```
                        .REM    %
```

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DVAAA-A

PRODUCT NAME: AAV11 DIAGNOSTIC TEST

DATE: OCTOBER 1976

MAINTAINER: DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

1.0 ABSTRACT

THE AAV11 DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED
TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE.
TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL
VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11/03 COMPUTER OR LSI-11 PROCESSOR
2. DLV11 WITH I/O TYPE TERMINAL
3. AAV11 DAC OPTION

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE

1. ASSURE THAT THE LSI-11 IS IN THE ODT MICROCODE STATE.
2. LOAD THE LOW OR HIGH SPEED READER WITH THE ABSOLUTE LOADER TAPE.
3. TYPE THE READER'S CSR ADDRESS (177560-LOW OR 177550-HIGH)
   AND CHARACTER 'L'.
4. AFTER TAPE IS LOADED, LOAD THE AAV11 BINARY TAPE INTO THE
   READER AND TYPE THE CHARACTER 'R'.
5. IF THE ABSOLUTE LOADER HAS ALREADY BEEN LOADED (STEPS 2 & 3),
   THEN ONLY THE STARTING ADDRESS OF THE ABSOLUTE LOADER AND
   THE CHARACTER 'G' NEED BE TYPED (WITH THE AAV11 BINARY TAPE
   IN THE APPROPRIATE READER).

4.0 STARTING PROCEDURE

1. MAKE SURE THE DEVICE BUS ADDRESS AGREE WITH THE
   DEFAULT VALUES DEFINED IN SECTION 7.1. IF NOT, CHANGE
   LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' ODT COMMAND.
2. INSURE THAT THE HALT SWITCH IS DISABLED (IF ANY).
4. TYPE THE STARTING ADDRESS OF 200 AND THE CHARACTER G.
5. THE PROGRAM WILL RESPOND BY TYPING THE PROGRAM TITLE.

4.1 PROGRAM START

200	STARTING ADDRESS OF THE LOGIC TEST (WITH UP TO FOUR AAV1
204	STARTING ADDRESS OF THE RAMP LOOP
210	STARTING ADDRESS OF THE STATIC CALIBRATION
214	STARTING ADDRESS OF THE DYNAMIC CALIBRATION
230	STARTING ADDRESS OF THE LOGIC TEST (WITH UP TO SIXTEEN A
240	STARTING ADDRESS FOR THE OPTION TESTER.

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR (7-0)

5.2 CONTROL

1. THE SOFTWARE SWITCH REGISTER 'SWREG' (LOC. 176) CAN BE CHANGED BY USING THE ODT FACILITIES.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CONTROL & G' KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.
3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING 'P' (CONTINUE).
4. IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL 'CONTROL & G' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE BY THE PROGRAM.

6.0 ERROR REPORTING

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF.

6.2 ERROR DATA

*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*BUSADR	AAV11 BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

*ALWAYS REPORTED

7.0 MISCELLANEOUS

7.1 AAV11 BUS ADDRESS MODIFICATION

MODIFY LOCATION '$BASE' (LOC. 1250) IF BASE BUS ADDRESS IS NOT 170440.

*NOTE: USE THE LSI-11 ODT FACILITIES TO MODIFY THIS LOCATIONS
AFTER PROGRAM LOAD.

7.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REQUIRES 8K OR MORE).
THIS DIAGNOSTIC DOES SUPPORT "APT" BUT HAS NOT BEEN RUN UNDER IT

7.3 POWER FAIL

A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT
WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH
NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).

7.4 MULTIPLE AAV11 INTERFACE TESTING

THIS PROGRAM DOES "AUTO-SIZE" THE NUMBER OF AAV11'S CONNECTED.
THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 4 AAV11 INTERFACES,
WHEN STARTED AT 200 AND 16. WHEN STARTED AT ADDRESS 230,
WITH CONTIGUOUS BUS ADDRESSES. THE "AUTO-SIZE" CAN BE INHIBITED
BY THE OPERATOR SETTING BIT 15 OF LOCATION "$ENV (LOC. 1214) AND
LOADING LOCATION '$BASE' WITH THE ADDRESS OF THE ONE UNIT TO BE TESTED.

7.5 RESTRICTIONS

NONE

8.0 EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS
TO ABOUT 20 SECONDS WITH ITERATIONS ENABLED WITH ONE AAV11 CONNECTED.
AN END PASS MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS.
END OF PASS WILL ALSO REPORT TOTAL ERROR COUNT AND ANY UNIT'S THAT HAD ERRORS

9.0 PROGRAM TEST DESCRIPTIONS

9.1 LOGIC TESTS (SA 200)

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE AAV11 DAC CONTROL. A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING.
WHEN STARTED AT LOCATION 200, THE PROGRAM WILL AUTO-SIZE UP TO 4 AAV11'S TO BE TESTED.

9.2 RAMP LOOP (SA 204)

THIS LOOP IS PROVIDED A METHOD FOR THE OPERATOR TO INSPECT AND VERIFY ANALOG OPERATION OF ALL DAC BITS. THE LOOP ALSO ENABLES THE OPERATOR TO VERIFY THAT NO TWO DAC'S ARE INTERCONNECTED.

9.3 STATIC CALIBRATION LOOP (SA 210)

THIS LOOP PROVIDES THE OPERATOR WITH A SIMPLE LOOP FOR VERIFYING THE INDIVIDUAL DAC BITS AND THE OPERATION OF DAC #3 DIGITAL OUTPUT BITS. THE VALUE OF THE SWITCH REGISTER IS LOADED INTO ALL DAC'S AND THE OUTPUT VOLTAGE CAN BE MONITORED.

9.4 DYNAMIC CALIBRATION LOOP (SA 214)

THIS PROVIDES THE OPERATOR WITH A LOOP THAT LOADS THE VALUE OF THE SWITCH REGISTER INTO THE DAC'S AND THEN AFTER A DELAY CLEARS THE DAC REGISTERS. THIS PROVIDES A SWITCHING PATTERN BETWEEN THE SELECTED VOLTAGE AND 0.

9.5 EXTENDED UNITS (SA 230)

SAME FUNCTION AS LOGIC TEST BUT ON 16. AAV11'S

9.6 TESTER SUPPORT (SA 240)

INITIALL PERFORMS THE LOGIC TESTS AND THEN EMPLOYS A KNOWN GOOD A TO D CONVERTER TO AID IN ADJUSTING THE POT'S ON THE AAV11 BOARD. THE OPERATOR IS INFORMED AS TO WHICH POT TO ADJUST AND WHICH D TO A CONVERTER IS TESTED.

10.0 LISTING

```
                        %
.TITLE  MAINDEC-11-DVAAA-A      AAV11   DIAGNOSTIC
;*COPYPIGHT (C) 1976
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY RAYMOND SHOOP
;*
```

```
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C1),MAR 24, 1976.
                        ;*
                        .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
                        .EQUIV  R6,SP           ;;STACK POINTER
                        .EQUIV  R7,PC           ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
```

```
                        SW06=   100
                        SW05=   40
                        SW04=   20
                        SW03=   10
                        SW02=   4
                        SW01=   2
                        SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
```

```
  404          000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
  405          000034                  TRAPVEC=34              ;;"TRAP" TRAP
  406          000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
  407          000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
  408          000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
  409
  410          170440                          ABASE=170440
```

```
                                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                                        ;*
                                        ;*      SWITCH                  USE
                                        ;*      ------          ------------------
                                        ;*        15            HALT ON ERROR
                                        ;*        14            LOOP ON TEST
                                        ;*        13            INHIBIT ERROR TYPEOUTS
                                        ;*        11            INHIBIT ITERATIONS
                                        ;*        10            BELL ON ERROR
                                        ;*         9            LOOP ON ERROR
                                        ;*         8            LOOP ON TEST IN SWR<7:0>
                                        .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  001450                  JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  006570                  JMP     FULRMP                  ;JUMP TO FULL RAMP LOOP
000210  000137  006656                  JMP     STATIC                  ;JUMP TO STATIC DAC CALIBRATION
000214  000137  006716                  JMP     DYNCAL                  ;JUMP TO DYNAMIC DAC CALIBRATION

                000230                  .=230
000230  000137  001440                  JMP     ADDOK                   ;JUMP AND ENABLE EXTENDED UNITS (16.)

                000240                  .=240
000240  000137  001432                  JMP     TESTER                  ;JUMP TO TESTER SA.

                000100                  .=100
000100  000104  000200  000002          104,200,2                       ;B EVENT SAFE GUARD
```

```
                                .SBTTL  ACT11 HOOKS

                                ;;*****************************************************************
                                ;HOOKS REQUIRED BY ACT11
                                        $SVPC=.                 ;SAVE PC
000106                                  .=46
000046                                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
000046  005542                          .=52
000052                                  .WORD   0               ;;2)SET LOC.52 TO ZERO
000052  000000                          .=$SVPC                 ;; RESTORE PC
000106                                  .=1000
001000                          .SBTTL  APT PARAMETER BLOCK

                                ;;*****************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;*****************************************************************
001000                                  .$X=.   ;;SAVE CURRENT LOCATION
000024                                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
000044                                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
001000                                  .=.$X   ;;RESET LOCATION COUNTER
                                ;;*****************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001174                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000030                  $TSTM:  .WORD   30      ;;RUN TIME OF LONGEST TEST
001006  000010                  $PASTM: .WORD   10      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000030                  $UNITM: .WORD   30      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000031                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS

                                ;;*****************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                  .WORD   0
001102  000                     $TSTNM: .BYTE   0       ;;CONTAINS THE TEST NUMBER
001103  000                     $ERFLG: .BYTE   0       ;;CONTAINS ERROR FLAG
001104  000000                  $ICNT:  .WORD   0       ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                  $LPADR: .WORD   0       ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                  $LPERR: .WORD   0       ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                  $ERTTL: .WORD   0       ;;CONTAINS TOTAL ERRORS DETECTED
001114  000                     $ITEMB: .BYTE   0       ;;CONTAINS ITEM CONTROL BYTE
001115  001                     $ERMAX: .BYTE   1       ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                  $ERRPC: .WORD   0       ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                  $GDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0       ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0       ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0       ;;CONTAINS 'BAD' DATA
001130  000000                          .WORD   0       ;;RESERVED--NOT TO BE USED
001132  000000                          .WORD   0
001134  000                     $AUTOB: .BYTE   0       ;;AUTOMATIC MODE INDICATOR
001135  000                     $INTAG: .BYTE   0       ;;INTERRUPT MODE INDICATOR
001136  000000                          .WORD   0
001140  177570                  SWR:    .WORD   SWR     ;;ADDRESS OF SWITCH REGISTER
001142  177570                  DISPLAY: .WORD  DDISP   ;;ADDRESS OF DISPLAY REGISTER
001144  177560                  $TKS:   177560          ;;TTY KBD STATUS
001146  177562                  $TKB:   177562          ;;TTY KBD BUFFER
001150  177564                  $TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566                  $TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000                     $NULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002                     $FILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012                     $FILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000                     $TPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000                  $TIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001162  000000                  $ESCAPE:0               ;;ESCAPE ON ERROR ADDRESS
001164  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001170  077                     $QUES:  .ASCII  /?/     ;;QUESTION MARK
001171  015                     $CRLF:  .ASCII  <15>    ;;CARRIAGE RETURN
001172  000012                  $LF:    .ASCIZ  <12>    ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*****************************************************************
                                .EVEN
001174                  $MAIL:                          ;;APT MAILBOX
001174  000000                  $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
001176  000000                  $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
001200  000000                  $TESTN: .WORD   ATESTN   ;;TEST NUMBER
001202  000000                  $PASS:  .WORD   APASS    ;;PASS COUNT
001204  000000                  $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
```

```
  533  001206  000000                $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
  534  001210  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
  535  001212  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
  536  001214                        $ETABLE:                ;;APT ENVIRONMENT TABLE
  537  001214      000               $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
  538  001215      000               $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
  539  001216  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
  540  001220  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
  541  001222  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
  542                                ;*                      BITS 15-11=CPU TYPE
  543                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
  544                                ;*                              11/70=06,PDQ=07,Q=10
  545                                ;*                      BIT 10=REAL TIME CLOCK
  546                                ;*                      BIT  9=FLOATING POINT PROCESSOR
  547                                ;*                      BIT  8=MEMORY MANAGEMENT
  548  001224      000               $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
  549  001225      000               $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
  550                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
  551                                ;*                              900 NSEC CORE=001
  552                                ;*                              300 NSEC BIPOLAR=002
  553                                ;*                              500 NSEC MOS=003
  554  001226  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
  555                                ;*                       MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
  556  001230      000               $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
  557  001231      000               $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
  558  001232  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
  559  001234      000               $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
  560  001235      000               $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
  561  001236  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
  562  001240      000               $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
  563  001241      000               $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
  564  001242  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
  565  001244  000000                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
  566  001246  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
  567  001250  170440                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
  568  001252  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
  569  001254  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
  570  001256                        $ETEND:
  571                                .MEXIT
```

```
                                .SBTTL  ERROR POINTER TABLE

                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS

                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT


001256                          $ERRTB:

                                ;ITEM   1
001256  007122                          EM1                     ;BUT TIME-OUT WHE  REF. A DAC ADDRESS
001260  010322                          DH2                     ;ERRPC  BUSADR
001262  011113                          DT1                     ;$ERRPC $BDDAT
001264  011204                          DF0

                                ,ITEM   2
001266  007176                          EM2                     ;DAC #0 REGISTER IN ERROR
001270  010267                          DH1                     ;ERRPC  BUSADR  GOOD    BAD
001272  011120                          DT2                     ;$ERRPC DAC0    $GDDAT  $BDDAT
001274  011204                          DF0

                                ;ITEM   3
001276  007225                          EM3                     ;DAC #1 REGISTER IN ERROR
001300  010267                          DH1                     ;ERRPC  BUSADR  GOOD    BAD
001302  011132                          DT3                     ;$ERRPC DAC1    $GDDAT  $BDDAT
001304  011204                          DF0

                                ;ITEM   4
001306  007254                          EM4                     ,DAC #2 REGISTER IN ERROR
001310  010267                          DH1                     ;ERRPC  BUSADR  GOOD    BAD
001312  011144                          DT4                     ;$ERRPC DAC2    $GDDAT  $BDDAT
001314  011204                          DF0

                                ;ITEM   5
001316  007303                          EM5                     ;DAC #3 REGISTER IN ERROR
001320  010267                          DH1                     ;ERRPC  BUSADR  GOOD    BAD
001322  011156                          DT5                     ;$ERRPC DAC3    $GDDAT  $BDDAT
001324  011204                          DF0

                                ;ITEM   6
001326  007332                          EM6                     ;SELECTED DAC OFFSET POT IS NOT ADJUSTED CORRECTLY
001330  010337                          DH6                     ;ERRPC  BUSADR  EXPECT  MAS     SPREAD
001332  011170                          DT6                     ;$ERRPC DACBAD  $GDDAT  $BDDAT  SPREAD
001334  011204                          DF0
```

ERROR POINTER TABLE

```
                                    ;ITEM   7
001340  007413                      EM7
001340  010337                      DH6
001342  011170                      DT6
001344  011204                      DF0

                                    ;ITEM   10
001346  007472                      EM10
001350  010337                      DH6
001352  011170                      DT6
001354  011204                      DF0

                                    ITEM    11
001356  007537                      EM11
001360  010337                      DH6
001362  011170                      DT6
001364  011204                      DF0

                                    ;ITEM   12
001366  007575                      EM12
001370  010337                      DH6
001372  011170                      DT6
001374  011204                      DF0

                                    ;ITEM   13
001376  007631                      EM13
001400  010267                      DH1
001402  011156                      DT5
001404  011204                      DF0

                                    ;ITEM   14
001406  007675                      EM14
001410  000000                      0
001412  000000                      0
001414  000000                      0

001416  000010              NXTADR: 10
001420  000000              EVER:   0
001422  170440              DAC0:   ABASE
001424  170442              DAC1:   ABASE+2
001426  170444              DAC2:   ABASE+4
001430  170446              DAC3:   ABASE+6
```

```
;SELECTED DAC GAIN POT IS NOT ADJUSTED CORRECTLY
;ERRPC  BUSADR  EXPECT  WAS     SPREAD
;$ERRPC DACBAD  $GDDAT  $BDDAT  SPREAD

;SELECTED DAC HAS A LINEARITY PROBLEM
;ERRPC  BUSADR  EXPECT  WAS     SPREAD
;$ERRPC DACBAD  $GDDAT  $BDDAT  SPREAD

;+15 VOLT SUPPLY IS INCORRECT
;ERRPC  BUSADR  EXPECT  WAS     SPREAD
;$ERRPC DACBAD  $GDDAT  $BDDAT  SPREAD

;-15 VOLT SUPPLY IS INCORRECT
;ERRPC  BUSADR  EXPECT  WAS     SPREAD
;$ERRPC DACBAD  $GDDAT  $BDDAT  SPREAD

;DAC #3 DIGITAL OUTPUT BITS IN ERROR
;ERRPC  BUSADR  GOOD    BAD
;$ERRPC DAC3    $GDDAT  $BDDAT

;WAKE UP OPERATOR AND ADJUST THE POT

;OFFSET TO NEXT ARV11 ADDRESS
```

```
  667  001432  005237  007020          TESTER: INC     @FTST           ;INDICATE TESTER MODE
  668  001436  000411                  BR      BEGIN1
  669  001440  012737  000021  007006  ADOOK:  MOV     #17.,NUMBOK     ;LOAD 16 MAX UNITS
  670  001446  000403                  BR      BEGINA
  671  001450  012737  000005  007006  BEGIN:  MOV     #5.,NUMBOK      ;LOAD 4 MAX UNITS
  672  001456  005037  007020          BEGINA: CLR     @FTST
  673  001462  005037  007012          BEGIN1: CLR     TEMP
  674  001466  005037  001420                  CLR     EVER
  675  001472  000005                          RESET
  676                                  .SBTTL  INITIALIZE THE COMMON TAGS
  677                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
  678  001474  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
  679  001500  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
  680  001502  022706  001140                  CMP     #SWR,R6 ;;DONE?
  681  001506  001374                          BNE     .-6             ;;LOOP BACK IF NO
  682  001510  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
  683                                  ;;INITIALIZE A FEW VECTORS
  684  001514  012737  011524  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
  685  001522  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
  686  001530  012737  012006  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
  687  001536  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
  688  001544  012737  014412  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
  689  001552  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
  690  001560  012737  012336  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
  691  001566  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
  692  001574  005037  001160                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
  693  001600  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
  694  001604  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
  695  001612  012737  001612  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
  696  001620  012737  001620  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
  697                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
  698                                  ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
  699  001626  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
  700  001632  012737  001666  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
  701  001640  012737  177570  001140          MOV     #OSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
  702  001646  012737  177570  001142          MOV     #ODISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
  703  001654  022777  177777  177256          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
  704  001662  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
  705                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
  706  001664  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
  707  001666  012716  001674          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
  708  001672  000002                          RTI
  709  001674  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
  710  001702  012737  000174  001142          MOV     #DISPREG,DISPLAY
  711  001710  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
  712
  713  001714  005037  001202                  CLR     $PASS           ;;CLEAR PASS COUNT
  714  001720  132737  000200  001215          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
  715  001726  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
  716  001730  012737  001216  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
  717  001736                          67$:
  718  001736  005037  007004                  CLR     BADUNT          ;RESET BAD INDICATOR
  719  001742  000137  002044                  JMP     INIT1
```

```
                                        ;SUBROUTINE TO LOAD A TRAP CATCHER
                                        ;
001746  012702  000252          LDTRAP: MOV     #252,R2                 ;LOAD R2
001752  012701  000250                  MOV     #250,R1                 ;LOAD R1
001756  010221                  5$:     MOV     R2,(R1)+                ;LOAD .+2
001760  005021                          CLR     (R1)+                   ;LOAD HALT
001762  010102                          MOV     R1,R2                   ;LOAD R2
001764  005722                          TST     (R2)+                   ;BUMP R2
001766  020227  001002                  CMP     R2,#1002                ;TEST FOR LAST
001772  001371                          BNE     5$                      ;BR UNTIL DONE
                                        ;
                                        ;AND LOAD DEVICE ADDRESSES LOCATIONS
                                        ;
001774  013700  001250                  MOV     @#BASE,R0               ;GET BASE ADDRESS
002000  010037  001422                  MOV     R0,DAC0         ;LOAD X ADDRESS
002004  010037  001424                  MOV     R0,DAC1         ;LOAD Y ADDRESS
002010  010037  001426                  MOV     R0,DAC2                 ;LOAD DAC #2
002014  010037  001430                  MOV     R0,DAC3                 ;LOAD DAC #3
002020  062737  000002  001424          ADD     #2,DAC1
002026  062737  000004  001426          ADD     #4,DAC2
002034  062737  000006  001430          ADD     #6,DAC3
002042  000207                          RTS     PC                      ;EXIT
                                        ;
002044  004737  001746          INIT1:  JSR     PC,LDTRAP
002050  005737  007012                  TST     TEMP                    ;TEST IF START OR RESTART
002054  001012                          BNE     MTEST                   ;RESTART
002056  005737  000042                  TST     @#42                    ;TEST IF MONITOR
002062  001007                          BNE     MTEST                   ;BR IF NOT
002064  005737  007020                  TST     WFTST                   ;TEST IF ON TESTER
002070  001402                          BEQ     1$                      ;BR IF NOT
002072  104401                          TYPE
002074  010011                          MSGSW                           ;TELL OPERATOR ABOUT TESTER SWITCHES
002076  104401                  1$:     TYPE                            ;CALL MESSAGE PRINTER VIA 'EMT'
002100  007040                          TITLE                           ;TYPE PROGRAM HEADER.
```

```
                                        .SBTTL  DETERMINE THE NUMBER OF AAV11 ON THIS SYSTEM
                                
002102  013737  001250  001126  MTEST:  MOV     $BASE,$BDOAT    ;GET THE BASE ADDRESS
002110  005037  007010          MTEST:  CLR     MASKNM
002114  005037  001206                  CLR     $UNIT           ;CLEAR UNIT #
002120  012737  002164  000004          MOV     #2$,ERR\EC      ;LOAD TRAP RETURN
002126  005777  176774          1$:     TST     @$BDOAT         ;TEST IF ADDR EXISTS
002132  063737  001416  001126          ADD     VADDR,$BDOAT    ;UPDATE THE BUS ADDRESS
002140  005237  001206                  INC     $UNIT           ;UPDATE UNIT COUNT
002144  005737  001214                  TST     $ENV            ;TEST IF "DO NOT SIZE"
002150  100413                          BMI     3$              ;BR IF NO SIZEING
002152  023737  007006  001206          CMP     NUMBOK,$UNIT    ;TEST IF MAX. NUMBER
002160  001362                          BNE     1$              ;BR IF NOT
002162  000406                          BR      3$              ;BR IF MAX.
002164  022626                  2$:     CMP     (SP)+,(SP)+     ;CLEAN THE STACK
002166  005737  001206                  TST     $UNIT           ;TEST IF ANY EXIST
002172  001002                          BNE     3$              ;BR IF SOME ARE THERE
002174  104001                          ERROR   1               ;BASE ADDRESS CAUSED AN BUS TRAP
002176  000443                          BR      TST1            ;IS $BASE CORRECT??
002200  005737  001420          3$:     TST     EVER            ;TEST IF # HAS BEEN REPORTED
002204  100422                          BMI     4$              ;BR IF IT HAS
002206  005737  007020                  TST     WFTST           ;TEST IF TESTER MODE
002212  001010                          BNE     6$              ;BR IF TESTER
002214  104401                          TYPE
002216  010224                          FOUND1                  ;TELL OPERATOR THE # OF AAV11'S
002220  013746  001206                  MOV     $UNIT,-(SP)
002224  104403                          TYPOS
002226     002                          .BYTE   2
002227     000                          .BYTE   0
002230  104401                          TYPE
002232  010250                          FOUND2
002234  013737  001206  001420  6$:     MOV     $UNIT,EVER      ;SAVE THE # OF AAV11'S FOR LATER
002242  052737  100000  001420          BIS     #BIT15,EVER     ;SET "REPORTED # FLAG"
002250  000405                          BR      5$              ;
002252  123737  001420  001206  4$:     CMPB    EVER,$UNIT      ;TEST IF ANY HAVE GONE AWAY
002260  001401                          BEQ     5$              ;BR IF ALL ARE STILL HERE
002262  104013                          ERROR   13              ;EXISTING UNIT FAILED TO RESPOND NOW
002264  005037  001206          5$:     CLR     $UNIT           ;RESET UNIT POINTER
002270  005037  001204                  CLR     $DEVCT          ;MAKE APT HAPPY
002274  004737  001746                  JSR     PC,LDTRAP       ;LOAD TRAP CATCHER AND BUS ADDRESSES
002300  012737  000001  007010          MOV     #BIT0,MASKNM    ;LOAD MASK NUMBER IF ERROR
```

```
                                        ;;*****************************************************************
                                        ;*TEST 1         TEST THAT THE AAV11 RESPONDS TO THE CPU
                                        ;;*****************************************************************
002306  000004                  TST1:   SCOPE
002310  012737  002340  000004          MOV     #1$,ERRVEC      ;LOAD BUS TRAP RETURN
002316  005777  177100                  TST     @DAC0           ;TEST DAC #0
002322  005777  177076                  TST     @DAC1           ;TEST DAC #1
002326  005777  177074                  TST     @DAC2           ;TEST DAC #2
002332  005777  177072                  TST     @DAC3           ;TEST DAC #3
002336  000407                        BR      2$              ;;BR AND RESTORE LOC. 4
002340  022626                  1$:     CMP     (SP)+,(SP)+     ;CLEAN THE STACK
002342  104001                          ERROR   1               ;ERROR, BUS TIMEOUT WHEN ADDRESSING THE AAV11
002344  012737  000006  000004          MOV     #6,ERRVEC       ;LOAD LOC 4
002352  000137  004530                  JMP     REMAIN          ;TEST IF ANY OTHER'S
002356  012737  000006  000004  2$:     MOV     #6,ERRVEC       ;LOAD RETURN
                                        ;;*****************************************************************
                                        ;*TEST 2         TEST THAT DAC0 REGISTER CAN BE CLEARED
                                        ;;*****************************************************************
002364  000004                  TST2:   SCOPE
002366  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED
002372  013777  001124  177022          MOV     $GDDAT,@DAC0            ;LOAD REG
002400  017737  177016  001126          MOV     @DAC0,$BDDAT            ;READ REG
002406  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
002414  001401                        BEQ     TST3            ;;BR IF EQUAL
002416  104002                        ERROR   2                       ;ERROR, DAC0 REGISTER NOT = 0

                                        ;;*****************************************************************
                                        ;*TEST 3         TEST THAT DAC0 REGISTER CAN BE LOADED WITH #7777
                                        ;;*****************************************************************
002420  000004                  TST3:   SCOPE
002422  012737  007777  001124          MOV     #7777,$GDDAT            ;LOAD EXPECTED
002430  013777  001124  176764          MOV     $GDDAT,@DAC0            ;LOAD REG
002436  017737  176760  001126          MOV     @DAC0,$BDDAT            ;READ REG
002444  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
002452  001401                        BEQ     TST4            ;;BR IF EQUAL
002454  104002                        ERROR   2                       ;ERROR, DAC0 REGISTER NOT = 7777

                                        ;;*****************************************************************
                                        ;*TEST 4         TEST THAT DAC0 REGISTER CAN HOLD A FLOATING 1 PATTERN
                                        ;;*****************************************************************
002456  000004                  TST4:   SCOPE
002460  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
002466  012737  004000  001124          MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
002474  013777  001124  176720  1$:     MOV     $GDDAT,@DAC0            ;LOAD DAC0 REGISTER
002502  017737  176714  001126          MOV     @DAC0,$BDDAT            ;READ THE REGISTER
002510  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE THE DATA
002516  001401                        BEQ     2$              ;;BR IF SAME
002520  104002                        ERROR   2                       ;ERROR, DAC0 REGISTER FAILED TO HOLD A FLOATING
002522  006237  001124          2$:     ASR     $GDDAT                  ;CHANGE THE DATA
002526  001362                        BNE     1$                      ;BR AND TEST MORE DATA
```

```
                                        ;;*****************************************************************
                                        ;*TEST 5         TEST THAT DAC0 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
                                        ;;*****************************************************************
002530  000004                  TST5:   SCOPE
002532  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
002540  012737  004000  001124          MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
002546  013777  001124  176646          MOV     $GDDAT,@DAC0            ;LOAD DAC0
002554  017737  176642  001126  1$:     MOV     @DAC0,$BDDAT            ;READ THE REGISTER
002562  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE THE GOOD TO DAC0
002570  001407                          BEQ     2$              ;;BR IF THE SAME
002572  017737  176624  001126          MOV     @DAC0,$BDDAT            ;SAVE FOR TYPEOUT
002600  104002                          ERROR   2               ;DAC0 FAILED TO HOLD A FLOATING 1 PATTERN
002602  013777  001124  176512          MOV     $GDDAT,@DAC0            ;LOAD DAC0 AGAIN
002610  006277  176606          2$:     ASR     @DAC0           ;CHANGE THE DATA
002614  006237  001124                  ASR     $GDDAT          ;CHANGE THE EXPECTED
002620  001355                          BNE     1$              ;BR IF MORE DATA
                                        ;;*****************************************************************
                                        ;*TEST 6         TEST THE "SUB" INSTRUCTION WORKS ON DAC0
                                        ;;*****************************************************************
002622  000004                  TST6:   SCOPE
002624  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
002632  012737  007777  001124          MOV     #7777,$GDDAT            ;LOAD EXPECTED
002640  013777  001124  176554          MOV     $GDDAT,@DAC0            ;LOAD DAC0
002646  162737  000001  001124  1$:     SUB     #1,$GDDAT               ;SUB A VALUE
002654  162777  000001  176540          SUB     #1,@DAC0                ;FROM EXPECTED AND DAC0
002662  017737  176534  001126          MOV     @DAC0,$BDDAT            ;READ THE REGISTER
002670  023737  001124  001126          CMP     $GDDAT,$BDDATI          ;COMPARE
002676  001404                          BEQ     2$              ;;BR IF SAME
002700  104002                          ERROR   2               ;THE SUB INSTRUCTION FAILED ON DAC0
002702  013777  001124  176512          MOV     $GDDAT,@DAC0            ;LOAD THE REGISTER AGAIN
002710  005737  001124          2$:     TST     $GDDAT                  ;TEST FOR MORE DATA
002714  001354                          BNE     1$              ;;BR IF MORE DATA
                                        ;;*****************************************************************
                                        ;*TEST 7         TEST THAT DAC1 REGISTER CAN BE CLEARED
                                        ;;*****************************************************************
002716  000004                  TST7:   SCOPE
002720  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED
002724  013777  001124  176472          MOV     $GDDAT,@DAC1            ;LOAD DAC1
002732  017737  176466  001126          MOV     @DAC1,$BDDAT            ;READ REG
002740  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
002746  001401                          BEQ     TST10           ;;BR IF EQUAL
002750  104003                          ERROR   3               ;ERROR, DAC1 REGISTER NOT = 0
                                        ;;*****************************************************************
                                        ;*TEST 10        TEST THAT DAC #1 REGISTER CAN BE LOADED WITH #7777
                                        ;;*****************************************************************
002752  000004                  TST10:  SCOPE
002754  012737  007777  001124          MOV     #7777,$GDDAT            ;LOAD EXPECTED
002762  013777  001124  176434          MOV     $GDDAT,@DAC1            ;LOAD DAC #1
002770  017737  176430  001126          MOV     @DAC1,$BDDAT            ;READ REG
002776  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003004  001401                          BEQ     TST11           ;;BR IF EQUAL
003006  104003                          ERROR   3               ;ERROR, DAC1 REGISTER NOT = 7777
```

```
                                        ;;*****************************************************************
                                        ;*TEST 11        TEST THAT DAC #1 RECISTER CAN HOLD A FLOATING 1 PATTERN
                                        ;;*****************************************************************
003010  000004                  TST11:  SCOPE
003012  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
003020  012737  004000  001124          MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
003026  013777  001124  176370  1$:     MOV     $GDDAT,@DAC1            ;LOAD THE REGISTER
003034  017737  176364  001126          MOV     @DAC1,$BDDAT            ;READ THE REGISTER
003042  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE THE DATA
003050  001401                          BEQ     2$              ;;BR IF DATA IS SAME
003052  104003                          ERROR   3               ;ERROR, DAC #1 REGISTER FAILED TO HOLD A FLOATIN
003054  006237  001124          2$:     ASR     $GDDAT                  ;CHANGE THE DATA
003060  001362                          BNE     1$              ;BR AND TEST MORE DATA
                                        ;;*****************************************************************
                                        ;*TEST 12        TEST THAT DAC1 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
                                        ;;*****************************************************************
003062  000004                  TST12:  SCOPE
003064  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
003072  012737  004000  001124          MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
003100  013777  001124  176316          MOV     $GDDAT,@DAC1            ;LOAD DAC1
003106  017737  176312  001126  1$:     MOV     @DAC1,$BDDAT            ;READ THE REGISTER
003114  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE THE GOOD TO DAC1
003122  001407                          BEQ     2$              ;;BR IF THE SAME
003124  017737  176274  001126          MOV     @DAC1,$BDDAT            ;SAVE FOR TYPEOUT
003132  104003                          ERROR   3               ;DAC1 FAILED TO HOLD A FLOATING 1 PATTERN
003134  013777  001124  176262          MOV     $GDDAT,@DAC1            ;LOAD DAC1 AGAIN
003142  006277  176256          2$:     ASR     @DAC1                   ;CHANGE THE DATA
003146  006237  001124                  ASR     $GDDAT                  ;CHANGE THE EXPECTED
003152  001355                          BNE     1$              ;BR IF MORE DATA
                                        ;;*****************************************************************
                                        ;*TEST 13        TEST THE "SUB" INSTRUCTION WORKS ON DAC1
                                        ;;*****************************************************************
003154  000004                  TST13:  SCOPE
003156  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003164  012737  007777  001124          MOV     #7777,$GDDAT            ;LOAD EXPECTED
003172  013777  001124  176224          MOV     $GDDAT,@DAC1            ;LOAD DAC1
003200  162737  000001  001124  1$:     SUB     #1,$GDDAT               ;SUB A VALUE
003206  162777  000001  176210          SUB     #1,@DAC1                ;FROM EXPECTED AND DAC1
003214  017737  176204  001126          MOV     @DAC1,$BDDAT            ;READ THE REGISTER
003222  023737  001124  001126          CMP     $GDDAT,$BDDATI          ;COMPARE
003230  001404                          BEQ     2$              ;;BR IF SAME
003232  104003                          ERROR   3               ;THE SUB INSTRUCTION FAILED ON DAC1
003234  013777  001124  176162          MOV     $GDDAT,@DAC1            ;LOAD THE REGISTER AGAIN
003242  005737  001124          2$:     TST     $GDDAT                  ;TEST FOR MORE DATA
003246  001354                          BNE     1$              ;;BR IF MORE DATA
```

```
                                        ;;*********************************************************
                                        ;*TEST 14       TEST THAT THE DAC #2 REGISTER CAN BE CLEARED
                                        ;;*********************************************************
003250 000004                           TST14:  SCOPE
003252 005037 001124                            CLR     $GDDAT                  ;LOAD EXPECTED
003256 013777 001124 176142                     MOV     $GDDAT,@DAC2            ;LOAD REG
003264 017737 176136 001126                     MOV     @DAC2,$BDDAT            ;READ REG
003272 023737 001124 001126                     CMP     $GDDAT,$BDDAT           ;COMPARE
003300 001401                                   BEQ     TST15           ;;BR IF EQUAL
003302 104004                                   ERROR   4                       ;ERROR,  DAC #2 REGISTER NOT = 0

                                        ;;*********************************************************
                                        ;*TEST 15       TEST THAT THE DAC #2 REGISTER CAN BE LOADED WITH #7777
                                        ;;*********************************************************
003304 000004                           TST15:  SCOPE
003306 012737 007777 001124                     MOV     #7777,$GDDAT            ;LOAD EXPECTED
003314 013777 001124 176104                     MOV     $GDDAT,@DAC2            ;LOAD REG
003322 017737 176100 001126                     MOV     @DAC2,$BDDAT            ;READ REG
003330 023737 001124 001126                     CMP     $GDDAT,$BDDAT           ;COMPARE
003336 001401                                   BEQ     TST16           ;;BR IF EQUAL
003340 104004                                   ERROR   4                       ;ERROR,  DAC #2 REGISTER NOT = 7777

                                        ;;*********************************************************
                                        ;*TEST 16       TEST THAT THE DAC #2 REGISTER CAN HOLD A FLOATING 1 PATTERN
                                        ;;*********************************************************
003342 000004                           TST16:  SCOPE
003344 012737 000100 001160                     MOV     #100,$TIMES     ;;DO 100 ITERATIONS
003352 012737 004000 001124                     MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
003360 013777 001124 176040 1$:         MOV     $GDDAT,@DAC2            ;LOAD DAC2 REGISTER
003366 017737 176034 001126                     MOV     @DAC2,$BDDAT            ;READ THE REGISTER
003374 023737 001124 001126                     CMP     $GDDAT,$BDDAT           ;COMPARE THE DATA
003402 001401                                   BEQ     2$              ;;BR IF SAME
003404 104004                                   ERROR   4                       ;ERROR, DAC #2 REGISTER FAILED TO HOLD A FLOATIN
003406 006237 001124                    2$      ASR     $GDDAT                  ;CHANGE THE DATA
003412 001362                                   BNE     1$                      ;BR AND TEST MORE DATA

                                        ;;*********************************************************
                                        ;*TEST 17       TEST THAT DAC2 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
                                        ;;*********************************************************
003414 000004                           TST17:  SCOPE
003416 012737 000100 001160                     MOV     #100,$TIMES     ;;DO 100 ITERATIONS
003424 012737 004000 001124                     MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
003432 013777 001124 175766                     MOV     $GDDAT,@DAC2            ;LOAD DAC2
003440 017737 175762 001126 1$:         MOV     @DAC2,$BDDAT            ;READ THE REGISTER
003446 023737 001124 001126                     CMP     $GDDAT,$BDDAT           ;COMPARE THE GOOD TO DAC2
003454 001407                                   BEQ     2$              ;;BR IF THE SAME
003456 017737 175744 001126                     MOV     @DAC2,$BDDAT            ;SAVE FOR TYPEOUT
003464 104004                                   ERROR   4                       ;DAC2 FAILED TO HOLD A FLOATING 1 PATTERN
003466 013777 001124 175732                     MOV     $GDDAT,@DAC2            ;LOAD DAC2 AGAIN
003474 006277 175726                    2$:     ASR     @DAC2                   ;CHANGE THE DATA
003500 006237 001124                            ASR     $GDDAT                  ;CHANGE THE EXPECTED
003504 001355                                   BNE     1$                      ;BR IF MORE DATA
```

```
                                        ;;*****************************************************************
                                        ;*TEST 20        TEST THE "SUB" INSTRUCTION WORKS ON DAC2
                                        ;;*****************************************************************
003506  000004                  TST20:  SCOPE
003510  012737  000010  001160          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003516  012737  007777  001124          MOV     #7777,$GDDAT            ;LOAD EXPECTED
003524  013777  001124  175674          MOV     $GDDAT,@DAC2            ;LOAD DAC2
003532  162737  000001  001124  1$:     SUB     #1,$GDDAT               ;SUB A VALUE
003540  162777  000001  175650          SUB     #1,@DAC2                ;FROM EXPECTED AND DAC2
003546  017737  175654  001126          MOV     @DAC2,$BDDAT            ;READ THE REGISTER
003554  023737  001124  001126          CMP     $GDDAT,$BDDATI          ;COMPARE
003562  001404                          BEQ     2$              ;;BR IF SAME
003564  104004                          ERROR   4               ;THE SUB INSTRUCTION FAILED ON DAC2
003566  013777  001124  175632          MOV     $GDDAT,@DAC2            ;LOAD THE REGISTER AGAIN
003574  005737  001124          2$:     TST     $GDDAT                  ;TEST FOR MORE DATA
003600  001354                          BNE     1$              ;;BR IF MORE DATA

                                        ;;*****************************************************************
                                        ;*TEST 21        TEST THAT THE DAC #3 REGISTER CAN BE CLEARED
                                        ;;*****************************************************************
003602  000004                  TST21:  SCOPE
003604  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED
003610  013777  001124  175612          MOV     $GDDAT,@DAC3            ;LOAD REG
003616  017737  175606  001126          MOV     @DAC3,$BDDAT            ;READ REG
003624  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003632  001401                          BEQ     TST22           ;;BR IF EQUAL
003634  104005                          ERROR   5               ;ERROR,  DAC #3 REGISTER NOT = 0

                                        ;;*****************************************************************
                                        ;*TEST 22        TEST THAT THE DAC #3 REGISTER CAN BE LOADED WITH #7777
                                        ;;*****************************************************************
003636  000004                  TST22:  SCOPE
003640  012737  007777  001124          MOV     #7777,$GDDAT            ;LOAD EXPECTED
003646  013777  001124  175554          MOV     $GDDAT,@DAC3            ;LOAD REG
003654  017737  175550  001126          MOV     @DAC3,$BDDAT            ;READ REG
003662  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003670  001401                          BEQ     TST23           ;;BR IF EQUAL
003672  104005                          ERROR   5               ;ERROR,  DAC #3 REGISTER NOT = 7777

                                        ;;*****************************************************************
                                        ;*TEST 23        TEST THAT THE DAC #3 REGISTER CAN HOLD A FLOATING 1 PATTERN
                                        ;;*****************************************************************
003674  000004                  TST23:  SCOPE
003676  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
003704  012737  004000  001124          MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
003712  013777  001124  175510  1$:     MOV     $GDDAT,@DAC3            ;LOAD DAC #3 REGISTER
003720  017737  175504  001126          MOV     @DAC3,$BDDAT            ;READ THE REGISTER
003726  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE THE DATA
003734  001401                          BEQ     2$              ;;BR IF SAME
003736  104005                          ERROR   5               ;ERROR, DAC #3 REGISTER FAILED TO HOLD A FLOATIN
003740  006237  001124          2$:     ASR     $GDDAT                  ;CHANGE THE DATA
003744  001362                          BNE     1$              ;BR AND TEST MORE DATA
```

```
1047                                    ;;*****************************************************************
1048                                    ;*TEST 24       TEST THAT DAC3 CAN HOLD A FLOATING 1 PATTERN (DYNAMICLY)
1049                                    ;;*****************************************************************
1050  003746  000004                    TST24:  SCOPE
1051  003750  012737  000100  001160            MOV     #100,$TIMES     ;;DO 100 ITERATIONS
1052  003756  012737  004000  001124            MOV     #BIT11,$GDDAT           ;LOAD EXPECTED
1053  003764  013777  001124  175436            MOV     $GDDAT,@DAC3            ;LOAD DAC3
1054  003772  017737  175432  001126  1$:       MOV     @DAC3,$BDDAT            ;READ THE REGISTER
1055  004000  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE THE GOOD TO DAC3
1056  004006  001407                            BEQ     2$              ;;BR IF THE SAME
1057  004010  017737  175414  001126            MOV     @DAC3,$BDDAT            ;SAVE FOR TYPEOUT
1058  004016  104005                            ERROR   5               ;DAC3 FAILED TO HOLD A FLOATING 1 PATTERN
1059  004020  013777  001124  175402            MOV     $GDDAT,@DAC3            ;LOAD DAC3 AGAIN
1060  004026  006277  175376            2$:     ASR     @DAC3                   ;CHANGE THE DATA
1061  004032  006237  001124                    ASR     $GDDAT                  ;CHANGE THE EXPECTED
1062  004036  001355                            BNE     1$                      ;BR IF MORE DATA
1063                                    ;;*****************************************************************
1064                                    ;*TEST 25       TEST THE "SUB" INSTRUCTION WORKS ON DAC3
1065                                    ;;*****************************************************************
1066  004040  000004                    TST25:  SCOPE
1067  004042  012737  000010  001160            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1068  004050  012737  007777  001124            MOV     #7777,$GDDAT            ;LOAD EXPECTED
1069  004056  013777  001124  175344            MOV     $GDDAT,@DAC3            ;LOAD DAC3
1070  004064  162737  000001  001124  1$:       SUB     #1,$GDDAT               ;SUB A VALUE
1071  004072  162777  000001  175330            SUB     #1,@DAC3                ;FROM EXPECTED AND DAC3
1072  004100  017737  175324  001126            MOV     @DAC3,$BDDAT            ;READ THE REGISTER
1073  004106  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1074  004114  001404                            BEQ     2$              ;;BR IF SAME
1075  004116  104005                            ERROR   5               ;THE SUB INSTRUCTION FAILED ON DAC3
1076  004120  013777  001124  175302            MOV     $GDDAT,@DAC3            ;LOAD THE REGISTER AGAIN
1077  004126  005737  001124            2$:     TST     $GDDAT                  ;TEST FOR MORE DATA
1078  004132  001354                            BNE     1$              ;;BR IF MORE DATA
```

```
1079                                    ;;*****************************************************************
1080                                    ;*TEST 26       TEST THAT THE FOUR DAC REGISTERS CAN HOLD DIFFERENT DATA
1081                                    ;;*****************************************************************
1082  004134  000004                    ↑ST26:  SCOPE
1083  004136  012777  001111  175256            MOV     #1111,@DACO             ;LOAD DAC #0
1084  004144  012777  002222  175252            MOV     #2222,@DAC1             ;LOAD DAC #1
1085  004152  012777  004444  175246            MOV     #4444,@DAC2             ;LOAD DAC #2
1086  004160  012777  007777  175242            MOV     #7777,@DAC3             ;LOAD DAC #3
1087  004166  012737  001111  001124            MOV     #1111,$GDDAT            ;LOAD EXPECTED
1088  004174  017737  175222  001126            MOV     @DACO,$BDDAT            ;READ REG
1089  004202  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1090  004210  001401                            BEQ     1$              ;;BR IF EQUAL
1091  004212  104002                            ERROR   2                       ;ERROR, SELECTED DAC #0 IN ERROR
1092
1093  004214  012737  002222  001124  1$:       MOV     #2222,$GDDAT            ;LOAD EXPECTED
1094  004222  017737  175176  001126            MOV     @DAC1,$BDDAT            ;READ REG
1095  004230  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1096  004236  001401                            BEQ     2$              ;;BR IF EQUAL
1097  004240  104003                            ERROR   3                       ;ERROR, SELECTED DAC #1 IN ERROR
1098
1099  004242  012737  004444  001124  2$:       MOV     #4444,$GDDAT            ;LOAD EXPECTED
1100  004250  017737  175152  001126            MOV     @DAC2,$BDDAT            ;READ REG
1101  004256  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1102  004264  001401                            BEQ     3$              ;;BR IF SAME
1103  004266  104004                            ERROR   4                       ;ERROR, SELECTED DAC #2 IN ERROR
1104
1105  004270  012737  007777  001124  3$:       MOV     #7777,$GDDAT            ;LOAD EXPECTED
1106  004276  017737  175126  001126            MOV     @DAC3,$BDDAT            ;READ REG
1107  004304  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1108  004312  001401                            BEQ     ↑ST27           ;;BR IF SAME
1109  004314  104005                            ERROR   5                       ;ERROR, SELECTED DAC #3 IN ERROR
```

```
1110                                    ;;*****************************************************************
1111                                    ;*TEST 27        TEST THAT RESET CLEARS DAC #0 REGISTER
1112                                    ;;*****************************************************************
1113  004316  000004                    TST27:  SCOPE
1114  004320  012737  000010  001160            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1115  004326  012777  177777  175066            MOV     #-1,@DAC0
1116  004334  005037  001124                    CLR     $GDDAT                  ;LOAD EXPECTED
1117  004340  000005                            RESET
1118  004342  017737  175054  001126            MOV     @DAC0,$BDDAT            ;READ REG
1119  004350  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1120  004356  001401                            BEQ     TST30           ;;BR IF EQUAL
1121  004360  104002                            ERROR   2                       ;ERROR, RESET FAILED TO CLEAR DAC #0
1122
1123                                    ;;*****************************************************************
1124                                    ;*TEST 30        TEST THAT RESET CLEARS DAC #1 REGISTER
1125                                    ;;*****************************************************************
1126  004362  000004                    TST30:  SCOPE
1127  004364  012737  000010  001160            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1128  004372  012777  177777  175024            MOV     #-1,@DAC1
1129  004400  005037  001124                    CLR     $GDDAT                  ;LOAD EXPECTED
1130  004404  000005                            RESET
1131  004406  017737  175012  001126            MOV     @DAC1,$BDDAT            ;READ REG
1132  004414  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1133  004422  001401                            BEQ     TST31           ;;BR IF EQUAL
1134  004424  104003                            ERROR   3                       ;ERROR, RESET FAILED TO CLEAR DAC #1
1135
1136
1137                                    ;;*****************************************************************
1138                                    ;*TEST 31        TEST THAT RESET CLEARS DAC #2 REGISTER
1139                                    ;;*****************************************************************
1140  004426  000004                    TST31:  SCOPE
1141  004430  012737  000010  001160            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1142  004436  012777  177777  174762            MOV     #-1,@DAC2               ;LOAD THE REGISTER
1143  004444  005037  001124                    CLR     $GDDAT                  ;CLEAR EXPECTED
1144  004450  000005                            RESET
1145  004452  017737  174750  001126            MOV     @DAC2,$BDDAT            ;READ THE REGISTER
1146  004460  001401                            BEQ     TST32           ;;BR IF CLEARED
1147  004462  104004                            ERROR   4                       ;ERROR, RESET FAILED TO CLEAR DAC #2
1148
1149                                    ;;*****************************************************************
1150                                    ;*TEST 32        TEST THAT RESET CLEARS DAC #3 REGISTER
1151                                    ;;*****************************************************************
1152  004464  000004                    TST32:  SCOPE
1153  004466  012737  000010  001160            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1154  004474  012777  177777  174726            MOV     #-1,@DAC3               ;LOAD THE REGISTER
1155  004502  005037  001124                    CLR     $GDDAT                  ;CLEAR THE EXPECTED
1156  004506  012737  000001  007012            MOV     #1,TEMP
1157  004514  000005                            RESET
1158  004516  017737  174706  001126            MOV     @DAC3,$BDDAT            ;READ THE REGISTER
1159  004524  001401                            BEQ     TST33           ;;BR IF CLEARED
1160  004526  104005                            ERROR   5                       ;ERROR, RESET FAILED TO CLEAR DAC #3
1161
```

```
                                REMAIN:
                                ;;*************************************************
                                ;*TEST 33       DETERMINE IF MORE AAV11'S REMAIN TO BE TESTED
                                ;;*************************************************
004530  000004                  TST33:  SCOPE
004532  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
004540  005237  001206                  INC     $UNIT                   ;UPDATE UNIT #
004544  123737  001206  001420          CMPB    $UNIT,EVEN              ;TEST IF MORE
004552  001424                          BEQ     TS~34           ;;BR IF NOT
004554  005237  001204                  INC     $DEVCT                  ;APT UNIT #
004560  063737  001416  001422          ADD     VADDR,DAC0              ;UPDATE BUS ADDRESS
004566  063737  001416  001424          ADD     VADDR,DAC1
004574  063737  001416  001426          ADD     VADDR,DAC2
004602  063737  001416  001430          ADD     VADDR,DAC3
004610  006337  007010                  ASL     MASKNM                  ;CHANGE THE ERROR FLAG BIT
004614  005037  001102                  CLR     $TSTNM
004620  000137  002306                  JMP     TST1                    ;TEST THE NEXT UNIT

                                ;;*************************************************
                                ;*TEST 34       DETERMINE IF RUNNING ON THE HARDWARE TESTER     (IF NOT REPORT END OF PA
                                ;;*************************************************
                                TST34:  SCOPE
004626  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
004634  005737  007320                  TST     @#7320                  ;TEST IF ON TESTER
004640  001002                          BNE     TST3[illegible]         ;;BR TO TEST
004642  010137  005454                  JMP     @#5[illegible]
```

```
                                        ;;*****************************************************
                                        ;*TEST 35       TEST THAT DAC #3 OUTPUT BITS (0-3) FUNCTION
                                        ;;*****************************************************
004646  000004                  T$T35:  SCOPE
004650  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
004656  012737  000010  007014          MOV     #BIT3,$TEMP             ;LOAD DAC PATTERN
004664  012737  004000  001124          MOV     #BIT11,$GDDAT           ;LOAD EXPECTED PATTERN

004672  013777  004014  174530  13:     MOV     $TEMP,@DAC3             ;LOAD DAC REGISTER
004700  017737  002120  001126          MOV     @DRIN,$BDDAT            ;READ THE REGISTER
004706  042737  170377  001126          BIC     #170377,$BDDAT          ;MASK OFF OTHER BITS
004714  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004722  001401                          BEQ     2$              ;;BR IF THE SAME
004724  104013                          ERROR   13              ;DAC #3 DIGITAL OUTPUT BITS IN ERROR

004726  006237  001124          2$:     ASR     $GDDAT                  ;ADJUST EXPECTED
004732  006237  007014                  ASR     $TEMP                   ;ADJUST LOADED PATTERN
004736  001355                          BNE     .$

                                        ;;*****************************************************
                                        ;*TEST 36       VERIFY THE AAV11 +15 SUPPLY
                                        ;;*****************************************************
004740  000004                  T$T36:  SCOPE
004742  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
004750  013737  007032  001124          MOV     V5744,$GDDAT            ;LOAD EXPECTED
004756  004537  006302                  JSR     R5,CONVRT               ;SAMPLE THE CHANNEL
004762  000012                          12
004764  013737  007034  007016          MOV     V144,SPREAD             ;LOAD TOLERANCE
004772  004737  006520                  JSR     PC,COMPAR               ;TEST IT
004776  000401                          BR      T$T37           ;;BR
005000  104011                          ERROR   11              ;+15 VOLT SUPPLY IS WRONG

                                        ;;*****************************************************
                                        ;*TEST 37       VERIFY THE AAV11 -15 SUPPLY
                                        ;;*****************************************************
005002  000004                  T$T37:  SCOPE
005004  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
005012  013737  007036  001124          MOV     V2034,$GDDAT            ;LOAD EXPECTED
005020  004537  006302                  JSR     R5,CONVRT               ;SAMPLE THE CHANNEL
005024  000011                          11
005026  013737  007034  007016          MOV     V144,SPREAD             ;LOAD TOLERANCE
005034  004737  006520                  JSR     PC,COMPAR               ;TEST IT
005040  000401                          BR      T$T40           ;;BR
005042  104012                          ERROR   12              ;-15 VOLT SUPPLY IS WRONG
```

```
1233                                    ;;**********************************************************************
1234                                    ;*TEST 40        DAC0 OFFSET ADJUSTMENT
1235                                    ;**********************************************************************
1236  005044  000004                    1$T40:  SCOPE
1237  005046  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
1238  005054  005737  001202                    TST     $PASS                   ;TEST IF FIRST PASS
1239  005060  001006                            BNE     1$T41           ;;BR IF NOT
1240
1241  005062  004537  005634                    JSR     R5,OFFDAC               ;LOAD AND EXECUTE DAC OFFSET ADJ.
1242  005066  001422                            DAC0                    ;DAC ADDRESS
1243  005070  010726                            SELDO                   ;TYPEOUT ADDRESS
1244  005072  010552                            ADJR46                  ;RES. TO ADJUST
1245  005074  000013                            13                      ;RESULT CHANNEL #
1246
1247                                    ;;**********************************************************************
1248                                    ;*TEST 41        DAC0 GAIN ADJUSTMENT
1249                                    ;**********************************************************************
1250  005076  000004                    1$T41:  SCOPE
1251  005100  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
1252  005106  005737  001202                    TST     $PASS                   ;TEST IF FIRST PASS
1253  005112  001005                            BNE     1$T42           ;;BR IF NOT
1254
1255  005114  004537  005764                    JSR     R5,GAIDAC               ;LOAD AND EXECUTE DAC GAIN ADJ.
1256  005120  001422                            DAC0                    ;DAC ADDRESS
1257  005122  010376                            ADJR34                  ;RES. TO ADJUST
1258  005124  000013                            13                      ;CHANNEL 1 FOR RESULTS
1259
1260                                    ;;**********************************************************************
1261                                    ;*TEST 42        DAC0 CALIBRATION
1262                                    ;**********************************************************************
1263  005126  000004                    1$T42:  SCOPE
1264  005130  012737  000001  001160            MOV     #1,$TIMES       ;,DO 1 ITERATION
1265  005136  004537  006174                    JSR     R5,CALDAC               ;LOAD AND EXECUTE CALIBRATION
1266  005142  001422                            DAC0                    ;DAC ADDRESS
1267  005144  000013                            13                      ;CHANNEL # FOR RESULTS
```

```
1268                                    ;;**********************************************************
1269                                    ;*TEST 43        DAC1 OFFSET ADJUSTMENT
1270                                    ;;**********************************************************
1271  005146  000004                    TST43:  SCOPE
1272  005150  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
1273  005156  005737  001202                    TST     $PASS                   ;TEST IF FIRST PASS
1274  005162  001006                    BNE     TST44           ;;BR IF NOT
1275
1276  005164  004537  005634            JSR     R5,OFFDAC               ;LOAD AND EXECUTE DAC OFFSET ADJ.
1277  005170  001424                    DAC1                            ;DAC ADDRESS
1278  005172  010744                    SELD1                           ;TYPEOUT ADDRESS
1279  005174  010605                    ADJR47                          ;RES. TO ADJUST
1280  005176  000014                    14                              ;RESULT CHANNEL #
1281
1282                                    ;;**********************************************************
1283                                    ;*TEST 44        DAC1 GAIN ADJUSTMENT
1284                                    ;;**********************************************************
1285  005200  000004                    TST44:  SCOPE
1286  005202  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
1287  005210  005737  001202                    TST     $PASS                   ;TEST IF FIRST PASS
1288  005214  001005                    BNE     TST45           ;;BR IF NOT
1289
1290  005216  004537  005764            JSR     R5,GAIDAC               ;LOAD AND EXECUTE DAC GAIN ADJ.
1291  005222  001424                    DAC                             ;DAC ADDRESS
1292  005224  010431                    ADJ 435                         ;RES. TO ADJUST
1293  005226  000014                    14                              ;CHANNEL # FOR RESULTS
1294
1295                                    ;;**********************************************************
1296                                    ;*TEST 45        DAC1 CALIBRATION
1297                                    ;;**********************************************************
1298  005230  000004                    TST45:  SCOPE
1299  005232  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
1300  005240  004537  006104            JSR     R5,CALDAC               ;LOAD AND EXECUTE CALIBRATION
1301  005244  001424                    DAC1                            ;DAC ADDRESS
1302  005246  000014                    14                              ;CHANNEL # FOR RESULTS
```

```
                                        ;;*****************************************************************
                                        ;*TEST 46        DAC2 OFFSET ADJUSTMENT
                                        ;;*****************************************************************
005250  000004                  ;ST46:  SCOPE
005252  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
005260  005737  001202                  TST     $PASS                   ;TEST IF FIRST PASS
005264  001006                  ;       BNE     ST47            ;;BR IF NOT
005266  004537  005634                  JSR     R5,OFFDAC               ;LOAD AND EXECUTE DAC OFFSET ADJ.
005272  001426                  ;       DAC2                            ;DAC ADDRESS
005274  010762                  ;       SELD2                           ;TYPEOUT ADDRESS
005276  010640                  ;       ADJR48                          ;RES. TO ADJUST
005300  000016                  ;       16                              ;RESULT CHANNEL #
                                        ;;*****************************************************************
                                        ;*TEST 47        DAC2 GAIN ADJUSTMENT
                                        ;;*****************************************************************
005302  000004                  ;ST47:  SCOPE
005304  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
005312  005737  001202                  TST     $PASS                   ;TEST IF FIRST PASS
005316  001005                  ;       BNE     ST50            ;;BR IF NOT
005320  004537  005764                  JSR     R5,GAIDAC               ;LOAD AND EXECUTE DAC GAIN ADJ.
005324  001426                  ;       DAC2                            ;DAC ADDRESS
005326  010464                  ;       ADJR36                          ;RES. TO ADJUST
005330  000016                  ;       16                              ;CHANNEL # FOR RESULTS
                                        ;;*****************************************************************
                                        ;*TEST 50        DAC2 CALIBRATION
                                        ;;*****************************************************************
005332  000004                  ;ST50:  SCOPE
005334  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
005342  004537  006104                  JSR     R5,CALDAC               ;LOAD AND EXECUTE CALIBRATION
005346  001426                  ;       DAC2                            ;DAC ADDRESS
005350  000016                  ;       16                              ;CHANNEL # FOR RESULTS
```

```
1338                                    ;;*************************************************************
1339                                    ;*TEST 51        DAC3 OFFSET ADJUSTMENT
1340                                    ;;*************************************************************
1341  005352  000004                    TST51:  SCOPE
1342  005354  012737  000001  001160            MOV     #1,$TIMES       ;;DO . ITERATION
1343  005362  005737  001202                    TST     $PASS                   ;TEST IF FIRST PASS
1344  005366  001006                    BNE     TST52           ;;BR IF NOT
1345
1346  005370  004537  005634                    JSR     R5,OFFDAC               ;LOAD AND EXECUTE DAC OFFSET ADJ.
1347  005374  001430                    DAC3                            ;DAC ADDRESS
1348  005376  011000                    SEL03                           ;TYPEOUT ADDRESS
1349  005400  010673                    ADJR49                          ;RES. TO ADJUST
1350  005402  000015                    15                              ;RESULT CHANNEL #
1351
1352                                    ;;*************************************************************
1353                                    ;*TEST 52        DAC3 GAIN ADJUSTMENT
1354                                    ;;*************************************************************
1355  005404  000004                    TST52:  SCOPE
1356  005406  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
1357  005414  005737  001202                    TST     $PASS                   ;TEST IF FIRST PASS
1358  005420  001005                    BNE     TST53           ;;BR IF NOT
1359
1360  005422  004537  005764                    JSR     R5,GAIDAC               ;LOAD AND EXECUTE DAC GAIN ADJ.
1361  005426  001430                    DAC3                            ;DAC ADDRESS
1362  005430  010517                    ADJR37                          ;RES. TO ADJUST
1363  005432  000015                    15                              ;CHANNEL # FOR RESULTS
1364
1365                                    ;;*************************************************************
1366                                    ;*TEST 53        DAC3 CALIBRATION
1367                                    ;;*************************************************************
1368  005434  000004                    TST53:  SCOPE
1369  005436  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
1370  005444  004537  006104                    JSR     R5,CALDAC               ;LOAD AND EXECUTE CALIBRATION
1371  005450  001430                    DAC3                            ;DAC ADDRESS
1372  005452  000015                    15                              ;CHANNEL # FOR RESULTS
```

```
                                        .SBTTL  END OF PASS ROUTINE
                                        ;;**********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERE'S A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO INIT7

005454                                  $EOP:
005454  000004                                SCOPE
005456  005037  001102                       CLR     $ISTNM          ;;ZERO THE TEST NUMBER
005462  005037  001160                       CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
005466  005237  001202                       INC     $PASS           ;;INCREMENT THE PASS NUMBER
005472  042737  100000  001202               BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
005500  005367  000000                       DEC     (PC)+           ;;LOOP?
005502  000001                          $EOPCT: .WORD   1
005504  003022                               BGT     $DOAGN          ;;YES
005506  012737                               MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
005510  000001                          $ENDCT: .WORD   1
005512  005502                               $EOPCT
005514  104401  005561                       TYPE    ,$ENDMG         ;;TYPE "END PASS #"
005520  013746  001202                       MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
005524  104405                               TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
005526  104401  005556                       TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
005532  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
005536  001405                               BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
005540  000005                               RESET                   ;;CLEAR THE WORLD
005542  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
005544  000240                               NOP                     ;;SAVE ROOM
005546  000240                               NOP                     ;;FOR
005550  000240                               NOP                     ;;ACT11
005552                                  $DOAGN:
005552  000137                               JMP     @(PC)+          ;;RETURN
005554  005576                          $RTNAD: .WORD   INIT7
005556     377     377     000          $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
005561     015  042412  042116          $ENDMG: .ASCIZ  <15><12>/END PASS #/
005566  050040  051501  020123
005574  000043
1411
005576  005737  007020                  INIT7:  TST     @#TST           ;TEST IF ON TESTER
005602  001012                               BNE     1$
005604  104401  010176                       TYPE,   ERRTOT
005610  013746  001112                       MOV     $ERTTL,-(SP)
005614  104405                               TYPDS
005616  104401  010210                       TYPE,   MESGD
005622  013746  007004                       MOV     BADUNT,-(SP)    ;;SAVE BADUNT FOR TYPEOUT
005626  104406                               TYPBN                   ;;GO TYPE--BINARY ASCII
005630  000137  002044                  1$:     JMP     INIT1           ;TEST IT AGAIN
```

```
                                        .SBTTL  SUBROUTINE TO ADJUST THE DAC'S OFFSET POTS
                                
005634 012537 005762            OFFDAC: MOV     (R5)+,10$               ;GET BUS ADDRESS
005640 017737 000116 005762             MOV     @10$,10$                ;
005646 013737 005762 007002             MOV     10$,DACBAD              ;LOAD BUS ADDRES IF ERROR
005654 012537 005672                    MOV     (R5)+,11$               ;GET POINTER TO ASCII MESSAGE
005660 012537 005712                    MOV     (R5)+,12$               ;GET POINTER TO RES. MESSAGE
005664 012537 005732                    MOV     (R5)+,13$               ;GET AND SAVE CHANNEL #
005670 104401                           TYPE                            ;TELL OPERATOR TO SELECT DAC N
005672 010726                    1$:    SELDO
005674 004537 006212                    JSR     R5,SNDVLT               ;LOAD A VOLTAGE OF M5.1200 VOLTS.
005700 011064                           M51200
005702 012777 000000 000052             MOV     #0000,@10$              ;LOAD THE SELECTED DAC TO NULL
005710 104401                           TYPE                            ;TELL OPERATOR TO ADJUST RXX FOR NULL
005712 010552                    12$:   ADJR46
005714 004737 006422             1$:    JSR     PC,CSPACE               ;WAIT UNTIL THE IS READY
005720 012737 000000 001124             MOV     #0000,$GDDAT            ;LOAD EXPECTED VALUE
005726 004537 006302                    JSR     R5,CONVRT               ;SAMPLE THE CHANNEL
005732 000013                    13$:   13
005734 012737 000002 007016             MOV     #2,SPREAD
005742 004737 006520                    JSR     PC,COMPAR               ;TEST RESULTS
005746 000404                           BR      2$              ;;BR IF WITHIN THE LIMIT
005750 104006                           ERROR   6       ;SELECTED DAC OFFSET POT WAS NOT ADJUSTED INCORRECTLY
005752 104401                           TYPE
005754 011016                           TRYAGN
005756 000756                           BR      1$                      ;LOOP AGAIN
005760 000205                    2$:    RTS     R5                      ;EXIT
005762 000000                    10$:   0

                                        .SBTTL  SUBROUTINE TO ADJUST THE GAIN ADJUSTMENT POTS

005764 012537 006102            GAIDAC: MOV     (R5)+,10$               ;GET BUS ADDRESS
005770 017737 000106 006102             MOV     @10$,10$                ;
005776 013737 006102 007002             MOV     10$,DACBAD              ;LOAD BUS ADDRESS IF ERROR
006004 012537 006032                    MOV     (R5)+,11$               ;GET ASCII RES. ADDRESS
006010 012537 006052                    MOV     (R5)+,12$               ;GET CHANNEL #
006014 004537 006212                    JSR     R5,SNDVLT               ;LOAD + 5.1175 VOLTS
006020 011077                           PS1175
006022 012777 007777 000052             MOV     #7777,@10$              ;LOAD THE DAC
006030 104401                           TYPE                            ;TELL OPERATOR WHICH RXX TO ADJUST FOR NULL
006032 010376                    11$:   ADJR34
006034 004737 006422             1$:    JSR     PC,CSPACE               ;WAIT FOR OPERATOR
006040 012737 007777 001124             MOV     #7777,$GDDAT            ;LOAD EXPECTED
006046 004537 006302                    JSR     R5,CONVRT               ;CONVERT THE VALUE
006052 000013                    12$:   13
006054 012737 000002 007016             MOV     #2,SPREAD               ;LOAD LIMIT
006062 004737 006520                    JSR     PC,COMPAR               ;TEST RESULTS
006066 000404                           BR      2$              ;;BR IF WITHIN LIMITS
006070 104007                           ERROR   7       ;SELECTED DAC GAIN POT WAS NOT ADJUSTED PROPERLY
006072 104401                           TYPE
006074 011016                           TRYAGN
006076 000756                           BR      1$
006100 000205                    2$:    RTS     R5                      ;EXIT
006102 000000                    10$:   0
```

```
                                        .SBTTL  SUBROUTINE TO TEST THE D/A CALIBRATION
                                        CALDAC: MOV     (R5)+,10$               ;GET BUS ADDRESS
006110 012737 000074 006210             MOV     @10$,10$                ;
006116 013737 006210 007002             MOV     10$,DACBAD              ;LOAD BUS ADDRESS IF ERROR
006124 012537 006150                    MOV     (R5)+,11$               ;GET CHANNEL #
006130 012777 007400 000052             MOV     #7400,@10$              ;LOAD THE DAC
006136 012737 007400 001124             MOV     #7400,$GDDAT            ;LOAD THE EXPECTED VALUE
006144 004537 006302                    1$:     JSR     R5,CONVRT               ;SAMPLE THE CHANNEL
006150 000013                           11$:    13
006152 012737 000003 007016             MOV     #3,SPREAD               ;LOAD TOLERANCE
006160 004737 006520                    JSR     PC,COMPAR               ;TEST THE RESULTS
006164 000401                           BR      2$              ;;BR
006166 104010                           ERROR   10              ;NON-LINEARITY IN DAC DETECTED
006170 162777 000400 000012             2$:     SUB     #400,@10$               ;ADJUST THE CONTENTS
006176 162737 000400 001124             SUB     #400,$GDDAT             ;ADJUST THE EXPECTED
006204 001357                           BNE     1$              ;;BR IF NOT DONE
006206 000205                           RTS     R5              ;EXIT
006210 000000                           10$:    0
                                        .SBTTL  SUBROUTINE TO LOAD A VOLTAGE INTO THE VOLTAGE SOURCE
006212 012500                           SNDVLT: MOV     (R5)+,R0                ;LOAD THE POINTER
006214 112001                           2$:     MOVB    (R0)+,R1                ;GET SOME DATA
006216 001421                           BEQ     3$              ;BR IF TERM
006220 110177 000576                    MOVB    R1,@FILZ                ;LOAD THE DATA
006224 012701 001000                    MOV     #1000,R1
006230 005301                           5$:     DEC     R1                      ;DELAY
006232 001376                           BNE     5$
006234 052777 000200 000560             BIS     #BIT7,@FILZ             ;SET BIT 7
006242 012701 001000                    MOV     #1000,R1                ;LOAD DELAY
006246 005301                           1$:     DEC     R1                      ;DELAY
006250 001376                           BNE     1$
006252 042777 000200 000542             BIC     #BIT7,@FILZ
006260 000755                           BR      2$
006262 012701 000000                    3$:     MOV     #0,R1                   ;LOAD DELAY
006266 152777 000177 000526             BISB    #177,@FILZ              ;DISABLE BITS
006274 005301                           4$:     DEC     R1                      ;DELAY
006276 001376                           BNE     4$
006300 000205                           RTS     R5                      ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CONVERT CHANNEL N ON THE TESTER A/D

006302  012537  006414          CONVRT: MOV     (R5)+,10$               ;GET THE CHANNEL #
006306  000337  006414                  SWAB    10$
006312  042737  170377  006414          BIC     #170377,10$             ;MASK OUT OTHER BITS
006320  013777  006414  000500          MOV     10$,@ADCS               ;SELECT CHANNEL
006326  005037  006416                  CLR     11$
006332  012737  000200  006420          MOV     #BIT7,12$               ;LOAD SHIFT COUNTER
006340  105277  000462          1$:     INCB    @ADCS                   ;CONVERT CHANNEL
006344  105777  000456          2$:     TSTB    @ADCS                   ;WAIT FOR DONE
006350  100375                  BPL     2$
006352  067737  000452  006416          ADD     @ADDBR,11$              ;UPDATE CONVERSION
006360  006237  006420                  ASR     12$                     ;FINISHED ?
006364  001365                  BNE     1$
006366  000257                  CCC
006370  006037  006416                  ROR     11$
006374  006237  006416                  ASR     11$
006400  006237  006416                  ASR     11$                     ;JUSTIFY DATA
006404  013737  006416  JC1126          MOV     11$,$BDDAT              ;LOAD ACTUAL (ADJUSTED)
006412  000205                  RTS     R5                              ;EXIT

006414  000000          10$:    0
006416  000000          11$:    0
006420  000000          12$:    0

                                        .SBTTL  SUBROUTINE TO LOOP UNTIL OPERATOR TYPES AN "SPACE"

006422  104401  007746          CSPACE: TYPE,   LDSPAC                  ;TELL OPERATOR TO HIT SPACE BAR
006426  012737  000014  006516  3$:     MOV     #14,11$                 ;LOAD DELAY COUNTER
006434  005037  006514                  CLR     10$
006440  105777  172500          1$:     TSTB    @#TKS                   ;WAIT FOR OPERATOR
006444  100410                  BMI     2$                              ;BR IF FLAG IS SET
006446  005337  006514                  DEC     10$                     ;DELAY
006452  001372                  BNE     1$                              ;BR IF NOT DONE
006454  005337  006516                  DEC     11$                     ;DELAY AGAIN
006460  001367                  BNE     1$
006462  104014                  ERROR   14                              ;WAKE UP OPERATOR
006464  000760                  BR      3$                              ;LOOP
006466  017737  172454  006514  2$:     MOV     @#TKB,10$               ;READ THE CHARACTER
006474  042737  177600  006514          BIC     #177600,10$             ;MASK OF OTHER BITS
006502  022737  000040  006514          CMP     #40,10$                 ;TEST FOR "SPACE"
006510  001346                  BNE     3$                              ;LOOP
006512  000207                  RTS     PC                              ;EXIT
006514  000000          10$:    0
006516  000010          11$:    BIT3
```

```
                                        .SBTTL  SUBROUTINE TO COMPARE TWO LOCATIONS BY THE SPREAD
                                
006520  010046                  COMPAR: MOV     R0,-(SP)                ;SAVE R0
006522  010146                          MOV     R1,-(SP)                ;SAVE R1
006524  013700  001124                  MOV     $GOODAT,R0              ;GET EXPECTED VALUE
006530  013701  001126                  MOV     $BADDAT,R1              ;GET THE UNKNOWN
006534  160100                          SUB     R1,R0                   ;SUBTRACT
006536  100001                          BPL     8$
006540  005400                          NEG     R0
006542  020037  007016          8$:     CMP     R0,SPREAD               ;TEST IF DIFFERENCE IF > THAN SPREAD
006546  003405                          BLE     10$
006550  012601                  9$:     MOV     (SP)+,R1                ;RESTORE R1
006552  012600                          MOV     (SP)+,R0                ;RESTORE R0
006554  062716  000002                  ADD     #2,(SP)                 ;MAKE AN ERROR EXIT
006560  000207                          RTS     PC                      ;EXIT
                                
006562  012601                  10$:    MOV     (SP)+,R1
006564  012600                          MOV     (SP)+,R0
006566  000207                          RTS     PC                      ;EXIT FOR GOOD LIMIT TEST
                                
                                        .SBTTL  FULL SCALE RAMP ON EACH RAMP
                                
006570  012706  001100          FULRMP: MOV     #STACK,SP               ;LOAD POINTER
006574  004737  001746                  JSR     PC,LOTRAP               ;LOAD BUS ADDRESS
006600  013700  001422          1$:     MOV     DAC0,R0         ;GET BUS ADDRESS
006604  004737  006642                  JSR     PC,10$                  ;LOAD THE RAMP ON DAC #1
006610  013700  001424                  MOV     DAC1,R0         ;GET BUS ADDRESS
006614  004737  006642                  JSR     PC,10$                  ;LOAD THE RAMP ON DAC #1
006620  013700  001426                  MOV     DAC2,R0         ;GET BUS ADDRESS
006624  004737  006642                  JSR     PC,10$                  ;LOAD THE RAMP ON DAC #2
006630  013700  001430                  MOV     DAC3,R0         ;GET THE BUS ADDRESS
006634  004737  006642                  JSR     PC,10$                  ;LOAD THE RAMP ON DAC #3
006640  000757                          BR      1$                      ;BR BACK
                                
006642  005010                  10$:    CLR     (R0)                    ;CLEAR DAC
006644  062710  000010          11$:    ADD     #10,(R0)                ;UPDATE THE DATA
006650  005710                          TST     (R0)                    ;TEST IF DONE
006652  001374                          BNE     11$                     ;BR IF NOT
006654  000207                          RTS     PC                      ;EXIT
```

```
1606
1607                                            .SBTTL  STATIC DAC CALIBRATION
1608
1609  006656  012706  001100        STATIC:   MOV     #STACK,SP             ;LOAD STACK POINTER
1610  006662  004737  001746                  JSR     PC,LDTRAP             ;LOAD BUS ADDRESSES
1611  006666  104410                1$:       CKSWR                         ;TEST FOR CTRL G
1612  006670  017700  172244                  MOV     @SWR,R0               ;READ SWITCHES
1613  006674  010077  172522                  MOV     R0,@DAC0              ;LOAD DAC #0
1614  006700  010077  172520                  MOV     R0,@DAC1              ;LOAD DAC #1
1615  006704  010077  172516                  MOV     R0,@DAC2              ;LOAD DAC #2
1616  006710  010077  172514                  MOV     R0,@DAC3              ;LOAD DAC #3
1617  006714  000764                          BR      1$
1618
1619                                            .SBTTL  DYNAMIC DAC CALIBRATION
1620
1621  006716  012706  001100        DYNCAL:   MOV     #STACK,SP             ;LOAD STACK POINTER
1622  006722  004737  001746                  JSR     PC,LDTRAP             ;LOAD BUS ADDRESSES
1623  006726  104410                1$:       CKSWR                         ;TEST FOR CTRL G
1624  006730  017700  172204                  MOV     @SWR,R0               ;READ SWR
1625  006734  004737  006750                  JSR     PC,10$                ;LOAD THE SWR VALUE TO ALL DACS
1626  006740  005000                          CLR     R0                    ;CLEAR R0
1627  006742  004737  006750                  JSR     PC,10$                ;LOAD ALL DAC'S WITH 0
1628  006746  000767                          BR      1$
1629
1630  006750  010077  172446        10$:      MOV     R0,@DAC0              ;LOAD DAC #0
1631  006754  010077  172444                  MOV     R0,@DAC1              ;LOAD DAC #1
1632  006760  010077  172442                  MOV     R0,@DAC2              ;LOAD DAC #2
1633  006764  010077  172440                  MOV     R0,@DAC3              ;LOAD DAC #3
1634  006770  012700  000020                  MOV     #20,R0                ;LOAD DELAY COUNTER
1635  006774  005300                11$:      DEC     R0                    ;DELAY
1636  006776  100376                          BPL     11$                   ;WAIT
1637  007000  000207                          RTS     PC                    ;EXIT
1638
1639  007002  170440                DACBAD:   ABASE
1640  007004  000000                BADUNT:   0
1641  007006  000004                NUMBOK    4.
1642  007010  000001                MASKNM:   BIT0
1643  007012  000000                TEMP:     0
1644  007014  000000                $TEMP:    0
1645  007016  000000                SPREAD:   0
1646  007020  000000                WFTST:    0
1647  007022  167772                FILZ:     167772
1648  007024  167774                DRIN:     167774
1649  007026  170500                ADCS:     170500
1650  007030  170502                ADBR:     170502
1651  007032  005744                V5744:    5744
1652  007034  000144                V144:     144
1653  007036  002034                V2034:    2034
1654
1655
```

```
1656
1657                                                    .SBTTL  ASCII MESSAGES
1658
1659  007040  005015  040412  053101  TITLE:  .ASCIZ  <15><12><12>'AAV11 DIAGNOSTIC TEST, (MAINDEC-11-DVAAA-A0)'<15><12>
1660  007046  030461  042040  040511
1661  007054  047107  051517  044524
1662  007062  020103  042524  052123
1663  007070  020054  046450  044501
1664  007076  042116  041505  030455
1665  007104  026461  053104  040501
1666  007112  026501  030101  006451
1667  007120  000012
1668  007122  052502  020123  044524  EM1:    .ASCIZ  /BUS TIME-OUT WHEN REFERENCING A DAC ADDRESS/
1669  007130  042515  047455  052125
1670  007136  053440  042510  020116
1671  007144  042522  042506  042522
1672  007152  041516  047111  020107
1673  007160  020101  040504  020103
1674  007166  042101  051104  051505
1675  007174  000123
1676  007176  040504  030103  051040  EM2:    .ASCIZ  /DAC0 REGISTER IN ERROR/
1677  007204  043505  051511  042524
1678  007212  020122  047111  042440
1679  007220  051122  051117     000
1680  007225     104  041501  020061  EM3:    .ASCIZ  /DAC1 REGISTER IN ERROR/
1681  007232  042522  044507  052123
1682  007240  051105  044440  020116
1683  007246  051105  047522  000122
1684  007254  040504  031103  051040  EM4:    .ASCIZ  /DAC2 REGISTER IN ERROR/
1685  007262  043505  051511  042524
1686  007270  020122  047111  042440
1687  007276  051122  051117     000
1688  007303     104  041501  020063  EM5:    .ASCIZ  /DAC3 REGISTER IN ERROR/
1689  007310  042522  044507  052123
1690  007316  051105  044440  020116
1691  007324  051105  047522  000122
1692  007332  042523  042514  052103  EM6:    .ASCIZ  /SELECTED DAC OFFSET POT WAS ADJUSTED INCORRECTLY/
1693  007340  042105  042040  041501
1694  007346  047440  043106  042523
1695  007354  020124  047520  020124
1696  007362  040527  020123  042101
1697  007370  052512  052123  042105
1698  007376  044440  041516  051117
1699  007404  042522  052103  054514
1700  007412     000
1701  007413     123  046105  041505  EM7:    .ASCIZ  /SELECTED DAC GAIN POT WAS ADJUSTED INCORRECTLY/
1702  007420  042524  020104  040504
1703  007426  020103  040507  047111
1704  007434  050040  052117  053440
1705  007442  051501  040440  045104
1706  007450  051525  042524  020104
1707  007456  047111  047503  051122
1708  007464  041505  046124  000131
1709  007472  042523  042514  052103  EM10:   .ASCIZ  /SELECTED DAC HAS A LINEARITY PROBLEM/
```

```
1710  007500  042105  042040  041501
1711  007506  044040  051501  040440
1712  007514  046040  047111  040505
1713  007526  044522  054524  050040
1714  007530  047522  046102  046505
1715  007536      000
1716  007537      053  032461  053040  EM11:   .ASCIZ  /+15 VOLT SUPPLY IS INCORRECT/
1717  007544  046117  020124  052523
1720  007552  050120  054514  044440
1721  007560  040123  047111  047503
1722  007566  041122  041505  000124
1723  007574  030455  020065  047526  EM12:   .ASCIZ  /-15 VOLT SUPPLY IS INCORRECT/
1724  007602  052114  051440  050125
1725  007610  046120  020131  051511
1726  007616  044440  041516  051117
1727  007624  042522  052103      000
1730  007631      124  041501  021442  EM13:   .ASCIZ  /DAC #3 DIGITAL OUTPUT BITS IN ERROR/
1731  007636  020063  044504  044507
1732  007644  040524  020114  052517
1733  007652  050124  052125  041040
1734  007660  052111  020123  047111
1735  007666  042440  051122  051117
1736  007674      000
1737  007675      007  003407  040527  EM14:   .ASCIZ  <7><7><7>/WAKE UP OPERATOR AND ADJUST THE POT/<7><7>
1740  007702  042513  052440  020120
1741  007710  050117  051105  052101
1742  007716  051117  040440  042116
1743  007724  040440  045124  051525
1744  007732  020124  044124  020105
1745  007740  047520  003524  000007
1746  007746  042011  050105  042522  LDSPAC: .ASCIZ  /       DEPRESS THE "SPACE-BAR" WHEN DONE/
1747  007754  051523  052040  042510
1750  007762  021040  050123  041501
1751  007770  026505  040502  021122
1752  007776  053440  042516  020116
1753  010004  047504  042516      000
1754  010011      007  005015  042524  MSGSW:  .ASCII  <7><15><12>/TESTER SW1-2 AND SW1-5 ONLY MUST BE ON/
1755  010016  052123  051105  051440
1756  010024  030527  031055  040440
1757  010032  042116  051445  030527
1760  010040  032455  047440  046116
1761  010046  020131  042515  052123
1762  010054  041040  020105  047117          .ASCII  <7><15><12>/SW2-2 SW2-4 AND SW2-5 MUST BE ON/
1763  010062  006407  051412  031127
1764  010070  031055  051440  031127
1765  010075  032055  040440  042116
1766  010104  051440  031127  032455
1767  010112  046440  051525  020124
1770  010120  042502  047440      116
1771  010126      015  003412  047503          .ASCIZ  <15><12><7>/CONNECT AAV11 TO J09 OF TESTER ONLY/<15><12>
1772  010134  047116  041505  020124
1773  010142  040501  030524  020061
1774  010146  047524  045040  034460
1775  010154  047440  020106  042524
```

```
1764  010162  052123  051105  047440
1765  010170  046116  006531  000012
1766  010176  021440  042440  051122  ERRTOT: .ASCIZ  / # ERRORS/
1767  010204  051117  000123
1768  010210  041040  042101  052440  MESG0:  .ASCIZ  / BAD UNITS /
1769  010216  044516  051524  000040
1770  010224     015     012          FOUND1: .BYTE   15,12
1771  010230  051120  043517  040522          .ASCIZ  /PROGRAM DETECTED /
1772  010234  020115  042504  046524
1773  010242  052103  042106  000040
1774  010250  034057  004451  040501  FOUND2: .ASCIZ  /(8)    AAV11(S)  /
1775  010256  030440  024061  024523
1776  010264  020040     000
1777  010267     106  051122  041520  DH1:    .ASCIZ  /ERRPC  BUSADR  EXPECT   WAS/
1778  010274  041011  051525  042101
1779  010302  020122  042440  050130
1780  010310  041505  020124  020040
1781  010316  040527  000123
1782  010322  051105  050122  004503  DH2:    .ASCIZ  /ERRPC  BUSADR/
1783  010330  052522  040523  051104
1784  010336     000
1785  010337     105  051122  041520  DH6:    .ASCIZ  /ERRPC  BUSADR  EXPECT  WAS     SPREAD/
1786  010344  041011  051525  042101
1787  010352  004522  054105  042520
1788  010360  052103  033411  051501
1789  010366  051411  051120  040505
1790  010374  000104
1791  010376  005015  040440  045104  ADJR34: .ASCIZ  <15><12>/ ADJUST R34 FOR A NULL  /
1792  010404  051525  020124  031522
1793  010412  020064  047506  020122
1794  010420  020101  052516  046114
1795  010426  020040     000
1796  010431     015  020012  042101  ADJR35: .ASCIZ  <15><12>/ ADJUST R35 FOR A NULL  /
1797  010436  052512  052123  051040
1798  010444  032463  043040  051117
1799  010452  040440  047040  046125
1800  010460  020114  000040
1801  010464  005015  040440  045104  ADJR36: .ASCIZ  <15><12>/ ADJUST R36 FOR A NULL  /
      010472  051525  020124  031522
      010500  020066  047506  020122
      010506  020101  052516  046114
1805  010514  020040     000
1806  010517     015  020012  042101  ADJR37: .ASCIZ  <15><12>/ ADJUST R37 FOR A NULL  /
1807  010524  052512  052123  051040
1808  010532  033463  043040  051117
1809  010540  040440  047040  046125
1810  010546  020114  000040
1811  010552  005015  040440  045104  ADJR46: .ASCIZ  <15><12>/ ADJUST R46 FOR A NULL  /
1812  010560  051525  020124  032122
1813  010566  020066  047506  020122
1814  010574  020101  052516  046114
1815  010602  020040     000
1816  010605     015  020012  042101  ADJR47: .ASCIZ  <15><12>/ ADJUST R47 FOR A NULL  /
1817  010612  052512  052123  051040
```

```
 1818  010620  033464  043040  051117
 1819  010626  040440  047040  046125
 1820  010634  020114  000040
 1821  010640  005015  040440  045104  ADJR48: .ASCIZ  <15><12>/ ADJUST R48 FOR A NULL  /
 1822  010646  051525  020124  032122
 1823  010654  020070  047506  020122
 1824  010662  020101  052516  046114
       010670  020040     000
       010673     015  020012  042101  ADJR49: .ASCIZ  <15><12>/ ADJUST R49 FOR A NULL  /
       010700  052512  052123  051040
       010706  034464  043040  051117
       010714  040440  047040  046125
       010722  020114  000040

       010726  005015  042523  042514  SELD0:  .ASCIZ  <15><12>/SELECT DAC0/
       010734  052103  042040  041501
 1834  010742  000060
 1835  010744  005015  042523  042514  SELD1:  .ASCIZ  <15><12>/SELECT DAC1/
 1836  010752  052103  042040  041501
 1837  010760  000061
 1838  010762  005015  042523  042514  SELD2:  .ASCIZ  <15><12>/SELECT DAC2/
 1839  010770  052103  042040  041501
 1840  010776  000062
 1841  011000  005015  042523  042514  SELD3:  .ASCIZ  <15><12>/SELECT DAC3/
 1842  011006  052103  042040  041501
 1843  011014  000063
 1844  011016  005015  042101  052512  TRYAGN: .ASCIZ  <15><12>/ADJUST THAT SAME POT AGAIN PLEASE/<15><12>
 1845  011024  052123  052040  040510
 1846  011032  020124  040523  042515
 1847  011040  050040  052117  040440
 1848  011046  040507  047111  050040
 1849  011054  042514  051501  006505
 1850  011062  000012
 1851          000001                        STX=1
 1852          000003                        ETX=3
 1853  011064     001                NS1200: .BYTE   STX
 1854  011065     116  030465  030062        .ASCII  /NS1200V/
 1855  011072  030060     126
 1856  011075     003     000                .BYTE   ETX,0
 1857  011077     001                PS1175: .BYTE   STX
 1858  011100  032520  030461  032467        .ASCII  /PS1175V/
 1859  011106  053060
 1860  011110     003     000                .BYTE   ETX,0
 1861                                        .EVEN
 1862  011112  001116  001126  000000  DT1:   SERRPC,$BDDAT,0
 1863  011120  001116  001422  001124  DT2:   SERRPC,DAC0,$GDDAT,$BDDAT,0
 1864  011126  001126  000000
 1865  011132  001116  001424  001124  DT3:   SERRPC,DAC1,$GDDAT,$BDDAT,0
 1866  011140  001126  000000
 1867  011144  001116  001426  001124  DT4:   SERRPC,DAC2,$GDDAT,$BDDAT,0
 1868  011152  001126  000000
 1869  011156  001116  001430  001124  DT5:   SERRPC,DAC3,$GDDAT,$BDDAT,0
 1870  011164  001126  000000
 1871  011170  001116  007002  001124  DT6:   SERRPC,DACBAD,$GDDAT,$BDDAT,SPREAD,0
```

```
 1872  011176  001126  002016  000000
 1873  011204  000000  000000  000000  DFD:   0,0,0,0,0,0,0,0
 1874  011212  000000  000000  000000
 1875  011220  000000  000000
 1876
 1877                                  .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
 1878
 1879                                  ;;*****************************************************************
 1880                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
 1881                                  ;*BINARY-ASCII NUMBER AND TYPE IT.
 1882                                  ;*CALL:
 1883                                  ;*      MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
 1884                                  ;*      TYPBN                   ;;TYPE IT
 1885
 1886  011224  010146                  $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
 1887  011226  016601  000006                  MOV     6(SP),R1        ;;GET THE INPUT NUMBER
 1888  011232  000261                          SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
 1889  011234  112737  000060  011276  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
 1890  011242  006101                          ROL     R1              ;;GET THIS BIT
 1891  011244  001406                          BEQ     2$              ;;DONE?
 1892  011246  105537  011276                  ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
 1893  011252  104401  011276                  TYPE    ,$BIN           ;;GO TYPE THIS BIT
 1894  011256  000241                          CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
 1895  011260  000765                          BR      1$              ;;GO DO THE NEXT BIT
 1896  011262  012601                  2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
 1897  011264  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
 1898  011272  012616                          MOV     (SP)+,(SP)
 1899  011274  000002                          RTI                     ;;RETURN TO USER
 1900  011276     000     000          $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
 1901                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 1902
 1903                                  ;;*****************************************************************
 1904                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 1905                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 1906                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 1907                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 1908                                  ;*REPLACED WITH SPACES.
 1909                                  ;*CALL:
 1910                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
 1911                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
 1912
 1913  011300                          $TYPDS:
 1914  011300  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 1915  011302  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 1916  011304  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 1917  011306  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 1918  011310  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 1919  011312  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 1920  011316  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 1921  011322  100004                          BPL     1$              ;;BR IF INPUT IS POS.
 1922  011324  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
 1923  011326  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
 1924  011334  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
 1925  011336  012703  011514                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
```

```
011342  112723  000040          MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
011346  005002                  CLR     R2              ;;CLEAR THE BCD NUMBER
011350  016001  011504          MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
011354  160105                  SUB     R1,R5           ;;FORM THIS BCD DIGIT
011356  002402                  BLT     4$              ;;BR IF DONE
011360  005202                  INC     R2              ;;INCREASE THE BCD DIGIT BY 1
011362  000774                  BR      3$
011364  060105                  ADD     R1,R5           ;;ADD BACK THE CONSTANT
011366  005702                  TST     R2              ;;CHECK IF BCD DIGIT=0
011370  001002                  BNE     5$              ;;FALL THROUGH IF 0
011372  105716                  TSTB    (SP)            ;;STILL DOING LEADING 0'S?
011374  100407                  BMI     7$              ;;BR IF YES
011376  106316                  ASLB    (SP)            ;;MSD?
011400  103003                  BCC     6$              ;;BR IF NO
011402  116663  000001  177777  MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
011410  052702  000060          BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
011414  052702  000040          BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
011420  110223                  MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
011422  005720                  TST     (R0)+           ;;JUST INCREMENTING
011424  020027  000010          CMP     R0,#10          ;;CHECK THE TABLE INDEX
011430  002746                  BLT     2$              ;;GO DO THE NEXT DIGIT
011432  003002                  BGT     8$              ;;GO TO EXIT
011434  010502                  MOV     R5,R2           ;;GET THE LSD
011436  000764                  BR      6$              ;;GO CHANGE TO ASCII
011440  105726                  TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
011442  100003                  BPL     9$              ;;BR IF NO
011444  116663  177777  177776  MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
011452  105013                  CLRB    (R3)            ;;SET THE TERMINATOR
011454  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
011456  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
011460  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
011462  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
011464  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
011466  104401  011514          TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
011472  016646  000002  000004  MOV     2(SP),4(SP)     ;;ADJUST THE STACK
011500  012616                  MOV     (SP)+,(SP)
011502  000002                  RTI                     ;;RETURN TO USER
011504  023420          $DTBL:  10000.
011506  001750                  1000.
011510  000144                  100.
011512  000012                  10.
011514  000004          $DBLK:  .BLKW   4
                        .SBTTL  SCOPE HANDLER ROUTINE

                        ;;**************************************************************
                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                        ;*AND LOAD THE ERROR FLAG (SERFLG) INTO DISPLAY<15:08>
                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                        ;*SW14=1        LOOP ON TEST
                        ;*SW11=1        INHIBIT ITERATIONS
                        ;*SW09=1        LOOP ON ERROR
                        ;*SW08=1        LOOP ON TEST IN SWR<7:0>
                        ;*CALL
```

```
 1980                                          .  SCOPE          ;;SCOPE=IOT
 1981
 1982  011524                                 $SCOPE:
 1983  011524  104410                                CKSWR
 1984  011526  104410                                CKSWR
 1985  011530  032777  040000  167402  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 1986  011536  001114                          BNE     $OVER           ;;YES IF SW14=1
 1987                                         ;#####START OF CODE FOR THE XOR TESTER#####
 1988  011540  000416                         $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
 1989                                                                         ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 1990  011542  013746  000004                         MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 1991  011546  012737  011566  000004                 MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 1992  011554  005737  177060                        TST     @#177060        ;;TIME OUT ON XOR?
 1993  011560  012637  000004                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 1994  011564  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
 1995  011566  022626                         5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 1996  011570  012637  000004                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 1997  011574  000423                          BR      7$              ;;LOOP ON THE PRESENT TEST
 1998  011576                                 6$:;#####END OF CODE FOR THE XOR TESTER#####
 1999  011576  032777  000400  167334                BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 2000  011604  001404                          BEQ     2$              ;;BR IF NO
 2001  011606  127737  167326  001102                CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR(7:0)
 2002  011614  001465                          BEQ     $OVER           ;;BR IF YES
 2003  011616  105737  001103                 2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 2004  011622  001421                          BEQ     3$              ;;BR IF NO
 2005  011624  123737  001115  001103                CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 2006  011632  101015                          BHI     3$              ;;BR IF NO
 2007  011634  032777  001000  167276                BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 2008  011642  001404                          BEQ     4$              ;;BR IF NO
 2009  011644  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 2010  011652  000446                          BR      $OVER
 2011  011654  105037  001103                 4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 2012  011660  005037  001160                        CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 2013  011664  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
 2014  011666  032777  004000  167244  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
 2015  011674  001011                          BNE     1$              ;;BR IF YES
 2016  011676  005737  001202                        TST     $PASS           ;;IF FIRST PASS OF PROGRAM
 2017  011702  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
 2018  011704  005237  001104                        INC     $ICNT           ;;INCREMENT ITERATION COUNT
 2019  011710  023737  001160  001104                CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
 2020  011716  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
 2021  011720  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
 2022  011726  013737  012004  001160                MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
 2023  011734  105237  001102                 $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 2024  011740  113737  001102  001200                MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 2025  011746  011637  001106                        MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 2026  011752  011637  001110                        MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 2027  011756  005037  001162                        CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 2028  011762  112737  000001  001115                MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(!) ERROR ON NEXT TEST
 2029  011770  013777  001102  167144  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 2030  011776  013716  001106                        MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 2031  012002  000002                          RTI                     ;;FIXES PS
 2032  012004  003720                         $MXCNT: 2000.           ;;MAX. NUMBER OF ITERATIONS
 2033                                         .SBTTL  ERROR HANDLER ROUTINE
```

```
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO $ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

012006                                  $ERROR:
012006  104410                                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
012010  053737  007010  007004                   BIS     MASKNM,BADUNT
012016  105237  001103                  7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
012022  001775                                  BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
012024  013777  001102  167110                   MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
012032  032777  002000  167100                   BIT     #BIT10,@SWR             ;;BELL ON ERROR?
012040  001402                                  BEQ     1$                      ;;NO - SKIP
012042  104401  001164                          TYPE    ,$BELL                  ;;RING BELL
012046  005237  001112                  1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
012052  011637  001116                          MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
012056  162737  000002  001116                   SUB     #2,$ERRPC
012064  117737  167026  001114                   MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
012072  032777  020000  167040                   BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
012100  001004                                  BNE     20$                     ;;SKIP TYPEOUTS
012102  004737  012202                          JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
012106  104401  001171                          TYPE    ,$CRLF
012112                                  20$:
012112  122737  000001  001214                   CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
012120  001007                                  BNE     2$                      ;;NO,SKIP APT ERROR REPORT
012122  113737  001114  012134                   MOVB    $ITEMB,21$              ;;SET ITEM NUMBER AS ERROR NUMBER
012130  004737  014162                          JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
012134  000                             21$:    .BYTE   0
012135  000                                     .BYTE   0
012136  000777                          22$:    BR      22$                     ;;APT ERROR LOOP
012140  005777  166774                  2$:     TST     @SWR                    ;;HALT ON ERROR
012144  100002                                  BPL     3$                      ;;SKIP IF CONTINUE
012146  000000                                  HALT                            ;;HALT ON ERROR!
012150  104410                                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
012152  032777  001000  166760          3$:     BIT     #BIT09,@SWR             ;;LOOP ON ERROR SWITCH SET?
012160  001402                                  BEQ     4$                      ;;BR IF NO
012162  013716  001110                          MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
012166  005737  001162                  4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
012172  001402                                  BEQ     5$                      ;;BR IF NONE
012174  013716  001162                          MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
012200                                  5$:
012200  000002                                  RTI                             ;;RETURN
                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                        ;;*****************************************************************
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
```

```
                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (SEPRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

012200                                  $ERRTYP:
012202  104401  001171                        TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
012206  010046                              MOV     R0,-(SP)        ;;SAVE R0
012210  005000                              CLR     R0              ;;PICKUP THE ITEM INDEX
012212  157700  001114                      BISB    @#$ITEMB,R0
012216  001004                              BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                    ;;TYPE THE PC OF THE ERROR
012220  013746  001116                      MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                    ;;ERROR ADDRESS
012224  104402                              TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012226  000426                              BR      6$              ;;GET OUT
012230  005300                      1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
012232  006300                              ASL     R0              ;;      WORK FOR THE ERROR TABLE
012234  006300                              ASL     R0
012236  006300                              ASL     R0
012240  062700  001256                      ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
012244  012037  012254                      MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
012250  001404                              BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
012252  104401                              TYPE                    ;;TYPE THE "ERROR MESSAGE"
012254  000000                      2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
012256  104401  001171                      TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
012262  012037  012272              3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
012266  001404                              BEQ     5$              ;;SKIP TYPEOUT IF 0
012270  104401                              TYPE                    ;;TYPE THE "DATA HEADER"
012272  000000                      4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
012274  104401  001171                      TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
012300  011000                      5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
012302  001004                              BNE     7$              ;;GO TYPE THE DATA
012304  012600                      6$:     MOV     (SP)+,R0        ;;RESTORE R0
012306  104401  001171                      TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
012312  000207                              RTS     PC              ;;RETURN
012314                                  7$:
012314  013046                              MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
012316  104402                              TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012320  005710                              TST     (R0)            ;;IS THERE ANOTHER NUMBER?
012322  001770                              BEQ     6$              ;;BR IF NO
012324  104401  012332                      TYPE    ,8$             ;;TYPE TWO(2) SPACES
012330  000771                              BR      7$              ;;LOOP
012332  020040    000           8$:     .ASCIZ  /  /                ;;TWO(2) SPACES
        012336                          .EVEN
                                        .SBTTL  POWER DOWN AND UP ROUTINES

                                        ;;**************************************************************
                                        ;POWER DOWN ROUTINE
012336  012737  012502  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
012344  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
012352  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
012354  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
012356  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
012360  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
012362  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
```

```
2142  012364  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2143  012366  017746  166546                   MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
2144  012372  010637  012506                   MOV     SP,$SAVR6       ;;SAVE SP
2145  012376  012737  012410  000024           MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
2146  012404  000000                          HALT
2147  012406  000776                          BR      .-2             ;;HANG UP
2148
2149                                    ;;**************************************************************
2150                                    ;POWER UP ROUTINE
2151  012410  012737  012502  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
2152  012416  013706  012506                   MOV     $SAVR6,SP       ;;GET SP
2153  012422  005037  012506                   CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
2154  012426  005237  012506           1$:     INC     $SAVR6          ;;WAIT FOR THE INC
2155  012432  001375                          BNE     1$              ;;OF  WORD
2156  012434  012677  166500                   MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
2157  012440  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
2158  012442  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
2159  012444  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
2160  012446  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
2161  012450  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2162  012452  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2163  012454  012737  012336  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
2164  012462  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
2165  012470  104401                          TYPE                    ;;REPORT THE POWER FAILURE
2166  012472  012510                  $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
2167  012474  012716                          MOV     (PC)+,(SP)      ;;RESTART AT BEGIN
2168  012476  001450                  $PWRAD: .WORD   BEGIN           ;;RESTART ADDRESS
2169  012500  000002                          RTI
2170  012502  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
2171  012504  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
2172  012506  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
2173  012510  005015  042522  052123  PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>
2174  012516  051101  044524  043516
2175  012524  040440  052040  051105
2176  012532  040440  050040  053517
2177  012540  051105  043040  044501
2178  012546  052514  042522  005015
2179  012554  000012
2180
2181                                    .EVEN
2182
2183                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
2184
2185                                    ;;**************************************************************
2186                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2187                                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2188                                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2189                                    ;*CALL:
2190                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2191                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
2192                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2193                                    ;*      .BYTE   M               ;;M=1 OR 0
2194                                    ;*                                      ;;1=TYPE LEADING ZEROS
2195                                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
```

```
2196                                    ;*STYPON- --ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2197                                    ;*STYPOS OR STYPOC
2198                                    ;*CALL:
2199                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2200                                    ;*      TYPON                   ;;CALL FOR TYPEOUT
2201                                    ;*
2202                                    ;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2203                                    ;*CALL:
2204                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2205                                    ;*      TYPOC                   ;;CALL FOR TYPEOUT
2206
2207  012556  017646  000000            $TYPOS: MOV     2(SP),-(SP)     ;;PICKUP THE MODE
2208  012562  116637  000001  013001            MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
2209  012570  116637  013003                    MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
2210  012574  062716  000002                    ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
2211  012600  000406                            BR      $TYPON
2212  012602  112737  000001  013001    $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
2213  012610  112737  000006  013003            MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
2214  012616  112737  000005  013000    $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
2215  012624  010346                            MOV     R3,-(SP)        ;;SAVE R3
2216  012626  010446                            MOV     R4,-(SP)        ;;SAVE R4
2217  012630  010546                            MOV     R5,-(SP)        ;;SAVE R5
2218  012632  113704  013003                    MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
2219  012636  005404                            NEG     R4
2220  012640  062704  000006                    ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
2221  012644  110437  013002                    MOVB    R4,$OMODE       ;;SAVE IT FOR USE
2222  012650  113704  013001                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
2223  012654  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
2224  012660  005003                            CLR     R3              ;;CLEAR THE OUTPUT WORD
2225  012662  006105                    1$:     ROL     R5              ;;ROTATE MSB INTO "C"
2226  012664  000404                            BR      3$              ;;GO DO MSB
2227  012666  006105                    2$:     ROL     R5              ;;FORM THIS DIGIT
2228  012670  006105                            ROL     R5
2229  012672  006105                            ROL     R5
2230  012674  010503                            MOV     R5,R3
2231  012676  006103                    3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
2232  012700  105337  013002                    DECB    $OMODE          ;;TYPE THIS DIGIT?
2233  012704  100016                            BPL     7$              ;;BR IF NO
2234  012706  042703  177770                    BIC     #177770,R3      ;;GET RID OF JUNK
2235  012712  001002                            BNE     4$              ;;TEST FOR 0
2236  012714  005704                            TST     R4              ;;SUPPRESS THIS 0?
2237  012716  001403                            BEQ     5$              ;;BR IF YES
2238  012720  005204                    4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
2239  012722  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
2240  012726  052703  000040            5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
2241  012732  110337  012776                    MOVB    R3,8$           ;;SAVE FOR TYPING
2242  012736  104401  012776                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
2243  012742  105337  013000            7$:     DECB    $OCNT           ;;COUNT BY 1
2244  012746  003347                            BGT     2$              ;;BR IF MORE TO DO
2245  012750  002402                            BLT     6$              ;;BR IF DONE
2246  012752  005204                            INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
2247  012754  000744                            BR      2$              ;;GO DO THE LAST DIGIT
2248  012756  012605                    6$:     MOV     (SP)+,R5        ;;RESTORE R5
2249  012760  012604                            MOV     (SP)+,R4        ;;RESTORE R4
```

```
                                        MOV     (SP)+,R3        ;;RESTORE R3
                                        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
                                        MOV     (SP)+,(SP)
                                        RTI                     ;;RETURN
                                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
                                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
                                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
                                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
                                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
                                .SBTTL  TYPE ROUTINE

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

013004  10573'  001157  $TYPE:  TSTB    @TPFLG          ;;IS THERE A TERMINAL?
013010  100002                  BPL     1$              ;;BR IF YES
013012  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
013014  000430                  BR      3$              ;;LEAVE
013016  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
013020  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
013024  122737  000001  001214  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
013032  001011                  BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
013034  132737  000100  001215  BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
013042  001405                  BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
013044  010037  013054          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
013050  004737  014152          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
013054  000000          61$:    .WORD   0               ;;MESSAGE ADDRESS
013056  132737  000040  001215  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
013064  001003                  BNE     60$             ;;YES,SKIP TYPE OUT
013066  112046          2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
013070  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
013072  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
013074  012600          60$:    MOV     (SP)+,R0        ;;RESTORE R0
013076  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
013102  000002                  RTI                     ;;RETURN
013104  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
013110  001430                  BEQ     8$
013112  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
013116  001006                  BNE     5$
013120  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
013122  104401                  TYPE                    ;;TYPE A CR AND LF
013124  001171                  $CRLF
```

```
2304  013126  105C37  013262            CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
2305  013132  000755                    BR      2$              ;;GET NEXT CHARACTER
2306  013134  004737  013216    5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
2307  013140  123726  001156    6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
2308  013144  001350                    BNE     2$              ;;IF NO GO GET NEXT CHAR.
2309  013146  013746  001154            MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
2310                                                            ;;AND THE NULL CHAR.
2311  013152  105366  000001    7$:     DECB    1 SP)           ;;DOES A NULL NEED TO BE TYPED?
2312  013156  002770                    BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
2313  013160  004737  013216            JSR     PC,$TYPEC       ;;GO TYPE A NULL
2314  013164  105337  013262            DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
2315  013170  000770                    BR      7$              ;;LOOP
2316
2317                                    ;HORIZONTAL TAB PROCESSOR
2318
2319  013172  112716  000040    8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
2320  013176  004737  013216    9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
2321  013202  132737  000007  013262    BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
2322  013210  001372                    BNE     9$              ;;TAB STOP
2323  013212  005726                    TST     (SP)+           ;;POP SPACE OFF STACK
2324  013214  000724                    BR      2$              ;;GET NEXT CHARACTER
2325  013216  105777  165726    $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
2326  013222  100375                    BPL     $TYPEC
2327  013224  116677  000002  165720    MOVB    2(SP),@$TPD     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2328  013232  122766  000015  000002    CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
2329  013240  001003                    BNE     1$              ;;BRANCH IF NO
2330  013242  105037  013262            CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
2331  013246  000406                    BR      $TYPEX          ;;EXIT
2332  013250  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
2333  013256  001402                    BEQ     $TYPEX          ;;BRANCH IF YES
2334  013260  105227                    INCB    (PC)+           ;;COUNT THE CHARACTER
2335  013262  000000    $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
2336  013264  000207    $TYPEX: RTS     PC
2337
2338                                    .SBTTL  TTY INPUT ROUTINE
2339
2340                                    ;;**************************************************************
2341                                    .ENABL  LSB
2342
2343                                    ;;**************************************************************
2344                                    ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2345                                    ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2346                                    ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2347                                    ;*WHEN OPERATING IN TTY FLAG MODE.
2348  013266  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
2349  013274  001074                    BNE     15$             ;;BRANCH IF NO
2350  013276  105777  165642            TSTB    @$TKS           ;;CHAR THERE?
2351  013302  100071                    BPL     15$             ;;IF NO, DON'T WAIT AROUND
2352  013304  117746  165636            MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
2353  013310  042716  177600            BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
2354  013314  022726  000007            CMP     #7,(SP)+        ;;IS IT A CONTROL G?
2355  013320  001062                    BNE     15$             ;;NO, RETURN TO USER
2356  013322  123727  001134  000001    CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
2357  013330  001456                    BEQ     15$             ;;BRANCH IF YES
```

```
2358
2359  013332  104401  014013            TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (↑G)
2360  013336  104401  014020  $GTSWR:   TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
2361  013342  013746  000176            MOV     $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
2362  013346  104402                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2363  013350  104401  014031            TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
2364  013354  005046          19$:      CLR     -(SP)           ;;CLEAR COUNTER
2365  013356  005046                    CLR     -(SP)           ;;THE NEW SWR
2366  013360  105777  165560  7$:       TSTB    @$TKS           ;;CHAR THERE?
2367  013364  100375                    BPL     7$              ;;IF NOT TRY AGAIN
2368
2369  013366  117746  165554            MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
2370  013372  042716  177600            BIC     #↑C177,(SP)     ;;MAKE IT 7-BIT ASCII
2371
2372
2373
2374  013376  021627  000025  9$:       CMP     (SP),#25        ;;IS IT A CONTROL-U?
2375  013402  001005                    BNE     10$             ;;BRANCH IF NOT
2376  013404  104401  014006            TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (↑U)
2377  013410  062706  000006  20$:      ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
2378  013414  000757                    BR      19$             ;;LET'S TRY IT AGAIN
2379
2380
2381  013416  021627  000015  10$:      CMP     (SP),#15        ;;IS IT A <CR>?
2382  013422  001022                    BNE     16$             ;;BRANCH IF NO
2383  013424  005766  000004            TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
2384  013430  001403                    BEQ     11$             ;;BRANCH IF YES
2385  013432  016677  000002  165500    MOV     2(SP),@SWR      ;;SAVE NEW SWR
2386  013440  062706  000006  11$:      ADD     #6,SP           ;;CLEAR UP STACK
2387  013444  104401  001171  14$:      TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
2388  013450  123727  001135  000001    CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
2389  013456  001003                    BNE     15$             ;;BRANCH IF NOT
2390  013460  012777  000100  165456    MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
2391  013466  000002          15$:      RTI                     ;;RETURN
2392  013470  004777  013216  16$:      JSR     PC,$TYPEC       ;;ECHO CHAR
2393  013474  021627  000060            CMP     (SP),#60        ;;CHAR < 0?
2394  013500  002420                    BLT     18$             ;;BRANCH IF YES
2395  013502  021627  000067            CMP     (SP),#67        ;;CHAR > 7?
2396  013506  003015                    BGT     18$             ;;BRANCH IF YES
2397  013510  042726  000060            BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2398  013514  005766  000002            TST     2(SP)           ;;IS THIS THE FIRST CHAR
2399  013520  001403                    BEQ     17$             ;;BRANCH IF YES
2400  013522  006316                    ASL     (SP)            ;;NO, SHIFT PRESENT
2401  013524  006316                    ASL     (SP)            ;;   CHAR OVER TO MAKE
2402  013526  006316                    ASL     (SP)            ;;   ROOM FOR NEW ONE.
2403  013530  005266  000002  17$:      INC     2(SP)           ;;KEEP COUNT OF CHAR
2404  013534  056616  177776            BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2405  013540  000707                    BR      7$              ;;GET THE NEXT ONE
2406  013542  104401  001170  18$:      TYPE    ,$QUES          ;;TYPE ?<CR><LF>
2407  013546  000720                    BR      20$             ;;SIMULATE CONTROL-U
2408                          .DSABL    LSB
2409
2410
2411                          ;;**************************************************************
```

```
                                        ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                        ;*CALL:
                                        ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                        ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                        ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                        ;
013550  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
013552  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
013560  105777  165360          1$:     TSTB    @$TKS           ;;WAIT FOR
013564  100375                          BPL     1$              ;;A CHARACTER
013566  117766  165354  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
013574  042766  177600  000034          BIC     #^C(177),4(SP)  ;;GET RID OF JUNK IF ANY
013602  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
013610  001013                          BNE     3$              ;;BRANCH IF NO
013612  105777  165326          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
013616  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
013620  117746  165322                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
013624  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
013630  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
013634  001366                          BNE     2$              ;;IF NOT DISCARD IT
013636  000750                          BR      1$              ;;YES, RESUME
013640  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
013646  002407                          BLT     4$              ;;BRANCH IF YES
013650  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
013656  003003                          BGT     4$              ;;BRANCH IF YES
013660  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
013666  000002                  4$:     RTI                     ;;GO BACK TO USER
                                        ;;**************************************************************
                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                        ;*CALL:
                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
                                        ;
013670  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
013672  012703  013776          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
013676  022703  014006          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
013702  101405                          BLOS    4$              ;;BR IF YES
013704  104411                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
013706  112313                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
013710  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
013714  001003                          BNE     3$              ;;SKIP IF NOT
013716  104401  001170          4$:     TYPE    $QUES           ;;TYPE A '?'
013722  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
013724  111337  013774          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
013730  104401  013774                  TYPE    ,9$
013734  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
013740  001356                          BNE     2$              ;;LOOP IF NOT RETURN
013742  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
013746  104401  001172                  TYPE    $LF             ;;TYPE A LINE FEED
013752  012603                          MOV     (SP)+,R3        ;;RESTORE R3
013754  010646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
013756  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
```

```
2466  013764  012766  013776  000004          MOV     #STTYIN,4(SP)
2467  013772  000002                          RTI                             ;;RETURN
2468  013774     000  9$:     .BYTE   0       ;;STORAGE FOR ASCII CHAR. TO TYPE
2469  013775     000          .BYTE   0       ;;TERMINATOR
2470  013776  000010  $TTYIN: .BLKB   8.      ;;RESERVE 8 BYTES FOR TTY INPUT
2471  014006  052536  005015  000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
2472  014013     136  006507  000012 $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL "G"
2473  014020  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
2474  014026  020075     000
2475  014031     040  047040  053505  $MNEW.  .ASCIZ  /  NEW = /
2476  014036  036440  000040
2477                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2478
2479                                  ;;**************************************************************
2480                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2481                                  ;*CHANGE IT TO BINARY.
2482                                  ;*CALL:
2483                                  ;*      RDOCT                           ;;READ AN OCTAL NUMBER
2484                                  ;*      RETURN HERE                     ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2485                                  ;*                                      ;;HIGH ORDER BITS ARE IN $HIOCT
2486
2487  014042  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2488  014044  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
2489  014052  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2490  014054  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2491  014056  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2492  014060  104412                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
2493  014062  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2494  014064  005001                          CLR     R1              ;;CLEAR DATA WORD
2495  014066  005002                          CLR     R2
2496  014070  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
2497  014072  001412                          BEQ     3$              ;;IF ZERO GET OUT
2498  014074  006301                          ASL     R1              ;;*2
2499  014076  006102                          ROL     R2
2500  014100  006301                          ASL     R1              ;;*4
2501  014102  006102                          ROL     R2
2502  014104  006301                          ASL     R1              ;;*8
2503  014106  006102                          ROL     R2
2504  014110  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
2505  014114  060201                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
2506  014116  000764                          BR      2$              ;;LOOP
2507  014120  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
2508  014122  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
2509  014126  010237  014142                  MOV     R2,$HIOCT
2510  014132  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
2511  014134  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2512  014136  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2513  014140  000002                          RTI                     ;;RETURN
2514  014142  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
2515                                  .SBTTL  APT COMMUNICATIONS ROUTINE
2516
2517                                  ;;**************************************************************
2518  014144  112737  000001  014410  $ATY1:   MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
2519  014152  112737  000001  014406  $ATY3:   MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
```

```
2520  014160  000403                          BR      $ATYC
2521  014162  112737  000001  014410  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
2522  014170                          $ATYC:
2523  014170  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2524  014172  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2525  014174  105737  014406                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
2526  014200  001450                          BEQ     5$              ;;IF NOT:  BR
2527  014202  122737  000001  001214          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
2530  014210  001031                          BNE     3$              ;;IF NOT:  BR
2531  014212  132737  000100  001215          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
2532  014220  001425                          BEQ     3$              ;;IF NOT:  BR
2533  014222  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
2534  014226  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
2535  014234  005737  001174          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
2536  014240  001375                          BNE     1$              ;;IF NOT:  WAIT
2537  014242  010037  001210                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
2540  014246  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
2541  014250  001376                          BNE     2$
2542  014252  163700  001210                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
2543  014256  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
2544  014260  010037  001212                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
2545  014264  012737  000004  001174          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
2546  014272  000413                          BR      5$
2547  014274  017637  000004  014320  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
2550  014302  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
2551  014310  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
2552  014314  004737  013004                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
2553  014320  000000                  4$:     .WORD   0
2554  014322                          5$:
2555  014322  105737  014410          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
2556  014326  001416                          BEQ     12$             ;;IF NOT:  BR
2557  014330  005737  001214                  TST     $ENV            ;;RUNNING UNDER APT?
2560  014334  001413                          BEQ     12$             ;;IF NOT:  BR
2561  014336  005737  001174          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
2562  014342  001375                          BNE     11$             ;;IF NOT:  WAIT
2563  014344  017637  000004  001176          MOV     @4(SP),$FATAL   ;;GET ERROR #
2564  014352  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
2565  014360  005237  001174                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
2566  014364  105037  014410          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
2567  014370  105037  014407                  CLRB    $LFLG           ;;CLEAR LOG FLAG
2570  014374  105037  014406                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
2571  014400  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2572  014402  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2573  014404  000207                          RTS     PC              ;;RETURN
2574  014406     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
2575  014407     000          $LFLG:  .BYTE   0               ;;LOG FLAG
2576  014410     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
2577                                  .EVEN
2600                          APTSIZE=200
2601                          APTENV=0C1
2602                          APTSPOOL=100
2603                          APTCSUF=040
2604
```

```
                                .SBTTL  TRAP DECODER

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

014412  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
014414  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
014420  005340                          TST     -(R0)           ;;BACKUP BY 2
014422  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
014424  006300                          ASL     R0              ;;POSITION FOR INDEXING
014426  016000  014446                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
014432  000200                          RTS     R0              ;;GO TO ROUTINE


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

014434  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
014436  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
014444  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
014446  014434                  $TRPAD: .WORD   $TRAP2
014450  013004                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
014452  012602                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
014454  012556                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
014456  012616                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
014460  011300                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
014462  011224                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

014464  013336                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING

014466  013266                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
014470  013650                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
014472  013670                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
014474  014042                          $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY

000001                          .END
```

```
ABASE = 170440            410#    526     567     663     664     665     666    1639
ACOM1 = 000000            526     569
ACOM2 = 000000            526
ACPUOP= 000000            526     541
ADBR    007030           1532    1650#
ADCS    007026           1529#   1530    1649#
ADDOK   001440           1639    669#
ADDMD = 000000            526
ADDM1 = 000000            526
ADDM10= 000000            526
ADDM11= 000000            526
ADDM12= 000000            526
ADDM13= 000000            526
ADDM14= 000000            526
ADDM15= 000000            526
ADDM2 = 000000            526
ADDM3 = 000000            526
ADDM4 = 000000            526
ADDM5 = 000000            526
ADDM6 = 000000            526
ADDM7 = 000000            526
ADDM8 = 000000            526
ADDM9 = 000000            526
ADEVCT= 000000            526     532
ADEVM = 000000            526     568
ADJR34  010376           1457    1461    1791#
ADJR35  010431           1292    1796#
ADJR36  010464           1327    1801#
ADJR37  010517           1362    1806#
ADJR46  010552           1244    1435    1811#
ADJR47  010605           1279    1816#
ADJR48  010640           1314    1821#
ADJR49  010673           1349    1826#
AEMV  = 000000            526     537
AEMVM = 000000            526     538
AFATAL= 000000            526     529
AMBDR1= 000000            526     554
AMBDR2= 000000            526     558
AMBDR3= 000000            526     561
AMBDR4= 000000            526     564
AMBMS1= 000000            526     548
AMBMS2= 000000            526     556
AMBMS3= 000000            526     559
AMBMS4= 000000            526     562
AMSGRD= 000000            526     534
AMSGLG= 000000            526     535
AMSGTY= 000000            526     528
AMTYP1= 000000            526     549
AMTYP2= 000000            526     557
AMTYP3= 000000            526     560
AMTYP4= 000000            526     563
APASS = 000000            526     531
APMTOP= 000000            526
APTCSU= 000040           2189    2571#
```

```
APTENV= 000001          2065    2282    2527    2569#
APTSIZ= 000200           714    2568#
APTSPO= 000100          2284    2529    2570#
ASWREG= 000000           526     539
ATESTN= 000000           526     530
AUNIT = 000000           526     533
AUSWR = 000000           526     540
AVECT1= 000000           526     565
AVECT2= 000000           526     566
BAOUNT  007004           718*   1418    1640#   2049*
BEGIN   001450           433     671#   2168
BEGINA  001456           670     672#
BEGIN1  001462           668     673#
BIT0  = 000001           394#    794    1642
BIT00 = 000001           384#    394
BIT01 = 000002           383#    393
BIT02 = 000004           382#    392
BIT03 = 000010           381#    391
BIT04 = 000020           380#    390
BIT05 = 000040           379#    389
BIT06 = 000100           378#    388
BIT07 = 000200           377#    387
BIT08 = 000400           376#    386    1999
BIT09 = 001000           375#    385    2007    2076
BIT1  = 000002           393#
BIT10 = 002000           374#   2053
BIT11 = 004000           373#    837     850     902     915     969     983    1038    1052    1194    2014
BIT12 = 010000           372#
BIT13 = 020000           371#   2060
BIT14 = 040000           370#   1985
BIT15 = 100000           369#    786
BIT2  = 000004           392#
BIT3  = 000010           391#   1193    1565
BIT4  = 000020           390#
BIT5  = 000040           389#
BIT6  = 000100           388#
BIT7  = 000200           387#   1508    1512    1528
BIT8  = 000400           386#
BIT9  = 001000           385#
BPTVEC= 000014           401#
CALDAC  006104          1265    1300    1335    1370    1477#
CKSWR = 104410          1611    1623    1983    1984    2048    2075    2613#
COMPAR  006520          1216    1259    1441    1467    1489    1569#
CONVRT  006302          1213    1256    1438    1464    1485    1523#
CR    = 000015           309#   2328    2338
CRLF  = 000200           310#   2299    2338
CSPACE  006422          1[illegible]    1462    1548#
DACBAD  007002          1425*   1454*   1479*   1639*   1871
DACO    001422           663#    734#    800     815*    816     826*    827     838*    839     851*    852     855     857*
                         858*    867#    869#    870     874*   1083*   1088    1115*   1118    1172*   1242    1256    1266
                        1591    1613#   1630#   1863
DAC1    001424           664#    735#    738#    801     882*    883     892*    893     903*    904     916*    917     920
                         922#    923#    932#    934#    935     939*   1084*   1094    1128*   1131    1173*   1277    1291
                        1301    1593    1614#   1631#   1865
```

```
DAC2    001426          665#    736*    739*    802     947*    948     958*    959     970*    971     984*    985     988
                        990*    991*   1000*   1002*   1003    1007*   1085*   1100    1142*   1145    1174*   1312    1326
                       1336    1595    1615*   1632*   1867
DAC3    001430          666#    737*    740*    803    1016*   1017    1027*   1028    1039*   1040    1053*   1054    1057
                       1059*   1060*   1069*   1071*   1072    1076*   1086*   1106    1154*   1158    1175*   1196*   1347
                       1361    1371    1597    1616*   1633*   1869
DDISP = 177570          316#    507     702
DF0     011204          593     599     605     611     617     623     629     635     641     647     653    1873#
DH1     010267          597     603     609     615     651    1777#
DH2     010322          591    1782#
DH6     010337          621     627     633     639     645    1785#
DISPLA  001142          507#    702*    710*   2029*   2052*
DISPRE  000174          430#    710
DRIN    007024         1197    1648#
DSWR  = 177570          315#    506     701
DT1     011112          592    1862#
DT2     011120          598    1863#
DT3     011132          604    1865#
DT4     011144          610    1867#
DT5     011156          616     652    1869#
DT6     011170          622     628     634     640     646    1871#
D/MCAL  006716          436    1621#
EMTVEC= 000030          404#    686*    687*
EM1     007122          590    1668#
EM10    007472          632    1709#
EM11    007537          638    1716#
EM12    007574          644    1721#
EM13    007631          650    1726#
EM14    007675          656    1733#
EM2     007176          596    1676#
EM3     007225          602    1680#
EM4     007254          608    1684#
EM5     007303          614    1688#
EM6     007332          620    1692#
EM7     007413          626    1701#
ERRTCT  010176         1414    1766#
ERRVEC= 000004          397#    699     700*    711*    759*    799*    807*    809*   1990    1991*   1993*   1996#
ETX   = 000003         1852#   1856    1860
EVER    001420          662#    674*    773     785*    786*    788    1169
FILZ    007022         1504*   1508*   1512*   1516*   1647#
FOUND1  010224          778    1770#
FOUND2  010250          784    1774#
FULRMP  006570          434    1589#
GAIDAC  005764         1255    1290    1325    1360    1452#
GNS   = ****** U        429    2604    2605    2606    2607    2608    2609    2611    2613    2614    2615    2616
GTSWR = 104407         2611#
HT    = 000011          307#   2297    2338
INIT1   002044          719     743#   1420
INIT7   005576         1406    1412#
IOTVEC= 000020          402#    684*    685*
LDSPAC  007746         1548    1740#
LDTRAP  001746          722#    743     793    1590    1610    1622
LF    = 000012          308#   2332    2338
MASKNM  007010          757*    794*   1176*   1642#   2049
```

DVAAAA.P11 CROSS REFERENCE TABLE

```
SELD1   010744          1278    1835#
SELD2   010762          1313    1836#
SELD3   011000          1348    1841#
SNOVLT  006212          1431    1457    1501#
SP      =X000006         327#    682#    699#    707#    711     768     779#    805     1394#   1415#   1418#   1569#   1570#
                        1578    1579    1580    1583    1584    1589#   1609#   1621#   1886#   1887    1896    1897#   1898#
                        1914#   1915#   1916    1917#   1918#   1919#   1920    1923#   1936    1938#   1940    1950    1952
                        1954    1955    1956    1957    1958    1960#   1961#   1990#   1993    1995    1996    2025    2026
                        2030#   2057    2078#   2081#   2093#   2098#   2119    2123#   2137#   2138#   2139#   2140#   2141#
                        2142#   2143#   2144    2152#   2156    2157    2158    2159    2160    2161    2162    2167#   2207#
                        2208    2209    2210#   2215#   2216#   2217#   2223    2248    2249    2250    2251#   2252#   2280#
                        2281    2291#   2293    2294    2295#   2297    2299    2301    2307    2309#   2311#   2319#   2323
                        2324    2327    2328    2332    2352#   2353#   2354    2361#   2364#   2365#   2369#   2370#   2374    2377#
                        2381    2383    2385    2386#   2393    2395    2397#   2398    2400#   2401#   2402#   2403#   2404#
                        2419#   2420#   2423#   2424#   2425    2429#   2430#   2431    2434    2436    2438#   2447#   2452
                        2463    2464#   2465#   2466#   2487#   2488#   2489#   2490#   2491#   2493    2496#   2504#   2505
                        2507    2508#   2510    2511    2512    2523#   2524#   2531    2532#   2543    2544#   2545#   2555
                        2556#   2561    2562    2581#   2582    2592#   2593#
SPREAD  000016          1215#   1229#   1440#   1466#   1488#   1576    1645#   1871
STACK = 001000           302#    682    1589    1609    1621
STATIC  006654           435    1609#
STKLMT= 177774           313#
STX     = 000001        1851#   1853    1857
SWR     001140           506#    680     701#    703     709#    716#    1612    1624    1985    1999    2001    2007    2014
                        2053    2060    2072    2076    2143    2156#   2348    2385#
SWREG   000176           431#    709     2348    2361
SW0     = 000001         366#
SW00    = 000001         356#    366
SW01    = 000002         355#    365
SW02    = 000004         354#    364
SW03    = 000010         353#    363
SW04    = 000020         352#    362
SW05    = 000040         351#    361
SW06    = 000100         350#    360
SW07    = 000200         349#    359
SW08    = 000400         348#    358
SW09    = 001000         347#    357
SW1     = 000002         365#
SW10    = 002000         346#
SW11    = 004000         345#
SW12    = 010000         344#
SW13    = 020000         343#
SW14    = 040000         342#
SW15    = 100000         341#
SW2     = 000004         364#
SW3     = 000010         363#
SW4     = 000020         362#
SW5     = 000040         361#
SW6     = 000100         360#
SW7     = 000200         359#
SW8     = 000400         358#
SW9     = 001000         357#
TBITAE= 000014           399#
TEMP    007012           673#    744     1156#   1643#
```

```
TESTER  001432            442    667#
TITLE   007040            753   1659#
TKVEC =  000060            406#
TPVEC =  000064            407#
TRAPVE=  000034            405#   688#   689#
TRTVEC=  000114            400#
TRYAGN  011016           1445   1471  1844#
TST1    002306            772    788#  1178
TST10   002752            885    890#
TST11   003010            895    90C#
TST12   003062            913#
TST13   003154            929#
TST14   003250            945#
TST15   003304            950    956#
TST16   003342            961    967#
TST17   003414            981#
TST2    002364            813#
TST20   003506            997#
TST21   003602           1014#
TST22   003636           1019   1025#
TST23   003674           1030   1036#
TST24   003746           1050#
TST25   004040           1066#
TST26   004134           1082#
TST27   004316           1108   1113#
TST3    002420            818    824#
TST30   004362           1120   1126#
TST31   004426           1133   1140#
TST32   004464           1146   1152#
TST33   004530           1159   1166#
TST34   004624           1170   1183#
TST35   004646           1186   1191#
TST36   004740           1210#
TST37   005002           1217   1223#
TST4    002456            829    835#
        005044           1230   1236#
        005076           1239   1250#
        005126           1253   1263#
                         1271#
                         1274   1285#
                         1288   1298#
        005250           1306#
        005302           1309   1320#
        002530            848#
TST50   005332           1323   1333#
TST51   005352           1341#
TST52   005404           1344   1355#
TST53   005434           1358   1368#
TST6    002620            864#
TST7    002716            880#
TYPBN =  104406           1419   2609#
TYPOS =  104405           1395   1416  2608#
TYPE  =  104401            750    752   777   783   1393  1396  1414  1417  1429  1434  1444  1460  1470
                          1548   1893  1954  2055  2063  2092  2109  2111  2114  2116  2120  2127  2165
```

```
                    2242    2302    2359    2360    2363    2376    2387    2406    2455    2458    2462
*YPOC = 104402      2100    2124    2362    2605#
TYPON = 104404      2607#
TYPOS = 104403       780    2606#
VADDR   001416       661#    761    1172    1173    1174    1175
V144    007034      1215    1228    1652#
V2034   007036      1225    1653#
VS744   007032      1212    1651#
WFTST   007020       667#    672#    748     775     1185    1412    1646#
$APTHD  001000       466     472#
$ASTAT= ****** U    2549    2564
$ATYC   014170      2520    2522#
$ATY1   014144      2518#
$ATY3   014152      2287    2519#
$ATY4   014162      2068    2521#
$AUTOB  001134       503#   2356    2477
$BASE   001250       567#    733     756
$BDADR  001122       498#
$BDCAT  001126       500#    756#    760     761#    815#    817     827#    828     839#    840     852#    853     855#
                     870#    871     883#    884     893#    894     904#    905     917#    918     920#    935#    936
                     948#    949     959#    960     971#    972     985#    986     988#    1003#   1004    1017#   1018
                    1028#   1029    1040#   1041    1054#   1055    1057#   1072#   1073    1088#   1089    1094#   1095
                    1100#   1101    1106#   1107    1118#   1119    1131#   1132    1145#   1158#   1197#   1198#   1199
                    1539#   1572    1862    1863    1865    1867    1869    1871
$BELL   001164       518#   2055    2084
$BIN    011276      1809#   1892#   1893    1900#
$CDW1   001254       569#
$CHARC  013262      2304#   2314#   2321    2330#   2335#
$CKSUM  013266      2348#   2613
$CMTAG  001100       486#    677     678     686     692     693
$CM3  = 000000       516#
$CNTLC  014013      2359    2472#
$CNTLU  014006      2376    2471#
$CPUOP  001222       541#
$CRLF   001171       520#   2063    2084    2092    2111    2116    2120    2303    2338    2387    2471
$DBLK   011514      1925    1959    1967#
$DEVCT  001204       532#    792#   1171#
$DEVM   001252       568#
$DORGN  005552      1389    1394    1404#
$DTBL   011504      1928    1963#
$ENDAD  005542       452    1400#
$ENDCT  005510      1391#
$ENDMG  005561      1393    1408#
$ENULL  005556      1396    1407#
$ENV    001214       537#    763    2065    2282    2527    255.
$ENVM   001215       538#    714    2284    2289    2529
$EOP    005454      1187    1381#
$EOPCT  005502      1388#   1392
$ERFLG  001103       489#   1973    2003    2005    2011#   2033    2050#   2084
$ERMAX  001115       495#    694#   2005    2028#   2033
$ERROR  012006       686    2047#
$ERRPC  001116       496#   1862    1863    1865    1867    1869    1871    2057#   2058#   2059    2084    2098
$ERRTB  001256       586#   2106
$ERRTY  012202      2062    2091#
```

```
$ERTTL  001112          493#  1415   2056*  2084
$ESCAP  001162          517#   693*  2027*  2079   2081   2084
$ETABL  001214          536#
$ETEND  001256          478    570#
$FATAL  001176          529#  2555*
$FFLG   014410         2518*  2521*  2549   2558*  2566#
$FILLC  001156          514#  2507   2338
$FILLS  001155          513#  2338
$GDADR  001120          497#
$GDDAT  001124          499#   814*   815    817    825*   826    828    837*   838    840    843*   850*   851
                        853    857    859*   866*   867    868*   871    874    875    881*   882    884    891*
                        892    894    902*   903    905    908*   915*   916    918    922    924*   931*   932
                        933*   936    939    940    946*   947    949    957*   958    960    969*   970    972
                        975*   983*   984    986    990    992*   999*  1000   1001*  1004   1007   1008   1015*
                       1016   1018   1026*  1027   1029   1038*  1039   1041   1044*  1052*  1053   1055   1059
                       1061*  1068*  1069   1070*  1073   1076   1077   1087*  1089   1093*  1095   1099*  1101
                       1105*  1107   1116*  1119   1129*  1132   1143*  1155*  1194*  1199   1203*  1212*  1225*
                       1437*  1463*  1483*  1493*  1571   1863   1865   1867   1869   1871
$GET42  005532          1397#
$GTSWR  013336          2360#  2611
$HD     = 000000         299
$HIBTS  001600          473#
$HIOCT  014142         2509*  2514#
$ICNT   001124          490#  2018*  2019   2021*  2032
$ILLUP  012502         2135   2151   2170#
$INTAG  001135          504#  2388   2477
$ITEMB  001114          495#  2059*  2067   2084   2095
$LF     001172          521#  2084   2338   2462   2471
$LFLG   014407         2559*  2565#
$LPADR  001106          491#   695*  2009*  2025*  2030   2032
$LPERR  001110          492#   696*  2009   2026*  2032   2078
$MADR1  001226          554#
$MADR2  001232          558#
$MADR3  001236          561#
$MADR4  001242          564#
$MAIL   001174          474    478    527#   713    2024   2065   2282
$MAMS1  001224          548#
$MAMS2  001230          556#
$MAMS3  001234          559#
$MAMS4  001240          562#
$MBADR  001002          474#
$MFLG   014406         2519*  2525   2560*  2564#
$MNEW   014031         2363   2475#
$MSGAD  001210          534*  2535*  2538
$MSGLG  001212          535#  2540*
$MSGTY  001174          528#  2533   2541*  2553   2557*
$MSWR   014020         2360   2473#
$MTYP1  001225          549#
$MTYP2  001231          557#
$MTYP3  001235          560#
$MTYP4  001241          563#
$MXCNT  012004         2022   2032#
$NULL   001154          512#  2309   2338
$NWTST= 000001          795#   810#   821#   832#   845#   861#   877#   887#   897#   910#   926#   942#   953#
```

```
                        964#  978#  994#  1011#  1022#  1033#  1047#  1063#  1079#  1110#  1123#  1137#  1149#
                       1163#  1180#  1188#  1207#  1220#  1233#  1247#  1260#  1268#  1282#  1295#  1303#  1317#
                       1330#  1338#  1352#  1365#
$OCNT   013000         2214#  2243#  2256#
$OMODE  013C02         2209#  2213#  2218   2221#  2232#  2258#
$OVER   011770         1986   2002   2010   2020   2029#
$PASS   001202          531#   713#  1238   1252   1273   1287   1308   1322   1343   1357   1385#  1386#  1394
                       1407   2016   2033
$PASTM  001006          476#
$PWRAD  012476         2168#
$PWRDN  012336          690   2135#  2163
$PWRNG  012472         2166#
$PWRUP  012410         2145   2151#
$QUES   001170          519#  2084   2338   2403   2455   2471
$RDCHR  013550         2419#  2614
$RDDEC= ****** U       2617
$RDLIN  013670         2447#  2615
$RDOCT  014042         2487#  2616
$RUSZ = 000010         2440#
$PTRAD  005554         1406#
$R2A  = ****** U       2617
$SAVRE= ****** U       2617
$SAVR6  012506         2144#  2152   2153#  2154#  2172#
$SCOPE  011524          684   1982#
$SETUP= 000117          588#   676#   683   684   686   688   690   692   693   695   1383   1983   2048
                       2075   2083   2343   2477
$STUP = 177777          588#   676#
$SVLAD  011734         1994   2023#
$SVPC = 000106          450#   455
$SWR  = 167400          289#   299   416   417   418   419   420   421   422   516   517   518   692
                        693   695   696   799   814   825   836   849   865   881   891   901   914
                        930   946   957   968   982   998  1015  1026  1037  1051  1067  1083  1114
                       1127  1141  1153  1167  1184  1192  1211  1224  1237  1251  1264  1272  1286
                       1299  1307  1321  1334  1342  1356  1369  1378  1384  1399  1405  1407  1974
                       1975  1976  1977  1978  1986  1997  1999  2000  2003  2004  2005  2012  2013
                       2014  2026  2029  2032  2039  2040  2041  2042  2043  2053  2060  2072  2076
                       2084  2169
$SWREG  001216          539#   716
$SWRMK= 000000          422   423  1978  1979  2001
$TEMP   007014         1193#  1196  1204#  1644#
$TESTN  001200          530#  2024#
$TIMES  001160          516#   692#   836#   849#   865#   901#   914#   930#   968#   982#   998#  1037#  1051#
                       1067#  1114#  1127#  1141#  1153#  1167#  1184#  1192#  1211#  1224#  1237#  1251#  1264#
                       1272#  1286#  1299#  1307#  1321#  1334#  1342#  1356#  1369#  1384#  2012#  2019   2022#
                       2032
$TKB    001146          509#  1559   2341   2352   2369   2423   2429
$TKS    001144          508#  1551   2341   2350   2366   2390#  2422   2427
$TN   = 000054          289#   299   772   795   799#   810   814#   818   821   825#   829   832   836#
                        845   849#   861   865#   877   881#   885   887   891#   895   897   901#   910
                        914#   926   930#   942   946#   950   953   957#   961   964   968#   978   982#
                        994   998#  1011  1015#  1019  1022  1026#  1030  1033  1037#  1047  1051#  1063
                       1067#  1079  1083#  1108  1110  1114#  1120  1123  1127#  1133  1137  1141#  1146
                       1149  1153#  1159  1163  1167#  1170  1180  1184#  1186  1188  1192#  1207  1211#
                       1217  1220  1224#  1230  1233  1237#  1239  1247  1251#  1253  1260  1264#  1268
```

CROSS REFERENCE TABLE

```
                        1272#   1274    1282    1286#   1288    1295    1239#   1303    1307#   1319    1317    1321#   1323
                        1330    1334#   1338    1342#   1344    1352    1356#   1358    1365    1379#
$TPB    001152           511#   2327*   2338
$TPFLG  001157           515#   2276    2338
$TPS    001150           510#   2325    2338
$TRAP   014412           688    2581#
$TRAP2  014434          2592#   2603
$TRP  = 000014          2536#   2605#   2606#   2607#   2608#   2609#   2610#   2611    2612#   2613    2614#   2615#   2616#
                        2617#
$TRPAD  014446          2586    2603#
$TSTM   001004           475#
$TSTMM  001102           488#   1177*   1383*   1973    2001    2023*   2024    2029    2033    2052    2084
$TYYIN  013776          2448    2449    2466    2470#
$TYPBM  011224          1886#   2609
$TYPOS  011300          1913#   2608
$TYPE   013004          2276#   2546    2596    2604
$TYPEC  013216          2306    2313    2320    2325#   2326    2392
$TYPEX  013264          2331    2333    2336#
$TYPOC  012602          2212#   2605
$TYPON  012616          2211    2214#   2607
$TYPOS  012556          2207#   2606
$UNIT   001206           533#    758*    762*    765     769     779     785     788     791*   1168*   1169
$UNITM  001010           477#
$USWR   001220           540#
$VECT1  001234           565#
$VECT2  001246           566#
$XTSTR  011540          1988#
$$GET4= 000000          1399#
$OFILL  013001          2208*   2212*   2222    2257#
$40CAT= ******  U       1905    2062
.     = 014476           425#    429#    438#    441#    444#    450     451#    453#    455#    456#    462     463#    465#
                         467#    485#    522     661     695     696    1407    1411    1967#   2032    2033    2084    2130#
                        2147    2171    2338    2341    2470#   2471    2477    2567#
.$ASTA= ******  U       2519    2522
.$$   = 001000           462#    467
```

```
ADJR      1791#  1796  1801  1806  1811  1816  1821  1826
COMMEN       1#  4098#
DYNIC     667#   845   910   978  1047
ENDCOM       1#  4098#
ERROR     303#   774   790   806   819   830   842   856   873   886   896   907   921   938   954
          962    974   989  1006  1020  1031  1043  1056  1075  1091  1097  1103  1109  1121  1134
         1147   1160  1201  1218  1231  1443  1469  1491  1557
ESCAPE       1#  4098#
GETPRI       1#  4098#
GETSWR       1#  4098#
MULT         1#  4098#
NEWTST       1#  4098#   795   810   821   832   845   861   877   887   897   910   926   942   958
          964    978   994  1011  1022  1033  1047  1063  1079  1110  1123  1137  1149  1163  1180
         1188   1207  1220  1233  1247  1260  1268  1282  1295  1303  1317  1330  1338  1352  1365
POP          1#  4098#  1954  2156  2157  2510  2561  2562
PUSH         1#  4098#  1913  2137  2143  2489  2522  2524  2545
REPORT       1#  4098#
SCOPE     304#   798   813   824   835   848   864   880   890   900   913   929   945   956   967
          981    997  1014  1025  1036  1050  1066  1082  1113  1126  1140  1152  1166  1183  1191
         1210   1223  1236  1250  1263  1271  1285  1298  1306  1320  1333  1341  1355  1368  1382
SELD      1832#  1835  1838  1841
SETPRI       1#  4098#
SETTRA    2596#  2605  2606  2607  2608  2609  2611  2613  2614  2615  2616
SETUP        1#  4098#   676
SKIP         1#  4098#   767   772   787   789   804   818   829   841   854   872   876   885   895
          906    919   937   941   950   961   973   987  1005  1009  1019  1030  1042  1056  1074
         1078   1090  1096  1102  1108  1120  1133  1146  1159  1170  1186  1200  1217  1230  1239
         1253   1274  1288  1309  1323  1344  1358  1442  1468  1490  1494
SLASH        1#  4098#
SPACE     409#
STARS        1#  4098#   448   459   461   468   481   522   525   795   797   810   812   821   823
          832    834   845   847   861   863   877   879   887   889   897   899   910   912   926
          928    942   944   953   955   964   966   978   980   994   996  1011  1013  1022  1024
         1033   1035  1047  1049  1063  1065  1079  1081  1110  1112  1123  1125  1137  1139  1149
         1151   1163  1165  1180  1182  1188  1190  1207  1209  1220  1222  1233  1235  1247  1249
         1260   1262  1268  1270  1282  1284  1295  1297  1303  1305  1317  1319  1330  1332  1338
         1340   1352  1354  1365  1367  1375  1879  1903  1970  2035  2086  2133  2149  2184  2261
         2340   2343  2411  2440  2479  2517  2575
SUBTST    667#   861   926   994  1063
SUPER    1180#  1233  1268  1303  1338
SWRSU        1#  4098#  6978
TRAPTRP   2596#
TYPBIN       1#  4098#  1418
TYPDEC       1#  4098#  1394
TYPNAM       1#  4098#
TYPNUM       1#  4098#
TYPOCS       1#  4098#
TYPOCT       1#  4098#  2098  2122  2361
TYPTXT       1#  4098#
$$CMRE    479#
$$CMTM    479#
$$ESCA       1#  4098#
$$NEWT       1#  4098#   795   810   821   832   845   861   877   887   897   910   926   942   953
          964    978   994  1011  1022  1033  1047  1063  1079  1110  1123  1137  1149  1163  1180
```

D06


```
        1188    1207    1220    1233    1247    1260    1268    1282    1295    1303    1317    1330    1338    1352    1365
$$SET   2596#   2605    2606    2607    2608    2609    2611    2613    2614    2615    2616
$$SETM   713#
$$SWIP     1#    409#    772     818     829     885     895     950     961    1019    1030    1108    1120    1133    1146
        1159    1170    1186    1217    1230    1239    1253    1274    1288    1309    1323    1344    1358
.EQUAT     1#    289#    299
.HEADE     1#    289#
.KT11      1#
.SETUP     1#    289#    508     676
.SWAHI     1#    289#    412
.SWLO    423#
.SYCTI     1#    289#    446
.SYSTB     1#    289#    523#
.SYPTH     1#    289#    457
.SYPTY     1#    289#   2515
.SYSTA     1#
.SCATC     1#    289#    423
.SCFIA     1#    289#    479
.SDB2D     1#
.SDB2O     1#
.SDIV      1#
.SEOP      1#    289#   1373
.SERRO     1#    289#   2033
.SETRT     1#    289#   2084
.SMULT     1#
.SPARM   289#
.SPOWE     1#    289#   2131
.SRAND     1#
.SRDOE     1#
.SRDOC     1#    289#   2477
.SREAD     1#    289#   2338
.SR2AZ     1#
.SSAVE     1#    289#
.SSB2D     1#
.SSB2O     1#
.SSCOP     1#    289#   1958
.SSIZE     1#
.SSPAC   289#
.SSUPR     1#
.SSWDC   289#
.STRAP     1#    289#   2573
.STYPB     1#    289#   1877
.STYPO     1#    289#   1901
.STYPE     1#    289#   2259
.STYPO     1#    289#   2182
.$4OCA     1#
.1170      1#
```

DVAAAA.P.. CROSS REFERENCE TABLE

ADCB	1892														
ADC	738	739	740	76.	1172	1173	1174	1175	1572	1580	1602	1933	2106	2210	2220
	2295	2377	2386	2505	2532	2544	2556								
	1176	2103	2104	2105	2400	2401	2402	2498	2500	2502	2585				
	1938														
	847	858	859	908	923	924	975	991	992	1044	1060	1061	1203	1204	1523
		538	2539												
		743	789	816	829	841	854	872	885	893	906	919	937	950	961
		987	1005	1019	1030	1042	1056	1074	1090	1096	1102	1108	1120	1133	1146
		1.00	1200	1398	1503	1891	2000	2002	2004	2008	2017	2051	2054	2077	2080
		21.3	2126	2237	2285	2298	2333	2357	2384	2399	2497	2526	2530	2550	2552
		384	1947	2244	2396	2437									
		306													
BIT	1198	386	1512	1525	1560	2234	2353	2370	2397	2424	2430	2438	2504		
BIS	796	1508	1941	1942	2049	2239	2240	2404							
BISB	1516	2095													
BIT	1985	1999	2007	2014	2053	2060	2076								
BITB	714	2284	2289	2321	2529										
BLE	1577														
BLOS	2450														
BLT	1930	1946	2245	2312	2394	2435									
BMI	764	774	1552	1637											
BNE	681	704	729	745	747	766	770	776	844	860	876	909	925	941	976
	993	1009	1045	1062	1078	1186	1205	1239	1253	1274	1288	1309	1323	1344	1358
	1413	1494	1507	1511	1518	1534	1554	1556	1562	1604	1935	1986	2015	2061	2066
	2096	2118	2155	2235	2283	2290	2292	2300	2308	2322	2329	2349	2355	2375	2382
	2389	2426	2432	2454	2460	2528	2534	2537	2554						
BPL	1531	1574	1636	1921	1951	2073	2233	2277	2326	2351	2367	2422	2428		
BP	668	670	706	767	772	787	804	1217	1230	1442	1446	1468	1472	1490	1513
	1558	1599	1617	1628	1895	1932	1949	1988	1994	1997	2010	2013	2071	2101	2128
	2147	2171	2211	2226	2247	2274	2305	2315	2324	2331	2378	2405	2407	2433	2456
	2506	2520	2542												
CCC	1535														
CLC	1894														
CLR	672	673	674	679	692	693	713	718	725	757	758	791	792	814	881
	946	1015	1116	1129	1143	1155	1177	1383	1384	1527	1550	1601	1626	1924	1927
	2012	2027	2094	2153	2224	2264	2365	2494	2495						
CLRB	1953	2011	2304	2330	2461	2558	2559	2560							
CMP	680	703	728	765	768	805	817	828	840	853	871	884	894	905	918
	936	949	960	972	986	1004	1018	1029	1041	1055	1073	1089	1095	1101	1107
	1115	1132	1199	1561	1576	1945	1995	2019	2348	2354	2374	2381	2393	2395	2425
	2431	2434	2436	2449											
CMPB	768	1169	2001	2005	2065	2282	2297	2299	2307	2328	2332	2356	2388	2453	2459
	2527														
DEC	1387	1506	1510	1517	1553	1555	1635	2102							
DECB	2232	2243	2311	2314											
EMT	303														
HALT	429	2074	2146	2170	2278										
INC	667	762	1168	1171	1385	1931	2018	2056	2154	2238	2246	2403	2557		
INCB	1525	2023	2050	2334											
IOT	304														
JMP	433	434	435	436	439	442	719	808	1178	1187	1405	1420			

CROSS REFERENCE TABLE

```
ASR      743   793  1213  1216  1226  1229  1241  1255  1265  1276  1290  1300  1311  1325  1335
        1346  1360  1370  1460  1431  1436  1438  1441  1457  1462  1464  1467  1485  1489  1590
        1592  1594  1596  1598  1610  1622  1625  1627  2062  2068  2287  2306  2313  2320  2392
        2546
ASR      669   671   678   682   684   685   686   687   688   689   690   691   695   696   699
         700   701   702   707   709   710   711   716   722   723   724   726   733   734   735
         736   737   756   759   779   785   794   799   807   809   815   816   825   826   827
         836   837   838   839   849   850   851   852   855   857   865   866   867   870   874
         882   883   891   892   893   901   902   903   904   914   915   916   917   920   922
         930   931   932   935   939   947   948   957   958   959   968   969   970   971   982
         983   984   985   988   990   998   999  1000  1003  1007  1016  1017  1026  1027  1028
        1037  1038  1039  1040  1051  1052  1053  1054  1057  1059  1067  1068  1069  1072  1076
        1083  1084  1085  1086  1087  1088  1093  1094  1099  1100  1105  1106  1114  1115  1118
        1127  1128  1131  1141  1142  1145  1153  1154  1156  1158  1167  1134  1192  1193  1194
        1196  1197  1211  1212  1215  1224  1225  1228  1237  1251  1264  1272  1286  1299  1307
        1321  1334  1342  1356  1369  1390  1394  1397  1415  1418  1423  1424  1425  1426  1427
        1428  1433  1437  1440  1452  1453  1454  1455  1456  1459  1463  1466  1477  1478  1479
        1480  1482  1483  1488  1501  1505  1509  1515  1523  1526  1528  1539  1549  1559  1569
        1570  1571  1572  1578  1579  1583  1584  1587  1591  1593  1595  1597  1609  1612  1613
        1614  1615  1616  1621  1624  1630  1631  1634  1633  1634  1886  1887  1896  1897  1898
        1914  1915  1916  1917  1918  1919  1920  1925  1928  1948  1954  1955  1956  1957  1958
        1960  1962  1990  1991  1993  1996  2009  2021  2022  2025  2026  2029  2030  2032  2057
        2078  2081  2093  2098  2107  2112  2117  2119  2123  2135  2136  2137  2138  2139  2140
        2141  2142  2143  2144  2145  2151  2152  2156  2157  2158  2159  2160  2161  2162  2163
        2164  2167  2207  2215  2216  2217  2223  2230  2248  2249  2250  2251  2252  2280  2281
        2286  2294  2305  2361  2385  2390  2419  2420  2447  2448  2463  2464  2465  2466  2487
        2488  2489  2490  2491  2493  2508  2509  2510  2511  2512  2523  2524  2531  2535  2540
        2541  2543  2545  2555  2561  2579  2581  2582  2586  2592  2593
MOVB     694  1502  1504  1869  1923  1925  1940  1943  1952  2024  2028  2059  2067  2208  2209
        2212  2213  2214  2218  2221  2222  2241  2291  2319  2327  2352  2369  2423  2429  2452
        2457  2496  2518  2519  2521  2584
NEG     1575  1922  2219
NOP     1401  1402  1403
RESET    675  1117  1130  1144  1157  1399
ROL     1890  2225  2227  2228  2229  2231  2499  2501  2503
ROR     1536
RTI      708  1899  1962  2031  2063  2169  2253  2296  2391  2439  2467  2513  2594
RTS      741  1447  1473  1495  1519  1540  1563  1581  1585  1605  1437  2121  2336  2563  2587
SEC     1888
SUB      868   869   933   934  1001  1002  1070  1071  1492  1493  1573  1929  2058  2538
SWAB    1524
TRAP    2596  2605  2606  2607  2608  2609  2611  2613  2614  2615  2616
TST      727   744   746   748   760   763   769   773   775   800   801   802   803   875   940
        1008  1077  1185  1238  1252  1273  1287  1308  1322  1343  1357  1412  1603  1934  1944
        1982  2016  2072  2079  2125  2236  2293  2301  2323  2383  2398  2507  2533  2551  2553
        2583
TSTB    1530  1551  1936  1950  2003  2276  2325  2350  2366  2421  2427  2525  2536  2549
.ASCII   519   520  1746  1753  1854  1858
.ASCIZ   518   521  1408  1659  1668  1676  1680  1684  1688  1692  1701  1709  1716  1721  1726
        1733  1740  1759  1766  1768  1771  1774  1777  1782  1785  1791  1796  1801  1809  1811
        1816  1821  1826  1832  1835  1838  1841  1844  2129  2173  2471  2472  2473  2475
.BLKB   2470
.BLKW   1967
.BYTE    488   489   494   495   503   504   512   513   514   515   537   538   548   549   556
```

DVAAAA.P11 CROSS REFERENCE TABLE

```
                1385    1997    1999    2000    2001    2003    2004    2005    2014    2016    2024    2026    2031    2032    2033
                2035    2038    2049    2053    2060    2062    2063    2065    2072    2076    2083    2084    2086    2101    2117
                2133    2143    2144    2149    2156    2157    2165    2167    2169    2173    2184    2261    2282    2340    2342
                2343    2344    2372    2411    2412    2440    2448    2449    2453    2454    2470    2471    2477    2479    2482
                2494    2517    2519    2522    2549    2564    2575    2581    2585    2596    2605    2606    2607    2608    2609
                2611    2613    2614    2615    2616    2617
.IFF             301     419     421     422     423     449     453     455     460     462     469     482     485     488     516
                 523     526     682     767     773     787     789     796     797     798     799     804     811     812     813
                 814     819     822     823     824     825     830     833     834     835     836     841     846     847     848
                 849     854     862     863     864     865     872     876     878     879     880     881     886     888     889
                 890     891     896     898     899     900     901     906     911     912     913     914     919     927     929
                 929     930     937     941     943     944     945     946     951     954     955     956     957     962     965
                 966     967     968     973     979     98[illegible]     981     982     987     995     996     997     998    1005    1009
                1012    1013    1014    1015    1020    1023    1024    1025    1026    1031    1034    1035    1036    1037    1042
                1048    1049    1050    1051    1056    1064    1065    1066    1067    1074    1078    1080    1081    1082    1083
                1090    1096    1102    1109    1111    1112    1113    1114    1121    1124    1125    1126    1127    1134    1138
                1139    1140    1141    1147    1150    1151    1152    1153    1160    1164    1165    1166    1167    1168    1171
                1181    1182    1183    1184    1185    1187    1189    1190    1191    1192    1193    1200    1208    1209    1210
                1211    1212    1218    1221    1222    1223    1224    1225    1231    1234    1235    1236    1237    1238    1240
                1248    1249    1250    1251    1252    1254    1261    1262    1263    1264    1265    1269    1270    1271    1272
                1273    1275    1283    1284    1285    1286    1287    1289    1296    1297    1298    1299    1300    1304    1305
                1306    1307    1308    1310    1318    1319    1320    1321    1322    1324    1331    1332    1333    1334    1335
                1339    1340    1341    1342    1343    1345    1353    1354    1355    1356    1357    1359    1366    1367    1368
                1369    1370    1376    1379    1383    1389    1392    1407    1442    1468    1490    1494    1880    1904    1971
                1998    2021    2006    2008    2032    2033    2036    2038    2053    2083    2084    2087    2102    2131    2134
                2150    2161    2185    2263    2341    2344    2412    2414    2419    2440    2441    2450    2454    2471    2480
                2516    2576    2582
.IFT            2013    2063    2414    2419    2498    2514    2515
.IFTF           2011    2062    2359    2412    2415    2494    2498    2514
.IIF             289     294     299     416     417     418     419     422     423     429     522     526     683     686     692
                 693     695     698    1377    1383    1384    1395    1407    1411    1419    1974    1975    1976    1977    1978
                1979    1983    2012    2013    2029    2032    2033    2039    2040    2041    2042    2043    2048    2075    2083
                2084    2099    2124    2338    2341    2362    2463    2471    2477    2604    2605    2606    2607    2608    2609
                2611    2613    2614    2615    2616
.IRP             588     676     795     810     821     832     845     861     877     887     897     910     926     942     953
                 964     978     994    1011    1022    1033    1047    1063    1079    1110    1123    1137    1149    1163    1180
                1188    1207    1220    1233    1247    1260    1268    1282    1295    1303    1317    1330    1338    1352    1365
                1914    1954    1984    2049    2137    2143    2156    2157    2489    2510    2523    2524    2545    2561    2562
.LIST              1     289     409     422     429     516     523     526     588     676     697     742     795     799     810
                 814     821     825     832     836     845     849     861     865     877     881     887     891     897     901
                 910     914     926     930     942     946     953     957     964     968     978     982     994     998    1011
                1015    1022    1026    1033    1037    1047    1051    1063    1067    1079    1083    1110    1114    1123    1127
                1137    1141    1149    1153    1163    1167    1180    1184    1188    1192    1207    1211    1220    1224    1233
                1237    1247    1251    1260    1264    1268    1272    1282    1286    1295    1299    1303    1307    1317    1321
                1330    1334    1338    1342    1352    1356    1365    1369    1383    1399    1587    1655    1978    2083    2440
                2596    2604    2605    2606    2607    2608    2609    2610    2611    2612    2613    2614    2615    2616    2617
.MACRO             1     423     479     667     713    1180    1791    1832    2596
.MCALL           289     409     523     597
.MEXIT           571
.NLIST             1     289     409     422     429     516     523     526     588     676     697     742     795     799     810
                 814     821     825     832     836     845     849     861     865     877     881     887     891     897     901
                 910     914     926     930     942     946     953     957     964     968     978     982     994     998    1011
                1015    1022    1026    1033    1037    1047    1051    1063    1067    1079    1083    1110    1114    1123    1127
                1137    1141    1149    1153    1163    1167    1180    1184    1188    1192    1207    1211    1220    1224    1233
```

MAINDEC-11-DVAAA-A AAV11 DIAGNOSTIC MACY11 27(665) 12-OCT-76 13:42 PAGE 73
DVAAAA.P11 CROSS REFERENCE TABLE

ERRORS DETECTED: 0

#,DVAAAA.SEQ/SOL/CRF/NL:TOC/DS:ERFZ=DVAAAA.SML,DVAAAA.P11
RUN-TIME: 54 64 6 SECONDS
CORE USED: 33K