUT ROM NUMBER IN ERROR MESSAGE BUF
004230  112124                    MOVB    (R1)+,          (R4)+
004232  000421                    BR      8$                      ;GO TO END OF ROUTINE
004234  112324            6$:     MOVB    (R3)+,          (R4)+   ;PUT BAD DEVICE CODE IN ERROR MESS. BUF.
004236  112324                    MOVB    (R3)+,          (R4)+
004240  000416                    BR      8$                      ;GO TO END OF ROUTINE
004242  016603  000006    7$:     MOV     6(R6),          R3      ;GET SAVED EXPECTED VALUE
004246  004767  000170            JSR     PC,             OCASC   ;GO COVERT OCTAL TO ASCII
004252  004767  000122            JSR     PC,             FILBUF  ;GO PUT DATA IN ERROR MESSAGE BUF.
004256  000011                    HT
004260  016603  000010            MOV     10(R6),         R3      ;GET SAVED RECEIVED VALUE
004264  004767  000152            JSR     PC,             OCASC   ;GO COVERT OCTAL TO ASCII
004270  004767  000104            JSR     PC,             FILBUF  ;GO PUT DATA IN ERROR MESSAGE BUFFER.
004274  000011                    HT
004276  112724  000015    8$:     MOVB    #CR,            (R4)+   ;PUT CR/LF AT END OF MESSAGE
004302  112724  000012            MOVB    #LF,            (R4)+
004306  112724  000000            MOVB    #0,             (R4)+   ;PUT TERMINATOR AT END OF MESSAGE
004312  005767  174640            TST     $ENV                    ;TST IF ON APT
```

```
1138  004316  001405                  BEQ     11$                     ;IF NO BRANCH
1139  004320  004767  000534          JSR     PC,     $ATY1           ;GO REPORT ERROR
1140  004324  010214                  ERRMSG
1141  004326  000000          9$:     .WORD   0
1142  004330  000000                  HALT                            ;THEN HALT
1143  004332  004767  000530  11$:    JSR     PC,     $ATY3           ;IF NOT ON APT JUST REPORT ERROR
1144  004336  010214                  ERRMSG
1145  004340  032777  100000  174536  10$:    BIT     #BIT15,  @SWR   ;IS HALT ON ERROR SET
1146  004346  001401                  BEQ     12$                     ;IF NO SKIP OVER HALT
1147  004350  000000                  HALT
1148  004352  005037  001110  12$:    CLR     @#ROMERR                ;CLEAR ERROR FLAGS
1149  004356  104412                  RESREG
1150  004360  000207                  RTS     PC
1151
1152  004362  006756          EHEADT: ER1MSG
1153  004364  006722                  ER2MSG
1154  004366  007014                  ER3MSG
1155  004370  007072                  ER4MSG
1156  004372  007146                  CRCMSG
1157  004374  007172                  PFMSG
1158  004376  007250                  NOROMM
1159
1160
1161
1162                          ;FILL BUFFER ROUTINE
1163                          ;THIS SUBROUTINE FILLS THE MESSAGE BUFFER WILL ASCII
1164                          ;CHARACTERS.
1165
1166  004400  022776  000011  000000  FILBUF: CMP     #HT,    @(R6)   ;IS FIRST CHARACTER A TAB OR CR
1167  004406  001003                  BNE     1$                      ;IF CR THEN GO PUT CR/LF IN BUFFER
1168  004410  112724  000011                  MOVB    #HT,    (R4)+   ;IF TAB THEN PUT TAB IN BUFFER
1169  004414  000404                  BR      2$                      ;GET OVER NEXT LINE
1170  004416  112724  000015          1$:     MOVB    #CR,    (R4)+   ;MOV CR/LF TO BUFFER
1171  004422  112724  000012                  MOVB    #LF,    (R4)+
1172  004426  112524                  2$:     MOVB    (R5)+,  (R4)+   ;PUT A CHARACTER IN MESSAGE BUFFER
1173  004430  105715                  TSTB    (R5)                    ;IS NEXT CHARACTER ZERO
1174  004432  001375                  BNE     2$                      ;IF NOT PUT IT IN MESSAGE BUFFER AND GET
1175  004434  062716  000002          ADD     #2,     (R6)            ;UPDATE RETURN POINTER TO GET OVER CHARA
1176  004440  000207                  RTS     PC                      ;THEN RETURN
1177
1178
1179
1180
1181                          ;OCTAL TO ASCII CONVERSION ROUTINE
1182                          ;THIS SUBROUTINE TAKES A SIXTEEN BIT OCTAL NUMBER AND
1183                          ;CONVERTS IT TO 6 ASCII CHARACTERS
1184
1185  004442  012700  007504  OCASC:  MOV     #OCTBUF,        R0      ;GET BUFFER ADDRESS
1186  004446  005020                  CLR     (R0)+                   ;CLEAR BUFFER
1187  004450  005020                  CLR     (R0)+
1188  004452  005020                  CLR     (R0)+
1189  004454  005020                  CLR     (R0)+
1190  004456  012700  007504          MOV     #OCTBUF,        R0      ;GET BUFFER ADDRESS
1191  004462  010337  004544          MOV     R3,             @#TEMP  ;GET OCTAL NUMBER
1192  004466  000241                  CLC                             ;CLEAR CARRY
1193  004470  006137  004544          ROL     @#TEMP                  ;ROTATE BIT INTO CARRY BIT
```

```
1194  004474  006110                        ROL     (R0)            ;ROTATE CARRY BIT INTO BUFFER
1195  004476  152710  000060        1$:     BISB    #60,    (R0)    ;MAKE IT ASCII
1196  004502  005200                        INC     R0              ;UPDATE BUFFER ADDRESS
1197  004504  020027  007512                CMP     R0,     #OCTBUF+6 ;HAVE WE CONVERTED ALL THE NUMBER
1198  004510  001003                        BNE     2$              ;IF NO CONTINUE
1199  004512  012705  007504                MOV     #OCTBUF,R5      ;IF YES PUT BUFFER ADDRESS IN REGISTER
1200  004516  000207                        RTS     PC              ;RETURN
1201  004520  006137  004544        2$:     ROL     @#TEMP          ;ROTATE BIT INTO CARRY BIT
1202  004524  106110                        ROLB    (R0)            ;ROTATE CARRY BIT INTO BUFFER
1203  004526  006137  004544                ROL     @#TEMP          ;
1204  004532  106110                        ROLB    (R0)            ;
1205  004534  006137  004544                ROL     @#TEMP          ;
1206  004540  106110                        ROLB    (R0)            ;
1207  004542  000755                        BR      1$              ;GO TO START OF LOOP
1208  004544  000000        TEMP:   0
1209
1210
1211
1212                                ;THIS ROUTINE IS CALLED BY THE PUT MESSAGE ROUTINE TO GET THE RIGHT
1213                                ;VALUE IN R3 SO THE OCTAL TO ASCII ROUTINE GETS THE RIGHT NUMBER.
1214
1215  004546  010346        OCADD:  MOV     R3,     -(R6)           ;SAVE THE VALUE OF R3
1216  004550  011303                MOV     (R3),   R3              ;PUT THE DATA TO BE CONVERTED IN R3
1217  004552  004767  177664        JSR     PC,     OCASC           ;GO CONVERT OCTAL TO ASCII
1218  004556  012603                MOV     (R6)+,  R3              ;RESTORE R3
1219  004560  000207                RTS     PC                      ;RETURN
1220
1221
1222
1223
1224
1225                                .SBTTL  TYPE ROUTINE
1226
1227                                ;;*****************************************************************
1228                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1229                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1230                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1231                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1232                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1233                                ;*
1234                                ;*CALL:
1235                                ;*1) USING A TRAP INSTRUCTION
1236                                ;*       TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1237                                ;*OR
1238                                ;*       TYPE
1239                                ;*       MESADR
1240                                ;*
1241
1242  004562  105767  000265        $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
1243  004566  100002                BPL     1$              ;;BR IF YES
1244  004570  000000                HALT                    ;;HALT HERE IF NO TERMINAL
1245  004572  000430                BR      3$              ;;LEAVE
1246  004574  010046        1$:     MOV     R0,-(SP)        ;;SAVE R0
1247  004576  017600  000002        MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
1248  004602  122767  000001  174346 CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
1249  004610  001011                BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
```

```
1250  004612  132767  000100  174337          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
1251  004620  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
1252  004622  010067  000004                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
1253  004626  004767  000234                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
1254  004632  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
1255  004634  132767  000040  174315  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
1256  004642  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
1257  004644  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1258  004646  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1259  004650  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1260  004652  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
1261  004654  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1262  004660  000002                          RTI                     ;;RETURN
1263  004662  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1264  004666  001430                          BEQ     8$
1265  004670  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
1266  004674  001006                          BNE     5$
1267  004676  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
1268  004700  104401                          TYPE                    ;;TYPE A CR AND LF
1269  004702  005055                          $CRLF
1270  004704  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
1271  004710  000755                          BR      2$              ;;GET NEXT CHARACTER
1272  004712  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1273  004716  126726  000130          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1274  004722  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
1275  004724  016746  000120                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1276                                                                  ;;AND THE NULL CHAR.
1277  004730  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1278  004734  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1279  004736  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
1280  004742  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
1281  004746  000770                          BR      7$              ;;LOOP
1282
1283                                  ;HORIZONTAL TAB PROCESSOR
1284
1285  004750  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1286  004754  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1287  004760  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1288  004766  001372                          BNE     9$              ;;TAB STOP
1289  004770  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
1290  004772  000724                          BR      2$              ;;GET NEXT CHARACTER
1291  004774  105777  000044          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
1292  005000  100375                          BPL     $TYPEC
1293  005002  116677  000002  000036          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1294  005010  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1295  005016  001003                          BNE     1$              ;;BRANCH IF NO
1296  005020  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1297  005024  000406                          BR      $TYPEX          ;;EXIT
1298  005026  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1299  005034  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
1300  005036  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
1301  005040  000000          $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
1302  005042  000207          $TYPEX: RTS     PC
1303
1304  005044  177564          $TPS:   .WORD   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
1305  005046  177566          $TPB:   .WORD   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
```

```
1306  005050  000          $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
1307  005051  002          $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
1308  005052  012          $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
1309  005053  000          $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
1310  005054  077          $QUES:  .ASCII  "?"             ;;QUESTION MARK
1311  005055  015          $CRLF:  .ASCII  <15>            ;;CARRAIGE RETURN
1312  005056  000012       $LF:    .ASCIZ  <12>            ;;LINEFEED
1313                        .SBTTL  APT COMMUNICATIONS ROUTINE
1314
1315                        ;;**************************************************************************
1316  005060  112767  000001  000236  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
1317  005066  112767  000001  000226  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
1318  005074  000403                   BR      $ATYC
1319  005076  112767  000001  000220  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
1320  005104                  $ATYC:
1321  005104  010046                   MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1322  005106  010146                   MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1323  005110  105767  000206          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
1324  005114  001450                   BEQ     5$              ;;IF NOT:  BR
1325  005116  122767  000001  174032  CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
1326  005124  001031                   BNE     3$              ;;IF NOT:  BR
1327  005126  132767  000100  174023  BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
1328  005134  001425                   BEQ     3$              ;;IF NOT:  BR
1329  005136  017600  000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
1330  005142  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
1331  005150  005767  173762    1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
1332  005154  001375                   BNE     1$              ;;IF NOT:  WAIT
1333  005156  010067  173770          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
1334  005162  105720            2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
1335  005164  001376                   BNE     2$
1336  005166  166700  173760          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
1337  005172  006200                   ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
1338  005174  010067  173754          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
1339  005200  012767  000004  173730  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
1340  005206  000413                   BR      5$
1341  005210  017667  000004  000016  3$:    MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
1342  005216  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
1343  005224  016746  172546          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
1344  005230  004767  177326          JSR     PC,$TYPE        ;;CALL TYPE MACRO
1345  005234  000000            4$:   .WORD   0
1346  005236                    5$:
1347  005236  105767  000062    10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
1348  005242  001416                   BEQ     12$             ;;IF NOT:  BR
1349  005244  005767  173706          TST     $ENV            ;;RUNNING UNDER APT?
1350  005250  001413                   BEQ     12$             ;;IF NOT:  BR
1351  005252  005767  173660    11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
1352  005256  001375                   BNE     11$             ;;IF NOT:  WAIT
1353  005260  017667  000004  173652  MOV     @4(SP),$FATAL   ;;GET ERROR #
1354  005266  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
1355  005274  005267  173636          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
1356  005300  105067  000020    12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
1357  005304  105067  000013          CLRB    $LFLG           ;;CLEAR LOG FLAG
1358  005310  105067  000006          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
1359  005314  012601                   MOV     (SP)+,R1        ;;POP STACK INTO R1
1360  005316  012600                   MOV     (SP)+,R0        ;;POP STACK INTO R0
1361  005320  000207                   RTS     PC              ;;RETURN
```

```
1362  005322      000                 $MFLG:  .BYTE   0               ;;MESSG. FLAG
1363  005323      000                 $LFLG:  .BYTE   0               ;;LOG FLAG
1364  005324      000                 $FFLG:  .BYTE   0               ;;FATAL FLAG
1365              005326                      .EVEN
1366              000200              APTSIZE=200
1367              000001              APTENV=001
1368              000100              APTSPOOL=100
1369              000040              APTCSUP=040
1370                                  .SBTTL  TTY INPUT ROUTINE
1371
1372                                  ;;**********************************************************************
1373  005326  177560                  $TKS:   .WORD   177560          ;;TTY KBD STATUS
1374  005330  177562                  $TKB:   .WORD   177562          ;;TTY KBD BUFFER
1375                                  .ENABL  LSB
1376
1377                                  ;;**********************************************************************
1378                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1379                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1380                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1381                                  ;*WHEN OPERATING IN TTY FLAG MODE.
1382  005332  022767  000176  173544  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
1383  005340  001074                          BNE     15$             ;;BRANCH IF NO
1384  005342  105777  177760                  TSTB    @$TKS           ;;CHAR THERE?
1385  005346  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
1386  005350  117746  177754                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
1387  005354  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
1388  005360  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
1389  005364  001062                          BNE     15$             ;;NO, RETURN TO USER
1390  005366  126727  173506  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
1391  005374  001456                          BEQ     15$             ;;BRANCH IF YES
1392
1393  005376  104401  006057                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
1394  005402  104401  006064  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
1395  005406  016746  172564          MOV     $SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
1396  005412  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1397  005414  104401  006075          TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
1398  005420  005046          19$:    CLR     -(SP)           ;;CLEAR COUNTER
1399  005422  005046                  CLR     -(SP)           ;;THE NEW SWR
1400  005424  105777  177676  7$:     TSTB    @$TKS           ;;CHAR THERE?
1401  005430  100375                  BPL     7$              ;;IF NOT TRY AGAIN
1402
1403  005432  117746  177672          MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
1404  005436  042716  177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1405
1406
1407
1408  005442  021627  000025  9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
1409  005446  001005                  BNE     10$             ;;BRANCH IF NOT
1410  005450  104401  006052          TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
1411  005454  062706  000006  20$:     ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
1412  005460  000757                  BR      19$             ;;LET'S TRY IT AGAIN
1413
1414
1415  005462  021627  000015  10$:     CMP     (SP),#15        ;;IS IT A <CR>?
1416  005466  0C1022                  BNE     16$             ;;BRANCH IF NO
1417  005470  005766  000004          TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
```

```
1418  005474  001403                        BEQ     11$                     ;;BRANCH IF YES
1419  005476  016677  000002  173400        MOV     2(SP),@SWR              ;;SAVE NEW SWR
1420  005504  062706  000006          11$:  ADD     #6,SP                   ;;CLEAR UP STACK
1421  005510  104401  005055          14$:  TYPE    ,$CRLF                  ;;ECHO <CR> AND <LF>
1422  005514  126727  173361  000001        CMPB    $INTAG,#1               ;;RE-ENABLE TTY KBD INTERRUPTS?
1423  005522  001003                        BNE     15$                     ;;BRANCH IF NOT
1424  005524  012777  000100  177574        MOV     #100,@$TKS              ;;RE-ENABLE TTY KBD INTERRUPTS
1425  005532  000002          15$:  RTI                                     ;;RETURN
1426  005534  004767  177234          16$:  JSR     PC,$TYPEC               ;;ECHO CHAR
1427  005540  021627  000060                CMP     (SP),#60                ;;CHAR < 0?
1428  005544  002420                        BLT     18$                     ;;BRANCH IF YES
1429  005546  021627  000067                CMP     (SP),#67                ;;CHAR > 7?
1430  005552  003015                        BGT     18$                     ;;BRANCH IF YES
1431  005554  042726  000060                BIC     #60,(SP)+               ;;STRIP-OFF ASCII
1432  005560  005766  000002                TST     2(SP)                   ;;IS THIS THE FIRST CHAR
1433  005564  001403                        BEQ     17$                     ;;BRANCH IF YES
1434  005566  006316                        ASL     (SP)                    ;;NO, SHIFT PRESENT
1435  005570  006316                        ASL     (SP)                    ;;   CHAR OVER TO MAKE
1436  005572  006316                        ASL     (SP)                    ;;   ROOM FOR NEW ONE.
1437  005574  005266  000002          17$:  INC     2(SP)                   ;;KEEP COUNT OF CHAR
1438  005600  056616  177776                BIS     -2(SP),(SP)             ;;SET IN NEW CHAR
1439  005604  000707                        BR      7$                      ;;GET THE NEXT ONE
1440  005606  104401  005054          18$:  TYPE    ,$QUES                  ;;TYPE ?<CR><LF>
1441  005612  000720                        BR      20$                     ;;SIMULATE CONTROL-U
1442                                .DSABL  LSB
1443
1444
1445                                ;;**************************************************************
1446                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1447                                ;*CALL:
1448                                ;*      RDCHR                           ;;INPUT A SINGLE CHARACTER FROM THE TTY
1449                                ;*      RETURN HERE                     ;;CHARACTER IS ON THE STACK
1450                                ;*                                      ;;WITH PARITY BIT STRIPPED OFF
1451                                ;
1452
1453  005614  011646          $RDCHR: MOV     (SP),-(SP)              ;;PUSH DOWN THE PC
1454  005616  016666  000004  000002        MOV     4(SP),2(SP)             ;;SAVE THE PS
1455  005624  105777  177476          1$:   TSTB    @$TKS                   ;;WAIT FOR
1456  005630  100375                        BPL     1$                      ;;A CHARACTER
1457  005632  117766  177472  000004        MOVB    @$TKB,4(SP)             ;;READ THE TTY
1458  005640  042766  177600  000004        BIC     #^C<177>,4(SP)          ;;GET RID OF JUNK IF ANY
1459  005646  026627  000004  000023        CMP     4(SP),#23               ;;IS IT A CONTROL-S?
1460  005654  001013                        BNE     3$                      ;;BRANCH IF NO
1461  005656  105777  177444          2$:   TSTB    @$TKS                   ;;WAIT FOR A CHARACTER
1462  005662  100375                        BPL     2$                      ;;LOOP UNTIL ITS THERE
1463  005664  117746  177440                MOVB    @$TKB,-(SP)             ;;GET CHARACTER
1464  005670  042716  177600                BIC     #^C177,(SP)             ;;MAKE IT 7-BIT ASCII
1465  005674  022627  000021                CMP     (SP)+,#21               ;;IS IT A CONTROL-Q?
1466  005700  001366                        BNE     2$                      ;;IF NOT DISCARD IT
1467  005702  000750                        BR      1$                      ;;YES, RESUME
1468  005704  026627  000004  000140  3$:   CMP     4(SP),#140              ;;IS IT UPPER CASE?
1469  005712  002407                        BLT     4$                      ;;BRANCH IF YES
1470  005714  026627  000004  000175        CMP     4(SP),#175              ;;IS IT A SPECIAL CHAR?
1471  005722  003003                        BGT     4$                      ;;BRANCH IF YES
1472  005724  042766  000040  000004        BIC     #40,4(SP)               ;;MAKE IT UPPER CASE
1473  005732  000002          4$:   RTI                                     ;;GO BACK TO USER
```

```
1474                                    ;;**********************************************************************
1475                                    ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1476                                    ;*CALL:
1477                                    ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
1478                                    ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1479                                    ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
1480
1481  005734  010346                   $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
1482  005736  012703  006042           1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
1483  005742  022703  006052           2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
1484  005746  101405                           BLOS    4$              ;;BR IF YES
1485  005750  104407                           RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
1486  005752  112613                           MOVB    (SP)+,(R3)      ;;GET CHARACTER
1487  005754  122713  000177           10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
1488  005760  001003                           BNE     3$              ;;SKIP IF NOT
1489  005762  104401  005054           4$:     TYPE    ,$QUES          ;;TYPE A '?'
1490  005766  000763                           BR      1$              ;;CLEAR THE BUFFER AND LOOP
1491  005770  111367  000044           3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
1492  005774  104401  006040                   TYPE    ,9$
1493  006000  122723  000015                   CMPB    #15,(R3)+       ;;CHECK FOR RETURN
1494  006004  001356                           BNE     2$              ;;LOOP IF NOT RETURN
1495  006006  105063  177777                   CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
1496  006012  104401  005056                   TYPE    ,$LF            ;;TYPE A LINE FEED
1497  006016  012603                           MOV     (SP)+,R3        ;;RESTORE R3
1498  006020  011646                           MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
1499  006022  016666  000004  000002           MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
1500  006030  012766  006042  000004           MOV     #$TTYIN,4(SP)
1501  006036  000002                           RTI                     ;;RETURN
1502  006040     000                   9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
1503  006041     000                           .BYTE   0               ;;TERMINATOR
1504  006042  000010                   $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
1505  006052  052536  005015     000   $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
1506  006057     136  006507  000012   $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
1507  006064  005015  053523  020122   $MSWR:  .ASCIZ  <15><12>/SWR = /
1508  006072  020075     000
1509  006075     040  047040  053505   $MNEW:  .ASCIZ  /  NEW = /
1510  006102  036440  000040
1511                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1512
1513                                    ;;**********************************************************************
1514                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1515                                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
1516                                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1517                                    ;*CALL:
1518                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1519                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
1520                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1521                                    ;*      .BYTE   M               ;;M=1 OR 0
1522                                    ;*                                      ;;1=TYPE LEADING ZEROS
1523                                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
1524                                    ;*
1525                                    ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1526                                    ;*$TYPOS OR $TYPOC
1527                                    ;*CALL:
1528                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
1529                                    ;*      TYPON                   ;;CALL FOR TYPEOUT
```

```
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
                                        $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
006106  017646  000000                  MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
006112  116667  000001  000211          MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
006120  112667  000207                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
006124  062716  000002                  BR      $TYPON
006130  000406                          MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
006132  112767  000001  000171  $TYPOC: MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
006140  112767  000006  000165          MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
006146  112767  000005  000154  $TYPON: MOV     R3,-(SP)        ;;SAVE R3
006154  010346                          MOV     R4,-(SP)        ;;SAVE R4
006156  010446                          MOV     R5,-(SP)        ;;SAVE R5
006160  010546                          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
006162  116704  000145                  NEG     R4
006166  005404                          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
006170  062704  000006                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
006174  110467  000132                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
006200  116704  000125                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
006204  016605  000012                  CLR     R3              ;;CLEAR THE OUTPUT WORD
006210  005003                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
006212  006105                          BR      3$              ;;GO DO MSB
006214  000404                  2$:     ROL     R5              ;;FORM THIS DIGIT
006216  006105                          ROL     R5
006220  006105                          ROL     R5
006222  006105                          MOV     R5,R3
006224  010503                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
006226  006103                          DECB    $OMODE          ;;TYPE THIS DIGIT?
006230  105367  000076                  BPL     7$              ;;BR IF NO
006234  100016                          BIC     #177770,R3      ;;GET RID OF JUNK
006236  042703  177770                  BNE     4$              ;;TEST FOR 0
006242  001002                          TST     R4              ;;SUPPRESS THIS 0?
006244  005704                          BEQ     5$              ;;BR IF YES
006246  001403                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
006250  005204                          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
006252  052703  000060          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
006256  052703  000040                  MOVB    R3,8$           ;;SAVE FOR TYPING
006262  110367  000040                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
006266  104401  006326          7$:     DECB    $OCNT           ;;COUNT BY 1
006272  105367  000032                  BGT     2$              ;;BR IF MORE TO DO
006276  003347                          BLT     6$              ;;BR IF DONE
006300  002402                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
006302  005204                          BR      2$              ;;GO DO THE LAST DIGIT
006304  000744                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
006306  012605                          MOV     (SP)+,R4        ;;RESTORE R4
006310  012604                          MOV     (SP)+,R3        ;;RESTORE R3
006312  012603                          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
006314  016666  000002  000004          MOV     (SP)+,(SP)
006322  012616                          RTI                     ;;RETURN
006324  000002                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
006326      000                 $OCNT:  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
006327      000                         .BYTE   0               ;;OCTAL DIGIT COUNTER
006330      000
```

```
1586   006331   000                  $OFILL: .BYTE   0                ;;ZERO FILL SWITCH
1587   006332   000000               $OMODE: .WORD   0                ;;NUMBER OF DIGITS TO TYPE
1588                                 .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
1589
1590                                 ;;*****************************************************************
1591                                 ;*SAVE R0-R5
1592                                 ;*CALL:
1593                                 ;*      SAVREG
1594                                 ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
1595                                 ;*
1596                                 ;*TOP---(+16)
1597                                 ;* +2---(+18)
1598                                 ;* +4---R5
1599                                 ;* +6---R4
1600                                 ;* +8---R3
1601                                 ;*+10---R2
1602                                 ;*+12---R1
1603                                 ;*+14---R0
1604
1605   006334                        $SAVREG:
1606   006334   010046                       MOV     R0,-(SP)         ;;PUSH R0 ON STACK
1607   006336   010146                       MOV     R1,-(SP)         ;;PUSH R1 ON STACK
1608   006340   010246                       MOV     R2,-(SP)         ;;PUSH R2 ON STACK
1609   006342   010346                       MOV     R3,-(SP)         ;;PUSH R3 ON STACK
1610   006344   010446                       MOV     R4,-(SP)         ;;PUSH R4 ON STACK
1611   006346   010546                       MOV     R5,-(SP)         ;;PUSH R5 ON STACK
1612   006350   016646   000022              MOV     22(SP),-(SP)     ;;SAVE PS OF MAIN FLOW
1613   006354   016646   000022              MOV     22(SP),-(SP)     ;;SAVE PC OF MAIN FLOW
1614   006360   016646   000022              MOV     22(SP),-(SP)     ;;SAVE PS OF CALL
1615   006364   016646   000022              MOV     22(SP),-(SP)     ;;SAVE PC OF CALL
1616   006370   000002                      RTI
1617
1618                                 ;*RESTORE R0-R5
1619                                 ;*CALL:
1620                                 ;*      RESREG
1621   006372                        $RESREG:
1622   006372   012666   000022              MOV     (SP)+,22(SP)     ;;RESTORE PC OF CALL
1623   006376   012666   000022              MOV     (SP)+,22(SP)     ;;RESTORE PS OF CALL
1624   006402   012666   000022              MOV     (SP)+,22(SP)     ;;RESTORE PC OF MAIN FLOW
1625   006406   012666   000022              MOV     (SP)+,22(SP)     ;;RESTORE PS OF MAIN FLOW
1626   006412   012605                       MOV     (SP)+,R5         ;;POP STACK INTO R5
1627   006414   012604                       MOV     (SP)+,R4         ;;POP STACK INTO R4
1628   006416   012603                       MOV     (SP)+,R3         ;;POP STACK INTO R3
1629   006420   012602                       MOV     (SP)+,R2         ;;POP STACK INTO R2
1630   006422   012601                       MOV     (SP)+,R1         ;;POP STACK INTO R1
1631   006424   012600                       MOV     (SP)+,R0         ;;POP STACK INTO R0
1632   006426   000002                      RTI
1633                                 .SBTTL  TRAP DECODER
1634
1635                                 ;;*****************************************************************
1636                                 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1637                                 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1638                                 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1639                                 ;*GO TO THAT ROUTINE.
1640
1641   006430   010046                $TRAP:  MOV     R0,-(SP)         ;;SAVE R0
```

```
1642  006432  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
1643  006436  005740                          TST     -(R0)           ;;BACKUP BY 2
1644  006440  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
1645  006442  006300                          ASL     R0              ;;POSITION FOR INDEXING
1646  006444  016000  006464                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
1647  006450  000200                          RTS     R0              ;;GO TO ROUTINE
1648
1649
1650                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
1651
1652  006452  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
1653  006454  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
1654  006462  000002                          RTI                     ;;RESTORE THE PSW
1655
1656                                  .SBTTL  TRAP TABLE
1657
1658                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1659                                  ;*BY THE "TRAP" INSTRUCTION.
1660
1661                                  ;       ROUTINE
1662                                  ;       -------
1663  006464  006452                  $TRPAD: .WORD   $TRAP2
1664  006466  004562                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
1665  006470  006132                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1666  006472  006106                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
1667  006474  006146                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
1668
1669  006476  005402                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING
1670
1671  006500  005332                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
1672  006502  005614                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
1673  006504  005734                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
1674  006506  006334                          $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
1675  006510  006372                          $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
1676
1677
1678                                  ;MESSAGES
1679
1680
1681  006512  030062                  ROMNUM: .ASCII  /20/
1682  006514  032463                          .ASCII  /35/
1683  006516  031463                          .ASCII  /33/
1684  006520  032063                          .ASCII  /34/
1685  006522  031063                          .ASCII  /32/
1686  006524  177607  000377          BELL:   .ASCIZ  <207><377><377>
1687
1688
1689  006530  044504  043501  020056  DRHEAD: .ASCIZ  /DIAG. ROM /
1690  006536  047522  020115    000
1691  006543    102   047517  051524  BRHEAD: .ASCII  /BOOTSTRAP ROM ENTRY POINTS AND DEVICE CODES/<15><12>
1692  006550  051124  050101  051040
1693  006556  046517  042440  052116
1694  006564  054522  050040  044517
1695  006572  052116  020123  047101
1696  006600  020104  042504  044526
1697  006606  042503  041440  042117
```

```
1698   006614  051505  005015
1699   006620  047011  020117  044504          .ASCIZ  /       NO DIAG.        RUN DIAG.       DEVICE CODE/
1700   006626  043501  004456  052522
1701   006634  020116  044504  043501
1702   006642  004456  042504  044526
1703   006650  042503  041440  042117
1704   006656  000105
1705   006660  051520  052505  047504  PFHEAD: .ASCIZ  /PSEUDO POWER-FAIL VECTOR ADDRESS /
1706   006666  050040  053517  051105
1707   006674  043055  044501  020114
1708   006702  042526  052103  051117
1709   006710  040440  042104  042522
1710   006716  051523  000040
1711   006722  047503  046125  020104  ER2MSG: .ASCIZ  /COULD NOT FIND DEVICE CODE /
1712   006730  047516  020124  044506
1713   006736  042116  042040  053105
1714   006744  041511  020105  047503
1715   006752  042504  000040
1716   006756  047506  047125  020104  ER1MSG: .ASCIZ  /FOUND UNEXPECTED DEVICE CODE /
1717   006764  047125  054105  042520
1718   006772  052103  042105  042040
1719   007000  053105  041511  020105
1720   007006  047503  042504  000040
1721   007014  047520  042527  026522  ER3MSG: .ASCII  /POWER-FAIL VECTOR ERROR/<15><12>
1722   007022  040506  046111  053040
1723   007030  041505  047524  020122
1724   007036  051105  047522  006522
1725   007044     012
1726   007045     011  054105  042520          .ASCIZ  /       EXPECTED        RECIEVED/<15><12>
1727   007052  052103  042105  051011
1728   007060  041505  042511  042526
1729   007066  006504  000012
1730   007072  047520  042527  026522  ER4MSG: .ASCII  /POWER-FAIL DATA ERROR/<15><12>
1731   007100  040506  046111  042040
1732   007106  052101  020101  051105
1733   007114  047522  006522     012
1734   007121     011  054105  042520          .ASCIZ  /       EXPECTED        RECIEVED/<15><12>
1735   007126  052103  042105  051011
1736   007134  041505  042511  042526
1737   007142  006504  000012
1738   007146  051103  020103  051105  CRCMSG: .ASCIZ  /CRC ERROR IN ROM E-/
1739   007154  047522  020122  047111
1740   007162  051040  046517  042440
1741   007170  000055
1742   007172  047503  046125  020104  PFMSG:  .ASCIZ  /COULD NOT DETERMINE POWER-FAIL VECTOR ADDRESS/
1743   007200  047516  020124  042504
1744   007206  042524  046522  047111
1745   007214  020105  047520  042527
1746   007222  026522  040506  046111
1747   007230  053040  041505  047524
1748   007236  020122  042101  051104
1749   007244  051505  000123
1750   007250  047516  051040  046517  NOROMM: .ASCII  /NO ROMS TEST ERROR/<15><12>
1751   007256  020123  042524  052123
1752   007264  042440  051122  051117
1753   007272  005015
```

```
1754  007274  042411  050130  041505          .ASCIZ  /       EXPECTED        RECIEVED/
1755  007302  042524  004504  042522
1756  007310  044503  053105  042105
1757  007316     000
1758  007317     015  042012  053105  SEQMSG: .ASCII  <15><12>/DEVICE CODES NOT IN ORDER SPECIFIED BY /
1759  007324  041511  020105  047503
1760  007332  042504  020123  047516
1761  007340  020124  047111  047440
1762  007346  042122  051105  051440
1763  007354  042520  044503  044506
1764  007362  042105  041040  020131
1765  007370  047111  052123  046101          .ASCIZ  /INSTALLATION PROCEDURE./
1766  007376  040514  044524  047117
1767  007404  050040  047522  042503
1768  007412  052504  042522  000056
1769                          .EVEN
1770
1771
1772                          ;BUFFERS
1773
1774  007420  000030          BUF1:
1775  007420  000000                  .WORD   0
1776  007422  000000                  .WORD   0
1777  007424  000000                  .WORD   0
1778  007426  000000                  .WORD   0
1779  007430  000000                  .WORD   0
1780  007432  000000                  .WORD   0
1781  007434  000000                  .WORD   0
1782  007436  000000                  .WORD   0
1783  007440  000000                  .WORD   0
1784  007442  000000                  .WORD   0
1785  007444  000000                  .WORD   0
1786  007446  000000                  .WORD   0
1787  007450  000000                  .WORD   0
1788  007452  000000                  .WORD   0
1789  007454  000000                  .WORD   0
1790  007456  000000                  .WORD   0
1791  007460  000000                  .WORD   0
1792  007462  000000                  .WORD   0
1793  007464  000000                  .WORD   0
1794  007466  000000                  .WORD   0
1795  007470  000000                  .WORD   0
1796  007472  000000                  .WORD   0
1797  007474  000000                  .WORD   0
1798  007476  000000                  .WORD   0
1799  007500  000000          BUF2:   .WORD   0
1800  007502  000000                  .WORD   0
1801  007504  000010          OCTBUF: .BLKB   10
1802  007514  000400          MES1:   .BLKB   400
1803  010114  000100          MES2:   .BLKB   100
1804  010214  000100          ERRMSG: .BLKB   100
1805
1806
1807          000001                  .END
```

```
ABASE  = 000000            506     547
ACDW1  = 000000            506     549
ACDW2  = 000000            506     550
ACPUOP= 000000             506     521
ADDW0  = 000000            506     551
ADDW1  = 000000            506     552
ADDW10= 000000             506     561
ADDW11= 000000             506     562
ADDW12= 000000             506     563
ADDW13= 000000             506     564
ADDW14= 000000             506     565
ADDW15= 000000             506     566
ADDW2  = 000000            506     553
ADDW3  = 000000            506     554
ADDW4  = 000000            506     555
ADDW5  = 000000            506     556
ADDW6  = 000000            506     557
ADDW7  = 000000            506     558
ADDW8  = 000000            506     559
ADDW9  = 000000            506     560
ADEVCT= 000000             506     512
ADEVM  = 000000            506     548
AENV   = 000000            506     517
AENVM  = 000000            506     518
AFATAL= 000000             506     509
AMADR1= 000000             506     534
AMADR2= 000000             506     538
AMADR3= 000000             506     541
AMADR4= 000000             506     544
AMAMS1= 000000             506     528
AMAMS2= 000000             506     536
AMAMS3= 000000             506     539
AMAMS4= 000000             506     542
AMSGAD= 000000             506     514
AMSGLG= 000000             506     515
AMSGTY= 000300             506     508
AMTYP1= 000000             506     529
AMTYP2= 000000             506     537
AMTYP3= 000000             506     540
AMTYP4= 000000             506     543
APASS  = 000000            506     511
APRIOR= 000000             506
APTCSU= 000040            1255    1369#
APTENV= 000001            1248    1325    1367#
APTER1= 000001             434#    826
APTER2= 000002             435#    840
APTER3= 000004             436#    851
APTER4= 000010             437#    857
APTSIZ= 000200             597    1366#
APTSP0= 000100            1250    1327    1368#
ASWREG= 000000             506     519
ATESTN= 000000             506     510
AUNIT  = 000000            506     513
AUSWR  = 000300            506     520
AVECT1= 000000             506     545
AVECT2= 000000             506     546
```

```
BELL    006524          1095    1686#
BIT0  = 000001           416#
BIT00 = 000001           406#     416
BIT01 = 000002           405#     415
BIT02 = 000004           404#     414
BIT03 = 000010           403#     413
BIT04 = 000020           402#     412
BIT05 = 000040           401#     411
BIT06 = 000100           400#     410
BIT07 = 000200           399#     409
BIT08 = 000400           398#     408
BIT09 = 001000           397#     407
BIT1  = 000002           415#
BIT10 = 002000           396#    1092
BIT11 = 004000           395#
BIT12 = 010000           394#
BIT13 = 020000           393#    1096
BIT14 = 040000           392#
BIT15 = 100000           391#    1145
BIT2  = 000004           414#
BIT3  = 000010           413#
BIT4  = 000020           412#
BIT5  = 000040           411#
BIT6  = 000100           410#
BIT7  = 000200           409#
BIT8  = 000400           408#
BIT9  = 001000           407#
BPTVEC= 000014           423#
BRHEAD  006543           963    1691#
BUF1    007420           622     817     832     879     953    1034    1774#
BUF2    007500           848     854     931*    932*    936*    937*    990     994    1799#
CALSUM  002462           730     743     766#
CHECKS  002244           657     724#
CKSWR = 104406          1671#
CR    = 000015           331#    958     965     966     993    1001    1108    1134    1170    1294    1304
CRCERR= 000020           438#    734     747
CRCLOP  002500           771#    780
CRCMSG  007146          1156    1738#
CRLF  = 000200           332#    621    1265    1304
DAFLAG  003252           886*    894     911*    917#   1025*   1028*   1029    1033*   1063    1066*   1069*
DDISP = 177570           338#    474     585
DEVCOD  003050           804     816     879#
DISPLA  001106           474#    585*    593*
DISPRE  000174           451#    593
DRHEAD  006530           956    1689#
DSWR  = 177570           337#    473     584
EHEADT  004362          1100    1152#
EMTVEC= 000030           426#
ERRCNT  001120           479#    1091*
ERRHAN  004062           711     735     748     810     827     841     852     858    1089#
ERRMSG  010214          1090    1140    1144    1804#
ERRVEC= 000004           419#    582     583*    594*    628     629*    641#
ER1MSG  006756          1152    1716#
ER2MSG  006722          1153    1711#
ER3MSG  007014          1154    1721#
ER4MSG  007072          1155    1730#
```

```
EXCADD  002554            728*   737*   750*   781    790#
FILBUF  004400            957    964    972    979    992    997    1107   1128   1132   1166#
FIRSTA  002552            729*   736*   749*   753    767    789#
FIRSTB  001114            477#   643    887
FRSCOD  003772           1042   1051#  1068
GETCOD  003776           1038   1041   1062#
GETEND  004060           1073   1080#
GNS   = ****** U          450    620    1664   1665   1666   1667   1669   1671   1672   1673   1674   1675
GTSWR = 104405            615   1669#
HT    = 000011            329#   973    980    984    998    1129   1133   1166   1168   1263   1304
IOTVEC= 000020            424#
LASCOD  003774           1042   1052#  1065
LASTAD  002556            727*   738*   751*   784    791#
LF    = 000012            330#   967    1002   1135   1171   1298   1304
LOOP    002472            769#   785
MESSAG  001112            476#   814*
MES1    007514            806   1802#
MES2    010114            813   1803#
NODATA  002242            707    716#
NOROME= 000100            440#   710
NOROMM  007250           1158   1750#
NOROMS  002204            655    706#
OCADD   004546            971    978    996   1215#
OCASC   004442           1127   1131   1185#  1217
OCTBUF  007504            624   1185   1190   1197   1199   1801#
PFERR = 000040            439#   808    939
PFHEAD  006660            991   1705#
PFMSG   007172           1157   1742#
PIRQ  = 177772            336#
PIRQVE= 000240            430#
PPFVAR  003254            807    846    929#
PROMP   002560            660    802#
PR0   = 000000            353#
PR1   = 000040            354#
PR2   = 000100            355#
PR3   = 000140            356#
PR4   = 000200            357#
PR5   = 000240            358#
PR6   = 000300            359#
PR7   = 000340            360#
PS    = 177776            333#   334
PSW   = 177776            334#
PUTMES  003340            805    812    948#
PWRVEC= 000024            425#
RDCHR = 104407           1485   1672#
RDLIN = 104410           1673#
RESREG= 104412           1149   1675#
RESVEC= 000010             420#
ROMCNT  002460            724*   725    740*   741    757#   1113
ROMERR  001110            475#   710*   734*   747*   808    826*   840*   851*   857*   939*   1098   1099   1109
                         1111   1148*
ROMFIN  001116            478#   626*   637*   648*   653    725    741    881    889    1022   1035
ROMNUM  006512           1114   1681#
RSTART  002044            456    653#   693
SAVREG= 104411           1089   1674#
SEQERR= 000200             441#
```

```
SEQEX   003770          1030    1032    1040    1045    1049#
SEQMSG  007317          1048    1758#
SEQTST  003646           661    1022#
STACK = 001100           324#    574
START   001400           454     572#
STKLMT= 177774           335#
SWR     001104           473#    584*    586     592*    599*    613     1092    1096    1145    1382    1419*
SWREG   000176           452#    592     613     1382    1395
SW0   = 000001           388#
SW00  = 000001           378#    388
SW01  = 000002           377#    387
SW02  = 000004           376#    386
SW03  = 000010           375#    385
SW04  = 000020           374#    384
SW05  = 000040           373#    383
SW06  = 000100           372#    382
SW07  = 000200           371#    381
SW08  = 000400           370#    380
SW09  = 001000           369#    379
SW1   = 000002           387#
SW10  = 002000           368#
SW11  = 004000           367#
SW12  = 010000           366#
SW13  = 020000           365#
SW14  = 040000           364#
SW15  = 100000           363#
SW2   = 000004           386#
SW3   = 000010           385#
SW4   = 000020           384#
SW5   = 000040           383#
SW6   = 000100           382#
SW7   = 000200           381#
SW8   = 000400           380#
SW9   = 001000           379#
TBITVE= 000014           421#
TEMP    004544          1191*   1193*   1201*   1203*   1205*   1208#
TESTAD  003250           643*    644     646     649*    651     883*    884     885     887*    891*    896     897     898*
                         900     901*    902     912     916#
TIMES   003644           627*    949     952*    1009#
TKVEC = 000060           428#
TPVEC = 000064           429#
TRAPVE= 000034           427#    576*    577*
TRTVEC= 000014           422#
TYPE  = 104401           607     682     683     1005    1047    1094    1268    1393    1394    1397    1410    1421    1440
                        1489    1492    1496    1571    1664#
TYPOC = 104402          1396    1665#
TYPON = 104404          1667#
TYPOS = 104403          1666#
$APTHD  001122           490     496#
$ASTAT= ****** U        1347    1362
$ATYC   005104          1318    1320#
$ATY1   005060          1139    1316#
$ATY3   005066          1143    1253    1317#
$ATY4   005076          1319#
$AUTOB  001100           470#    617*    1390    1511
$BASE   001212           547#    847
```

```
$CDW1   001216            549#    853
$CDW2   001220            550#
$CHARC  005040           1270*   1280*   1287    1296*   1301#
$CKSWR  005332           1382#   1671
$CMTAG= ****** U          574     578
$CNTLG  006057           1393    1506#
$CNTLU  006052           1410    1505#
$CPUOP  001164            521#
$CRLF   005055           1269    1311#   1421    1505
$DDW0   001222            551#    818     831
$DDW1   001224            552#
$DDW10  001246            561#
$DDW11  001250            562#
$DDW12  001252            563#
$DDW13  001254            564#
$DDW14  001256            565#
$DDW15  001260            566#
$DDW2   001226            553#
$DDW3   001230            554#
$DDW4   001232            555#
$DDW5   001234            556#
$DDW6   001236            557#
$DDW7   001240            558#
$DDW8   001242            559#
$DDW9   001244            560#
$DEVCT  001146            512#
$DEVM   001214            548#
$DOAGN  002162            678     685     691#
$ENDAD  002152            464     605     687#
$ENDCT  002126            579     680#
$ENDMG  002171            682     695#
$ENULL  002166            683     694#
$ENV    001156            517#    611     1137    1248    1325    1349
$ENVM   001157            518#    597     802     1250    1255    1327
$EOP    002102            656     659     672#
$EOPCT  002120            579*    677#    681
$ETABL  001156            516#
$ETEND  001262            502     569#
$FATAL  001140            509#   1353*
$FFLG   005324           1316*   1319*   1347    1356*   1364#
$FILLC  005052           1273    1308#
$FILLS  005051           1307#
$GET42  002142            684#
$GTSWR  005402           1394#   1669
$HD     = 000003          312     313
$HIBTS  001122            497#
$INTAG  001101            471#   1422    1511
$LF     005056           1312#   1496    1505
$LFLG   005323           1357*   1363#
$MADR1  001170            534#
$MADR2  001174            538#
$MADR3  001200            541#
$MADR4  001204            544#
$MAIL   001136            498     502     507#    596     611     1248
$MAMS1  001166            528#
$MAMS2  001172            536#
```

```
$MAMS3  001176            539#
$MAMS4  001202            542#
$MBADR  001124            498#
$MFLG   005322           1317*  1323   1358*  1362#
$MNEW   006075           1397   1509#
$MSGAD  001152            514#  1333*  1336
$MSGLG  001154            515#  1338*
$MSGTY  001136            508#  1331   1339*  1351   1355*
$MSWR   006064           1394   1507#
$MTYP1  001167            529#
$MTYP2  001173            537#
$MTYP3  001177            540#
$MTYP4  001203            543#
$NULL   005050           1275   1306#
$OCNT   006330           1543*  1572*  1585#
$OMODE  006332           1538*  1542*  1547   1550*  1561*  1587#
$PASS   001144            511#   578*   596*   658    674*   675*   694
$PASTM  001130            500#
$QUES   005054           1310#  1440   1489   1505
$RDCHR  005614           1453#  1672
$RDDEC= ****** U         1674
$RDLIN  005734           1481#  1673
$RDOCT= ****** U         1674
$RDSZ = 000010           1474#
$RESRE  006372           1621#  1675
$RTNAD  002164            693#
$R2A  = ****** U         1676
$SAVRE  006334           1605#  1674
$SETUP= 000124            455#   575    576    578    580    605    608    674   1377   1511
$STUP = 177777            455#
$SVPC = 000210            462#   467
$SWR  = 160000            312    313#   318    319    320    321    580   667    674    686    692    694
$SWREG  001160            519#   599
$TESTN  001142            510#
$TKB    005330           1374#  1386   1403   1457   1463
$TKS    005326           1373#  1384   1400   1424*  1455   1461
$TN   = 000001            312#
$TPB    005046           1293*  1305#
$TPFLG  005053           1242   1309#
$TPS    005044           1291   1304#
$TRAP   006430            576   1641#
$TRAP2  006452           1652#  1663
$TRP  = 000013           1656#  1665#  1666#  1667#  1668#  1669   1670#  1671   1672#  1673#  1674#  1675#  1676#
$TRPAD  006464           1646   1663#
$TSTM   001126            499#
$TTYIN  006042           1482   1483   1500   1504#
$TYPBN= ****** U         1668
$TYPDS= ****** U         1668
$TYPE   004562           1242#  1344   1656   1664
$TYPEC  004774           1272   1279   1286   1291#  1292   1426
$TYPEX  005042           1297   1299   1302#
$TYPOC  006132           1541#  1665
$TYPON  006146           1540   1543#  1667
$TYPOS  006106           1536#  1666
$UNIT   001150            513#
$UNITM  001132            501#
```

```
$USWR   001162            520#
$VECT1  001206            545#
$VECT2  001210            546#
$SGET4= 000000            686#
$OFILL  006331           1537*  1541*  1551   1586#
.       = 010314            443#   446#   450#   455#   462   463#   465#   467#   469#   486   487#   489#   491#
                            571#   694    697   1304   1305  1306  1307  1308  1309  1310  1311  1312  1313
                           1365#  1373   1374   1504#  1505  1511  1801#  1802#  1803#  1804#
.$ASTA= ****** U           1317   1320
.$X   = 001122              486#   491
```

```
COMMEN    431#
ENDCOM    431#
ERROR     325#
ESCAPE    431#
GETPRI    431#
GETSWR    431#    608#
MULT      431#
NEWTST    431#
POP       302#    431#   1359   1360   1626   1633
PUSH      302#    431#   1320   1322   1343   1606   1633
REPORT    302#    431#
SCOPE     326#
SETPRI    431#
SETTRA   1656#   1665   1666   1667   1669   1671   1672   1673   1674   1675
SETUP     302#    431#    572
SKIP      431#
SLASH     431#
SOFTSW    302#
SPACE     431#
STARS     431#    433    460    483    485    492    505    664   1227   1315   1372   1377   1445   1474   1513
         1590   1635
SWRSU     431#    580#
TRMTRP   1656#
TYPBIN    431#
TYPDEC    431#
TYPNAM    302#    431#    601
TYPNUM    431#
TYPOCS    431#
TYPOCT    431#   1395
TYPTXT    431#
$$ESCA    431#
$$NEWT    431#
$$SET    1656#   1665   1666   1667   1669   1671   1672   1673   1674   1675
$$SETM    596#
$$SKIP    431#
.EQUAT    302#    321
.HEADE    302#
.SETUP    302#    455
.STARS    302#
.SWRHI    302#    314
.SWRLO    321#
.$ACTI    302#    458
.$APTB    302#    503
.$APTH    302#    481
.$APTY    302#   1313
.$ASTA    302#
.$CATC    302#    444
.$CMTA    302#
.$EOP     302#    662
.$READ    302#   1370
.$SAVE    302#   1588
.$TRAP    302#   1633
.$TYPE    302#   1225
.$TYPO    302#   1511
```

. ABS. 010314 000

ERRORS DETECTED: 0

CZM9BA,CZM9BA/SOL/CRF/NL:TOC=CZM9BA.P11
RUN-TIME: 20 5 .7 SECONDS
RUN-TIME RATIO: 97/27=3.5
CORE USED: 20K (39 PAGES)

E04