AR11

WRAPAROUND TEST 2

MD-11-DZARC-B

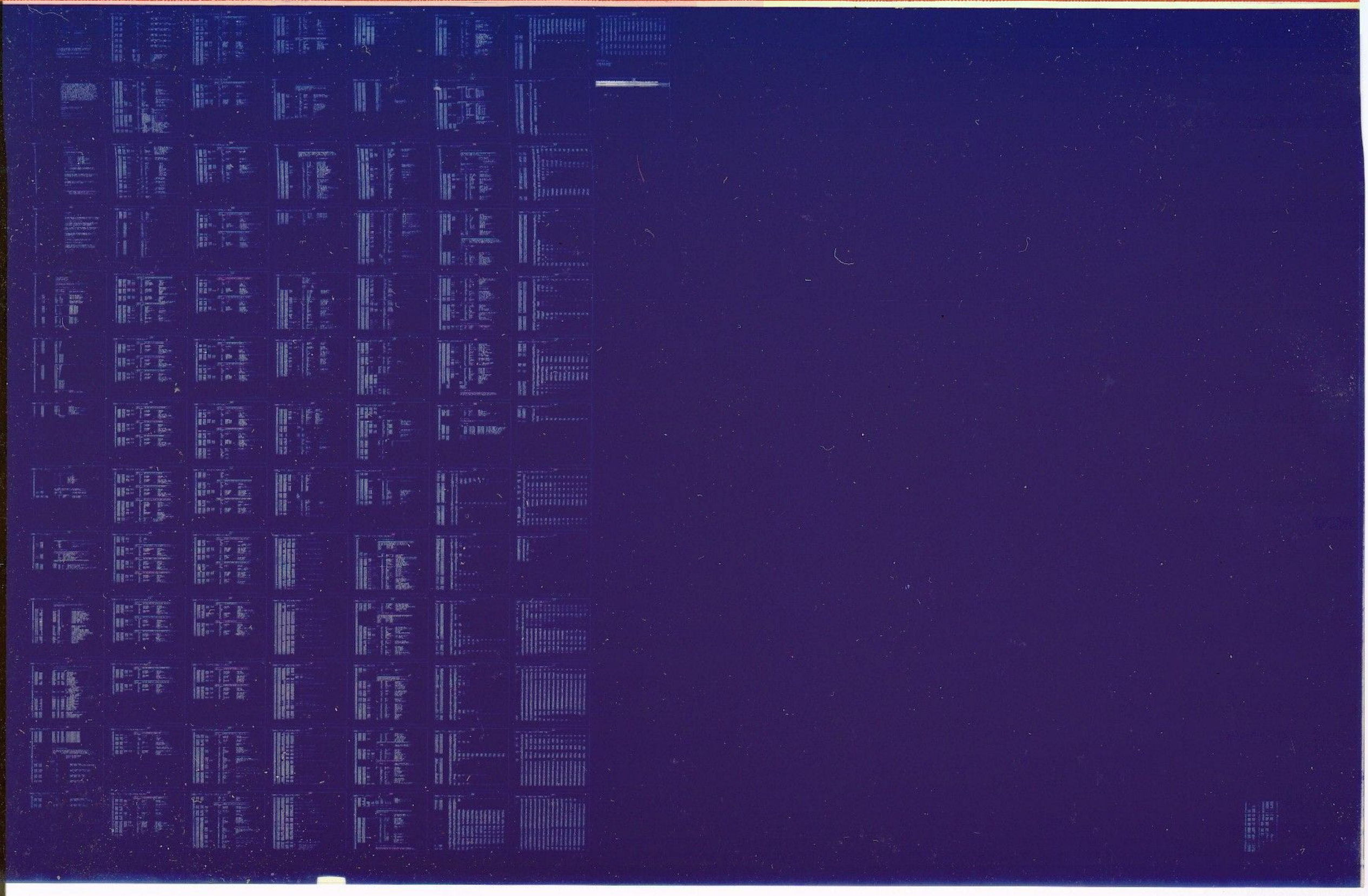
EP-DZARC-B-DL-A

FICHE 1 OF 1

NOV 1976

digital

MADE IN USA



.REM %

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZARC-B
PRODUCT NAME:	AR11 DIAGNOSTIC TEST III (WRAPAROUND TEST)
DATE:	MAY 21, 1976
MAINTAINER:	DIAGNOSTIC GROUP

1. ABSTRACT

THE WRAPAROUND DIAGNOSTIC ENABLES THE AR11 TO TEST ITSELF, IN CONJUCTION WITH A G5036 WRAPAROUND MODULE, WHICH IS CONNECTED TO THE AR11 THROUGH A BC08-R CABLE. THE TWO D/A CONVERTERS ARE COMBINED IN VARIOUS MANNERS USING RESISTIVE DIVIDERS ON THE G5036 MODULE AND SENT BACK INTO THE A/D INPUTS. CONSEQUENTLY, BOTH X AND Y D/A'S ARE TESTED BY THE A/D CONVERTER, AND THE A/D IS TESTED BY THE D/A CONVERTERS. BECAUSE THE D/A'S ARE DIVIDED DOWN BEFORE GOING INTO THE A/D, TEST RESOLUTION FAR BETTER THAN 1 LSB IS ATTAINED. IN ADDITION, THE ANALOG POWER SUPPLY LEVELS (+/- 14 VOLTS) ARE DIVIDED DOWN AND USED AS INPUTS, ALLOWING TEST OF THE DC-TO-DC CONVERTER; THE FOUR SCOPE CONTROL LOGIC OUTPUTS (ERASE L, NON-STORE L, CHD2 L, AND WRITE-THRU L) ARE USED AS INPUTS, ALLOWING PARAMETRIC TESTS OF THEIR HIGH AND LOW OUTPUT LEVELS; THE CHD2 L OUTPUT IS FED BACK INTO THE ERASE RET INPUT, ALLOWING TEST OF THE STORAGE SCOPE ERASE RETURN HANDSHAKING LOGIC; AND THE SCOPE INTENSIFY OUTPUT IS FED BACK INTO THE A/D EXTERNAL START INPUT, ALLOWING TESTS OF BOTH FUNCTIONS. SINCE ALL A/D INPUT CHANNELS ARE USED (AT DIFFERENT INPUT VOLTAGE LEVELS), THE WRAPAROUND MODULE ALSO PROVIDES A COMPLETE FUNCTIONAL CHECK ON THE A/D INPUT MULTIPLEXER.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 FAMILY COMPUTER WITH 8K WORDS OF MEMORY
AR11 HEX OPTION MODULE INSTALLED
G5036 WRAPAROUND MODULE AND BC08-R CABLE
TELETYPE

2.2 STORAGE

THIS PROGRAM USES LESS THAN 8K OF MEMORY.

3. LOADING PROCEDURE

PROCEDURE FOR NORMAL BINARY TAPES SHOULD BE FOLLOWED.

```
                                4.      STARTING PROCEDURE
                                        -----------------

                                4.1     CONTROL SWITCH SETTINGS

                                        THIS PROGRAM HAS BEEN MODIFIED TO RUN WITH OR
                                        WITHOUT A HARDWARE SWITCH REGISTER.

                                        STANDARD PDP-11 FORMAT

                                                SW 15 = 1       HALT ON ERROR
                                                SW 14 = 1       LOOP ON TEST
                                                SW 13 = 1       INHIBIT ERROR TYPEOUTS
                                                SW 12 = 1       FORCED PRINTOUT ENABLE
                                                SW 11 = 1       INHIBIT ITERATIONS
                                                SW 10 = 1       BELL ON ERROR
                                                SW 09 = 1       LOOP ON ERROR
                                                SW 08 = 1       LOOP ON TEST IN SWR <7:0>
                                        REFER TO 9. FOR SOFTWARE SWITCH REGISTER CONTROL

                                4.2     STARTING ADDRESS OR ADDRESSES

                                        200     IS THE STARTING ADDRESS OF THE WRAPAROUND TEST.
                                        204     IS THE RESTART ADDRESS OF THE WRAPAROUND TEST.
                                        210     IS THE STARTING ADDRESS FOR THE OPTION TEST AREA.

                                5.      OPERATING PROCEDURE
                                        -------------------

                                        THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH
                                        REGISTER WHEN REQUESTED BY THE PROGRAM OR AN ERROR WILL OCCUR.
                                        ONCE STARTED THE TEST WILL RUN IN ITS NORMAL MANNER
                                        WITHOUT OPERATOR INTERVENTION OR SWITCH SELECTION.

                                6.      ERRORS
                                        ------

                                        THIS PROGRAM USES THE DIAGNOSTIC 'SYSMAC' PACKAGE FOR
                                        ERROR REPORTING AND TYPEOUT. REFER TO THE "ERROR POINTER TABLE"
                                        FOR TYPE AND DESCRIPTION OF ERRORS.

                                7.      RESTRICTIONS
                                        ------------

                                        SWITCH ON G5036 MUST BE IN THE '0' POSITION.


                                        ************************************************************
                                        THE WRAPAROUND (G5036) MODULE MUST BE CONNECTED AS FOLLOWS:
                                                AR11 TO BC08R CONNECTION A-A, VV-VV
                                                BC08R TO G5036 CONNECTION "UPSIDE-DOWN" A-VV, VV-A
                                        ************************************************************
```

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE FIRST PASS WILL TAKE APPROXIMATELY 45 SECONDS FOR COMPLETION.
ADDITIONAL PASSES WILL TAKE APPROXIMATELY 4 MIN. FOR COMPLETION.
THE END OF A PASS IS INDICATED BY THE 'END PASS #X' MESSAGE.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS

LOCATION $BASE CONTAINS THE AR11 BASE DEVICE ADDRESS <170400>
LOCATION $VECT1 CONTAINS THE AR11 BASE INTERRUPT VECTOR <340>
LOCATION $FILLS CONTAINS THE TTY FILLER CHARACTER COUNT
LOCATION $NULL CONTAINS THE TTY FILLER CHARACTER

*NOTE: IF THESE LOCATIONS ARE CHANGED, THE OPERATOR MUST START
THE TEST AGAIN AT LOC. 200. THE PROGRAM WILL USE THE BASE
ADDRESS AND VECTOR AND UPDATE THE ACTUAL PROGRAM VALUES.

8.3 MULTIPLE AR-11 TESTING

A PROVISION IS MADE FOR TESTING SEQUENTIAL AR-11. STARTING
AT BUS ADDRESS/VECTOR DEFINED BY $BASE AND $VECT1.
THE HEADER TYPEOUT WILL INFORM THE OPERATOR OF THE NUMBER
OF AR-11'S FOUND.

8.4 XXDP/ACT/APT NOTES

THIS PROGRAM IS A CHAINABLE PROGRAM UNDER XXDP/ACT.
THE APT HOOKS HAVE BEEN INSTALLED BUT NOT TESTED.

9. SOFTWARE SWITCH REGISTER OPERATION

THE PROGRAM SUPPORTS NON-SWITCH REGISTER CPU TYPES.
A CHANGE IN SWR VALUE IS ACCOMPLISHED BY TYPING A "CTRL G".
THE RESPONSE WILL BE "SWR = " AND WAIT FOR A NEW VALUE.
THE OPERATOR NOW INPUTS THE NEW VALUE AND TERMINATES WITH A "CR".

10. TABLE OF CONTENTS

ATTACHED

```
                                        %
                                .TITLE  MAINDEC-11-DZARC-B
                                ;*COPYRIGHT (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-B2),NOV 21, 1975.
                                ;*

                                .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
                000100          SW06=   100
                000040          SW05=   40
                000020          SW04=   20
                000010          SW03=   10
                000004          SW02=   4
                000002          SW01=   2
                000001          SW00=   1
                                .EQUIV  SW09,SW9
                                .EQUIV  SW08,SW8
                                .EQUIV  SW07,SW7
                                .EQUIV  SW06,SW6
                                .EQUIV  SW05,SW5
                                .EQUIV  SW04,SW4
                                .EQUIV  SW03,SW3
                                .EQUIV  SW02,SW2
                                .EQUIV  SW01,SW1
                                .EQUIV  SW00,SW0

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

                                ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
```

```
                                TBITVEC=14              ;;"T" BIT
                                TRTVEC= 14              ;;TRACE TRAP
                                BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                                IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                                PWRVEC= 24              ;;POWER FAIL
                                EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                                TRAPVEC=34              ;;"TRAP" TRAP
                                TKVEC=  60              ;;TTY KEYBOARD VECTOR
                                TPVEC=  64              ;;TTY PRINTER VECTOR
                                PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
                                        ABASE=170400
                                        AVECT1=340
                                        APRIOR=200
```

```
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            FORCED PRINTOUT
                                ;*        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  001516          JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001536          JMP     BEGIN1          ;RESTART ADDRESS
000210  000137  001530          JMP     BEGIN2          ;STARTING ADDRESS OF OPTION TEST AREA
```

```
                                        .SBTTL  ACT11 HOOKS

                                        ;;******************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                 000214                         $SVPC=.                 ;SAVE PC
                 000046                         .=46
000046  010436                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
                 000052                         .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
                 000214                         .=$SVPC                 ;; RESTORE PC
                 001000                         .=1000

                                        .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  001200                          $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000020                          $TSTM:  .WORD   20      ;;RUN TIM OF LONGEST TEST
001006  000030                          $PASTM: .WORD   30      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000120                          $UNITM: .WORD   120     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000052                                  .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  000000                          .WORD   0
001136  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001140  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001142  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001144  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001146  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001150  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001152  000                     $NULL:  .BYTE   0       ;;CONTAINS NULL CHARACTER FOR FILLS
001153  002                     $FILLS: .BYTE   2       ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001154  012                     $FILLC: .BYTE   12      ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001155  000                     $TPFLG: .BYTE   0       ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001156  000000                  $REGAD: .WORD   0       ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REGO) WAS OBTAINED
001160  000000                  $REGO:  .WORD   0       ;;CONTAINS (($REGAD)+0)
001162  000000                  $REG1:  .WORD   0       ;;CONTAINS (($REGAD)+2)
001164  000000                  $TIMES: 0               ;;MAX. NUMBER OF ITERATIONS
001166  000000                  $ESCAPE:0               ;;ESCAPE ON ERROR ADDRESS
001170  177607  000377           $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001174  077                     $QUES:  .ASCII  /?/             ;;QUESTION MARK
001175  015                     $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001176  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
```

```
                                ;;**************************************************************
                                .SBTTL  APT MAILBOX-ETABLE
                                ;;**************************************************************
                                .EVEN
001200                          $MAIL:                          ;;APT MAILBOX
001200  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001202  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001204  000000                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001206  000000                  $PASS:  .WORD   APASS           ;;PASS COUNT
001210  000000                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001212  000000                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001214  000000                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
001216  000000                  $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
001220                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
001220  000                     $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
001221  000                     $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
001222  000000                  $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
001224  000000                  $USWR:  .WORD   AUSWR           ;;USER SWITCHES
001226  000000                  $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
                                ;*                              BITS 15-11=CPU TYPE
                                ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                                      11/70=06,PDQ=07,Q=10
                                ;*                              BIT 10=REAL TIME CLOCK
                                ;*                              BIT  9=FLOATING POINT PROCESSOR
                                ;*                              BIT  8=MEMORY MANAGEMENT
001230  000                     $MAMS1: .BYTE   AMAMS1          ;;HIGH ADDRESS,M.S. BYTE
001231  000                     $MTYP1: .BYTE   AMTYP1          ;;MEM. TYPE,BLK#1
                                ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                                      900 NSEC CORE=001
                                ;*                                      300 NSEC BIPOLAR=002
                                ;*                                      500 NSEC MOS=003
001232  000000                  $MADR1: .WORD   AMADR1          ;;HIGH ADDRESS,BLK#1
                                ;*                              MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001234  000                     $MAMS2: .BYTE   AMAMS2          ;;HIGH ADDRESS,M.S. BYTE
001235  000                     $MTYP2: .BYTE   AMTYP2          ;;MEM.TYPE,BLK#2
001236  000000                  $MADR2: .WORD   AMADR2          ;;MEM.LAST ADDRESS,BLK#2
001240  000                     $MAMS3: .BYTE   AMAMS3          ;;HIGH ADDRESS,M.S.BYTE
001241  000                     $MTYP3: .BYTE   AMTYP3          ;;MEM.TYPE,BLK#3
001242  000000                  $MADR3: .WORD   AMADR3          ;;MEM.LAST ADDRESS,BLK#3
001244  000                     $MAMS4: .BYTE   AMAMS4          ;;HIGH ADDRESS,M.S.BYTE
001245  000                     $MTYP4: .BYTE   AMTYP4          ;;MEM.TYPE,BLK#4
001246  000000                  $MADR4: .WORD   AMADR4          ;;MEM.LAST ADDRESS,BLK#4
001250  340                     $VECT1: .BYTE   AVECT1          ;;INTERRUPT VECTOR#1
001251  000                     $VECT2: .BYTE   AVECT2          ;;INTERRUPT VECTOR#2
001252  200                     $PRIOR: .BYTE   APRIOR          ;;BUS PRIORITY #1,#2
001253  000                             .BYTE   0               ;;SPARE,NOT USED
                                        .EVEN
001254  170400                  $BASE:  .WORD   ABASE           ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001256  000000                  $DEVM:  .WORD   ADEVM           ;;DEVICE MAP
001260  000000                  $CDW1:  .WORD   ACDW1           ;;CONTROLLER DESCRIPTION WORD#1
001262  000000                  $CDW2:  .WORD   ACDW2           ;;CONTROLLER DESCRIPTION WORD#2
001264  000000                  $DDW0:  .WORD   ADDW0           ;;DEVICE DESCRIPTOR WORD#0
001266  000000                  $DDW1:  .WORD   ADDW1           ;;DEVICE DESCRIPTOR WORD#1
001270  000000                  $DDW2:  .WORD   ADDW2           ;;DEVICE DESCRIPTOR WORD#2
```

```
001272  000000                  $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001274  000000                  $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001276  000000                  $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001300  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001302  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001304  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001306  000000                  $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001310  000000                  $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001312  000000                  $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001314  000000                  $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001316  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001320  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001322  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15

001324                          $ETEND:


                                .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


001324                          $ERRTB:

                                ;ITEM   1
001324  011736                          EM1             ;ERROR ON A/D CHANNEL
001326  012451                          DH1             ;ERRPC  ARADD   CHANNEL NOMINAL TOLERANCE ACTUAL
001330  014572                          DT1             ;$ERRPC ARBADD  CHANL   $GDDAT  SPREAD  $BDDAT
001332  000000                          0

                                ;ITEM   2
001334  011763                          EM2             ;VC LOGIC SIGNAL HIGH OUTPUT TOO LOW
001336  012530                          DH2             ;ERRPC  ARADD   CHANNEL 3V LEV. OUTPUT
001340  014610                          DT2             ;$ERRPC ARBADD  CHANL   $GDDAT  $BDDAT
001342  000000                          0

                                ;ITEM   3
001344  012027                          EM3             ;VC STATUS REGISTER IN ERROR
001346  012577                          DH3             ;ERRPC  ARADD   GOOD    BAD
001350  014624                          DT3             ;$ERRPC ARBADD  $GDDAT  $BDDAT
001352  000000                          0

                                ;ITEM   4
001354  012063                          EM4             ;EXTERNAL AD START FAILED
001356  012577                          DH3             ;ERRPC  ARADD   GOOD    BAD
001360  014624                          DT3             ;$ERRPC ARBADD  $GDDAT  $BDDAT
```

```
001362  000000                   0

                        ;ITEM    5
001364  012120                   EM5              ;A/D DIFFERENTIAL LINEARITY ERROR
001366  012634                   DH5              ;ERRPC  ARADD   # OF STATES     ALLOWED
001370  014636                   DT5              ;$ERRPC ARBADD    $BDDAT        $GDDAT
001372  000000                   0

                        ;ITEM    6
001374  012161                   EM6              ;NOISE LEVEL EXCEEDED LIMIT
001376  012674                   DH6              ;ERRPC  ARADD   LIMIT   MEASURED
001400  014624                   DT3              ;$ERRPC ARBADD  $GDDAT  $BDDAT
001402  000000                   0
```

```
                                ;ITEM   7
001404  012214                          EM7             ;VC LOGIC SIGNAL LOW OUTPUT TOO HIGH
001406  012735                          DH7             ;ERRPC  ARADD   A/D CHAN +.4V LEV. OUTPUT
001410  014610                          DT2             ;$ERRPC ARBADD  CHANL   $GDDAT   $BDDAT
001412  000000                          0

                                ;ITEM   10
001414  012260                          EM10            ;D/A DIFFERENTIAL LINEARITY ERROR
001416  013004                          DH10            ;ERRPC  ARADD   STEP    NOMINAL SPREAD  ACTUAL
001420  014650                          DT10            ;$ERRPC ARBADD  EDGE    $GDDAT  SPREAD  $BDDAT
001422  000000                          0

                                ;ITEM   11
001424  012321                          EM11            ;A/D INTER-CHANNEL SETTLING ERROR
001426  013063                          DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001430  014666                          DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001432  000000                          0

                                ;ITEM   12
001434  012362                          EM12            ;OFFSET ERROR
001436  013063                          DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001440  014666                          DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001442  000000                          0

                                ;ITEM   13
001444  012377                          EM13            ;CALIBRATION ERROR
001446  013063                          DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001450  014666                          DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001452  000000                          0

                                ;ITEM   14
001454  012421                          EM14            ;LINEARITY ERROR AT XXOO
001456  013063                          DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001460  014666                          DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001462  000000                          0

001464  170400          ARBADD: 170400
001466  000340          ARBVCT: 340
001470  000000          NMBEXT: 0
001472  000000          NBEXT:  0

001474  170400          ADCS:   170400          ;A TO D STATUS/CONTROL REGISTER
001476  170401          ADCS1:  170401          ;A TO D STATUS REGISTER (HIGH BYTE)
001500  170402          ADDBR:  170402          ;A TO D CONVERTED VALUE (READ)
001502  170404          CSR:    170404          ;CLOCK STATUS REGISTER
001504  170406          CSB:    170406          ;CLOCK PRESET BUFFER
001506  170410          VCSTAT: 170410          ;DAC STATUS REGISTER
001510  170412          VCXREG: 170412          ;X BUFFER
001512  170414          VCYREG: 170414          ;Y BUFFER
001514  170416          CSC:    170416          ;CLOCK COUNTER
                        .SBTTL  PROGRAM START-UP
```

```
                                    BEGIN:  CLR     R0                      ;CLEAR R0
001516  005000
                                            CLR     WFTEST                  ;CLEAR TOLERANCE FLAG
001520  005037  016762
                                            BR      RBEG
001524  000406
                                            BR      RBEG
001526  000405
                                    BEGIN2: MOV     #1,WFTEST               ;SET TOLERANCE FLAG
001530  012737  000001  016762
                                    BEGIN1: MOV     #-1,R0                  ;LOAD R0
001536  012700  177777
                                    RBEG:   RESET
001542  000005
                                            CLR     PS
001544  005037  177776
                                            MOV     #STACK,SP               ;LOAD STACK
001550  012706  001100
                                            MOV     #1$,@#4                 ;LOAD BUS ERROR
001554  012737  001602  000004
                                            MOV     $BASE,R2                ;LOAD STARTING ADDRESS
001562  013702  001254
                                            CLR     R3                      ;CLEAR COUNT
001566  005003
                                    2$:     TST     (R2)                    ;TEST IF EXISTENT
001570  005712
                                            ADD     #20,R2                  ;EXIST, UPDATE TEST ADDRESS
001572  062702  000020
                                            INC     R3                      ;UPDATE # OF AR11'S
001576  005203
                                            BR      2$
001600  000773
                                    1$:     CMP     (SP)+,(SP)+             ;POP STACK
001602  022626
                                            TST     R3                      ;TEST IF FIRST DOES EXIST
001604  005703
                                            BNE     3$                      ;BR
001606  001002
                                            HALT                            ;FIRST AR11 DOES NOT EXIST
001610  000000
                                            BR      BEGIN                   ;CHECK THE PROGRAM DEVICE ADDRESS
001612  000741
                                    3$:     DEC     R3                      ;ADJUST R3
001614  005303
                                            MOV     R3,NMBEXT               ;SAVE THE NUMBER OF ADDITIONAL AR11'S
001616  010337  001470
                                            MOV     #6,@#4                  ;RESET BUS ERROR
001622  012737  000006  000004
                                            CLR     @#6
001630  005037  000006
                                            MOV     $BASE,ARBADD            ;LOAD FIRST ADDRESS
001634  013737  001254  001464
                                            MOV     $VECT1,ARBVCT           ;LOAD FIRST VECTOR
001642  013737  001250  001466
                                            MOV     NMBEXT,NBEXT            ;LOAD NUMBER OF ADDITIONAL AR11'S
001650  013737  001470  001472
                                            RESET
001656  000005
                                    ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
                                            MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001660  012706  001100
                                            CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001664  005026
                                            CMP     #$BDDAT,R6      ;;DONE?
001666  022706  001126
                                            BNE     .-6             ;;LOOP BACK IF NO
001672  001374
                                            MOV     #STACK,SP       ;;SETUP THE STACK POINTER
001674  012706  001100
                                    ;;INITIALIZE A FEW VECTORS
                                            MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001700  012737  017172  000020
                                            MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001706  012737  000340  000022
                                            MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001714  012737  017452  000030
                                            MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001722  012737  000340  000032
                                            MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001730  012737  021774  000034
                                            MOV     #340,@#TRAPVEC+2;LEVEL 7
001736  012737  000340  000036
                                            MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001744  012737  020576  000024
                                            MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001752  012737  000340  000026
                                            MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
001760  013737  010404  010376
                                            CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001766  005037  001164
                                            CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001772  005037  001166
                                            MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
001776  112737  000001  001115
                                            MOV     #..$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
002004  012737  002004  001106
                                            MOV     #..$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
002012  012737  002012  001110
                                    ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                    ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
                                            MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
002020  013746  000004
                                            MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
002024  012737  002062  000004
                                            MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
002032  012737  177570  001136
                                            MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
002040  012737  177570  001140
```

```
   659  002046  022777  177777  177062          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
   660  002054  001013                          BNE     65$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
   661                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
   662  002056  005737  000001                  TST     @#1             ;;FORCE A TRAP THROUGH ERRVEC
   663  002062  012737  000176  001136  64$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
   664  002070  012737  000174  001140          MOV     #DISPREG,DISPLAY ;;POINT TO SOFTWARE DISPLAY REG
   665  002076  012716  002104                  MOV     #65$,(SP)       ;;REPLACE OLD PC WITH NEW
   666  002102  000002                          RTI                     ;;RESTORE PC AND PSW
   667  002104  012637  000004          65$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
   668
   669  002110                          $ARG1:
   670  002110  005037  001206                  CLR     $PASS           ;;CLEAR PASS COUNT
   671  002114  132737  000200  001221          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
   672  002122  001403                          BEQ     64$             ;;YES,USE NON-APT SWITCH
   673  002124  012737  001222  001136          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
   674  002132                          64$:
   675  002132  000005                  RBEG2:  RESET
   676  002134  012702  000232                  MOV     #232,R2                 ;LOAD R2
   677  002140  012701  000230                  MOV     #230,R1                 ;LOAD R1
   678  002144  010221                  5$:     MOV     R2,(R1)+                ;LOAD .+2
   679  002146  005021                          CLR     (R1)+                   ;LOAD HALT
   680  002150  010102                          MOV     R1,R2                   ;LOAD R2
   681  002152  005722                          TST     (R2)+                   ;BUMP R2
   682  002154  020227  001002                  CMP     R2,#1002                ;TEST FOR LAST
   683  002160  001371                          BNE     5$                      ;BR UNTIL DONE
   684  002162  004737  016646                  JSR     PC,WFADJ                ;ADJUST SOME TOLERANCES
   685  002166  005700                          TST     R0                      ;TEST R0
   686  002170  001402                          BEQ     2$                      ;BR IF CLEARED
   687  002172  000137  002476                  JMP     4$                      ;INHIBIT TYPOUT
   688  002176  005737  000042          2$:     TST     @#42                    ;TEST ACT-11 OR DDP
   689  002202  001402                          BEQ     3$                      ;BR IF CLEARED
   690  002204  000137  002476                  JMP     4$                      ;INHIBIT TYPOUT
   691  002210                          3$:
   692  002210  104400  002216                  TYPE    ,65$                    ;;TYPE ASCIZ STRING
   693  002214  000422                          BR      64$                     ;;GET OVER THE ASCIZ
   694                                  ;;65$:  .ASCIZ  <15><12>/AR-11 DIAGNOSTIC WRAPAROUND TEST/
   695  002262                          64$:
   696  002262  104400  002270                  TYPE    ,67$                    ;;TYPE ASCIZ STRING
   697  002266  000414                          BR      66$                     ;;GET OVER THE ASCIZ
   698                                  ;;67$:  .ASCIZ  <15><12>/MAINDEC-11-DZARC-B/<15><12>
   699  002320                          66$:
   700  002320  013746  001470                  MOV     NMBEXT,-(SP)            ;PUSH ON STACK
   701  002324  104402                          TYPOS                           ;TYPE OCTAL
   702  002326  000002                          .WORD   2
   703  002330  104400  002336                  TYPE    ,69$                    ;;TYPE ASCIZ STRING
   704  002334  000420                          BR      68$                     ;;GET OVER THE ASCIZ
   705                                  ;;69$:  .ASCIZ  /(8) ADDITIONAL AR11'S CONNECTED/
   706  002376                          68$:
   707  002376  104400  002404                  TYPE    ,71$                    ;;TYPE ASCIZ STRING
   708  002402  000435                          BR      70$                     ;;GET OVER THE ASCIZ
   709                                  ;;71$:  .ASCIZ  <15><12>/ALL AR-11'S MUST HAVE G5036 WRAPAROUND MODULE INSTALLED/
   710  002476                          70$:
   711
   712  002476  012700  001474          4$:     MOV     #ADCS,R0                ;LOAD POINTER
   713  002502  013720  001464          10$:    MOV     ARBADD,(R0)+
   714  002506  022700  001516                  CMP     #BEGIN,R0               ;TEST FOR END
```

```
                                        BNE     10$
                                        INC     ADCS1
                                        MOV     #ADDBR,R0
                                        MOV     #2,R1
                                12$:    ADD     R1,(R0)+
                                        TST     (R1)+
                                        CMP     #20,R1
                                        BNE     12$
                                        .SBTTL  A TO D CHANNEL NUMBERS

                                ;BIPOLAR CHANNEL NUMBERS

                                        CH0=    0
                                        CH1=    1
                                        CH2=    2
                                        CH3=    3
                                        CH4=    4
                                        CH5=    5
                                        CH6=    6
                                        CH7=    7
                                        CH10=   10
                                        CH11=   11
                                        CH12=   12
                                        CH13=   13
                                        CH14=   14
                                        CH15=   15
                                        CH16=   16
                                        CH17=   17

                                ;UNIPOLAR CHANNEL NUMBERS

                                        CH40=   40
                                        CH41=   41
                                        CH42=   42
                                        CH43=   43
                                        CH44=   44
                                        CH45=   45
                                        CH46=   46
                                        CH47=   47
                                        CH50=   50
                                        CH51=   51
                                        CH52=   52
                                        CH53=   53
                                        CH54=   54
                                        CH55=   55
                                        CH56=   56
                                        CH57=   57
                                        .SBTTL
                                        .SBTTL  TEST NUMBER                     ABSTRACT
                                        .SBTTL
```

```
                                ;;*****************************************************************
                                ;*TEST 1        TEST THAT "ERASE" AND DONE CAN BE SET AND CLEARED
                                ;;*****************************************************************
002542  000004          TST1:   SCOPE
002544  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
002552  005077  176730                  CLR     @VCSTAT                 ;CLEAR
002556  012737  010000  001124          MOV     #BIT12,$GDDAT           ;LOAD EXPECTED
002564  013777  001124  176714          MOV     $GDDAT,@VCSTAT          ;LOAD REG
002572  017737  176710  001126          MOV     @VCSTAT,$BDDAT          ;READ REG
002600  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
002606  001401                          BEQ     1$              ;;BR IF SET
002610  104003                          ERROR   3                       ;ERASE FAILED TO SET, CHECK G5036-
                                                                        ;BC08R CONNECTION: A-VV, VV-A
002612  012737  001000  001124  1$:     MOV     #BIT9,$GDDAT            ;LOAD EXPECTED
002620  052777  001000  176660          BIS     #BIT9,@VCSTAT           ;SET CH 2 BIT
002626  017737  176654  001126          MOV     @VCSTAT,$BDDAT          ;READ REG
002634  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
002642  001401                          BEQ     2$              ;;BR IF CLEARED
002644  104003                          ERROR   3                       ;ERASE BIT FAILED TO CLEAR BY ERASE
                                                                        ;RETURN (SETTING OF THE CH 2 BIT)
002646  012737  000200  001124  2$:     MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
002654  042777  001000  176624          BIC     #BIT9,@VCSTAT           ;CLEAR CH 02
002662  017737  176620  001126          MOV     @VCSTAT,$BDDAT          ;READ REG
002670  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
002676  001401                          BEQ     TST2            ;;BR IF EQUAL
002700  104003                          ERROR   3                       ;READY FAILED TO SET ON THE END
                                                                        ; OF ERASE RETURN (CLEARING CH 2)
                                ;;*****************************************************************
                                ;*TEST 2        EXTERNAL A TO D START FROM INTENSIFY
                                ;;*****************************************************************
002702  000004          TST2:   SCOPE
002704  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
002712  005077  176570                  CLR     @VCSTAT
002716  017777  176556  176554          MOV     @ADDBR,@ADDBR           ;CLEAR DONE
002724  005077  176544                  CLR     @ADCS
002730  012737  000220  001124          MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPECTED
002736  013777  001124  176530          MOV     $GDDAT,@ADCS            ;LOAD EXTERNAL A/D START ENABLE
002744  005277  176536                  INC     @VCSTAT                 ;INTENSIFY
002750  105777  176532          1$:     TSTB    @VCSTAT                 ;WAIT FOR READY
002754  100375                          BPL     1$
002756  012737  000400  015216          MOV     #BIT8,TEMP              ;SET UP A COUNTER
002764  005337  015216          2$:     DEC     TEMP                    ;DELAY
002770  100375                          BPL     2$
002772  017737  176476  001126          MOV     @ADCS,$BDDAT            ;READ REG.
003000  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
003006  001401                          BEQ     3$              ;;BR IF EQUAL
003010  104004                          ERROR   4                       ;INTENSIFY PULSE FAILED TO START
003012  017777  176462  176460  3$:     MOV     @ADDBR,@ADDBR           ; AN A TO D CONVERSION
003020  005077  176462                  CLR     @VCSTAT
003024  005077  176444                  CLR     @ADCS
```

```
                                ;;*****************************************************************
                                ;*TEST 3         TEST THAT CH 0 IS A BIPOLAR GROUND
                                ;;*****************************************************************
003030  000004                  TST3:   SCOPE
003032  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003040  012737  001000  001124          MOV     #1000,$GDDAT            ;LOAD EXPECTED VALUE
003046  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003052  000000                          CH0                             ;CH 0
003054  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003062  012737  000001  011664          MOV     #1,SPREAD               ;LOAD + OR - COUNT SPREAD
003070  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003074  000401                          BR      TST4            ;;BR IF WITHIN TOLERANCE
003076  104001                          ERROR   1                       ;CH 0 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE


                                ;;*****************************************************************
                                ;*TEST 4         TEST THAT CH 40 IS A UNIPOLAR GROUND
                                ;;*****************************************************************
003100  000004                  TST4:   SCOPE
003102  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003110  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED VALUE
003116  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003122  000040                          CH40                            ;CH 40
003124  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003132  012737  000001  011664          MOV     #1,SPREAD               ;LOAD + OR - COUNT SPREAD
003140  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003144  000401                          BR      TST5            ;;BR IF WITHIN TOLERANCE
003146  104001                          ERROR   1                       ;CH 40 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE


                                ;;*****************************************************************
                                ;*TEST 5         TEST THAT CH 2 IS AT +1 VOLT BIPOLAR
                                ;;*****************************************************************
003150  000004                  TST5:   SCOPE
003152  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003160  012737  001315  001124          MOV     #1315,$GDDAT            ;LOAD EXPECTED VALUE
003166  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003172  000002                          CH2                             ;CH 2
003174  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003202  012737  000024  011664          MOV     #24,SPREAD              ;LOAD + OR - COUNT SPREAD
003210  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003214  000401                          BR      TST6            ;;BR IF WITHIN TOLERANCE
003216  104001                          ERROR   1                       ;CH 2 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
```

```
                                ;;*****************************************************************
                                ;*TEST 6         TEST THAT CH 42 IS AT +1 VOLT UNIPOLAR
                                ;;*****************************************************************
003220  000004          TST6:   SCOPE
003222  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003230  012737  000315  001124          MOV     #315,$GDDAT             ;LOAD EXPECTED VALUE
003236  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003242  000042                          CH42                    ;CH 42
003244  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003252  012737  000024  011664          MOV     #24,SPREAD              ;LOAD + OR - COUNT SPREAD
003260  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003264  000401                          BR      TST7            ;;BR IF WITHIN TOLERANCE
003266  104001                          ERROR   1                       ;CH 42 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
                                ;;*****************************************************************
                                ;*TEST 7         TEST THAT CH 3 IS AT +2.5 BIPOLAR
                                ;;*****************************************************************
003270  000004          TST7:   SCOPE
003272  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003300  013737  016764  001124          MOV     V1754,$GDDAT            ;LOAD EXPECTED VALUE
003306  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003312  000003                          CH3                     ;CH 3
003314  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003322  013737  016766  011664          MOV     V24,SPREAD              ;LOAD + OR - COUNT SPREAD
003330  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003334  000401                          BR      TST10           ;;BR IF WITHIN TOLERANCE
003336  104001                          ERROR   1                       ;CH 3 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
                                ;;*****************************************************************
                                ;*TEST 10        TEST THAT CH 43 IS AT +2.5 UNIPOLAR
                                ;;*****************************************************************
003340  000004          TST10:  SCOPE
003342  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003350  012737  001000  001124          MOV     #1000,$GDDAT            ;LOAD EXPECTED VALUE
003356  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003362  000043                          CH43                    ;CH 43
003364  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003372  013737  016770  011664          MOV     V50,SPREAD              ;LOAD + OR - COUNT SPREAD
003400  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003404  000401                          BR      TST11           ;;BR IF WITHIN TOLERANCE
003406  104001                          ERROR   1                       ;CH 435FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
```

```
                                ;;*****************************************************************
                                ;*TEST 11       TEST THAT CH 4 IS AT -2.5 BIPOLAR
                                ;;*****************************************************************
003410  000004                  TST11:  SCOPE
003412  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003420  013737  016766  001124          MOV     V24,$GDDAT              ;LOAD EXPECTED VALUE
003426  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003432  000004                          CH4                             ;CH 4
003434  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003442  013737  016766  011664          MOV     V24,SPREAD              ;LOAD + OR - COUNT SPREAD
003450  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003454  000401                          BR      TST12           ;;BR IF WITHIN TOLERANCE
003456  104001                          ERROR   1                       ;CH 4 FAILED TO EQUAL EXPECTED VALUE

                                ;;*****************************************************************
                                ;*TEST 12       TEST THAT CH 57 IS AT +4 VOLTS UNIPOLAR
                                ;;*****************************************************************
003460  000004                  TST12:  SCOPE
003462  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003470  012737  001463  001124          MOV     #1463,$GDDAT            ;LOAD EXPECTED VALUE
003476  004537  011534                  JSR     R5,CONVRT               ;CONVERT
003502  000057                          CH57                            ;CH 57
003504  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003512  012737  000120  011664          MOV     #120,SPREAD             ;LOAD + OR - COUNT SPREAD
003520  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003524  000401                          BR      TST13           ;;BR IF WITHIN TOLERANCE
003526  104001                          ERROR   1                       ;CH 57 FAILED TO EQUAL EXPECTED VALUE

                                ;;*****************************************************************
                                ;*TEST 13       TEST THAT CH 41 IS LESS THAN +200 MVOLTS UNIPOLAR
                                ;;*****************************************************************
003530  000004                  TST13:  SCOPE
003532  012737  000004  001164          MOV     #4,$TIMES       ;;DO 4 ITERATIONS
003540  012737  000041  011662          MOV     #CH41,CHANL             ;LOAD CHANNEL # FOR ERROR TYPOUT
003546  112777  000041  175722          MOVB    #CH41,@ADCS1            ;LOAD MUX
003554  013737  016772  001124          MOV     VA24,$GDDAT             ;LOAD EXPECTED VALUE
003562  013737  016772  011664          MOV     VA24,SPREAD             ;LOAD + OR - COUNT SPREAD
003570  012737  000200  015216          MOV     #BIT7,TEMP              ;LOAD DELAY
003576  005337  015216          2$:     DEC     TEMP                    ;DELAY
003602  001375                          BNE     2$
003604  005277  175664                  INC     @ADCS                   ;CONVERT CH 41
003610  105777  175660          3$:     TSTB    @ADCS                   ;DONE ?
003614  100375                          BPL     3$
003616  017737  175656  001126          MOV     @ADDBR,$BDDAT           ;READ RESULTS
003624  013737  001126  015164          MOV     $BDDAT,BIASC1           ;SAVE RESULTS
003632  004737  011666                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003636  000401                          BR      1$              ;;BR IF WITHIN TOLERANCE
003640  104001                          ERROR   1                       ;CH 41 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE. BIAS CURRENT TOO HIGH
003642  032777  010000  175266  1$:     BIT     #BIT12,@SWR             ;TEST FORCED SW
003650  001432                          BEQ     TST14           ;;BR IF NOT FORCED PRINTOUT
003652  104400  003660                  TYPE    ,65$            ;;TYPE ASCIZ STRING
003656  000412                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12><12>/AR-11 ADDRESS = /
003704                          64$:
003704  013746  001464                  MOV     ARBADD,-(SP)            ;SAVE ADDRESS
```

```
                                        TYPOC
                                        TYPE    ,ACRLF
                                        TYPE    ,BIASST
                                        MOV     BIASC1,-(SP)
                                        TYPOS
                                        .BYTE   3,0
                                        TYPE    ,BIASDN

                        ;;**************************************************************
                        ;*TEST 14       TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH
                        ;;**************************************************************
TST14:  SCOPE
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                        MOV     #BIT11!BIT9,@VCSTAT     ;LOAD VC STATUS
                                        MOV     #1146,$GDDAT            ;LOAD EXPECTED VALUE
                                        JSR     R5,CONVRT               ;CONVERT
                                        CH53                            ;CH 53
                                        MOV     ADEND,$BDDAT            ;LOAD VALUE READ
                                        CMP     $GDDAT,$BDDAT           ;COMPARE
                                        BLE     TST15           ;;BR IF LESS THAN OR EQUAL
                                        ERROR   2                       ;CH 53 FAILED TO EQUAL EXPECTED VALUE

                        ;;**************************************************************
                        ;*TEST 15       TEST THAT ERASE L (CH 53) IS LESS THAN +400 MVOLTS WHEN LOW
                        ;;**************************************************************
TST15:  SCOPE
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                        CLR     @VCSTAT                 ;CLEAR VC STATUS
                                        MOV     #BIT12!BIT10,@VCSTAT    ;SET ERASE
                                        MOV     #120,$GDDAT             ;LOAD EXPECTED VALUE
                                        JSR     R5,CONVRT               ;CONVERT
                                        CH53                            ;CH 53
                                        MOV     ADEND,$BDDAT            ;LOAD VALUE READ
                                        CMP     $GDDAT,$BDDAT           ;COMPARE
                                        BGE     TST16           ;;BR IF GREATER THAN OR EQUAL
                                        ERROR   7                       ;CH 53 FAILED TO EQUAL EXPECTED VALUE

                        ;;**************************************************************
                        ;*TEST 16       TEST THAT WRITE-THRU (CH 54) IS GREATER THAN +3 VOLTS WHEN HIGH
                        ;;**************************************************************
TST16:  SCOPE
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                        MOV     #BIT12!BIT9,@VCSTAT     ;LOAD VC STATUS
                                        MOV     #1146,$GDDAT            ;LOAD EXPECTED VALUE
                                        JSR     R5,CONVRT               ;CONVERT
                                        CH54                            ;CH 54
                                        MOV     ADEND,$BDDAT            ;LOAD VALUE READ
                                        CMP     $GDDAT,$BDDAT           ;COMPARE
                                        BLE     TST17           ;;BR IF LESS THAN OR EQUAL
                                        ERROR   2                       ;CH 54 FAILED TO EQUAL EXPECTED VALUE
```

```
                                ;;*****************************************************************
                                ;*TEST 17        TEST THAT WRITE-THRU (CH 54) IS LESS THAN +400 MVOLTS WHEN LOW
                                ;;*****************************************************************
004140  000004          TST17:  SCOPE
004142  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
004150  012777  006000  175330          MOV     #BIT11!BIT10,@VCSTAT    ;SET WRITE-THRU
004156  012737  000120  001124          MOV     #120,$GDDAT             ;LOAD EXPECTED VALUE
004164  004537  011534                  JSR     R5,CONVRT               ;CONVERT
004170  000054                          CH54                            ;CH 54
004172  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
004200  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004206  002001                          BGE     TST20           ;;BR IF GREATER THAN OR EQUAL
004210  104007                          ERROR   7                       ;CH 54 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
                                ;;*****************************************************************
                                ;*TEST 20        TEST THAT NON-STORE (CH 55) IS LESS THAN +400 MVOLTS WHEN LOW
                                ;;*****************************************************************
004212  000004          TST20:  SCOPE
004214  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
004222  005077  175260                  CLR     @VCSTAT                 ;CLEAR STORE MODE
004226  012737  000120  001124          MOV     #120,$GDDAT             ;LOAD EXPECTED VALUE
004234  004537  011534                  JSR     R5,CONVRT               ;CONVERT
004240  000055                          CH55                            ;CH 55
004242  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
004250  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004256  002001                          BGE     TST21           ;;BR IF GREATER THAN OR EQUAL
004260  104007                          ERROR   7                       ;CH 54 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
                                ;;*****************************************************************
                                ;*TEST 21        TEST THAT NON-STORE L (CH 55) IS GREATER THAN +3 VOLTS WHEN HIGH
                                ;;*****************************************************************
004262  000004          TST21:  SCOPE
004264  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
004272  012777  017000  175206          MOV     #BIT12!BIT11!BIT10!BIT9,@VCSTAT         ;SET STORE MODE
004300  012737  001146  001124          MOV     #1146.$GDDAT            ;LOAD EXPECTED VALUE
004306  004537  011534                  JSR     R5,CONVRT               ;CONVERT
004312  000055                          CH55                            ;CH 55
004314  013737  011656  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
004322  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004330  003401                          BLE     TST22           ;;BR IF LESS THAN OR EQUAL
004332  104002                          ERROR   2                       ;CH 55 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
```

```
1058                                    ;;*****************************************************************
1059                                    ;*TEST 22       TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH
1060                                    ;;*****************************************************************
1061  004334  000004                    TST22:  SCOPE
1062  004336  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1063  004344  012777  014000  175134            MOV     #BIT12!BIT11,@VCSTAT    ;LOAD VC STATUS
1064  004352  012737  001146  001124            MOV     #1146,$GDDAT            ;LOAD EXPECTED VALUE
1065  004360  004537  011534                    JSR     R5,CONVRT               ;CONVERT
1066  004364  000056                            CH56                            ;CH 56
1067  004366  013737  011656  001126            MOV     ADEND,$BDDAT            ;LOAD VALUE READ
1068  004374  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1069  004402  003401                            BLE     TST23           ;;BR IF LESS THAN OR EQUAL
1070  004404  104002                            ERROR   2                       ;CH 56 FAILED TO EQUAL EXPECTED
1071                                                                            ; VALUE
1072
1073                                    ;;*****************************************************************
1074                                    ;*TEST 23       TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW
1075                                    ;;*****************************************************************
1076  004406  000004                    TST23:  SCOPE
1077  004410  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1078  004416  012777  003000  175062            MOV     #BIT9!BIT10,@VCSTAT     ;SET CHANNEL 2
1079  004424  012737  000120  001124            MOV     #120,$GDDAT             ;LOAD EXPECTED VALUE
1080  004432  004537  011534                    JSR     R5,CONVRT               ;CONVERT
1081  004436  000056                            CH56                            ;CH 56
1082  004440  013737  011656  001126            MOV     ADEND,$BDDAT            ;LOAD VALUE READ
1083  004446  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1084  004454  002001                            BGE     1$              ;;BR IF GREATER THAN
1085  004456  104007                            ERROR   7                       ;CH 56 FAILED TO EQUAL EXPECTED
1086                                                                            ; VALUE
1087  004460  005077  175022            1$:     CLR     @VCSTAT                 ;CLEAR BIT 9
1088
```

```
1089                                    ;;*****************************************************************
1090                                    ;*TEST 24        A TO D DIFFERENTIAL LINEARITY TEST
1091                                    ;;*****************************************************************
1092  004464  000004                    TST24:  SCOPE
1093  004466  012737  000001  001164            MOV     #1,$TIMES       ;;DO 1 ITERATION
1094
1095  004474  004537  015220                    JSR     R5,DIFLIN               ;START TEST
1096  004500  001510                            VCXREG                  ;X COARSE
1097  004502  001512                            VCYREG                  ;Y FINE
1098  004504     001     005                     .BYTE   1,5                     ;GO AND CHANNEL 5
1099
1100  004506  005037  001124                    CLR     $GDDAT                  ;CLEAR EXPECTED
1101  004512  013737  015740  001126            MOV     SKIPST,$BDDAT           ;LOAD # OF SKIPPED STATES
1102  004520  001401                            BEQ     1$              ;;BR IF NO SKIPPED STATES
1103  004522  104005                            ERROR   5                       ;SKIPPED STATE(S) DETECTED
1104
1105  004524  013737  015744  001126  1$:      MOV     EXCESS,$BDDAT           ;LOAD # OF >2LSB STATES
1106  004532  001401                            BEQ     2$              ;;BR IF NO GREATER THAN 2LSB WIDE STATES
1107  004534  104005                            ERROR   5                       ;> 2LSB WIDE STATE(S) DETECTED
1108
1109  004536  013737  015742  001126  2$:      MOV     DIFERR,$BDDAT           ;LOAD # OUTSIDE +- 5/2 LSB
1110  004544  012737  000064  001124            MOV     #52.,$GDDAT             ;LOAD MAX # OF ALLOWABLE STATES OUTSIDE +- 1/2 L
1111  004552  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;ANY ERRORS
1112  004560  002001                            BGE     TST25           ;;BR IF NO ERROR
1113  004562  104005                            ERROR   5                       ;DIFFERENTIAL LINEARITY ERROR, >5% OF STATE
1114                                                                            ;WIDTHS OUTSIDE +- 1/2 LSB
1115
```

```
1116                                    ;;*****************************************************************
1117                                    ;*TEST 25       X D/A DIFFERENTIAL LINEARITY USING CH 5
1118                                    ;;*****************************************************************
1119  004564  000004                    TST25:  SCOPE
1120  004566  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1121  004574  012701  000001                    MOV     #1,R1                   ;LOAD EDGE VALUE
1122  004600  012702  015022                    MOV     #RSLT2,R2               ;LOAD RESULT POINTER
1123
1124  004604  010177  174700            1$:     MOV     R1,@VCXREG              ;LOAD X DAC WITH PRESET
1125  004610  010137  004624                    MOV     R1,10$                  ;LOAD EDGE VALUE
1126
1127  004614  004437  010716                    JSR     R4,SAR                  ;SOFTWARE SUCCESSIVE APPROX. ROUTINE
1128  004620  001512                            VCYREG
1129  004622     000     005                    .BYTE   0,CH5
1130  004624  000000                    10$:    0                               ;EDGE VALUE
1131  004626  000200                            BIT7                            ;50-50
1132
1133  004630  013737  015122  004746            MOV     DACSAV,12$              ;SAVE VALUE
1134  004636  005377  174646                    DEC     @VCXREG                 ;LOAD X DAC
1135  004642  010137  004656                    MOV     R1,11$                  ;LOAD SAME EDGE
1136
1137  004646  004437  010716                    JSR     R4,SAR                  ;SOFTWARE S.A.R.
1138  004652  001512                            VCYREG
1139  004654     000     005                    .BYTE   0,CH5
1140  004656  000000                    11$:    0
1141  004660  000200                            BIT7
1142
1143  004662  013737  015122  004750            MOV     DACSAV,13$              ;SAVE VALUE
1144  004670  013737  015122  001126            MOV     DACSAV,$BDDAT           ;LOAD VALUE INTO $BDDAT
1145  004676  163737  004746  001126            SUB     12$,$BDDAT              ;STEP HEIGHT IN $BDDAT
1146  004704  013722  001126                    MOV     $BDDAT,(R2)+            ;SAVE RESULT
1147  004710  012737  000062  001124            MOV     #62,$GDDAT              ;LOAD EXPECTED VALUE FOR 1 LSB STEP
1148  004716  013737  016774  011664            MOV     V62,SPREAD              ;LOAD TOLERANCE
1149  004724  004737  011666                    JSR     PC,COMPAR               ;TEST LIMITS
1150  004730  000401                            BR      2$              ;;BR IF WITHIN LIMITS
1151  004732  104010                            ERROR   10                      ;D/A DIFFERENTIAL LINEARITY ERROR
1152  004734  006301                    2$:     ASL     R1                      ;MOVE LEFT
1153  004736  020127  002000                    CMP     R1,#2000                ;DONE
1154  004742  001320                            BNE     1$              ;;BR OF NOT DONE
1155  004744  000402                            BR      TST26           ;;
1156
1157  004746  000000                    12$:    0
1158  004750  000000                    13$:    0
```

```
                                        ;;**********************************************************************
                                        ;*TEST 26       Y D/A DIFFERENTIAL LINEARITY USING CH 6
                                        ;;**********************************************************************
004752  000004                  TST26:  SCOPE
004754  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
004762  012701  000001                  MOV     #1,R1                   ;LOAD EDGE VALUE
004766  012702  015060                  MOV     #RSLT3,R2               ;LOAD RESULT POINTER

004772  010177  174514          1$:     MOV     R1,@VCYREG              ;LOAD Y DAC WITH PRESET
004776  010137  005012                  MOV     R1,10$                  ;LOAD EDGE VALUE

005002  004437  010716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
005006  001510                          VCXREG
005010     000     006                  .BYTE   0,CH6
005012  000000                  10$:    0                               ;EDGE VALUE
005014  000200                          BIT7

005016  013737  015122  005200          MOV     DACSAV,12$              ;SAVE VALUE
005024  005377  174462                  DEC     @VCYREG                 ;LOAD Y DAC
005030  010137  005044                  MOV     R1,11$                  ;LOAD EDGE

005034  004437  010716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
005040  001510                          VCXREG
005042     000     006                  .BYTE   0,CH6
005044  000000                  11$:    0
005046  000200                          BIT7

005050  013737  015122  005202          MOV     DACSAV,13$              ;SAVE RESULTS
005056  013737  015122  001126          MOV     DACSAV,$BDDAT           ;LOAD $BDDAT
005064  163737  005200  001126          SUB     12$,$BDDAT              ;STEP HEIGHT IN $BDDAT
005072  013722  001126                  MOV     $BDDAT,(R2)+            ;SAVE RESULT
005076  012737  000062  001124          MOV     #62,$GDDAT              ;LOAD EXPECTED
005104  013737  016774  011664          MOV     V62,SPREAD              ;LOAD TOLERANCE
005112  004737  011666                  JSR     PC,COMPAR               ;TEST IF WITHIN LIMITS
005116  000401                          BR      2$              ;;BR IF WITHIN LIMITS
005120  104010                          ERROR   10                      ;D/A DIFFERENTIAL LINEARITY ERROR
005122  006301                  2$:     ASL     R1
005124  020127  002000                  CMP     R1,#2000                ;DONE ?
005130  001320                          BNE     1$              ;;BR IF NOT DONE
005132  032777  010000  173776          BIT     #BIT12,@SWR             ;TEST FORCED PRINTOUT
005140  001421                          BEQ     TST27           ;;BR IF NOT FORCED PRINTOUT
005142  104400  013203                  TYPE    ,DIFMSG
005146  004537  011510                  JSR     R5,TYPSTG               ;TYPE A STRING OF OCTAL #
005152  015022                          RSLT2                           ;STARTING
005154  000012                          10.                             ; # OF LOCATIONS
005156  104400  013367                  TYPE    ,DIYMSG
005162  004537  011510                  JSR     R5,TYPSTG               ;TYPE A STRING OF OCTAL #
005166  015060                          RSLT3
005170  000012                          10.                             ; # OF LOCATIONS
005172  104400  013377                  TYPE    ,DIEMSG
005176  000402                          BR      TST27           ;;
005200  000000                  12$:    0
005202  000000                  13$:    0
```

```
1212                                    ;;*********************************************************************
1213                                    ;*TEST 27       A TO D POSITIVE MULTIPLEXER SETTLING TEST
1214                                    ;;*********************************************************************
1215  005204  000004                   TST27:  SCOPE
1216  005206  012737  000010  001164           MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1217  005214  012777  001770  174270           MOV     #1770,@VCYREG           ;LOAD Y DAC
1218  005222  004437  010716                   JSR     R4,SAR          ;SOFTWARE S.A.R.
1219  005226  001510                           VCXREG
1220  005230      000      006                  .BYTE   0,6
1221  005232  001771                           1771
1222  005234  000200                           BIT7
1223
1224  005236  013737  015122  005330           MOV     DACSAV,10$              ;SAVE RESULTS
1225  005244  004437  010716                   JSR     R4,SAR          ;SOFTWARE S.A.R.
1226  005250  001510                           VCXREG
1227  005252      004      006                  .BYTE   CH4,CH5         ;-2.5V  CH AND CH 6
1228  005254  001771                           1771                    ;EDGE
1229  005256  000200                           BIT7
1230
1231  005260  013737  015122  001126           MOV     DACSAV,$BDDAT           ;SAVE RESULT IN $BDDAT
1232  005266  163737  005330  001126           SUB     10$,$BDDAT              ;SUB FIRST VALUE
1233  005274  013737  001126  015160           MOV     $BDDAT,ADPMUX           ;SAVE RESULT
1234  005302  012737  000000  001124           MOV     #0,$GDDAT               ;LOAD EXPECTED SETTLING ERROR
1235  005310  012737  000077  011664           MOV     #77,SPREAD              ;LOAD TOLERANCE
1236  005316  004737  011666                   JSR     PC,COMPAR               ;TEST IF WITHIN LIMITS
1237  005322  000403                           BR      TST30           ;;BR IF WITHIN
1238  005324  104011                           ERROR   11                      ;A/D POSITIVE SETTLING ERROR
1239  005326  000401                           BR      TST30           ;;
1240  005330  000000                   10$:    0
1241
```

```
1242                                    ;;*****************************************************************
1243                                    ;*TEST 30       A TO D NEGATIVE MULTIPLEXER SETTLING TEST
1244                                    ;;*****************************************************************
1245  005332  000004                    TST30:  SCOPE
1246  005334  012737  000010  001164            MOV     #10.$TIMES      ;;DO 10 ITERATIONS
1247  005342  012777  000010  174142            MOV     #10,@VCYREG             ;LOAD Y DAC
1248  005350  004437  010716                    JSR     R4,SAR                  ;SOFTWARE S.A.R.
1249  005354  001510                            VCXREG
1250  005356     000     006                    .BYTE   0,6
1251  005360  000010                            10
1252  005362  000200                            BIT7
1253
1254  005364  013737  015122  005522            MOV     DACSAV,10$              ;SAVE RESULTS
1255  005372  004437  010716                    JSR     R4,SAR                  ;SOFTWARE S.A.R.
1256  005376  001510                            VCXREG
1257  005400     003     006                    .BYTE   CH3,CH6                 ;+2.5V  CH AND CH 6
1258  005402  000010                            10                              ;EDGE
1259  005404  000200                            BIT7
1260
1261  005406  013737  015122  001125            MOV     DACSAV,$BDDAT           ;SAVE RESULT IN $BDDAT
1262  005414  163737  005522  001126            SUB     10$,$BDDAT              ;SUB FIRST VALUE
1263  005422  013737  001126  015162            MOV     $BDDAT,ADNMUX           ;SAVE RESULTS
1264  005430  012737  000000  001124            MOV     #0,$GDDAT               ;LOAD EXPECTED ZERO SETTLING ERROR
1265  005436  012737  000077  011664            MOV     #77,SPREAD              ;LOAD TOLERANCE
1266  005444  004737  011666                    JSR     PC,COMPAR               ;TEST IF WITHIN LIMITS
1267  005450  000401                            BR      1$              ;;BR IF WITHIN
1268  005452  104011                            ERROR   11                      ;A/D NEGATIVE SETTLING ERROR
1269  005454  032777  010000  173454  1$:       BIT     #BIT12,@SWR             ;TEST FOR FORCED TYPEOUT
1270  005462  001420                            BEQ     TST31           ;;BR IF NOT FORCED
1271  005464  104400  013436                    TYPE    ADSMSG
1272  005470  013746  015160                    MOV     ADPMUX,-(SP)            ;SAVE MUX SETTLING
1273  005474  104402                            TYPOS
1274  005476     003     000                    .BYTE   3,0
1275  005500  104400  013525                    TYPE    ADEMSG
1276  005504  013746  015162                    MOV     ADNMUX,-(SP)            ;SAVE MUX SETTLING
1277  005510  104402                            TYPOS
1278  005512     003     000                    .BYTE   3,0
1279  005514  104400  016536                    TYPE    ACRLF
1280  005520  000401                            BR      TST31           ;;
1281
1282  005522  000000                    10$:    0
1283
```

```
1284                                    ;;*****************************************************************
1285                                    ;*TEST 31       A/D RMS NOISE TEST ON CHANNEL 7 - BIPOLAR
1286                                    ;;*****************************************************************
1287  005524  000004                    TST31:  SCOPE
1288  005526  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1289
1290  005534  004437  010716                    JSR     R4,SAR                  ;S A R  ROUTINE
1291  005540  001512                            VCYREG                  ;Y AXIS
1292  005542      000     007                   .BYTE   0,7                     ;CHANNEL 7, FINE Y
1293  005544  001000                            1000                    ;EDGE VALUE
1294  005546  000000                            0                       ;RMS RESULTS
1295
1296  005550  010537  001126                    MOV     R5,$BDDAT               ;SAVE RESULTS (2X RMS NOISE, 1=1/50 LSB)
1297  005554  010537  015130                    MOV     R5,CH7RMS               ;SAVE RESULTS (RMS NOISE, 1 = 1/100 LSB)
1298  005560  013737  015124  001124            MOV     RMSMAX,$GDDAT           ;LOAD EXPECTED
1299  005566  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1300  005574  002001                            BGE     TST32           ;;BR IF NOISE WITHIN SPEC
1301  005576  104006                            ERROR   6                       ;A/D RMS NOISE (BIPOLAR) IS GREATER
1302                                                                    ; THAN SPEC
1303
1304                                    ;;*****************************************************************
1305                                    ;*TEST 32       A/D PEAK NOISE TEST ON CHANNEL 7 - BIPOLAR
1306                                    ;;*****************************************************************
1307  005600  000004                    TST32:  SCOPE
1308  005602  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1309
1310  005610  004437  010716                    JSR     R4,SAR                  ;S A R  ROUTINE
1311  005614  001512                            VCYREG                  ;Y AXIS
1312  005616      000     007                   .BYTE   0,7                     ;CHANNEL 7, FINE Y
1313  005620  001000                            1000                    ;EDGE VALUE
1314  005622  100000                            BIT15                   ;PEAK RESULTS
1315
1316  005624  010537  001126                    MOV     R5,$BDDAT               ;SAVE RESULTS (PEAK NOISE, 1 = .01 LSB)
1317  005630  010537  015132                    MOV     R5,CH7PEK               ;SAVE RESULTS
1318  005634  013737  015126  001124            MOV     PEAKMAX,$GDDAT          ;LOAD EXPECTED
1319  005642  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1320  005650  002001                            BGE     TST33           ;;BR IF NOISE WITHIN SPEC
1321  005652  104006                            ERROR   6                       ;A/D PEAK NOISE (BIPOLAR) IS GREATER
1322                                                                    ; THAN SPEC
1323
```

```
1324                                    ;;*****************************************************************
1325                                    ;*TEST 33       A/D RMS NOISE TEST ON CHANNEL 47 - UNIPOLAR
1326                                    ;;*****************************************************************
1327  005654  000004                    TST33:  SCOPE
1328  005656  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1329
1330  005664  004437  010716                    JSR     R4,SAR                  ;S A R  ROUTINE
1331  005670  001512                            VCYREG                          ;Y AXIS
1332  005672     000     047                    .BYTE   0,CH47                  ;CHANNEL 47, FINE Y
1333  005674  000003                            3                               ;EDGE VALUE
1334  005676  000000                            0                               ;RMS RESULTS
1335
1336  005700  010537  001126                    MOV     R5,$BDDAT               ;SAVE RESULTS (RMS NOISE, 1=.01 LSB)
1337  005704  010537  015134                    MOV     R5,CH47RM               ;SAVE RESULT
1338  005710  013737  015124  001124            MOV     RMSMAX,$GDDAT           ;LOAD EXPECTED
1339  005716  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1340  005724  002001                            BGE     TST34           ;;BR IF NOISE WITHIN SPEC
1341  005726  104006                            ERROR   6                       ;A/D RMS NOISE (UNIPOLAR) IS GREATER
1342                                                                            ; THAN SPEC
1343
1344                                    ;;*****************************************************************
1345                                    ;*TEST 34       A/D PEAK NOISE TEST ON CHANNEL 47 - UNIPOLAR
1346                                    ;;*****************************************************************
1347  005730  000004                    TST34:  SCOPE
1348  005732  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1349
1350  005740  004437  010716                    JSR     R4,SAR                  ;S A R  ROUTINE
1351  005744  001512                            VCYREG                          ;Y AXIS
1352  005746     000     047                    .BYTE   0,CH47                  ;CHANNEL 47, FINE Y
1353  005750  000003                            3                               ;EDGE VALUE
1354  005752  100000                            BIT15                           ;PEAK RESULTS
1355
1356  005754  010537  001126                    MOV     R5,$BDDAT               ;SAVE RESULTS (PEAK NOISE, 1 = .01 LSB)
1357  005760  010537  015136                    MOV     R5,CH47PK               ;SAVE RESULT
1358  005764  013737  015126  001124            MOV     PEKMAX,$GDDAT           ;LOAD EXPECTED
1359  005772  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1360  006000  002001                            BGE     TST35           ;;BR IF NOISE WITHIN SPEC
1361  006002  104006                            ERROR   6                       ;A/D PEAK NOISE (UNIPOLAR) IS GREATER
1362                                                                            ; THAN SPEC
1363
```

```
1364                                    ;;*****************************************************************
1365                                    ;*TEST 35       X DAC RMS NOISE TEST ON CHANNEL 5
1366                                    ;;*****************************************************************
1367  006004  000004               TST35:  SCOPE
1368  006006  012737  000010  001164        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1369  006014  012777  001000  173466        MOV     #1000,@VCXREG           ;LOAD X POS
1370
1371  006022  004437  010716                JSR     R4,SAR                  ;S A R  ROUTINE
1372  006026  001512                       VCYREG                          ;Y AXIS
1373  006030     000     005                .BYTE   0,5                     ;CHANNEL 5, COARSE X AND FINE Y
1374  006032  001000                       1000                            ;EDGE VALUE
1375  006034  000000                       0                               ;RMS RESULTS
1376
1377  006036  010537  001126               MOV     R5,$BDDAT               ;SAVE RESULTS (X DAC + A/D RMS NOISE, 1=.01 LSB)
1378  006042  163737  015130  001126        SUB     CH7RMS,$BDDAT           ;XDAC RMS NOISE ONLY
1379  006050  013737  001126  015140        MOV     $BDDAT,XDACRM           ;SAVE RESULT
1380  006056  013737  015124  001124        MOV     RMSMAX,$GDDAT           ;LOAD EXPECTED
1381  006064  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1382  006072  002001                       BGE     TST36           ;;BR IF NOISE WITHIN SPEC
1383  006074  104006                       ERROR   6                       ;X D/A RMS NOISE  IS GREATER
1384                                                                        ; THAN SPEC
1385
1386                                    ;;*****************************************************************
1387                                    ;*TEST 36       X DAC PEAK NOISE TEST ON CHANNEL 5
1388                                    ;;*****************************************************************
1389  006076  000004               TST36:  SCOPE
1390  006100  012737  000010  001164        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1391
1392  006106  012777  001000  173374        MOV     #1000,@VCXREG           ;LOAD X DAC
1393  006114  004437  010716                JSR     R4,SAR                  ;S A R  ROUTINE
1394  006120  001512                       VCYREG                          ;Y AXIS
1395  006122     000     005                .BYTE   0,5                     ;CHANNEL 5, COARSE X AND FINE Y
1396  006124  001000                       1000                            ;EDGE VALUE
1397  006126  100000                       BIT15                           ;PEAK RESULTS
1398
1399  006130  010537  001126               MOV     R5,$BDDAT               ;SAVE RESULTS (X DAC + A/D PEAK NOISE, 1=.01 LSB
1400  006134  163737  015132  001126        SUB     CH7PEK,$BDDAT           ;X DAC PEAK NOISE ONLY
1401  006142  013737  001126  015142        MOV     $BDDAT,XDACPK           ;SAVE RESULT
1402  006150  013737  015126  001124        MOV     PEKMAX,$GDDAT           ;LOAD EXPECTED
1403  006156  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1404  006164  002001                       BGE     TST37           ;;BR IF NOISE WITHIN SPEC
1405  006166  104006                       ERROR   6                       ;X D/A PEAK NOISE  IS GREATER
1406                                                                        ; THAN SPEC
1407
```

```
1408                                    ;;*****************************************************************
1409                                    ;*TEST 37       Y DAC RMS NOISE TEST ON CHANNEL 6
1410                                    ;;*****************************************************************
1411  006170  000004                    TST37:  SCOPE
1412  006172  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1413
1414  006200  012777  001000  173304            MOV     #1000,@VCYREG           ;LOAD Y REG
1415  006206  004437  010716                    JSR     R4,SAR                  ;S A R  ROUTINE
1416  006212  001510                            VCXREG                          ;X AXIS
1417  006214     000     006                    .BYTE   0,6                     ;CHANNEL 6 ,COARSE Y AND FINE X
1418  006216  001000                            1000                            ;EDGE VALUE
1419  006220  000000                            0                               ;RMS RESULTS
1420
1421  006222  010537  001126                    MOV     R5,$BDDAT               ;SAVE RESULTS (Y DAC + A/D RMS NOISE, 1=.01 LSB)
1422  006226  163737  015130  001126            SUB     CH7RMS,$BDDAT           ;Y DAC RMS NOISE ONLY
1423  006234  013737  001126  015144            MOV     $BDDAT,YDACRM           ;SAVE RESULT
1424  006242  013737  015124  001124            MOV     RMSMAX,$GDDAT           ;LOAD EXPECTED
1425  006250  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1426  006256  002001                            BGE     TST40           ;;BR IF NOISE WITHIN SPEC
1427  006260  104006                            ERROR   6                       ;Y D/A RMS NOISE  IS GREATER
1428                                                                            ; THAN SPEC
1429
1430                                    ;;*****************************************************************
1431                                    ;*TEST 40       Y DAC PEAK NOISE TEST ON CHANNEL 6
1432                                    ;;*****************************************************************
1433  006262  000004                    TST40:  SCOPE
1434  006264  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1435
1436  006272  012777  001000  173212            MOV     #1000,@VCYREG           ;LOAD Y AXIS
1437  006300  004437  010716                    JSR     R4,SAR                  ;S A R  ROUTINE
1438  006304  001510                            VCXREG                          ;X AXIS
1439  006306     000     006                    .BYTE   0,6                     ;CHANNEL 6, COARSE Y AND FINE X
1440  006310  001000                            1000                            ;EDGE VALUE
1441  006312  100000                            BIT15                           ;PEAK RESULTS
1442
1443  006314  010537  001126                    MOV     R5,$BDDAT               ;SAVE RESULTS (Y DAC + A/D PEAK NOISE, 1=.01 LSB
1444  006320  163737  015132  001126            SUB     CH7PEK,$BDDAT           ;Y DAC PEAK NOISE ONLY
1445  006326  013737  001126  015146            MOV     $BDDAT,YDACPK           ;SAVE RESULT
1446  006334  013737  015126  001124            MOV     PEKMAX,$GDDAT           ;LOAD EXPECTED
1447  006342  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1448  006350  002001                            BGE     1$              ;;BR IF NOISE WITHIN SPEC
1449  006352  104006                            ERROR   6                       ;Y DAC PEAK NOISE  IS GREATER
1450                                                                            ; THAN SPEC
1451
1452  006354  032777  010000  172554  1$:       BIT     #BIT12,@SWR             ;TEST FORCED PRINTOUT
1453  006362  001432                            BEQ     TST41           ;;BR IF NOT FORCED
1454  006364  104400  013543                    TYPE    ,NOIMSG
1455  006370  004537  011466                    JSR     R5,BYTWO                ;TYPE 2 OCTAL COL.
1456  006374  015130                            CH7RMS
1457  006376  015132                            CH7PEK
1458  006400  104400  013664                    TYPE    ,NOIMG1
1459  006404  004537  011466                    JSR     R5,BYTWO                ;TYPE 2 OCTAL COL.
1460  006410  015134                            CH47RM
1461  006412  015136                            CH47PK
1462  006414  104400  013707                    TYPE    ,NOIMG2
1463  006420  004537  011466                    JSR     R5,BYTWO                ;TYPL 2 OCTAL COL.
```

```
                                        XDACRM
                                        XDACPK
                                        TYPE    ,NOIMG3
                                        JSR     R5,BYTWO
                                        YDACRM
                                        YDACPK
                                        TYPE    ,ACRLF

                        ;;**************************************************************
                        ;*TEST 41        BIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
                        ;;**************************************************************
TST41:  SCOPE
        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
        JSR     R4,$AR                  ;SOFTWARE S.A.R.
        VCYREG
        .BYTE   0,CH7                   ;CHANNEL 7, FINE Y
        1001                            ;EDGE, 1000/1001 TRANSITION
        BIT7                            ;50-50

        MOV     DACSAV,OFSTBX           ;SAVE A/D OFFSET BIPOLAR
        MOV     DACSAV,$BDDAT           ;LOAD VALUE READ
        MOV     #1014,$GDDAT            ;LOAD EXPECTED
        MOV     V62,$PREAD              ;LOAD +- VARIABLE, 1LSB
        JSR     PC,COMPAR               ;TEST IF WITHIN +- VALUE
        BR      TST42           ;;BR IF WITHIN LIMITS
        ERROR   12                      ;A/D BIPOLAR OFFSET ERROR

                        ;;**************************************************************
                        ;*TEST 42        UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
                        ;;**************************************************************
TST42:  SCOPE
        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
        JSR     R4,$AR                  ;SOFTWARE S.A.R.
        VCYREG
        .BYTE   0,CH47                  ;CHANNEL 7, UNIPOLAR FINE Y
        1                               ;EDGE, 0/1 TRANSITION
        BIT7                            ;50-50

        MOV     DACSAV,OFSTUX           ;SAVE A/D OFFSET UNIPOLAR
        MOV     DACSAV,$BDDAT           ;LOAD VALUE READ
        MOV     #1014,$GDDAT            ;LOAD EXPECTED
        MOV     V62,$PREAD              ;LOAD +- VARIABLE, 1 LSB
        JSR     PC,COMPAR               ;TEST IF WITHIN +- VALUE
        BR      TST43           ;;BR IF WITHIN LIMITS
        ERROR   12                      ;A/D UNIPOLAR OFFSET ERROR
```

```
1509                                    ;;*****************************************************************
1510                                    ;*TEST 43       BIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
1511                                    ;;*****************************************************************
1512  006620  000004                    TST43:  SCOPE
1513  006622  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1514  006630  004437  010716                    JSR     R4,SAR                  ;SOFTWARE S.A.R.
1515  006634  001512                            VCYREG
1516  006636     000     007                    .BYTE   0,CH7                           ;CHANNEL 7, FINE Y
1517  006640  001001                            1001                    ;EDGE, 1000/1001 TRANSITION
1518  006642  000201                            BIT7!BIT0               ;50-50, WAIT LOOP
1519
1520  006644  013737  015122  015154            MOV     DACSAV,OFSLBX           ;SAVE A/D OFFSET BIPOLAR
1521  006652  013737  015122  001126            MOV     DACSAV,$BDDAT           ;LOAD VALUE READ
1522  006660  012737  001014  001124            MOV     #1014,$GDDAT            ;LOAD EXPECTED
1523  006666  013737  016774  011664            MOV     V62,SPREAD              ;LOAD +- VARIABLE, 1 LSB
1524  006674  004737  011666                    JSR     PC,COMPAR               ;TEST IF WITHIN +- VALUE
1525  006700  000401                            BR      1$              ;;BR IF WITHIN LIMITS
1526  006702  104012                            ERROR   12                      ;A/D BIPOLAR OFFSET ERROR
1527
1528  006704  163737  015150  001126  1$:       SUB     OFSTBX,$BDDAT           ;OFFSET DUE TO WAIT LOOP
1529  006712  005037  001124                    CLR     $GDDAT                  ;
1530  006716  012737  000031  011664            MOV     #31,SPREAD              ;1/2 LSB ALLOWED
1531  006724  004737  011666                    JSR     PC,COMPAR               ;WITHIN LIMITS
1532  006730  000401                            BR      TST44           ;;BR IF WITHIN LIMITS
1533  006732  104012                            ERROR   12                      ;A/D BIPOLAR OFFSET ERROR DUE TO WAIT LOOP
1534
1535                                    ;;*****************************************************************
1536                                    ;*TEST 44       UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
1537                                    ;;*****************************************************************
1538  006734  000004                    TST44:  SCOPE
1539  006736  012737  000010  001164            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1540  006744  004437  010716                    JSR     R4,SAR                  ;SOFTWARE S.A.R.
1541  006750  001512                            VCYREG
1542  006752     000     047                    .BYTE   0,CH47                  ;CHANNEL 7, UNIPOLAR, FINE Y
1543  006754  000001                            1                       ;EDGE, 0/1 TRANSITION
1544  006756  000201                            BIT7!BIT0               ;50-50, WAIT LOOP
1545
1546  006760  013737  015122  015156            MOV     DACSAV,OFSLUX           ;SAVE A/D OFFSET UNIPOLAR
1547  006766  013737  015122  001126            MOV     DACSAV,$BDDAT           ;LOAD VALUE READ
1548  006774  012737  001014  001124            MOV     #1014,$GDDAT            ;LOAD EXPECTED
1549  007002  013737  016774  011664            MOV     V62,SPREAD              ;LOAD +- VARIABLE, 1 LSB
1550  007010  004737  011666                    JSR     PC,COMPAR               ;TEST IF WITHIN +- VALUE
1551  007014  000401                            BR      1$              ;;BR IF WITHIN LIMITS
1552  007016  104012                            ERROR   12                      ;A/D UNIPOLAR OFFSET ERROR
1553
1554  007020  163737  015152  001126  1$:       SUB     OFSTUX,$BDDAT           ;OFFSET DUE TO WAIT LOOP
1555  007026  005037  001124                    CLR     $GDDAT                  ;
1556  007032  012737  000031  011664            MOV     #31,SPREAD              ;1/2 LSB ALLOWED
1557  007040  004737  011666                    JSR     PC,COMPAR               ;WITHIN LIMITS
1558  007044  000401                            BR      TST45           ;;BR IF WITHIN LIMITS
1559  007046  104012                            ERROR   12                      ;A/D UNIPOLAR OFFSET ERROR DUE TO WAIT LOOP
```

```
1560                                    ;;*****************************************************************
1561                                    ;*TEST 45        X D/A OFFSET USING CH 5
1562                                    ;;*****************************************************************
1563  007050  000004                   TST45:  SCOPE
1564  007052  012737  000010  001164           MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1565  007060  012777  001000  172422           MOV     #1000,@VCXREG           ;LOAD X REG
1566  007066  004437  010716                   JSR     R4,SAR                  ;SOFTWARE S.A.R.
1567  007072  001512                           VCYREG                          ;Y AXIS
1568  007074     000     005                   .BYTE   0,CH5                   ;CHANNEL 5, COARSE X AND FINE Y
1569  007076  001001                           1001                            ;EDGE VALUE, 1000/1001 TRANSITION
1570  007100  000200                           BIT7                            ;50-50
1571  007102  013737  015122  001126           MOV     DACSAV,$BDDAT           ;SAVE RESULT (X DAC + A/D OFFSET)
1572  007110  163737  015150  001126           SUB     OFSTBX,$BDDAT           ;OFFSET DUE TO X DAC
1573  007116  005037  001124                   CLR     $GDDAT                  ;CLEAR EXPECTED
1574  007122  013737  016776  011664           MOV     V77,SPREAD              ;TOLERANCE DUE TO X DAC
1575  007130  004737  011666                   JSR     PC,COMPAR               ;TEST IF WITHIN +- LIMITS
1576  007134  000401                           BR      TST46           ;;BR IF WITHIN LIMITS
1577  007136  104012                           ERROR   12                      ;X DAC OFFSET ERROR
1578
1579                                    ;;*****************************************************************
1580                                    ;*TEST 46        Y D/A OFFSET USING CH 6
1581                                    ;;*****************************************************************
1582  007140  000004                   TST46:  SCOPE
1583  007142  012737  000010  001164           MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1584  007150  012777  001000  172334           MOV     #1000,@VCYREG           ;LOAD Y REG
1585  007156  004437  010716                   JSR     R4,SAR                  ;SOFTWARE S.A.R.
1586  007162  001510                           VCXREG                          ;X AXIS
1587  007164     000     006                   .BYTE   0,CH6                   ;CHANNEL 6, COARSE Y AND FINE X
1588  007166  001001                           1001                            ;EDGE VALUE, 1000/1001 TRANSITION
1589  007170  000200                           BIT7                            ;50-50
1590  007172  013737  015122  001126           MOV     DACSAV,$BDDAT           ;SAVE RESULT (Y DAC + A/D OFFSET)
1591  007200  163737  015150  001126           SUB     OFSTBX,$BDDAT           ;OFFSET DUE TO Y DAC
1592  007206  005037  001124                   CLR     $GDDAT                  ;CLEAR EXPECTED
1593  007212  013737  016776  011664           MOV     V77,SPREAD              ;TOLERANCE DUE TO Y DAC
1594  007220  004737  011666                   JSR     PC,COMPAR               ;TEST IF WITHIN +- LIMITS
1595  007224  000401                           BR      TST47           ;;BR IF WITHIN LIMITS
1596  007226  104012                           ERROR   12                      ;Y DAC OFFSET ERROR
```

```
                                        ;;*****************************************************************
                                        ;*TEST 47        CALIBRATION USING CHANNEL 11 - X DAC VS. A/D
                                        ;;*****************************************************************
007230  000004                  TST47:  SCOPE
007232  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
007240  012777  001300  172242          MOV     #1300,@VCXREG           ;LOAD X DAC
007246  012737  001600  001124          MOV     #1600,$GDDAT            ;LOAD EXPECTED
007254  004537  011534            2$:   JSR     R5,CONVRT               ;CONVERT 32X AND AVERAGE
007260  000011                          CH11
007262  013737  011656  001126          MOV     ADEND,$BDDAT            ;READ VALUE OBTAINED
007270  012737  000003  011664          MOV     #3,SPREAD               ;LOAD DEVIATION
007276  004737  011666                  JSR     PC,COMPAR               ;TEST IF WITHIN
007302  000401                          BR      1$                      ;BR IF WITHIN
007304  104013                          ERROR   13                      ;CALIBRATION ERROR - X DAC VS. A/D
007306  162777  000100  172174  1$:     SUB     #100,@VCXREG            ;DECREMENT DAC
007314  162737  000200  001124          SUB     #200,$GDDAT             ;DEC. EXPECTED
007322  100354                          BPL     2$                      ;BR UNTIL DONE

                                        ;;*****************************************************************
                                        ;*TEST 50        CALIBRATION USING CHANNEL 12 - Y DAC VS. A/D
                                        ;;*****************************************************************
007324  000004                  TST50:  SCOPE
007326  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
007334  012777  001300  172150          MOV     #1300,@VCYREG           ;LOAD Y DAC
007342  012737  001600  001124          MOV     #1600,$GDDAT            ;LOAD EXPECTED
007350  004537  011534            2$:   JSR     R5,CONVRT               ;CONVERT 32X AND AVERAGE
007354  000012                          CH12
007356  013737  011656  001126          MOV     ADEND,$BDDAT            ;READ VALUE OBTAINED
007364  012737  000003  011664          MOV     #3,SPREAD               ;LOAD DEVIATION
007372  004737  011666                  JSR     PC,COMPAR               ;TEST IF WITHIN
007376  000401                          BR      1$                      ;BR IF WITHIN
007400  104013                          ERROR   13                      ;CALIBRATION ERROR - Y DAC VS. A/D
007402  162777  000100  172102  1$:     SUB     #100,@VCYREG            ;DECREMENT DAC
007410  162737  000200  001124          SUB     #200,$GDDAT             ;DEC. EXPECTED
007416  100354                          BPL     2$                      ;BR UNTIL DONE
```

```
                                        ;;*****************************************************************
                                        ;*TEST 51       LINEARITY TEST USING CH 6 - Y DAC VS. A/D
                                        ;;*****************************************************************
007420  000004                  TST51:  SCOPE
007422  012737  000010  001164          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
007430  012700  014702                  MOV     #NUMBF2,R0              ;LOAD EDGE VALUE POINTER
007434  012702  014726                  MOV     #RSLT0,R2               ;LOAD RESULT POINTER

007440  011037  007460          1$:     MOV     (R0),10$                ;LOAD EDGE EXPECTED
007444  012077  172042                  MOV     (R0)+,@VCYREG           ;LOAD DAC
007450  004437  010716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
007454  001510                          VCXREG
007456     000     006                  .BYTE   0,CH6                   ;CHANNEL 6, COARSE Y AND FINE X
007460  000000                  10$:    0                               ;EDGE VALUE
007462  000200                          BIT7                            ;50-50

007464  013722  015122                  MOV     DACSAV,(R2)+            ;SAVE RESULTS
007470  005710                          TST     (R0)                    ;LAST RESULT ?
007472  100362                          BPL     1$                      ;BR UNTIL DONE
007474  005037  001124                  CLR     $GDDAT                  ;ZERO NOMINAL LINEARITY ERROR
007500  013737  016776  011664          MOV     V77,SPREAD              ;TOLERANCE FOR A/D + DAC
007506  013700  014746                  MOV     RSLT0+20,R0             ;GET LAST VALUE
007512  163700  014726                  SUB     RSLT0,R0                ;SUBTRACT FIRST VALUE
007516  006200                          ASR     R0
007520  006200                          ASR     R0
007522  006200                          ASR     R0
007524  005500                          ADC     R0                      ;AVERAGE DIFFERENCE IN R0
007526  012701  000002                  MOV     #2,R1
007532  005037  014746                  CLR     RSLT0+20                ;START RUNNING SUM OF ERRORS
007536  060037  014726          2$:     ADD     R0,RSLT0                ;COMPUTE NOM. VALUE ON LINE BETW. END PTS.
007542  163761  014726  014726          SUB     RSLT0,RSLT0(R1)         ;COMPUTE ERROR WITH RESP. TO END PT. LINE
007550  066137  014726  014746          ADD     RSLT0(R1),RSLT0+20      ;KEEP RUNNING SUM
007556  005721                          TST     (R1)+                   ;BUMP R1
007560  020127  000020                  CMP     R1,#20                  ;DONE ?
007564  001364                          BNE     2$              ;BR IF NOT
007566  006237  014746                  ASR     RSLT0+20
007572  006237  014746                  ASR     RSLT0+20
007576  006237  014746                  ASR     RSLT0+20
007602  005537  014746                  ADC     RSLT0+20                ;AVERAGE ERROR WITH RESP. TO END PT. LINE
007606  005037  014726                  CLR     RSLT0
007612  005001                          CLR     R1
007614  163761  014746  014726  3$:     SUB     RSLT0+20,RSLT0(R1)      ;ERROR WITH RESP. TO "BEST STRAIGHT LINE"

007622  016137  014726  001126          MOV     RSLT0(R1),$BDDAT
007630  004737  011666                  JSR     PC,COMPAR               ;COMPARE LINEARITY ERROR WITH + OR - TOLERANCE
007634  000422                          BR      4$
007636  010102                          MOV     R1,R2
007640  042702  177770                  BIC     #177770,R2              ;MASK BITS 0-2
007644  062702  000060                  ADD     #60,R2                  ;CONVERT TO ASCII
007650  110237  012445                  MOVB    R2,EM14A
007654  010102                          MOV     R1,R2                   ;LOAD R2
007656  006202                          ASR     R2
007660  006202                          ASR     R2
007662  006202                          ASR     R2
007664  042702  177770                  BIC     #177770,R2              ;MASK
007670  062702  000060                  ADD     #60,R2                  ;MAKE ASCII
```

```
1688  007674  110237  012444              MOVB    R2,EM14B                ;SAVE #
1689  007700  104014                      ERROR   14                      ;LINEARITY ERROR USING CH 6 - YDAC VS. A/D
1690  007702  005721              4$:     TST     (R1)+                   ;BUMP R1
1691  007704  020127  000020              CMP     R1,#20                  ;DONE ?
1692  007710  001341                      BNE     3$              ;;BR IF NOT DONE
1693
1694                                      ;;**********************************************************************
1695                                      ;*TEST 52        LINEARITY TEST USING CH 5 - X DAC VS. A/D
1696                                      ;;**********************************************************************
1697  007712  000004              TST52:  SCOPE
1698  007714  012737  000010  001164      MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1699  007722  012700  014702              MOV     #NUMBF2,R0              ;LOAD EDGE VALUE POINTER
1700  007726  012702  014764              MOV     #RSLT1,R2               ;LOAD RESULT POINTER
1701
1702  007732  011037  007752      1$:     MOV     (R0),10$                ;LOAD EDGE EXPECTED
1703  007736  012077  171546              MOV     (R0)+,@VCXREG           ;LOAD DAC
1704  007742  004437  010716              JSR     R4,SAR                  ;SOFTWARE S.A.R.
1705  007746  001512                      VCYREG
1706  007750     000     005              .BYTE   0,CH5                   ;CHANNEL 5, COARSE X AND FINE Y
1707  007752  000000              10$:    0                               ;EDGE VALUE
1708  007754  000200                      BIT7                                    ;50-50
1709
1710  007756  013722  015122              MOV     DACSAV,(R2)+            ;SAVE RESULTS
1711  007762  005710                      TST     (R0)                    ;LAST RESULT ?
1712  007764  100362                      BPL     1$                      ;BR UNTIL DONE
1713  007766  005037  001124              CLR     $GDDAT                  ;ZERO NOMINAL LINEARITY ERROR
1714  007772  013737  016776  011664      MOV     V77,SPREAD              ;TOLERANCE FOR A/D + DAC
1715  010000  013700  015004              MOV     RSLT1+20,R0             ;GET LAST VALUE
1716  010004  163700  014764              SUB     RSLT1,R0                ;SUBTRACT FIRST VALUE
1717  010010  006200                      ASR     R0
1718  010012  006200                      ASR     R0
1719  010014  006200                      ASR     R0
1720  010016  005500                      ADC     R0                      ;AVERAGE DIFFERENCE IN R0
1721  010020  012701  000002              MOV     #2,R1
1722  010024  005037  015004              CLR     RSLT1+20                ;START RUNNING SUM OF ERRORS
1723  010030  060037  014764      2$:     ADD     R0,RSLT1                ;COMPUTE NOM. VALUE ON LINE BETW. END PTS.
1724  010034  163761  014764  014764      SUB     RSLT1,RSLT1(R1)         ;COMPUTE ERROR WITH RESP. TO END PT. LINE
1725  010042  066137  014764  015004      ADD     RSLT1(R1),RSLT1+20      ;KEEP RUNNING SUM
1726  010050  005721                      TST     (R1)+                   ;BUMP R1
1727  010052  020127  000020              CMP     R1,#20                  ;DONE ?
1728  010056  001364                      BNE     2$                      ;BR IF NOT
1729  010060  006237  015004              ASR     RSLT1+20
1730  010064  006237  015004              ASR     RSLT1+20
1731  010070  006237  015004              ASR     RSLT1+20
1732  010074  005537  015004              ADC     RSLT1+20                ;AVERAGE ERROR WITH RESP. TO END PT. LINE
1733  010100  005037  014764              CLR     RSLT1
1734  010104  005001                      CLR     R1
1735  010106  163761  015004  014764  3$: SUB     RSLT1+20,RSLT1(R1)      ;ERROR WITH RESP. TO "BEST STRAIGHT LINE"
1736  010114  016137  014764  001126      MOV     RSLT1(R1),$BDDAT
1737  010122  004737  011666              JSR     PC,COMPAR               ;COMPARE LINEARITY ERROR WITH + OR - TOLERANCE
1738  010126  000422                      BR      4$
1739  010130  010102                      MOV     R1,R2
1740  010132  042702  177770              BIC     #177770,R2              ;MASK BITS 0-2
1741  010136  062702  000060              ADD     #60,R2                  ;CONVERT TO ASCII
1742  010142  110237  012445              MOVB    R2,EM14A
1743  010146  010102                      MOV     R1,R2                   ;LOAD R2
```

```
                                        ASR     R2
                                        ASR     R2
                                        ASR     R2
                                        BIC     #177770,R2              ;MASK
                                        ADD     #60,R2                  ;MAKE ASCII
                                        MOVB    R2,EM14B                ;SAVE #
                                        ERROR   14                      ;LINEARITY ERROR USING CH 5 - XDAC VS. A D
                                4$:     TST     (R1)+                   ;BUMP R1
                                        CMP     R1,#20                  ;DONE ?
                                        BNE     3$              ;;BR IF NOT DONE
                                        BIT     #BIT12,@SWR             ;TEST IF FORCED TYPEOUT
                                        BEQ     TST53           ;;BR IF NOT FORCED
                                        TYPE    ,LINMSG
                                        JSR     R5,TYPSTG               ;TYPE OCTAL STRING
                                        RSLT0                           ;STARTING AT
                                        8.                              ; # OF LOCATIONS
                                        TYPE    ,LINMG1
                                        JSR     R5,TYPSTG               ;TYPE OCTAL STRING
                                        RSLT1                           ;STARTING AT
                                        8.                              ; # OF LOCATIONS
                                        TYPE    ,ACRLF

                                ;;**************************************************************
                                ;*TEST 53       DETERMINE IF MORE AR11'S ARE TO BE TESTED
                                ;;**************************************************************
                                TST53:  SCOPE
                                        MOV     #1,$TIMES       ;;DO 1 ITERATION
                                        TST     NBEXT                   ;TEST IF ANY
                                        BEQ     1$                      ;BR IF NONE
                                        ADD     #20,ARBADD              ;UPDATE DEVICE ADDRESS
                                        ADD     #20,ARBVCT              ;UPDATE DEVICE VECTOR
                                        DEC     NBEXT                   ;ANOTHER ONE ?
                                        BR      BYPAS1                  ;BR IF ANOTHER
                                1$:     MOV     $BASE,ARBADD            ;RELOAD ADDRESS
                                        MOV     $VECT1,ARBVCT           ;RELOAD VECTOR
                                        MOV     NMBEXT,NBEXT            ;RELOAD NUMBER
                                        RESET
                                        JMP     $EOP                    ;DONE
                                BYPAS1: MOV     #-1,R0
                                        JMP     RBEG2                   ;TEST ANOTHER UNIT
                                        .SBTTL
```

```
1795
1796
1797                                    .SBTTL  END OF PASS ROUTINE
1798
1799
1800                                    ;;**********************************************************************
1801                                    ;*INCREMENT THE PASS NUMBER ($PASS)
1802                                    ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
1803                                    ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
1804                                    ;*IF THERES A MONITOR GO TO IT
1805                                    ;*IF THERE ISN'T JUMP TO BYPAS1
1806
1807  010350                            $EOP:
1808  010350  000004                           SCOPE
1809  010352  005037  001102                   CLR     $TSTNM          ;;ZERO THE TEST NUMBER
1810  010356  005037  001164                   CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
1811  010362  005237  001206                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
1812  010366  042737  100000  001206          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
1813  010374  005327                           DEC     (PC)+           ;;LOOP?
1814  010376  000001            $EOPCT: .WORD   1
1815  010400  003022                           BGT     $DOAGN          ;;YES
1816  010402  012737                           MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
1817  010404  000001            $ENDCT: .WORD   1
1818  010406  010376                           $EOPCT
1819  010410  104400  010455                   TYPE    $ENDMG          ;;TYPE "END PASS #"
1820  010414  013746  001206                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
1821  010420  104404                           TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
1822  010422  104400  010452                   TYPE    $ENULL          ;;TYPE A NULL CHARACTER
1823  010426  013700  000042            $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
1824  010432  001405                           BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
1825  010434  000005                           RESET                   ;;CLEAR THE WORLD
1826  010436  004710                    $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
1827  010440  000240                           NOP                     ;;SAVE ROOM
1828  010442  000240                           NOP                     ;;FOR
1829  010444  000240                           NOP                     ;;ACT11
1830  010446                            $DOAGN:
1831  010446  000137                           JMP     @(PC)+          ;;RETURN
1832  010450  010340                    $RTNAD: .WORD   BYPAS1
1833  010452      377       377       000    $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
1834  010455      015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
1835  010462  050040  051501  020123
1836  010470  000043
1837
```

```
                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;*REPLACED WITH SPACES.
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                ;*      TYPDS                   ;;GO TO THE ROUTINE

010472                          $TYPDS:
010472  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
010474  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
010476  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
010500  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
010502  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
010504  012746  020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
010510  016605  000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
010514  100004                         BPL     1$              ;;BR IF INPUT IS POS.
010516  005405                         NEG     R5              ;;MAKE THE BINARY NUMBER POS.
010520  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
010526  005000                 1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
010530  012703  010706                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
010534  112723  000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
010540  005002                 2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
010542  016001  010676                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
010546  160105                 3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
010550  002402                         BLT     4$              ;;BR IF DONE
010552  005202                         INC     R2              ;;INCREASE THE BCD DIGIT BY 1
010554  000774                         BR      3$
010556  060105                 4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
010560  005702                         TST     R2              ;;CHECK IF BCD DIGIT=0
010562  001002                         BNE     5$              ;;FALL THROUGH IF 0
010564  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
010566  100407                         BMI     7$              ;;BR IF YES
010570  106316                 5$:     ASLB    (SP)            ;;MSD?
010572  103003                         BCC     6$              ;;BR IF NO
010574  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
010602  052702  000060         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
010606  052702  000040         7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
010612  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
010614  005720                         TST     (R0)+           ;;JUST INCREMENTING
010616  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
010622  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
010624  003002                         BGT     8$              ;;GO TO EXIT
010626  010502                         MOV     R5,R2           ;;GET THE LSD
010630  000764                         BR      6$              ;;GO CHANGE TO ASCII
010632  105726                 8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
010634  100003                         BPL     9$              ;;BR IF NO
010636  116663  177777  177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
010644  105013                 9$:     CLRB    (R3)            ;;SET THE TERMINATOR
010646  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
```

```
 1883  010650  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 1884  010652  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
 1885  010654  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 1886  010656  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 1887  010660  104400  010706                  TYPE    $DBLK           ;;NOW TYPE THE NUMBER
 1888  010664  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
 1889  010672  012616                          MOV     (SP)+,(SP)
 1890  010674  000002                          RTI                     ;;RETURN TO USER
 1891  010676  023420                  $DTBL:  10000.
 1892  010700  001750                          1000.
 1893  010702  000144                          100.
 1894  010704  000012                          10.
 1895  010706  000004                  $DBLK:  .BLKW   4
 1896                                          .SBTTL  SUCCESSIVE APPROX. SUBROUTINE
```

```
                                        ;S A R SUBROUTINE
                                        ;       JSR     R4,SAR
                                        ;       AXIS POINTER
                                        ;       CH. IN HIGH BYTE / DUMMY CH. IN LOW BYTE
                                        ;       EDGE VALUE
                                        ;       RESULT/MODE
                                        ;       RETURN WITH R5 = DIFFERENCE
                                        ;
                                        ;       BIT15 = 0       RMS
                                        ;       BIT15 = 1       PEAK
                                        ;       BIT7 = 1        50-50
                                        ;       BIT0 = 1        ;WAIT LOOP
010716  012437  015120          SAR:    MOV     (R4)+,ADAC              ;SAVE DAC ADDRESS POINTER
010722  012437  015166                  MOV     (R4)+,SARCHN            ;SAVE CHANNEL
010726  012437  015170                  MOV     (R4)+,EDGE              ;SAVE VALUE EDGE
010732  012437  015172                  MOV     (R4)+,CONS              ;SAVE RMS/PEAK SW
010736  010046                          MOV     R0,-(SP)                ;SAVE GPR
010740  010146                          MOV     R1,-(SP)
010742  010246                          MOV     R2,-(SP)
010744  010346                          MOV     R3,-(SP)
010746  113737  015167  011662          MOVB    SARCHN+1,CHANL
010754  005037  011660                  CLR     FCHANL
010760  113737  015167  011661          MOVB    SARCHN+1,FCHANL+1
010766  005237  011660                  INC     FCHANL                  ;SET START BIT
010772  017737  004122  015120          MOV     @ADAC,ADAC              ;GET BUS ADDRESS OF ACTIVE DAC
011000  012777  011132  170460          MOV     #11$,@ARBVCT            ;LOAD VECTOR
011006  005037  015174                  CLR     FINS                    ;CLEAR 2 PASS COUNTER FLAG
011012  005737  015172                  TST     CONS                    ;TEST IF RMS
011016  100003                          BPL     10$                     ;BR IF RMS
011020  012703  000002                  MOV     #2,R3                   ;LOAD A = .4% OF 512
011024  000407                          BR      15$
011026  012703  000121          10$:    MOV     #121,R3                 ;LOAD A = 16% OF 512
011032  105737  015172                  TSTB    CONS                    ;TEST FOR 50-50
011036  100002                          BPL     15$
011040  012703  000400                  MOV     #400,R3                 ;LOAD 50-50, A = 50% COUNT
011044  012702  001000          15$:    MOV     #1000,R2                ;SET UP MSB
011050  005077  004044                  CLR     @ADAC                   ;CLEAR DAC
011054  005001                  14$:    CLR     R1                      ;SET HIGH = 0
011056  012700  001000                  MOV     #1000,R0                ;SET UP 512 CONVERSIONS
011062  060277  004032                  ADD     R2,@ADAC                ;NEXT BIT OF ACTIVE DAC
011066  032737  000001  015172  6$:     BIT     #BIT0,CONS              ;TEST BIT 0
011074  001407                          BEQ     20$
011076  013777  011660  170370          MOV     FCHANL,@ADCS            ;START CONVERSION
011104  105777  170364          27$:    TSTB    @ADCS
011110  100375                          BPL     27$
011112  000410                          BR      12$
011114  052737  000100  011660  20$:    BIS     #BIT6,FCHANL            ;SET INTR. ENABLE
011122  013777  011660  170344          MOV     FCHANL,@ADCS            ;START CONVERSION
011130  000001                          WAIT
011132  022626                  11$:    CMP     (SP)+,(SP)+
011134  027737  170340  015170  12$:    CMP     @ADDBR,EDGE             ;IS RESULT < EDGE VALUE ?
011142  002401                          BLT     1$                      ;YES, BR
011144  005201                          INC     R1                      ;NO, INC HIGH FOR RESULT > OR = EDGE VALUE
011146  105737  015166          1$:     TSTB    SARCHN                  ;TEST IF DUMMY CH. IS SET
011152  001420                          BEQ     3$                      ;BR IF NOT
```

```
 1953  011154  012777  011176  170304          MOV     #4$,@ARBVCT             ;LOAD VECTOR
 1954  011162  113777  015166  170306          MOVB    SARCHN,@ADCS1           ;LOAD MUX
 1955  011170  005277  170300                  INC     @ADCS                   ;CONVERT
 1956  011174  000001                          WAIT
 1957  011176  022626                  4$:     CMP     (SP)+,(SP)+             ;POP STACK
 1958  011200  027777  170274  170272  13$:    CMP     @ADDBR,@ADDBR           ;FAKE READ
 1959  011206  012777  011132  170252          MOV     #11$,@ARBVCT            ;RESET VECTOR
 1960  011214  005300                  3$:     DEC     R0                      ;DONE 512 TIMES ?
 1961  011216  001323                          BNE     6$                      ;NO
 1962  011220  020103                          CMP     R1,R3                   ;IS HIGH > OR = A ?
 1963  011222  002402                          BLT     2$                      ;NO, LEAVE BIT ON
 1964  011224  160277  003670                  SUB     R2,@ADAC                ;YES, TAKE IT OUT
 1965  011230  022702  000001          2$:     CMP     #1,R2                   ;TEST FOR Z = 1, S.A.R. DONE ?
 1966  011234  001402                          BEQ     21$                     ;BR IF DONE
 1967  011236  006202                          ASR     R2                      ;NO SET Z /2
 1968  011240  000705                          BR      14$                     ;TRY AGAIN
 1969  011242  017737  003652  015122  21$:    MOV     @ADAC,DACSAV
 1970  011250  005737  015174                  TST     FINS                    ;FIRST OR SECOND TIME ?
 1971  011254  001020                          BNE     45$                     ;SECOND, BR
 1972  011256  017705  003636                  MOV     @ADAC,R5                ;READ VALUE
 1973  011262  005237  015174                  INC     FINS                    ;SET FLAG
 1974  011266  005737  015'^2                  TST     CONS                    ;TEST IF PEAK/RMS
 1975  011272  100003                          BPL     41$                     ;BR IF RMS
 1976  011274  012703  000776                  MOV     #776,R3                 ;LOAD 99.6% OF 512
 1977  011300  000405                          BR      42$
 1978  011302  012703  000657          41$:    MOV     #657,R3                 ;SET A = 84% OF 512
 1979  011306  105737  015172          43$:    TSTB    CONS
 1980  011312  100404                          BMI     5$
 1981  011314  000653                  42$:    BR      15$                     ;DO MORE TESTING
 1982  011316  167705  003576          45$:    SUB     @ADAC,R5                ;SUBTRACT DIFF
 1983  011322  005405                          NEG     R5
 1984  011324  005077  170144          5$:     CLR     @ADCS                   ;RETURN WITH 2 X NOISE IN R5
 1985  011330  012603                          MOV     (SP)+,R3
 1986  011332  012602                          MOV     (SP)+,R2
 1987  011334  012601                          MOV     (SP)+,R1
 1988  011336  012600                          MOV     (SP)+,R0
 1989  011340  000204                          RTS     R4                      ;EXIT
 1990
 1991                                          .SBTTL  MISC. SUBROUTINES
```

```
                                        ;CONVERT R2 INTO DECIMAL DIGITS
                                        
011342  012737  177774  011446  DECPRT: MOV     #-4,DIGCNT      ;LOAD COUNT
011350  010146                          MOV     R1,-(SP)        ;SAVE R1
011352  012701  011462                  MOV     #DIGT1-1,R1     ;LOAD POINTER
011356  012737  011452  011450          MOV     #DECPNT+2,DECPNT        ;LOAD POINTER
011364  012737  177777  011444  1$:     MOV     #-1,DIGIT       ;LOAD #
011372  005237  011444          2$:     INC     DIGIT           ;UPDATE IT
011376  167702  000046                  SUB     @DECPNT,R2      ;SUB MAGNITUDE
011402  100373                          BPL     2$              ;BR IF +
011404  067702  000040                  ADD     @DECPNT,R2      ;RESTORE
011410  052737  000060  011444          BIS     #60,DIGIT       ;MAKE A #
011416  113721  011444                  MOVB    DIGIT,(R1)+     ;LOAD INTO STRING
011422  005237  011446                  INC     DIGCNT          ;UPDATE COUNT
011426  001002                          BNE     3$              ;BR IF NOT DONE
011430  012601                          MOV     (SP)+,R1        ;RESTORE R1
011432  000207                          RTS     PC              ;EXIT
011434  062737  000002  011450  3$:     ADD     #2,DECPNT       ;UPDATE POINTER
011442  000750                          BR      1$              ;BR BACK
                                        
011444  000000                  DIGIT:  0
011446  000000                  DIGCNT: 0
011450  011452                  DECPNT: .+2
011452  001750                          1000.
011454  000144                          100.
011456  000012                          10.
011460  000001                          1.
                                        
011462      000                 DIGT0:  .BYTE   0
011463      000                 DIGT1:  .BYTE   0
011464      000                 DIGT2:  .BYTE   0
011465      000                 DIGT3:  .BYTE   0
                                        
011466  013546                  BYTWO:  MOV     @(R5)+,-(SP)            ;SAVE ON STACK
011470  104402                          TYPOS
011472      004     001                 .BYTE   4,1
011474  104400  014541                  TYPE    ,MULTSP
011500  013546                          MOV     @(R5)+,-(SP)            ;SAVE ON STACK
011502  104402                          TYPOS
011504      004     001                 .BYTE   4,1
011506  000205                          RTS     R5                      ;EXIT
                                        
011510  012500                  TYPSTG: MOV     (R5)+,R0                ;GET ADDRESS POINTER
011512  012501                          MOV     (R5)+,R1                ;LOAD # OF LOC.
011514  012046                  1$:     MOV     (R0)+,-(SP)             ;SAVE ON STACK
011516  104402                          TYPOS
011520      004     001                 .BYTE   4,1
011522  104400  016543                  TYPE    ,SP3MSG
011526  005301                          DEC     R1                      ;DONE ?
011530  001371                          BNE     1$
011532  000205                  2$:     RTS     R5                      ;EXIT
```

```
                                        ;SUBROUTINE TO GET AVERAGE OF 32 CONVERSIONS

011534  012537  011652          CONVRT: MOV     (R5)+,10$
011540  013737  011652  011662          MOV     10$,CHANL
011546  000337  011652                  SWAB    10$
011552  012737  000040  011654          MOV     #32.,11$
011560  005037  011656                  CLR     ADEND
011564  013777  011652  167702          MOV     10$,@ADCS
011572  005277  167676          2$:     INC     @ADCS
011576  105777  167672          1$:     TSTB    @ADCS
011602  100375                  BPL     1$
011604  067737  167670  011656          ADD     @ADDBR,ADEND
011612  005337  011654                  DEC     11$
011616  001365                  BNE     2$
011620  006237  011656                  ASR     ADEND
011624  006237  011656                  ASR     ADEND
011630  006237  011656                  ASR     ADEND
011634  006237  011656                  ASR     ADEND
011640  006237  011656                  ASR     ADEND
011644  005537  011656                  ADC     ADEND                   ;ROUND UP
011650  000205                  RTS     R5

011652  000000          10$:    0
011654  000000          11$:    0
011656  000000          ADEND:  0
011660  000000          FCHANL: 0
011662  000000          CHANL:  0
011664  000000          SPREAD: 0

011666  010046          COMPAR: MOV     R0,-(SP)
011670  010146                  MOV     R1,-(SP)
011672  013700  001124          MOV     $GDDAT,R0               ;LOAD R0
011676  013701  001126          MOV     $BDDAT,R1               ;LOAD R1
011702  160100                  SUB     R1,R0                   ;SUBTRACT
011704  100001                  BPL     8$                      ;
011706  005400                  NEG     R0
011710  020037  011664  8$:     CMP     R0,SPREAD               ;MAGNITUDE OF DIFF. IN R0
011714  003405                  BLE     10$
011716  012601          9$:     MOV     (SP)+,R1
011720  012600                  MOV     (SP)+,R0
011722  062716  000002          ADD     #2,(SP)
011726  000207                  RTS     PC

011730  012601          10$:    MOV     (SP)+,R1
011732  012600                  MOV     (SP)+,R0
011734  000207                  RTS     PC
                                .SBTTL  ASCII MESSAGES AND ERROR POINTERS
```

```
2090
2091  011736  051105  047522  020122  EM1:    .ASCIZ  \ERROR ON A/D CHANNEL\
2092  011744  047117  040440  042057
2093  011752  041440  040510  047116
2094  011760  046105     000
2095  011763     126  020103  047514  EM2:    .ASCIZ  /VC LOGIC SIGNAL HIGH OUTPUT TOO LOW/
2096  011770  044507  020103  044523
2097  011776  047107  046101  044040
2098  012004  043511  020110  052517
2099  012012  050124  052125  052040
2100  012020  047517  046040  053517
2101  012026     000
2102  012027     126  020103  052123  EM3:    .ASCIZ  /VC STATUS REGISTER IN ERROR/
2103  012034  052101  051525  051040
2104  012042  043505  051511  042524
2105  012050  020122  047111  042440
2106  012056  051122  051117     000
2107  012063     105  052130  051105  EM4:    .ASCIZ  /EXTERNAL A TO D START FAILED/
2108  012070  040516  020114  020101
2109  012076  047524  042040  051440
2110  012104  040524  052122  043040
2111  012112  044501  042514  000104
2112  012120  027501  020104  044504  EM5:    .ASCIZ  \A/D DIFFERENTIAL LINEARITY ERROR\
2113  012126  043106  051105  047105
2114  012134  044524  046101  046040
2115  012142  047111  040505  044522
2116  012150  054524  042440  051122
2117  012156  051117     000
2118  012161     116  044517  042523  EM6:    .ASCIZ  /NOISE LEVEL EXCEEDED LIMIT/
2119  012166  046040  053105  046105
2120  012174  042440  041530  042505
2121  012202  042504  020104  044514
2122  012210  044515  000124
2123  012214  041526  046040  043517  EM7:    .ASCIZ  /VC LOGIC SIGNAL LOW OUTPUT TOO HIGH/
2124  012222  041511  051440  043511
2125  012230  040516  020114  047514
2126  012236  020127  052517  050124
2127  012244  052125  052040  047517
2128  012252  044040  043511  000110
2129  012260  027504  020101  044504  EM10:   .ASCIZ  \D/A DIFFERENTIAL LINEARITY ERROR\
2130  012266  043106  051105  047105
2131  012274  044524  046101  046040
2132  012302  047111  040505  044522
2133  012310  054524  042440  051122
2134  012316  051117     000
2135  012321     101  042057  044440  EM11:   .ASCIZ  \A/D INTER-CHANNEL SETTLING ERROR\
2136  012326  052116  051105  041455
2137  012334  040510  047116  046105
2138  012342  051440  052105  046124
2139  012350  047111  020107  051105
2140  012356  047522  000122
2141  012362  043117  051506  052105  EM12:   .ASCIZ  /OFFSET ERROR/
2142  012370  042440  051122  051117
2143  012376     000
2144  012377     103  046101  041111  EM13:   .ASCIZ  /CALIBRATION ERROR/
2145  012404  040522  044524  047117
```

```
2146  012412  042440  051122  051117
2147  012420     000
2148  012421     114  047111  040505  EM14:   .ASCII  /LINEARITY ERROR AT /
2149  012426  044522  054524  042440
2150  012434  051122  051117  040440
2151  012442  020124
2152  012444     060                  EM14B:  .BYTE   60
2153  012445     060     060     060  EM14A:  .BYTE   60,60,60,0
2154  012450     000
2155  012451     105  051122  041520  DH1:    .ASCIZ  /ERRPC   ARADD   CHANL   NOMINAL SPREAD  ACTUAL/
2156  012456  020040  040440  040522
2157  012464  042104  020040  041440
2158  012472  040510  046116  020040
2159  012500  047040  046517  047111
2160  012506  046101  051440  051120
2161  012514  040505  020104  040440
2162  012522  052103  040525  000114
2163  012530  051105  050122  020103  DH2:    .ASCIZ  /ERRPC   ARADD    CHANL  3V LEV. OUTPUT/
2164  012536  020040  051101  042101
2165  012544  020104  020040  041440
2166  012552  040510  046116  020040
2167  012560  053063  046040  053105
2168  012566  020056  052517  050124
2169  012574  052125     000
2170  012577     105  051122  041520  DH3:    .ASCIZ  /ERRPC   ARADD    GOOD    BAD/
2171  012604  020040  040440  040522
2172  012612  042104  020040  020040
2173  012620  047507  042117  020040
2174  012626  020040  040502  000104
2175  012634  051105  050122  020103  DH5:    .ASCIZ  /ERRPC   ARADD  # OF ST. ALLOWED/
2176  012642  020040  051101  042101
2177  012650  020104  021440  047440
2178  012656  020106  052123  020056
2179  012664  046101  047514  042527
2180  012672  000104
2181  012674  051105  050122  020103  DH6:    .ASCIZ  /ERRPC   ARADD   LIMIT   MEASURED/
2182  012702  020040  051101  042101
2183  012710  020104  020040  044514
2184  012716  044515  020124  020040
2185  012724  042515  051501  051125
2186  012732  042105     000
2187  012735     105  051122  041520  DH7:    .ASCIZ  \ERRPC   ARADD   A/D CH. .4 LEV  OUTPUT\
2188  012742  020040  040440  040522
2189  012750  042104  020040  040440
2190  012756  042057  041440  027110
2191  012764  027040  020064  042514
2192  012772  020126  047440  052125
2193  013000  052520  000124
2194  013004  051105  050122  020103  DH10:   .ASCIZ  \ERRPC   ARADD    STEP   NOMINAL SPREAD  ACTUAL\
2195  013012  020040  051101  042101
2196  013020  020104  020040  051440
2197  013026  042524  020120  020040
2198  013034  047516  044515  040516
2199  013042  020114  050123  042522
2200  013050  042101  020040  041501
2201  013056  052524  046101     000
```

```
2202  013063    105  051122  041520  DH11:   .ASCIZ  /ERRPC   ARADD   NOMINAL SPREAD  ACTUAL/
2203  013070 020040  040440  040522
2204  013076 042104  020040  047040
2205  013104 046517  047111  046101
2206  013112 051440  051120  040505
2207  013120 020104  040440  052103
2208  013126 040525  000114
2209  013132 005015  020111  044502  BIASST: .ASCIZ  <15><12>/I BIAS = /
2210  013140 051501  036440  000040
2211  013146 024040                  BIASDN: .ASCII  / (/
2212  013150 027062                  BASBT1: .ASCII  /2./
2213  013152 020060  044515  051103  BASBT2: .ASCII  /0 MICROAMP LIMIT = /
2214  013160 040517  050115  046040
2215  013166 046511  052111  036440
2216  013174    040
2217  013175    065  024460  005015  BASBT3: .ASCIZ  /50)/<15><12>
2218  013202    000
2219  013203    015  005012  027504  DIFMSG: .ASCII  <15><12><12>\D/A DIFFERENTIAL LINEARITY\
2220  013210 020101  044504  043106
2221  013216 051105  047105  044524
2222  013224 046101  046040  047111
2223  013232 040505  044522  054524
2224  013240 005015  052123  050105          .ASCII  <15><12>\STEP 0/1    1/2    3/4    7/10   17/20  37/40\
2225  013246 030040  030457  020040
2226  013254 020040  027461  020062
2227  013262 020040  031440  032057
2228  013270 020040  020040  027467
2229  013276 030061  020040  030440
2230  013304 027467  030062  020040
2231  013312 033463  032057    060
2232  013317    040  033467  030457          .ASCII  \ 77/100 177/200 377/400 777/1000\
2233  013324 030060  030440  033467
2234  013332 031057  030060  031440
2235  013340 033467  032057  030060
2236  013346 033440  033467  030457
2237  013354 030060    060
2238  013357    015  020012  020130          .ASCIZ  <15><12>\ X   \
2239  013364 020040    000
2240  013367    015  020012  020131  DIYMSG: .ASCIZ  <15><12>/ Y   /
2241  013374 020040    000
2242  013377    015  024012  047516  DIEMSG: .ASCIZ  <15><12>/(NOMINAL STEP HEIGHT = 62)/<15><12>
2243  013404 044515  040516  020114
2244  013412 052123  050105  044040
2245  013420 044505  044107  020124
2246  013426 020075  031066  006451
2247  013434 000012
2248  013436 005015  027501  020104  ADSMSG: .ASCII  <15><12>\A/D INTER-CHANNEL SETTLING (1 LSB = 62)\
2249  013444 047111  042524  026522
2250  013452 044103  047101  042516
2251  013460 020114  042523  052124
2252  013466 044514  043516  024040
2253  013474 020061  051514  020102
2254  013502 020075  031066    051
2255  013507    015  050012  051517          .ASCIZ  <15><12>/POSITIVE = /
2256  013514 052111  053111  020105
2257  013522 020075    000
```

```
2258  013525  040     047040  043505  ADEMSG: .ASCIZ  /  NEGATIVE = /
2259  013532  052101  053111  020105
2260  013540  020075  000
2261  013543  015     047012  044517  NOIMSG: .ASCII  <15><12>\NOISE:    RMS (1/4 LSB LIMIT = 31)  PEAK (2 LSB LIMIT = 144)\
2262  013550  042523  020072  020040
2263  013556  051040  051515  024040
2264  013564  027461  020064  051514
2265  013572  020102  044514  044515
2266  013600  020124  020075  030463
2267  013606  020051  050040  040505
2268  013614  020113  031050  046040
2269  013622  041123  046040  046511
2270  013630  052111  036440  030440
2271  013636  032064  051
2272  013641  015     040412  042057          .ASCIZ  <15><12>\A/D BIPOLAR     \
2273  013646  041040  050111  046117
2274  013654  051101  020040  020040
2275  013662  000040
2276  013664  005015  027501  020104  NOIMG1: .ASCIZ  <15><12>\A/D UNIPOLAR    \
2277  013672  047125  050111  046117
2278  013700  051101  020040  020040
2279  013706  000
2280  013707  015     054012  042040  NOIMG2: .ASCIZ  <15><12>\X D/A           \
2281  013714  040457  020040  020040
2282  013722  020040  020040  020040
2283  013730  000040
2284  013732  005015  020131  027504  NOIMG3: .ASCIZ  <15><12>\Y D/A           \
2285  013740  020101  020040  020040
2286  013746  020040  020040  020040
2287  013754  000
2288  013755  015     005012  044514  LINMSG: .ASCII  <15><12><12>/LINEARITY (1 LSB = 62)/<15><12>
2289  013762  042516  051101  052111
2290  013770  020131  030450  046040
2291  013776  041123  036440  033040
2292  014004  024462  005015
2293  014010  020040  020040  020040          .ASCII  /                0      200    400    600    1000   1200   1400   1600/
2294  014016  020040  020040  020040
2295  014024  020040  020040  020060
2296  014032  020040  020040  031040
2297  014040  030060  020040  020040
2298  014046  030064  020060  020040
2299  014054  033040  030060  020040
2300  014062  020040  030061  030060
2301  014070  020040  030440  030062
2302  014076  020060  020040  032061
2303  014104  030060  020040  030440
2304  014112  030066  060
2305  014115  015     054412  042040          .ASCIZ  <15><12>\Y D/A VS. A/D   \
2306  014122  040457  053040  027123
2307  014130  040440  042057  020040
2308  014136  000040
2309  014140  005015  020130  027504  LINMG1: .ASCIZ  <15><12>\X D/A VS. A/D   \
2310  014146  020101  051526  020056
2311  014154  027501  020104  020040
2312  014162  000
2313  014163  015     005012  027501  ADDIFM: .ASCIZ  <15><12><12>\A/D DIFFERENTIAL LINEARITY:\<15><12>
```

```
2314  014170  020104  044504  043106
2315  014176  051105  047105  044524
2316  014204  046101  046040  047111
2317  014212  040505  044522  054524
2318  014220  006472  000012
2319  014224  052123  052101  026505  ADDIF:  .ASCIZ  /STATE-WIDTH/<15><12>
2320  014232  044527  052104  006510
2321  014240  000012
2322  014242  034050  020051  045523  SKPMSG: .ASCIZ  /(8) SKIPPED STATE(S)/<15><12>
2323  014250  050111  042520  020104
2324  014256  052123  052101  024105
2325  014264  024523  005015     000
2326  014271     050  024470  037040  GRT2MG: .ASCIZ  /(8) > 2 LSB STATE(S)/<15><12>
2327  014276  031040  046040  041123
2328  014304  051440  040524  042524
2329  014312  051450  006451  000012
2330  014320  034050  020051  020074  NARMSG: .ASCIZ  \(8) < 1/2 LSB NARROW STATE(S)\<15><12>
2331  014326  027461  020062  051514
2332  014334  020102  040516  051122
2333  014342  053517  051440  040524
2334  014350  042524  051450  006451
2335  014356  000012
2336  014360  034050  020[illegible]  020076  WIDMSG: .ASCIZ  \(8) > 1 1/2 LSB WIDE STATE(S)\<15><12>
2337  014366  020061  027[illegible]  020062
2338  014374  051514  020102  044527
2339  014402  042504  051440  040524
2340  014410  042524  051450  006451
2341  014416  000012
2342  014420  054130  054056  020045  PERHLF: .ASCIZ  \XX.X% OF STATES WITHIN 1/2 LSB  (SPEC = 95%)\<15><12>
2343  014426  043117  051440  040524
2344  014434  042524  020123  044527
2345  014442  044124  047111  030440
2346  014450  031057  046040  041123
2347  014456  020040  051450  042520
2348  014464  020103  020075  032471
2349  014472  024445  005015     000
2350  014477     130  027130  022530  PERQRT: .ASCIZ  \XX.X% OF STATES WITHIN 1/4 LSB \<15><12>
2351  014504  047440  020106  052123
2352  014512  052101  051505  053440
2353  014520  052111  044510  020116
2354  014526  027461  020064  051514
2355  014534  020102  005015     000
2356  014541     040  020040  020040  MULTSP: .ASCIZ  /                    /
2357  014546  020040  020040  020040
2358  014554  020040  020040  020040
2359  014562  020040  020040  020040
2360  014570  000040
2361                                          .EVEN
2362  014572  001116  001464  011662  DT1:    $ERRPC,ARBADD,CHANL,$GDDAT,SPREAD,$BDDAT,0
2363  014600  001124  011664  001126
2364  014606  000000
2365  014610  001116  001464  011662  DT2:    $ERRPC,ARBADD,CHANL,$GDDAT,$BDDAT,0
2366  014616  001124  001126  000000
2367  014624  001116  001464  001124  DT3:    $ERRPC,ARBADD,$GDDAT,$BDDAT,0
2368  014632  001126  000000
2369  014636  001116  001464  001126  DT5:    $ERRPC,ARBADD,$BDDAT,$GDDAT,0
```

```
2370  014644  001124  000000
2371  014650  001116  001464  015170  DT10:   $ERRPC,ARBADD,EDGE,$GDDAT,SPREAD,$BDDAT,0
2372  014656  001124  011664  001126
2373  014664  000000
2374  014666  001116  001464  001124  DT11:    .RRPC,ARBADD,$GDDAT,SPREAD,$BDDAT,0
2375  014674  011664  001126  000000
2376  014702  000004  000203  000402  NUMBF2: 4,203,402,601,1000,1177,1376,1575,1774,BIT15
2377  014710  000601  001000  001177
2378  014716  001376  001575  001774
2379  014724  100000
2380  014726  000000  000000  000000  RSLT0:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2381  014734  000000  000000  000000
2382  014742  000000  000000  000000
2383  014750  000000  000000  000000
2384  014756  000000  000000  000000
2385  014764  000000  000000  000000  RSLT1:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2386  014772  000000  000000  000000
2387  015000  000000  000000  000000
2388  015006  000000  000000  000000
2389  015014  000000  000000  000000
2390  015022  000000  000000  000000  RSLT2:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2391  015030  000000  000000  000000
2392  015036  000000  000000  000000
2393  015044  000000  000000  000000
2394  015052  000000  000000  000000
2395  015060  000000  000000  000000  RSLT3:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2396  015066  000000  000000  000000
2397  015074  000000  000000  000000
2398  015102  000000  000000  000000
2399  015110  000000  000000  000000
2400  015116  000000
2401
                                      .SBTTL  DIFFERENTIAL LINEARITY SUBROUTINE
```

```
 2402  015120  170412                ADAC:   170412                  ;BUS ADDRESS OF DAC
 2403  015122  000000                DACSAV: 0
 2404  015124  000031                RMSMAX: 25.                     ;1/4 LSB
 2405  015126  000310                PEKMAX: 200.                    ;2 LSB
 2406  015130  000000                CH7RMS: 0
 2407  015132  000000                CH7PEK: 0
 2408  015134  000000                CH47RM: 0
 2409  015136  000000                CH47PK: 0
 2410  015140  000000                XDACRM: 0
 2411  015142  000000                XDACPK: 0
 2412  015144  000000                YDACRM: 0
 2413  015146  000000                YDACPK: 0
 2414  015150  000000                OFSTBX: 0
 2415  015152  000000                OFSTUX: 0
 2416  015154  000000                OFSLBX: 0
 2417  015156  000000                OFSLUX: 0
 2418  015160  000000                ADPMUX: 0
 2419  015162  000000                ADNMUX: 0
 2420  015164  000000                BIASC1: 0
 2421  015166     000      D06       SARCHN: .BYTE   0,6             ;CHANNEL # IN HIGH BYTE
 2422  015170  001000                EDGE:   1000                    ;CONVERTED VALUE EDGE
 2423  015172  000000                CONS:   0                       ;0 = RMS BIT 15 = 1 PEAK
 2424  015174  000000                FINS:   0                       ;2 PASS COUNTER
 2425  015176  000031                LOLIM1: 31
 2426  015200  000046                LOLIM2: 46
 2427  015202  000113                HILIM1: 113
 2428  015204  000076                HILIM2: 76
 2429  015206  000000                QRTOK:  0
 2430  015210  000000                NARROW: 0
 2431  015212  000000                WIDE:   0
 2432  015214  000062                AMEAN:  62
 2433  015216  000000                TEMP:   0
```

```
2434
2435  015220  012537  016640          DIFLIN: MOV     (R5)+,VCREG1        ;LOAD COARSE REGISTER ADDRESS
2436  015224  012537  016642                  MOV     (R5)+,VCREG2        ;LOAD FINE ADDRESS
2437  015230  012537  016644                  MOV     (R5)+,DIFCHN        ;LOAD CHANNEL
2438  015234  010546                          MOV     R5,-(SP)
2439  015236  005077  164244                  CLR     @VCSTAT
2440  015242  017737  001372  016640          MOV     @VCREG1,VCREG1      ;GET BUS ADDRESS
2441  015250  017737  001366  016642          MOV     @VCREG2,VCREG2
2442  015256  012777  000000  001354          MOV     #0,@VCREG1          ;CLEAR COARSE
2443  015264  012777  000000  001350          MOV     #0,@VCREG2          ;CLEAR FINE
2444  015272  012701  022040                  MOV     #ADBUFF,%1
2445  015276  005000                          CLR     %0
2446  015300  005021                  1$:     CLR     (1)+                ;CLEAR MEMORY
2447  015302  022701  024504                  CMP     #LAST,%1
2448  015306  001374                          BNE     1$
2449
2450  015310  004737  016340          CONVR:  JSR     PC,CONVT
2451  015314  001410                          BEQ     2$                  ;BRANCH IF RESULT = 0
2452  015316  020500                          CMP     R5,R0               ;COMPARE RESULT AND POINT
2453  015320  100406                          BMI     2$                  ;BRANCH FOR RESULT < POINT
2454  015322  105265  022040                  INCB    ADBUFF(%5)          ;INCREMENT CONTENTS OF Z
2455  015326  103003                          BCC     2$
2456  015330  112765  000377  022040          MOVB    #377,ADBUFF(R5)
2457  015336  062777  000001  001276  2$:     ADD     #1,@VCREG2          ;UPDATE FINE REG.
2458  015344  005777  001272                  TST     @VCREG2
2459  015350  001357                          BNE     CONVR               ;BRANCH UNTIL FINISH
2460  015352  022777  001774  001260          CMP     #1774,@VCREG1       ;TEST COARSE REG.
2461  015360  001422                          BEQ     READ
2462  015362  062777  000014  001250          ADD     #14,@VCREG1         ;UPDATE COARSE REG.
2463  015370  005077  001246                  CLR     @VCREG2             ;CLEAR FINE REG.
2464  015374  004737  016340                  JSR     PC,CONVT
2465  015400  062705  000004                  ADD     #4,R5               ;(4+RESULT) IN R5
2466  015404  010500                          MOV     %5,%0               ;LOAD POINT
2467  015406  062705  022040                  ADD     #ADBUFF,%5          ;
2468  015412  012704  000010                  MOV     #10,%4
2469  015416  105025                  1$:     CLRB    (R5)+               ;CLEAR 8. MEM BYTE LOCATIONS
2470  015420  005304                          DEC     %4
2471  015422  001375                          BNE     1$
2472  015424  000731                          BR      CONVR
2473  015426  032777  010000  163502  READ:   BIT     #BIT12,@SWR
2474  015434  001402                          BEQ     13$
2475  015436  104400  014163                  TYPE    ,ADDIFM
2476  015442  005000                  13$:    CLR     R0
2477  015444  012702  001776                  MOV     #1776,%2
2478  015450  012703  022041                  MOV     #ADBUFF+1,%3
2479  015454  012737  000001  015216          MOV     #1,TEMP
2480  015462  005037  015206                  CLR     QRTOK
2481  015466  005037  015742                  CLR     DIFERR
2482  015472  005037  015210                  CLR     NARROW
2483  015476  005037  015212                  CLR     WIDE
2484  015502  005037  015736                  CLR     20$
2485  015506  005037  015740                  CLR     SKIPST
2486  015512  005037  015744                  CLR     EXCESS
2487  015516  111300                  2$:     MOVB    (R3),R0
2488  015520  020027  000144                  CMP     R0,#144             ;TEST IF STATE WIDTH > 2LSB
2489  015524  003024                          BGT     1$                  ;YES, BR
```

```
2490  015526  105260  024102                  INCB    ABUFF4(R0)              ;UPDATE STATE-WIDTH BUFFER
2491  015532  020037  015202                  CMP     R0,HILIM1               ;TEST IF STATE WIDTH > 1 1/2 LSB
2492  015536  003021                          BGT     3$                      ;YES, BR
2493  015540  020037  015204                  CMP     R0,HILIM2               ;TEST IF STATE WIDTH > 1 1/4 LSB
2494  015544  003061                          BGT     4$                      ;YES, BR
2495  015546  020037  017000                  CMP     R0,V1                   ;IS IT A SKIPPED STATE?
2496  015552  103416                          BLO     10$                     ;YES, BR
2497  015554  020037  015176                  CMP     R0,LOLIM1               ;TEST IF STATE WIDTH < 1/2 LSB
2498  015560  002415                          BLT     11$                     ;YES BR
2499  015562  020037  015200                  CMP     R0,LOLIM2               ;TEST IF STATE WIDTH < 3/4 LSB
2500  015566  002450                          BLT     4$                      ;YES, BR
2501  015570  005237  015206                  INC     QRTOK                   ;STATE WIDTH BETWEEN 3/4 AND 1 1/4 LSB
2502  015574  000445                          BR      4$
2503  015576  005237  015744          1$:     INC     EXCESS                  ;UPDATE > 2 LSB WIDE STATE COUNT
2504  015602  005237  015212          3$:     INC     WIDE                    ;UPDATE > 1 1/2 LSB WIDE STATE COUNT
2505  015606  000406                          BR      12$
2506  015610  005237  015740          10$:    INC     SKIPST                  ;UPDATE SKIPPED STATE COUNT
2507  015614  105700                  11$:    TSTB    R0
2508  015616  100767                          BMI     1$
2509  015620  005237  015210                  INC     NARROW                  ;UPDATE < 1/2 LSB NARROW STATE COUNT
2510  015624  005237  015742          12$:    INC     DIFERR                  ;UPDATE OUTSIDE + OR - 1/2 LSB COUNT
2511  015630  032777  010000  163300  5$:     BIT     #BIT12,@SWR             ;TEST FOR FORCED TYPEOUT
2512  015636  001424                          BEQ     4$              ;;BR IF NOT
2513  015640  005737  015736                  TST     20$                     ;TEST IF FIRST TIME ?
2514  015644  001004                          BNE     6$                      ;BR IF YES
2515  015646  005237  015736                  INC     20$
2516  015652  104400  014224                  TYPE    ,ADDIF
2517  015656  013746  015216          6$:     MOV     TEMP,-(SP)              ;SAVE CURRENT STATE
2518  015662  104402                          TYPOS
2519  015664     004     001                  .BYTE   4,1
2520  015666  104400  016552                  TYPE    ,TYANXX
2521  015672  042700  177400                  BIC     #177400,R0
2522  015676  010046                          MOV     R0,-(SP)
2523  015700  104402                          TYPOS
2524  015702     003     001                  .BYTE   3,1
2525  015704  104400  016536                  TYPE    ,ACRLF
2526  015710  005237  015216          4$:     INC     TEMP
2527  015714  105723                          TSTB    (R3)+
2528  015716  005302                          DEC     %2
2529  015720  001276                          BNE     2$
2530  015722  032777  010000  163206          BIT     #BIT12,@SWR             ;TEST BIT 12
2531  015730  001006                          BNE     SWDIST                  ;BR IF FORCED PRINTOUT
2532  015732  012605                          MOV     (SP)+,R5
2533  015734  000205                          RTS     R5
2534  015736  000000                  20$:    0
2535  015740  000000                  SKIPST: 0                               ;SKIPPED STATE COUNT
2536  015742  000000                  DIFERR: 0                               ;OUTSIDE + OR - 1/2 LSB STATE COUNT
2537  015744  000000                  EXCESS: 0                               ;> 2 LSB STATE COUNT
2538
2539  015746  012703  024102          SWDIST: MOV     #ABUFF4,%3
2540  015752  005037  015216                  CLR     TEMP
2541  015756  104400  016536                  TYPE    ,ACRLF
2542  015762  013746  015740                  MOV     SKIPST,-(SP)            ;SAVE SKIPPED STATES
2543  015766  104402                          TYPOS
2544  015770     003     000                  .BYTE   3,0
2545  015772  104400  014242                  TYPE    ,SKPMSG
```

```
2546  015776  013746  015744                    MOV     EXCESS,-(SP)            ;SAVE EXCESSIVE WIDE STATES
2547  016002  104402                            TYPOS
2548  016004     003     000                    .BYTE   3,0
2549  016006  104400  014271                    TYPE    ,GRT2MG
2550  016012  013746  015210                    MOV     NARROW,-(SP)
2551  016016  104402                            TYPOS
2552  016020     003     000                    .BYTE   3,0
2553  016022  104400  014320                    TYPE    ,NARMSG
2554  016026  013746  015212                    MOV     WIDE,-(SP)              ;SAVE WIDE STATES
2555  016032  104402                            TYPOS
2556  016034     003     000                    .BYTE   3,0
2557  016036  104400  014360                    TYPE    ,WIDMSG
2558  016042  012700  001776                    MOV     #1022.,R0
2559  016046  163700  015212                    SUB     WIDE,R0
2560  016052  163700  015210                    SUB     NARROW,R0
2561  016056  004537  017044                    JSR     R5,PRCNT                ;CONVERT TO PERCENT
2562  016062  014420                            PERHLF
2563  016064  104400  014420                    TYPE    ,PERHLF
2564  016070  013700  015206                    MOV     QRTOK,R0
2565  016074  004537  017044                    JSR     R5,PRCNT
2566  016100  014477                            PERQRT
2567  016102  104400  014477                    TYPE    ,PERQRT
2568  016106  104400                            TYPE
2569  016110  016554                            STDMSG
2570  016112  013702  015216          6$:       MOV     TEMP,R2
2571  016116  006302                            ASL     R2
2572  016120  004737  011342                    JSR     PC,DECPRT               ;CONVERT R2 TO DEC.
2573  016124  113737  011463  016400            MOVB    DIGT1,RSV1
2574  016132  113737  011464  016402            MOVB    DIGT2,RSV2
2575  016140  113737  011465  016403            MOVB    DIGT3,RSV3
2576
2577  016146  111302                            MOVB    (R3),R2
2578  016150  004737  011342                    JSR     PC,DECPRT               ;CONVERT R2 TO DEC.
2579  016154  113737  011462  016405            MOVB    DIGT0,RSX1
2580  016162  113737  011463  016406            MOVB    DIGT1,RSX2
2581  016170  113737  011464  016407            MOVB    DIGT2,RSX3
2582  016176  113737  011465  016410            MOVB    DIGT3,RSX4
2583  016204  104400                            TYPE
2584  016206  016376                            RSVMSG
2585  016210  111305                            MOVB    (%3),R5
2586  016212  005305                2$:         DEC     R5
2587  016214  100403                            BMI     1$
2588  016216  104400                            TYPE
2589  016220  016541                            TYANX
2590  016222  000773                            BR      2$
2591  016224  023737  015176  015216  1$:       CMP     LOLIM1,TEMP
2592  016232  001413                            BEQ     4$
2593  016234  023737  015202  015216            CMP     HILIM1,TEMP
2594  016242  001412                            BEQ     5$
2595  016244  023737  015214  015216            CMP     AMEAN,TEMP
2596  016252  001010                            BNE     3$
2597  016254  104400  016472                    TYPE    ,TYMEAN
2598  016260  000405                            BR      3$
2599  016262  104400  016414                4$:  TYPE    ,TYLOW
2600  016266  000402                            BR      3$
2601  016270  104400  016442                5$:  TYPE    ,TYHIGH
```

```
2602  016274  005237  015216          3$:     INC     TEMP
2603  016300  105723                          TSTB    (3)+
2604  016302  022737  000145  015216          CMP     #145,TEMP
2605  016310  001300                          BNE     6$
2606  016312  104400                          TYPE
2607  016314  016536                          ACRLF
2608  016316  104400  016516                  TYPE    ,OVERNG
2609  016322  013746  015744                  MOV     EXCESS,-(SP)
2610  016326  104402                          TYPOS
2611  016330     004     001                  .BYTE   4,1
2612  016332  012605                          MOV     (SP)+,R5
2613  016334  000205                          RTS     R5
2614
2615  016336  000000                  12$:    0
2616
2617  016340  005277  163142          CONVT:  INC     @VCSTAT
2618  016344  105777  163136          2$:     TSTB    @VCSTAT
2619  016350  100375                          BPL     2$
2620  016352  005005                          CLR     %5
2621  016354  013777  016644  163112          MOV     DIFCHN,@ADCS    ;CONVERT
2622  016362  105777  163106          1$:     TSTB    @ADCS           ;WAIT FOR DONE
2623  016366  100375                          BPL     1$
2624  016370  017705  163104                  MOV     @ADDBR,R5       ;READ VALUE
2625  016374  000207                          RTS     PC
2626
2627  016376     015     012          RSVMSG: .BYTE   15,12
2628  016400     060     056          RSV1:   .BYTE   60,56
2629  016402     060                  RSV2:   .BYTE   60
2630  016403     060     055          RSV3:   .BYTE   60,55
2631  016405     060                  RSX1:   .BYTE   60
2632  016406     060                  RSX2:   .BYTE   60
2633  016407     060                  RSX3:   .BYTE   60
2634  016410     060     040     111  RSX4:   .BYTE   60,40,111,0
2635  016413     000
2636  016414  026455  026455  026455  TYLOW:  .ASCIZ  "----------  (1/2 LSB)"
2637  016422  026455  026455  020040
2638  016430  030450  031057  046040
2639  016436  041123  000051
2640  016442  026455  026455  026455  TYHIGH: .ASCIZ  "----------  (1 1/2 LSB)"
2641  016450  026455  026455  020040
2642  016456  030450  030440  031057
2643  016464  046040  041123  000051
2644  016472  026455  026455  026455  TYMEAN: .ASCIZ  "----------  (1 LSB)"
2645  016500  026455  026455  020040
2646  016506  030450  046040  041123
2647  016514  000051
2648  016516  052517  020124  043117  OVERNG: .ASCIZ  /OUT OF RANGE - /
2649  016524  051040  047101  042507
2650  016532  026440  000040
2651  016536     015     012     000  ACRLF:  .BYTE   15,12,0
2652  016541     052     000          TYANX:  .BYTE   52,0
2653  016543     040  020040     000  SP3MSG: .ASCIZ  /   /
2654  016547     000     000     000          .BYTE   0,0,0
2655  016552     055     000          TYANXX: .BYTE   55,0
2656  016554     015     012          STDMSG: .BYTE   15,12
2657  016556  052123  052101  026505          .ASCII  /STATE-WIDTH DISTRIBUTION/
```

```
 2658  016564  044527  052104  020110
 2659  016572  044504  052123  044522
 2660  016600  052502  044524  047117
 2661  016606     015     012                .BYTE   15,12
 2662  016610  044527  052104  026510        .ASCIZ  /WIDTH-NUMBER OF STATES/
 2663  016616  052516  041115  051105
 2664  016624  047440  020106  052123
 2665  016632  052101  051505     000
 2666          016640                        .EVEN
 2667  016640  000000                VCREG1: 0
 2668  016642  000000                VCREG2: 0
 2669  016644  000000                DIFCHN: 0
 2670
 2671                                        .SBTTL  PARAMETER ADJUSTMENT ROUTINE
 2672  016646  012701  016764        WFADJ:  MOV     #V1754,R1               ;LOAD PARM. POINTER
 2673  016652  005737  016762                TST     WFTEST                  ;TEST IF OPTION TEST AREA
 2674  016656  001017                        BNE     1$                      ;BR IF IT WAS
 2675  016660  012702  017004                MOV     #VARLT1,R2              ;LOAD "STANDARD" PARM. VALUES
 2676  016664  112737  000062  013150        MOVB    #'2,BASBT1              ;LOAD BIAS MESSAGE
 2677  016672  112737  000060  013152        MOVB    #'0,BASBT2              ;
 2678  016700  112737  000065  013175        MOVB    #'5,BASBT3              ;LOAD BIAS LIMIT
 2679  016706  112737  000060  013176        MOVB    #'0,BASBT3+1
 2680  016714  000416                        BR      2$
 2681  016716  012702  017022        1$:     MOV     #VARLT2,R2              ;LOAD "OPTION TEST AREA" PARM. VALUES
 2682  016722  112737  000061  013150        MOVB    #'1,BASBT1              ;LOAD BIAS MESSAGE
 2683  016730  112737  000070  013152        MOVB    #'8,BASBT2
 2684  016736  112737  000064  013175        MOVB    #'4,BASBT3              ;LOAD BIAS LIMIT
 2685  016744  112737  000064  013176        MOVB    #'4,BASBT3+1
 2686  016752  012221                2$:     MOV     (R2)+,(R1)+             ;LOAD INTO PARM. LIST
 2687  016754  005711                        TST     (R1)                    ;TEST FOR LAST
 2688  016756  100375                        BPL     2$                      ;BR IF NOT
 2689  016760  000207                        RTS     PC                      ;EXIT
 2690
 2691  016762  000000                WFTEST: 0
 2692
 2693  016764  001754                V1754:  1754            ;1760 IF OPTION TEST AREA SELECTED
 2694  016766  000024                V24:    24              ;20     "       "       "       "
 2695  016770  000050                V50:    50              ;40     "       "       "       "
 2696  016772  000024                VA24:   24              ;22     "       "       "       "
 2697  016774  000062                V62:    62              ;45     "       "       "       "
 2698  016776  000077                V77:    77              ;45     "       "       "       "
 2699  017000  000001                V1:     1               ;15     "       "       "       "
 2700  017002  100000                        BIT15              RM.
 2701
 2702  017004  001754                VARLT1: 1754
 2703  017006  000024                        24
 2704  017010  000050                        50
 2705  017012  000024                        24
 2706  017014  000062                        62
 2707  017016  000077                        77
 2708  017020  000001                        1
 2709
 2710  017022  001760                VARLT2: 1760
 2711  017024  000020                        20
 2712  017026  000040                        40
 2713  017030  000022                        22
```

```
 2714  017032  000045                          45
 2715  017034  000045                          45
 2716  017036  000015                          15
 2717
 2718                                          .SBTTL  SUBROUTINE TO CONVERT R0 TO DECIMAL PERCENTAGE OF 1022.
 2719
 2720  017040  000000                  MSGPNT: 0
 2721  017042  000000                  PCT:    0
 2722  017044  012537  017040          PRCNT:  MOV     (R5)+,MSGPNT            ;GET MESSAGE POINTER
 2723  017050  010037  017042                  MOV     R0,PCT                  ;
 2724  017054  006200                          ASR     R0
 2725  017056  006200                          ASR     R0
 2726  017060  006200                          ASR     R0
 2727  017062  006200                          ASR     R0
 2728  017064  006200                          ASR     R0
 2729  017066  006200                          ASR     R0
 2730  017070  160037  017042                  SUB     R0,PCT
 2731  017074  006200                          ASR     R0
 2732  017076  160037  017042                  SUB     R0,PCT                  ;PCT CONTAINS OCTAL %
 2733  017102  013702  017042                  MOV     PCT,R2                  ;LOAD R2 WITH PERCENTAGE
 2734  017106  004737  011342                  JSR     PC,DECPRT               ;CONVERT TO DEC.
 2735  017112  013700  017040                  MOV     MSGPNT,R0               ;LOAD MSG. POINTER
 2736  017116  022737  001750  017042          CMP     #1000.,PCT              ;TEST FOR 100 %
 2737  017124  001411                          BEQ     1$                      ;BR IF 100 %
 2738  017126  113720  011463                  MOVB    DIGT1,(R0)+             ;LOAD A DIGIT
 2739  017132  113720  011464                  MOVB    DIGT2,(R0)+
 2740  017136  112720  000056                  MOVB    #56,(R0)+               ;LOAD "."
 2741  017142  113720  011465                  MOVB    DIGT3,(R0)+             ;LOAD 1/10 % DIGIT
 2742  017146  000205                          RTS     R5                      ;EXIT
 2743
 2744  017150  112720  000040          1$:     MOVB    #40,(R0)+               ;LOAD "SPACE"
 2745  017154  112720  000061                  MOVB    #'1,(R0)+               ;LOAD '1' 100%
 2746  017160  112720  000060                  MOVB    #'0,(R0)+
 2747  017164  112720  000060                  MOVB    #'0,(R0)+
 2748  017170  000205                          RTS     R5                      ;EXIT
```

```
                                        .SBTTL  SCOPE HANDLER ROUTINE
                                        ;;*********************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) 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
                                        ;*      SCOPE           ;;SCOPE=IOT

017172                                  $SCOPE:
017172  104405                          CKSWR
017174  032777  040000  161734  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
017202  001114                          BNE     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
017204  000416                          $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
017206  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
017212  012737  017232  000004                  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
017220  005737  177060                          TST     @#177060        ;;TIME OUT ON XOR?
017224  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
017230  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
017232  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
017234  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
017240  000423                          BR      7$              ;;LOOP ON THE PRESENT TEST
017242                          6$:;#####END OF CODE FOR THE XOR TESTER#####
017242  032777  000400  161666          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
017250  001404                          BEQ     2$              ;;BR IF NO
017252  127737  161660  001102          CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
017260  001465                          BEQ     $OVER           ;;BR IF YES
017262  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
017266  001421                          BEQ     3$              ;;BR IF NO
017270  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
017276  101015                          BHI     3$              ;;BR IF NO
017300  032777  001000  161630          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
017306  001404                          BEQ     4$              ;;BR IF NO
017310  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
017316  000446                          BR      $OVER
017320  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
017324  005037  001164                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
017330  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
017332  032777  004000  161576  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
017340  001011                          BNE     1$              ;;BR IF YES
017342  005737  001206                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
017346  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
017350  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
017354  023737  001164  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
017362  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
017364  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
017372  013737  017450  001164          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
017400  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
```

```
 2805  017404  113737  001102  001204          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 2806  017412  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 2807  017416  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 2808  017422  005037  001166                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 2809  017426  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 2810  017434  013777  001102  161476  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 2811  017442  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 2812  017446  000002                          RTI                     ;;FIXES PS
 2813  017450  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
 2814
 2815                                  .SBTTL  ERROR HANDLER ROUTINE
 2816
 2817                                  ;;**************************************************************
 2818                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 2819                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 2820                                  ;*AND GO TO $ERRTYP ON ERROR
 2821                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 2822                                  ;*SW15=1         HALT ON ERROR
 2823                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
 2824                                  ;*SW10=1         BELL ON ERROR
 2825                                  ;*SW09=1         LOOP ON ERROR
 2826                                  ;*CALL
 2827                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 2828
 2829  017452                          $ERROR:
 2830  017452  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 2831  017456  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 2832  017460  013777  001102  161452          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 2833  017466  032777  002000  161442          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
 2834  017474  001402                          BEQ     1$              ;;NO - SKIP
 2835  017476  104400  001170                  TYPE    ,$BELL          ;;RING BELL
 2836  017502  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 2837  017506  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 2838  017512  162737  000002  001116          SUB     #2,$ERRPC
 2839  017520  117737  161372  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
 2840  017526  032777  020000  161402          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
 2841  017534  001004                          BNE     20$             ;;SKIP TYPEOUTS
 2842  017536  004737  020442                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
 2843  017542  104400  001175                  TYPE    ,$CRLF
 2844  017546                          20$:
 2845  017546  122737  000001  001220          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 2846  017554  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
 2847  017556  113737  001114  017570          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
 2848  017564  004737  021544                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
 2849  017570     000                  21$:    .BYTE   0
 2850  017571     000                          .BYTE   0
 2851  017572  000777                  22$:    BR      22$             ;;APT ERROR LOOP
 2852  017574  005777  161336          2$:     TST     @SWR            ;;HALT ON ERROR
 2853  017600  100001                          BPL     3$              ;;SKIP IF CONTINUE
 2854  017602  000000                          HALT                    ;;HALT ON ERROR!
 2855  017604  032777  001000  161324  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
 2856  017612  001402                          BEQ     4$              ;;BR IF NO
 2857  017614  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
 2858  017620  005737  001166          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
 2859  017624  001402                          BEQ     5$              ;;BR IF NONE
 2860  017626  013716  001166                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
```

```
                                        5$:
 2861  017632
 2862  017632  022737  010436  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
 2863  017640  001001                          BNE     6$              ;;BRANCH IF NO
 2864  017642  000000                          HALT                    ;;YES
 2865  017644                                6$:
 2866  017644  000002                          RTI                     ;;RETURN
 2867
 2868                                  .SBTTL  TTY INPUT ROUTINE
 2869
 2870                                  ;;**********************************************************************
 2871                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
 2872                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
 2873                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
 2874                                  ;*WHEN OPERATING IN TTY FLAG MODE.
 2875  017646  022737  000176  001136  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
 2876  017654  001073                          BNE     14$             ;;BRANCH IF NO
 2877  017656  105777  161260                  TSTB    @$TKS           ;;CHAR THERE?
 2878  017662  100070                          BPL     14$             ;;IF NO, DON'T WAIT AROUND
 2879  017664  117746  161254          2$:     MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
 2880  017670  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
 2881  017674  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
 2882  017700  001061                          BNE     14$             ;;NO, RETURN TO USER
 2883  017702  104400  020311                  TYPE    ,$CNTLG         ;;YES, ECHO CONTROL G
 2884
 2885  017706  104400  020316          6$:     TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
 2886  017712  013746  000176                  MOV     $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
 2887  017716  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2888  017720  104400  020327                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
 2889  017724  005046                          CLR     -(SP)           ;;CLEAR COUNTER
 2890  017726  005046                          CLR     -(SP)           ;;THE NEW SWR
 2891  017730  104406          7$:     RDCHR                           ;;GET NEXT CHAR
 2892
 2893  017732  022716  000025          8$:     CMP     #25,(SP)        ;;IS IT A CONTROL U?
 2894  017736  001005                          BNE     9$              ;;BRANCH IF NO
 2895  017740  104400  020304                  TYPE    ,$CNTLU         ;;YES, ECHO IT
 2896  017744  062706  000006                  ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
 2897  017750  000756                          BR      6$              ;;LET'S TRY IT AGAIN
 2898
 2899  017752  022716  000015          9$:     CMP     #15,(SP)        ;;IS IT A <CR>?
 2900  017756  001011                          BNE     11$             ;;BRANCH IF NO
 2901  017760  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
 2902  017764  001403                          BEQ     10$             ;;BRANCH IF YES
 2903  017766  016677  000002  161142          MOV     2(SP),@SWR      ;;SAVE NEW SWR
 2904  017774  062706  000006          10$:    ADD     #6,SP           ;;CLEAR UP STACK
 2905  020000  000417                          BR      13$             ;;RETURN TO USER
 2906  020002  022716  000012          11$:    CMP     #12,(SP)        ;;IS IT A <LF>
 2907  020006  001017                          BNE     15$             ;;BRANCH IF NO
 2908  020010  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
 2909  020014  001403                          BEQ     12$             ;;YES
 2910  020016  016677  000002  161112          MOV     2(SP),@SWR      ;;SAVE NEW SWR
 2911  020024  062706  000006          12$:    ADD     #6,SP           ;;CLEAR UP STACK
 2912  020030  013716  000046                  MOV     @#46,(SP)       ;;GET RESTART
 2913  020034  062716  000010                  ADD     #10,(SP)        ;;ADDRESS
 2914  020040  104400  001175          13$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
 2915  020044  000002                  14$:    RTI                     ;;RETURN
 2916  020046  004737  021456          15$:    JSR     PC,$TYPEC       ;;ECHO CHAR
```

```
 2917  020052  042726  177770                BIC     #177770,(SP)+   ;;RESTRICT TO 0-7
 2918  020056  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
 2919  020062  001403                        BEQ     16$             ;;BRANCH IF YES
 2920  020064  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
 2921  020066  006316                        ASL     (SP)            ;;   CHAR OVER TO MAKE
 2922  020070  006316                        ASL     (SP)            ;;   ROOM FOR NEW ONE.
 2923  020072  005266  000002          16$:  INC     2(SP)           ;;KEEP COUNT OF CHAR
 2924  020076  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
 2925  020102  000712                        BR      7$              ;;GET THE NEXT ONE
 2926                                  ;;**************************************************************
 2927                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
 2928                                  ;*CALL:
 2929                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
 2930                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
 2931                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
 2932                                  ;
 2933
 2934  020104  011646                $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
 2935  020106  016666  000004  000002        MOV     4(SP),2(SP)     ;;SAVE THE PS
 2936  020114  105777  161022          1$:   TSTB    @$TKS           ;;WAIT FOR
 2937  020120  100375                        BPL     1$              ;;A CHARACTER
 2938  020122  117766  161015  000004        MOVB    @$TKB,4(SP)     ;;READ THE TTY
 2939  020130  042766  177600  000004        BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
 2940  020136  026627  000004  000140        CMP     4(SP),#140      ;;IS IT UPPER CASE?
 2941  020144  002407                        BLT     2$              ;;BRANCH IF YES
 2942  020146  026627  000004  000175        CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
 2943  020154  003003                        BGT     2$              ;;BRANCH IF YES
 2944  020156  042766  000040  000004        BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
 2945  020164  000002                  2$:   RTI                     ;;GO BACK TO USER
 2946                                  ;;**************************************************************
 2947                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
 2948                                  ;*CALL:
 2949                                  ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
 2950                                  ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
 2951                                  ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
 2952
 2953  020166  010346                $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
 2954  020170  012703  020274          1$:   MOV     #$TTYIN,R3      ;;GET ADDRESS
 2955  020174  022703  020304          2$:   CMP     #$TTYIN+80.,R3  ;;BUFFER FULL?
 2956  020200  101405                        BLOS    4$              ;;BR IF YES
 2957  020202  104406                        RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
 2958  020204  112613                        MOVB    (SP)+,(R3)      ;;GET CHARACTER
 2959  020206  122713  000177          10$:  CMPB    #177,(R3)       ;;IS IT A RUBOUT
 2960  020212  001003                        BNE     3$              ;;SKIP IF NOT
 2961  020214  104400  001174          4$:   TYPE    $QUES           ;;TYPE A '?'
 2962  020220  000763                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
 2963  020222  111337  020272          3$:   MOVB    (R3),9$         ;;ECHO THE CHARACTER
 2964  020226  104400  020272                TYPE    9$
 2965  020232  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
 2966  020236  001356                        BNE     2$              ;;LOOP IF NOT RETURN
 2967  020240  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
 2968  020244  104400  001176                TYPE    $LF             ;;TYPE A LINE FEED
 2969  020250  012603                        MOV     (SP)+,R3        ;;RESTORE R3
 2970  020252  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
 2971  020254  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
 2972  020262  012766  020274  000004        MOV     #$TTYIN,4(SP)
```

```
                                        RTI                     ;;RETURN
                                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
                                        .BYTE   0               ;;TERMINATOR
                                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
                                $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
                                $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
                                $MSWR:  .ASCIZ  <15><12>/SWR = /
                                $MNEW:  .ASCIZ  /  NEW = /

                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT

                                $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
                                        MOV     4(SP),2(SP)     ;;INPUT NUMBER
                                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
                                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
                                1$:     RDLIN                   ;;READ AN ASCIZ LINE
                                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
                                        CLR     R1              ;;CLEAR DATA WORD
                                        CLR     R2
                                2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
                                        BEQ     3$              ;;IF ZERO GET OUT
                                        ASL     R1              ;;*2
                                        ROL     R2
                                        ASL     R1              ;;*4
                                        ROL     R2
                                        ASL     R1              ;;*8
                                        ROL     R2
                                        BIC     #↑C7,(SP)       ;;STRIP THE ASCII JUNK
                                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
                                        BR      2$              ;;LOOP
                                3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
                                        MOV     R1,12(SP)       ;;SAVE THE RESULT
                                        MOV     R2,$HIOCT
                                        MOV     (SP)+,R2        ;;POP STACK INTO R2
                                        MOV     (SP)+,R1        ;;POP STACK INTO R1
                                        MOV     (SP)+,R0        ;;POP STACK INTO R0
                                        RTI                     ;;RETURN
                                $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE


                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                ;;**********************************************************************
                                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
```

```
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
                                        $ERRTYP:
020442  104400  001175                          TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
020446  010046                                  MOV     R0,-(SP)        ;;SAVE R0
020450  005000                                  CLR     R0              ;;PICKUP THE ITEM INDEX
020452  153700  001114                          BISB    @#$ITEMB,R0
020456  001004                                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                        ;;TYPE THE PC OF THE ERROR
020460  013746  001116                          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
020464  104401                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
020466  000426                                  BR      6$              ;;GET OUT
020470  005300                          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
020472  006300                                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
020474  006300                                  ASL     R0
020476  006300                                  ASL     R0
020500  062700  001324                          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
020504  012037  020514                          MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
020510  001404                                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
020512  104400                                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
020514  000000                          2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
020516  104400  001175                          TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
020522  012037  020532                  3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
020526  001404                                  BEQ     5$              ;;SKIP TYPEOUT IF 0
020530  104400                                  TYPE                    ;;TYPE THE "DATA HEADER"
020532  000000                          4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
020534  104400  001175                          TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
020540  011000                          5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
020542  001004                                  BNE     7$              ;;GO TYPE THE DATA
020544  012600                          6$:     MOV     (SP)+,R0        ;;RESTORE R0
020546  104400  001175                          TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
020552  000207                                  RTS     PC              ;;RETURN
020554                                  7$:
020554  013046                                  MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
020556  104401                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
020560  005710                                  TST     (R0)            ;;IS THERE ANOTHER NUMBER?
020562  001770                                  BEQ     6$              ;;BR IF NO
020564  104400  020572                          TYPE    ,8$             ;;TYPE TWO(2) SPACES
020570  000771                                  BR      7$              ;;LOOP
020572  020040  000             8$:     .ASCIZ  /  /                    ;;TWO(2) SPACES
020576                                  .EVEN
```

```
                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;*****************************************************************
                                ;POWER DOWN ROUTINE
020576  012737  020742  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
020604  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
020612  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
020614  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
020616  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
020620  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
020622  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
020624  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
020626  017746  160304                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
020632  010637  020746                MOV     SP,$SAVR6       ;;SAVE SP
020636  012737  020650  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
020644  000000                        HALT
020646  000776                        BR      .-2             ;;HANG UP

                                ;;*****************************************************************
                                ;POWER UP ROUTINE
020650  012737  020742  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
020656  013706  020746                MOV     $SAVR6,SP       ;;GET SP
020662  005037  020746                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
020666  005237  020746          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
020672  001375                        BNE     1$              ;;OF  WORD
020674  012677  160236                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
020700  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
020702  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
020704  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
020706  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
020710  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
020712  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
020714  012737  020576  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
020722  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
020730  104400                        TYPE                    ;;REPORT THE POWER FAILURE
020732  020750                  $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
020734  012716                        MOV     (PC)+,(SP)      ;;RESTART AT BEGIN
020736  001516                  $PWRAD: .WORD   BEGIN           ;;RESTART ADDRESS
020740  000002                        RTI
020742  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
020744  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
020746  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
020750  005015  042522  052123  PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>
020756  051101  044524  043516
020764  040440  052106  051105
020772  040440  050040  053517
021000  051105  043040  044501
021006  052514  042522  005015
021014  000012

                                .EVEN
```

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

021016  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
021022  116637  000001  021241          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
021030  112637  021243                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
021034  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
021040  000406                          BR      $TYPON
021042  112737  000001  021241  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
021050  112737  000006  021243          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
021056  112737  000005  021240  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
021064  010346                          MOV     R3,-(SP)        ;;SAVE R3
021066  010446                          MOV     R4,-(SP)        ;;SAVE R4
021070  010546                          MOV     R5,-(SP)        ;;SAVE R5
021072  113704  021243                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
021076  005404                          NEG     R4
021100  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
021104  110437  021242                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
021110  113704  021241                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
021114  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
021120  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
021122  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
021124  000404                          BR      3$              ;;GO DO MSB
021126  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
021130  006105                          ROL     R5
021132  006105                          ROL     R5
021134  010503                          MOV     R5,R3
021136  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
021140  105337  021242                  DECB    $OMODE          ;;TYPE THIS DIGIT?
021144  100016                          BPL     7$              ;;BR IF NO
021146  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
021152  001002                          BNE     4$              ;;TEST FOR 0
```

```
3179   021154   005704                          TST     R4              ;;SUPPRESS THIS 0?
3180   021156   001403                          BEQ     5$              ;;BR IF YES
3181   021160   005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
3182   021162   052703   000060                 BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
3183   021166   052703   000040         5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3184   021172   110337   021236                 MOVB    R3,8$           ;;SAVE FOR TYPING
3185   021176   104400   021236                 TYPE    ,8$             ;;GO TYPE THIS DIGIT
3186   021202   105337   021240         7$:     DECB    $OCNT           ;;COUNT BY 1
3187   021206   003347                          BGT     2$              ;;BR IF MORE TO DO
3188   021210   002402                          BLT     6$              ;;BR IF DONE
3189   021212   005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3190   021214   000744                          BR      2$              ;;GO DO THE LAST DIGIT
3191   021216   012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
3192   021220   012604                          MOV     (SP)+,R4        ;;RESTORE R4
3193   021222   012603                          MOV     (SP)+,R3        ;;RESTORE R3
3194   021224   016666   000002  000004         MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3195   021232   012616                          MOV     (SP)+,(SP)
3196   021234   000002                          RTI                     ;;RETURN
3197   021236      000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3198   021237      000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3199   021240      000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3200   021241      000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
3201   021242   000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3202
3203                                           .SBTTL  TYPE ROUTINE
3204
3205                                           ;;**************************************************************
3206                                           ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3207                                           ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3208                                           ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3209                                           ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3210                                           ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3211                                           ;*
3212                                           ;*CALL:
3213                                           ;*1) USING A TRAP INSTRUCTION
3214                                           ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3215                                           ;*OR
3216                                           ;*      TYPE
3217                                           ;*      MESADR
3218                                           ;*
3219
3220   021244   105737   001155         $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
3221   021250   100002                          BPL     1$              ;;BR IF YES
3222   021252   000000                          HALT                    ;;HALT HERE IF NO TERMINAL
3223   021254   000430                          BR      3$              ;;LEAVE
3224   021256   010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
3225   021260   017600   000002                 MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
3226   021264   122737   000001  001220         CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
3227   021272   001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
3228   021274   132737   000100  001221         BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
3229   021302   001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
3230   021304   010037   021314                 MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
3231   021310   004737   021534                 JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
3232   021314   000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
3233   021316   132737   000040  001221  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
3234   021324   001003                          BNE     60$             ;;YES,SKIP TYPE OUT
```

```
3235  021326  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
3236  021330  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
3237  021332  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
3238  021334  012600                  6O$:    MOV     (SP)+,R0        ;;RESTORE R0
3239  021336  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
3240  021342  000002                          RTI                     ;;RETURN
3241  021344  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
3242  021350  001430                          BEQ     8$
3243  021352  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
3244  021356  001006                          BNE     5$
3245  021360  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
3246  021362  104400                          TYPE                    ;;TYPE A CR AND LF
3247  021364  001175                          $CRLF
3248  021366  105037  021522                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
3249  021372  000755                          BR      2$              ;;GET NEXT CHARACTER
3250  021374  004737  021456          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
3251  021400  123726  001154          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
3252  021404  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
3253  021406  013746  001152                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
3254                                                                  ;;AND THE NULL CHAR.
3255  021412  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
3256  021416  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
3257  021420  004737  021456                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
3258  021424  105337  021522                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
3259  021430  000770                          BR      7$              ;;LOOP
3260
3261                                  ;HORIZONTAL TAB PROCESSOR
3262
3263  021432  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
3264  021436  004737  021456          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
3265  021442  132737  000007  021522          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
3266  021450  001372                          BNE     9$              ;;TAB STOP
3267  021452  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
3268  021454  000724                          BR      2$              ;;GET NEXT CHARACTER
3269  021456  105777  157464          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
3270  021462  100375                          BPL     $TYPEC
3271  021464  116677  000002  157456          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3272  021472  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
3273  021500  001003                          BNE     1$              ;;BRANCH IF NO
3274  021502  105037  021522                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
3275  021506  000406                          BR      $TYPEX          ;;EXIT
3276  021510  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
3277  021516  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
3278  021520  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
3279  021522  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
3280  021524  000207          $TYPEX: RTS     PC
3281
3282
3283                                  .SBTTL  APT COMMUNICATIONS ROUTINE
3284
3285                                  ;;**************************************************************************
3286  021526  112737  000001  021772  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
3287  021534  112737  000001  021770  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
3288  021542  000403                          BR      $ATYC
3289  021544  112737  000001  021772  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
3290  021552          $ATYC:
```

```
3291  021552  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3292  021554  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3293  021556  105737  021770                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
3294  021562  001450                          BEQ     5$              ;;IF NOT:  BR
3295  021564  122737  000001  001220          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
3296  021572  001031                          BNE     3$              ;;IF NOT:  BR
3297  021574  132737  000100  001221          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
3298  021602  001425                          BEQ     3$              ;;IF NOT:  BR
3299  021604  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
3300  021610  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
3301  021616  005737  001200          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
3302  021622  001375                          BNE     1$              ;;IF NOT:  WAIT
3303  021624  010037  001214                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
3304  021630  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
3305  021632  001376                          BNE     2$
3306  021634  163700  001214                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
3307  021640  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
3308  021642  010037  001216                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
3309  021646  012737  000004  001200          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
3310  021654  000413                          BR      5$
3311  021656  017637  000004  021702  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
3312  021664  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
3313  021672  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
3314  021676  004737  021244                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
3315  021702  000000                  4$:     .WORD   0
3316  021704                          5$:
3317  021704  105737  021772          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
3318  021710  001416                          BEQ     12$             ;;IF NOT:  BR
3319  021712  005737  001220                  TST     $ENV            ;;RUNNING UNDER APT?
3320  021716  001413                          BEQ     12$             ;;IF NOT:  BR
3321  021720  005737  001200          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
3322  021724  001375                          BNE     11$             ;;IF NOT:  WAIT
3323  021726  017637  000004  001202          MOV     @4(SP),$FATAL   ;;GET ERROR #
3324  021734  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
3325  021742  005237  001200                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
3326  021746  105037  021772          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
3327  021752  105037  021771                  CLRB    $LFLG           ;;CLEAR LOG FLAG
3328  021756  105037  021770                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
3329  021762  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3330  021764  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3331  021766  000207                          RTS     PC              ;;RETURN
3332  021770     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
3333  021771     000          $LFLG:  .BYTE   0               ;;LOG FLAG
3334  021772     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
3335        021774                          .EVEN
3336        000200          APTSIZE=200
3337        000001          APTENV=001
3338        000100          APTSPOOL=100
3339        000040          APTCSUP=040
3340
3341                        .SBTTL  TRAP DECODER
3342
3343                        ;;**************************************************************************
3344                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3345                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3346                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
```

```
                                        ;*GO TO THAT ROUTINE.

                                $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
                                        MOV     2(SP),R0        ;;GET TRAP ADDRESS
                                        TST     -(R0)           ;;BACKUP BY 2
                                        MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
                                        ASL     R0              ;;POSITION FOR INDEXING
                                        MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
                                        RTS     R0              ;;GO TO ROUTINE

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
022016                          $TRPAD:
022016  021244                          $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
022020  021042                          $TYPOC  ;;CALL=TYPOC    TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
022022  021016                          $TYPOS  ;;CALL=TYPOS    TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
022024  021056                          $TYPON  ;;CALL=TYPON    TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
022026  010472                          $TYPDS  ;;CALL=TYPDS    TRAP+4(104404)  TYPE DECIMAL NUMBER (WITH SIGN)
022030  017646                          $CKSWR  ;;CALL=CKSWR    TRAP+5(104405)  TEST FOR CHANGE IN SOFT-SWR
022032  020104                          $RDCHR  ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
022034  020166                          $RDLIN  ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
022036  020340                          $RDOCT  ;;CALL=RDOCT    TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY

022040  000000                  ADBUFF: 0
022042  001020                          .BLKW   1020
024102  000000                  ABUFF4: 0
024104  000200                          .BLKW   200
024504  000000                  LAST:   0
000001                                  .END
```

```
ABASE = 170400           311#   430    474
ABUFF4  024102          2490*  2539   3378#
ACDW1 = 000000           430    476
ACDW2 = 000000           430    477
ACPUOP= 000000           430    445
ACRLF   016536           963   1279   1470   1764   2525   2541   2607   2651#
ADAC    015120          1910*  1922*  1934*  1937*  1964*  1969   1972   1982   2402#
ADBUFF  022040          2444   2454*  2456*  2467   2478   3376#
ADCS    001474           592#   712    800*   802*   809    815*   946*   947   1940*  1941   1945*  1955*  1924*
                        2050*  2051*  2052   2621*  2622
ADCS1   001476           593#   716*   940*  1954*
ADCBR   001500           594#   717    799*   813*   949    1948   1958   2054   2624
ADDIF   014224          2319#  2516
ADDIFM  014163          2313#  2475
ADDW0 = 000000           430    478
ADDW1 = 000000           430    479
ADDW10= 000000           430    488
ADDW11= 000000           430    489
ADDW12= 000000           430    490
ADDW13= 000000           430    491
ADDW14= 000000           430    492
ADDW15= 000000           430    493
ADDW2 = 000000           430    480
ADDW3 = 000000           430    481
ADDW4 = 000000           430    482
ADDW5 = 000000           430    483
ADDW6 = 000000           430    484
ADDW7 = 000000           430    485
ADDW8 = 000000           430    486
ADDW9 = 000000           430    487
ADEMSG  013525          1275   2258#
ADEND   011656           824    839    854    869    884    899    914    928    979    994   1008   1022   1037
                        1052   1067   1082   1606   1624   2049*  2054*  2057*  2058*  2059*  2060*  2061*  2062*
                        2067#
ADEVCT= 000000           430    436
ADEVM = 000000           430    475
ADNMUX  015162          1263*  1276   2419#
ADPMUX  015160          1233*  1272   2418#
ADSMSG  013436          1271   2248#
AENV  = 000000           430    441
AENVM = 000000           430    442
AFATAL= 000000           430    433
AMADR1= 000000           430    458
AMADR2= 000000           430    462
AMADR3= 000000           430    465
AMADR4= 000000           430    468
AMAMS1= 000000           430    452
AMAMS2= 000000           430    460
AMAMS3= 000000           430    463
AMAMS4= 000000           430    466
AMEAN   015214          2432#  2595
AMSGAD= 000000           430    438
AMSGLG= 000000           430    439
AMSGTY= 000000           430    432
AMTYP1= 000000           430    453
AMTYP2= 000000           430    461
```

```
AMTYP3= 000000           430     464
AMTYP4= 000000           430     467
APASS = 000000           430     435
APRIOR= 000200           313#    430     471
APTCSU= 000C40          3233    3339#
APTENV= 000001          2845    3226    3295    3337#
APTSIZ= 000200           671    3336#
APTSPO= 000100          3228    3297    3338#
ARBADD  001464           587#    628*    713     961     1773*   1777*   2362    2365    2367    2369    2371    2374
ARBVCT  001466           588#    629*   1774*   1778*   1923*   1953*   1959*
ASWREG= 000000           430     443
ATESTN= 000000           430     434
AUNIT = 000000           430     437
AUSWR = 000000           430     444
AVECT1= 000340           312#    430     469
AVECT2= 000000           430     470
BASBT1  013150          2212#   2676*   2682*
BASBT2  013152          2213#   2677*   2683*
BASBT3  013175          2217#   2678*   2679*   2684*   2685*
BEGIN   001516           340     603#    623     714     3110
BEGIN1  001536           341     608#
BEGIN2  001530           342     607#
BIASC1  015164           950*    965    2420#
BIASDN  013146           968    2211#
BIASST  013132           964    2209#
BIT0  = 000001           296#   1518    1544    1938
BIT00 = 000001           286#    296
BIT01 = 000002           285#    295
BIT02 = 000004           284#    294
BIT03 = 000010           283#    293
BIT04 = 000020           282#    292
BIT05 = 000040           281#    291
BIT06 = 000100           280#    290
BIT07 = 000200           279#    289
BIT08 = 000400           278#    288    2780
BIT09 = 001000           277#    287    2788    2855
BIT1  = 000002           295#
BIT10 = 002000           276#    990    1018    1048    1078    2833
BIT11 = 004000           275#    975    1018    1048    1063    2795
BIT12 = 010000           274#    771     955     990    1004    1048    1063    1198    1269    1452    1754    2473    2511
                        2530
BIT13 = 020000           273#   2840
BIT14 = 040000           272#   2766
BIT15 = 100000           271#   1314    1354    1397    1441    2376    2700
BIT2  = 000004           294#
BIT3  = 000010           293#
BIT4  = 000020           292#    801
BIT5  = 000040           291#
BIT6  = 000100           290#   1944
BIT7  = 000200           289#    786     801     943    1131    1141    1174    1184    1222    1229    1252    1259.   1481
                        1500    1518    1544    1570    1589    1646    1708
BIT8  = 000400           288#    806
BIT9  = 001000           287#    778     779     787     975    1004    1048    1078
BPTVEC= 000014           303#
BYPAS1  010340          1776    1782#   1821
BYTWO   011466          1455    1459    1463    1467    2025#
```

```
CHANL   011662            939*   1918*   2046*   2069#   2362    2365
CH0   = 000000            727#    823
CH1   = 000001            728#
CH10  = 000010            735#
CH11  = 000011            736#   1605
CH12  = 000012            737#   1623
CH13  = 000013            738#
CH14  = 000014            739#
CH15  = 000015            740#
CH16  = 000016            741#
CH17  = 000017            742#
CH2   = 000002            729#    853
CH3   = 000003            730#    883    1257
CH4   = 000004            731#    913    1227
CH40  = 000040            746#    838
CH41  = 000041            747#    939     940
CH42  = 000042            748#    868
CH43  = 000043            749#    898
CH44  = 000044            750#
CH45  = 000045            751#
CH46  = 000046            752#
CH47  = 000047            753#   1332    1352    1498    1542
CH47PK  015136           1357*   1461    2409#
CH47RM  015134           1337*   1460    2408#
CH5   = 000005            732#   1129    1139    1568    1705
CH50  = 000050            754#
CH51  = 000051            755#
CH52  = 000052            756#
CH53  = 000053            757#    978     993
CH54  = 000054            758#   1007    1021
CH55  = 000055            759#   1036    1051
CH56  = 000056            760#   1066    1081
CH57  = 000057            761#    927
CH6   = 000006            733#   1172    1182    1227    1257    1587    1644
CH7   = 000007            734#   1479    1516
CH7PEK  015132           1317*   1400    1444    1457    2407#
CH7RMS  015130           1297*   1378    1422    1456    2406#
CKSWR = 104405           2765    3371#
COMPAR  011666            826     841     856     871     886     901     916     930     951    1149    1192    1236    1266
                         1487    1506    1524    1531    1550    1557    1575    1594    1608    1626    1676    1737    2072#
CONS    015172           1913*   1925    1930    1938    1974    1979    2423#
CONVR   015310           2450#   2459    2472
CONVRT  011534            822     837     852     867     882     897     912     926     977     992    1006    1020    1035
                         1050    1065    1080    1604    1622    2045#
CONVT   016340           2450    2464    2617#
CR    = 000015            211#   3272    3282
CRLF  = 000200            212#   3243    3282
CSB     001504            596#
CSC     001514            600#
CSR     001502            595#
DACSAV  015122           1133    1143    1144    1176    1186    1187    1224    1231    1254    1261    1483    1484    1502
                         1503    1520    1521    1546    1547    1571    1590    1648    1710    1969*   2403#
DDISP = 177570            218#    406     658
DECPNT  011450           1997*   2000    2002    2009*   2014#
DECPRT  011342           1994#   2572    2578    2734
DH1     012451            517    2155#
```

L06


```
DH10    013004            559    2194#
DH11    013063            565     571     577     583    2202#
DH2     012530            523    2163#
DH3     012577            529     535    2170#
DH5     012634            541    2175#
DH6     012674            547    2181#
DH7     012735            553    2187#
DIEMSG  013377           1208    2242#
DIFCHN  016644           2437*   2621    2669#
DIFERR  015742           1109    2481*   2510*   2536#
DIFLIN  015220           1095    2435#
DIFMSG  013203           1200    2219#
DIGCNT  011446           1994*   2005*   2013#
DIGIT   011444           1998*   1999*   2003*   2004    2012#
DIGT0   011462           2020#   2579
DIGT1   011463           1996    2021#   2573    2580    2738
DIGT2   011464           2022#   2574    2581    2739
DIGT3   011465           2023#   2575    2582    2741
DISPLA  001140            406#    658*    664*   2810*   2832*
DISPRE  000174            336#    664
DIYMSG  013367           1204    2240#
DSWR  = 177570            217#    405     657
DT1     014572            518    2362#
DT10    014650            560    2371#
DT11    014666            566     572     578     584    2374#
DT2     014610            524     554    2365#
DT3     014624            530     536     548    2367#
DT5     014636            542    2369#
EDGE    015170           1912*   1948    2371    2422#
EMTVEC= 000030            306#    641*    642*
EM1     011736            516    2091#
EM10    012260            558    2129#
EM11    012321            564    2135#
EM12    012362            570    2141#
EM13    012377            576    2144#
EM14    012421            582    2148#
EM14A   012445           1681*   1742*   2153#
EM14B   012444           1688*   1749*   2152#
EM2     011763            522    2095#
EM3     012027            528    2102#
EM4     012063            534    2107#
EM5     012120            540    2112#
EM6     012161            546    2118#
EM7     012214            552    2123#
ERRVEC= 000004            299#    655     656*    667*   2771    2772*   2774*   2777*
EXCESS  015744           1105    2486*   2503*   2537#   2546    2609
FCHANL  011660           1919*   1920*   1921*   1940    1944*   1945    2068#
FINS    015174           1924*   1970    1973*   2424#
GNS   = ****** U          335     694     698     705     709     959    3366    3367    3368    3369    3370    3371    3372
                         3373    3374
GRT2MG  014271           2326#   2549
HILIM1  015202           2427#   2491    2593
HILIM2  015204           2428#   2493
HT    = 000011            209#   3241    3282
IOTVEC= 000020            304#    639*    640*
LAST    024504           2447    3380#
```

```
LF     = 000012            210*   3276   3282
LINMG1   014140           1760   2309*
LINMSG   013755           1756   2288*
LOLIM1   015176           2425*  2497   2591
LOLIM2   015200           2426*  2499
MSGPNT   017040           2720*  2722*  2735
MULTSP   014541           2028   2356*
NARMSG   014320           2330*  2553
NARROW   015210           2430*  2482*  2509*  2550   2560
NBEXT    001472            590*   630*  1771   1775*  1779*
NMBEXT   001470            589*   625*   630    700   1779
NOIMG1   013664           1458   2276*
NOIMG2   013707           1462   2280*
NOIMG3   013732           1466   2284*
NOIMSG   013543           1454   2261*
NUMBF2   014702           1637   1699   2376*
OFSLBX   015154           1520*  2416*
OFSLUX   015156           1546*  2417*
OFSTBX   015150           1483*  1528   1572   1591   2414*
OFSTUX   015152           1502*  1554   2415*
OVERNG   016516           2608   2648*
PC     =%000007            230*   684*   826*   841*   856*   871*   886*   901*   916*   930*   951*  1149*  1192*
                          1236*  1266*  1487*  1506*  1524*  1531*  1550*  1557*  1575*  1594*  1608*  1626*  1676*
                          1737*  1802*  1805*  1815*  1820   2008*  2084*  2088*  2450*  2464*  2572*  2578*  2625*
                          2689*  2734*  2842*  2848*  2916*  3061*  3109   3231*  3250*  3257*  3264*  3278*  3280*
                          3314*  3331*
PCT      017042           2721*  2723*  2730*  2732*  2733   2736
PEKMAX   015126           1318   1358   1402   1446   2405*
PERHLF   014420           2342*  2562   2563
PERQRT   014477           2350*  2566   2567
PIRQ   = 177772            216*
PIRQVE= 000240             310*
PRCNT    017044           2561   2565   2722*
PR0    = 000000            233*
PR1    = 000040            234*
PR2    = 000100            235*
PR3    = 000140            236*
PR4    = 000200            237*
PR5    = 000240            238*
PR6    = 000300            239*
PR7    = 000340            240*
PS     = 177776            213*   214    610*
PSW    = 177776            214*
PWRMSG   020750           3108   3115*
PWRVEC= 000024             305*   645*   646*  3077*  3078*  3087*  3093*  3105*  3106*
QRTOK    015206           2429*  2480*  2501*  2564
RBEG     001542            605    606    609*
RBEG2    002132            675*  1783
RDCHR  = 104406           2891   2957   3372*
RDLIN  = 104407           2999   3373*
RDOCT  = 104410           3374*
READ     015426           2461   2473*
RESVEC= 000010             300*
RMSMAX   015124           1298   1338   1380   1424   2404*
RSLTO    014726           1638   1653   1654   1660*  1661*  1662*  1663*  1667*  1668*  1669*  1670*  1671*  1673*
                          1675   1758   2380*
```

DZARCB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
RSLT1   014764          1700    1715    1716    1722*   1723*   1724*   1725*   1729*   1730*   1731*   1732*   1733*   1735*
                        1736    1762    2385#
RSLT2   015022          1122    1202    2390#
RSLT3   015060          1165    1206    2395#
RSVMSG  016376          2584    2627#
RSV1    016400          2573*   2628#
RSV2    016402          2574*   2629#
RSV3    016403          2575*   2630#
RSX1    016405          2579*   2631#
RSX2    016406          2580*   2632#
RSX3    016407          2581*   2633#
RSX4    016410          2582*   2634#
R0      =%000000         221#    603*    608*    685     712*    713*    714     717*    719*   1637*   1640    1641    1649
                        1653*   1654*   1655*   1656*   1657*   1658*   1661    1699*   1702    1703    1711    1715*   1716*
                        1717*   1718*   1719*   1720*   1723    1782*   1812*   1815    1842    1852*   1856    1872    1873
                        1886*   1914    1936*   1960*   1988*   2034*   2036    2072    2074*   2076*   2078*   2079    2082*
                        2087*   2452    2476*   2487*   2488    2490*   2491    2493    2495    2497    2499    2507    2521*
                        2522    2558*   2559*   2560*   2564*   2723    2724*   2725*   2726*   2727*   2728*   2729*   2730
                        2731*   2732    2735*   2738*   2739*   2740*   2741*   2744*   2745*   2746*   2747*   2996    3000*
                        3003    3019*   3033    3034*   3035*   3042*   3043*   3044*   3045*   3046*   3047    3052    3057*
                        3059*   3063    3065    3079    3104*   3224    3225*   3230    3235    3238*   3291    3299*   3303
                        3304    3306*   3307*   3308    3330*   3349    3350*   3351    3352*   3353*   3354*   3355*
R1      =%000001         222#    677*    678*    679*    680     718*    719     720     721    1121*   1124    1125    1135
                        1152*   1153    1164*   1167    1168    1178    1195*   1196    1659*   1662*   1663    1664    1665
                        1672*   1673*   1675    1678    1682    1690    1691    1721*   1724*   1725    1726    1727    1734*
                        1735*   1736    1739    1743    1751    1752    1843    1856*   1857    1861    1885*   1915    1935*
                        1950*   1962    1987*   1995    1996*   2004*   2007*   2035*   2040*   2073    2075*   2076    2081*
                        2086*   2672*   2686*   2687    2997    3001*   3005*   3007*   3009*   3012*   3015    3018*   3090
                        3103*   3292    3329*
R2      =%000002         223#    613*    615     616*    676*    678     680*    681     682    1122*   1146*   1165*   1189*
                        1638*   1648*   1678*   1679*   1680*   1681    1682*   1683*   1684*   1685*   1686*   1697*   1698
                        1700*   1710*   1739*   1740*   1741*   1742    1743*   1744*   1745*   1746*   1747*   1748*   1749
                        1844    1855*   1859*   1862    1869*   1870*   1871    1876*   1884*   1916    1933*   1937    1964
                        1965    1967*   1986*   2000*   2002*   2570*   2571*   2577*   2675*   2681*   2686    2733*   2998
                        3002*   3006*   3008*   3010*   3016    3017*   3091    3102*
R3      =%000003         224#    614*    617*    620     624*    625     1845    1853*   1854*   1868*   1871*   1880*   1881*
                        1883*   1917    1927*   1929*   1932*   1962    1976*   1978*   1995*   6497    6527    6577    6753
                        2954*   2955    2958*   2959    2963    2965    2967*   2969*   3082    3101*   3158    3167*   3173*
                        3174*   3177*   3182*   3183*   3184    3193*
R4      =%000004         225#    1127*   1137*   1170*   1180*   1218*   1225*   1248*   1255*   1290*   1310*   1330*   1350*
                        1371*   1393*   1415*   1437*   1477*   1496*   1514*   1540*   1566*   1595*   1642*   1704*   1910
                        1911    1912    1913    1989*   3083    3100*   3159    3161*   3162*   3163*   3164    3165*   3179
                        3181*   3185*   3192*
R5      =%000005         226#    822*    837*    852*    867*    882*    897*    912*    926*    977*    982*   1006*   1020*
                        1035*   1050*   1065*   1080*   1095*   1201*   1205*   1296    1297    1316    1317    1336    1337
                        1356    1357    1377    1399    1421    1443    1455*   1459*   1463*   1467*   1604*   1622*   1767*
                        1761*   1846    1848*   1850*   1857*   1861*   1876    1982*   1972*   1982*   1983*   2025    2029
                        2032*   2034    2035    2042*   2045    2063*   2435    2436    2437    2438    2452    2456*   2455*
                        2469*   2532*   2533*   2561*   2565*   2585*   2586*   2612*   2613*   2624*   2722    2742*   2748*
                        3084    3099*   3160    3166*   3168*   3170*   3171*   3172*   3173    3191*
R6      =%000006         227#    229     633*    634*    635
R7      =%000007         228#    230
SAR       010716        1127    1137    1170    1180    1218    1225    1248    1255    1290    1310    1330    1350    1371
                        1393    1415    1437    1477    1496    1514    1540    1566    1595    1642    1704    1910#
SARCHN  015166          1911*   1918    1920    1951    1954    2421#
SKIPST  015740          1101    2485*   2506*   2535#   2542
```

```
SKPMSG  014242          2322#  2545
SP     =000006           229#   611*   619    637*   655*   665*   667    700*   961*   965*  1272*  1276*  1809*
                        1842*  1843*  1844*  1845*  1846*  1847*  1848   1851*  1864   1866*  1868   1878   1880
                        1882   1883   1884   1885   1886   1888*  1889*  1914*  1915*  1916*  1917*  1947   1957
                        1985   1986   1987   1988   1995*  2007   2025*  2029*  2036*  2072*  2073*  2081   2082
                        2083*  2086   2087   2438*  2517*  2522*  2532   2542*  2546*  2550*  2554*  2609*  2612
                        2771*  2774   2776   2777   2806   2807   2811*  2837   2857*  2860*  2879*  2880*  2881
                        2886*  2889*  2890*  2893   2896*  2899   2901   2903   2904*  2906   2908   2910   2911*
                        2912*  2913*  2917*  2918   2920*  2921*  2922*  2923*  2924*  2934*  2935*  2938*  2939*
                        2940   2942   2944*  2953*  2958   2969   2970*  2971*  2972*  2994*  2995*  2996*  2997*
                        2998*  3000   3003*  3011*  3012   3014   3015*  3017   3018   3019   3033*  3038*  3059
                        3063*  3079*  3080*  3081*  3082*  3083*  3084*  3085*  3086   3094*  3098   3099   3100
                        3101   3102   3103   3104   3109*  3150*  3151   3152   3153*  3158*  3159*  3160*  3166
                        3191   3192   3193   3194*  3195*  3224*  3225   3235*  3237   3238   3239*  3241   3243
                        3245   3251   3253*  3255*  3263*  3267   3271   3272   3276   3291*  3292*  3299   3300*
                        3311   3312*  3313*  3323   3324*  3329   3330   3349*  3350
SPREAD  011664           825*   840*   855*   870*   885*   900*   915*   929*   942*  1148*  1191*  1235*  1265*
                        1486*  1505*  1523*  1530*  1549*  1556*  1574*  1593*  1607*  1625*  1652*  1714*  2070#
                        2079   2362   2371   2374
SP3MSG  016543          2039   2653#
STACK = 001100           204#   611    637
STDMSG  016554          2569   2656#
STKLMT= 177774           215#
SWDIST  015746          2531   2539#
SWR     001136           405#   657*   659    663*   673*   955   1198   1269   1452   1754   2473   2511   2530
                        2766   2780   2782   2788   2795   2833   2840   2852   2855   2875   2903*  2910*  3085
                        3098*
SWREG   000176           337#   663    2875   2886
SWD    = 000001           268#
SWD0   = 000001           258#   268
SWD1   = 000002           257#   267
SWD2   = 000004           256#   266
SWD3   = 000010           255#   265
SWD4   = 000020           254#   264
SWD5   = 000040           253#   263
SWD6   = 000100           252#   262
SWD7   = 000200           251#   261
SWD8   = 000400           250#   260
SWD9   = 001000           249#   259
SW1    = 000002           267#
SW10   = 002000           248#
SW11   = 004000           247#
SW12   = 010000           246#
SW13   = 020000           245#
SW14   = 040000           244#
SW15   = 100000           243#
SW2    = 000004           266#
SW3    = 000010           265#
SW4    = 000020           264#
SW5    = 000040           263#
SW6    = 000100           262#
SW7    = 000200           261#
SW8    = 000400           260#
SW9    = 001000           259#
T3ITVE= 000014           301#
TEMP    015216           806*   807*   943*   944*  2433#  2479*  2517   2526*  2540*  2570   2591   2593   2595
```

```
                        2602*   2604
TKVEC  =  000060         308#
TPVEC  =  000064         309#
TRAPVE=   000034         307#    643*    644*
TRTVEC=   000014         302#
TST1      002542         758#
TST10     003340         887     894#
TST11     003410         902     909#
TST12     003460         917     923#
TST13     003530         931     937#
TST14     003736         956     973#
TST15     004010         981     987#
TST16     004066         996    1002#
TST17     004140        1010    1016#
TST2      002702         790     796#
TST20     004212        1024    1031#
TST21     004262        1039    1046#
TST22     004334        1054    1061#
TST23     004406        1069    1076#
TST24     004464        1092#
TST25     004564        1112    1119#
TST26     004752        1155    1162#
TST27     005204        1199    1209    1215#
TST3      003030         819#
TST30     005332        1237    1239    1245#
TST31     005524        1270    1280    1287#
TST32     005600        1300    1307#
TST33     005654        1320    1327#
TST34     005730        1340    1347#
TST35     006004        1360    1367#
TST36     006076        1382    1389#
TST37     006170        1404    1411#
TST4      003100         827     834#
TST40     006262        1426    1433#
TST41     006450        1453    1475#
TST42     006534        1488    1494#
TST43     006620        1507    1512#
TST44     006734        1532    1538#
TST45     007050        1558    1563#
TST46     007140        1576    1582#
TST47     007230        1595    1600#
TST5      003150         842     849#
TST50     007324        1618#
TST51     007420        1635#
TST52     007712        1697#
TST53     010250        1755    1769#
TST6      003220         857     864#
TST7      003270         872     879#
TYANX     016541        2589    2652#
TYANXX    016552        2520    2655#
TYHIGH    016442        2601    2640#
TYLOW     016414        2599    2636#
TYMEAN    016472        2597    2644#
TYPDS  =  104404        1810    3370#
TYPE   =  104400         692     696     703     707     957     963     964     968    1200    1204    1208    1271    1275
                        1279    1454    1458    1462    1466    1470    1756    1760    1764    1809    1811    1897    2028
```

```
                    2039    2475    2516    2520    2525    2541    2545    2549    2553    2557    2563    2567    2568
                    2583    2588    2597    2599    2601    2606    2608    2835    2843    2883    2885    2888    2895
                    2914    2961    2964    2968    3032    3049    3051    3054    3056    3060    3067    3107    3185
                    3246    3366#
TYPOC = 104401       962    2887    3040    3064    3367#
TYPON = 104403      3369#
TYPOS = 104402       701     966    1273    1277    2026    2030    2037    2518    2523    2543    2547    2551    2555
                    2610    3368#
TYPSTG  011510      1201    1205    1757    1761    2034#
VARLT1  017004      2675    2702#
VARLT2  017022      2681    2710#
VA24    016772       941     942    2696#
VCREG1  016640      2435*   2440*   2442*   2460    2462*   2667#
VCREG2  016642      2436*   2441*   2443*   2457*   2458    2463*   2668#
VCSTAT  001506       597#    770*    772*    773     779*    780     787*    788     798*    803*    804     814*    975*
                     989*    990*   1004*   1018*   1033*   1048*   1063*   1078*   1087*   2439*   2617*   2618
VCXREG  001510       596#   1096    1124*   1134*   1171    1181    1219    1226    1249    1256    1369*   1392*   1416
                    1438    1565*   1586    1602*   1611*   1643    1703*
VCYREG  001512       599#   1097    1128    1138    1167*   1177*   1217*   1247*   1291    1311    1331    1351    1372
                    1394    1414*   1436*   1478    1497    1515    1541    1567    1584*   1620*   1629*   1641*   1705
V1      017000      2495    2699#
V1754   016764       881    2672    2693#
V24     016766       885     911     915    2694#
V50     016770       900    2695#
V62     016774      1148    1191    1486    1505    1523    1549    2697#
V77     016776      1574    1593    1652    1714    2698#
WFADJ   016646       684    2672#
WFTEST  016762       604*    607*   2673    2691#
WIDE    015212      2431#   2483*   2504*   2554    2559
WIDMSG  014360      2336#   2557
XDACPK  015142      1401*   1465    2411#
XDACRM  015140      1379*   1464    2410#
YDACPK  015146      1445*   1469    2413#
YDACRM  015144      1423*   1468    2412#
$APTHD  001000       366     372#
$ARG1   002110       669#
$ASTAT= ****** U    3317    3332
$ATYC   021552      3288    3290#
$ATY1   021526      3286#
$ATY3   021534      3231    3287#
$ATY4   021544      2848    3289#
$BASE   001254       474#    613     628    1777
$BDADR  001122       399#
$BDDAT  001126       401#    635     773*    774     780*    781     788*    789     809*    810     824*    839*    854*
                     869*    884*    899*    914*    928*    949*    950     979*    980     994*    995    1008*   1009
                    1022*   1023    1037*   1038    1052*   1053    1067*   1068    1082*   1083    1101*   1105*   1109*
                    1111    1144*   1145*   1146    1187*   1188*   1189    1231*   1232*   1233    1261*   1262*   1263
                    1296*   1299    1316*   1319    1336*   1339    1356*   1359    1377*   1378*   1379    1381    1399*
                    1400*   1401    1403    1421*   1422*   1423    1425    1443*   1444*   1445    1447    1494*   1503*
                    1521*   1528*   1547*   1554*   1571*   1572*   1590*   1591*   1606*   1624*   1675*   1736*   2075
                    2362    2365    2367    2369    2371    2374
$BELL   001170       421#   2835    2867
$CDW1   001260       476#
$CDW2   001262       477#
$CHARC  021522      3248*   3258*   3265    3274*   3279#
$CKSWR  017646      2875#   3371
```

```
$CMTAG  001100            387#    632     633     641     647     648     649
$CM1   =  000002          417#    418#    419#
$CM2   =  000004          417#    418#    419#
$CM3   =  000002          415#    417
$CNTLG  020311           2883    2978#
$CNTLU  020304           2895    2977#
$CPUOP  001226            445#
$CRLF   001175            423#   2843    2867    2914    2977    3032    3051    3056    3060    3247    3282
$DBLK   010706           1853    1887    1895#
$DDW0   001264            478#
$DDW1   001266            479#
$DDW10  001310            488#
$DDW11  001312            489#
$DDW12  001314            490#
$DDW13  001316            491#
$DDW14  001320            492#
$DDW15  001322            493#
$DDW2   001270            480#
$DDW3   001272            481#
$DDW4   001274            482#
$DDW5   001276            483#
$DDW6   001300            484#
$DDW7   001302            485#
$DDW8   001304            486#
$DDW9   001306            487#
$DEVCT  001210            436#
$DEVM   001256            475#
$DOAGN  010446           1804    1813    1819#
$DTBL   010676           1856    1891#
$ENDAD  010436            351    1815#   2862
$ENDCT  010404            647    1806#
$ENDMG  010455           1808    1823#
$ENULL  010452           1811    1822#
$ENV    001220            441#   2845    3226    3295    3319
$ENVM   001221            442#    671    3228    3233    3297
$EOP    010350           1781    1796#
$EOPCT  010376            647*   1803#   1807
$ERFLG  001103            390#   2755    2784    2786    2792*   2[illegible]4   2830*   2867
$ERMAX  001115            396#    650*   2786    2809*   2814
$ERROR  017452            641    2829#
$ERRPC  001116            397#   2362    2365    2367    2369    2371    2374    2837*   2838*   2839    2867    3039
$ERRTB  001324            513#   3046
$ERRTY  020442           2842    3031#
$ERTTL  001112            394#   2836*   2867
$ESCAP  001166            420#    649*   2808*   2858    2860    2867
$ETABL  001220            440#
$ETEND  001324            378     496#
$FATAL  001202            433#   3323*
$FFLG   021772           3286*   3289*   3317    3326*   3334#
$FILLC  001154            413#   3251    3282
$FILLS  001153            412#   3282
$GDADR  001120            398#
$GDDAT  001124            400#    771*    772     774     778*    781     786*    789     801*    802     810     821*    836*
                          851*    866*    881*    896*    911*    925*    941*    976*    980     991*    995    1005*   1009
                         1019*   1023    1034*   1038    1049*   1053    1064*   1068    1079*   1083    1100*   1110*   1111
                         1147*   1190*   1234*   1264*   1298*   1299    1318*   1319    1338*   1339    1358*   1359    1390*
```

```
                    1381    1402*   1403    1424*   1425    1446*   1447    1485*   1504*   1522*   1529*   1548*   1555*
                    1573*   1592*   1603*   1612*   1621*   1630*   1651*   1713*   2074    2362    2365    2367    2369
                    2371    2374
$GET42  010426      1812#
$HD   = 000000       200
$HIBTS  001000       373#
$HIOCT  020440      3016*   3021#
$ICNT   001104       391#   2799*   2800    2802*   2813
$ILLUP  020742      3077    3093    3112#
$ITEMB  001114       395#   2839*   2847    2867    3035
$LF     001176       424#   2867    2968    2977    3282
$LFLG   021771      3327*   3333#
$LPADR  001106       392#    651*   2790*   2806*   2811    2813
$LPERR  001110       393#    652*   2790    2807*   2813    2857
$MADR1  001232       458#
$MADR2  001236       462#
$MADR3  001242       465#
$MADR4  001246       468#
$MAIL   001200       374     378     431#    669     2805    2845    3226
$MAMS1  001230       452#
$MAMS2  001234       460#
$MAMS3  001240       463#
$MAMS4  001244       466#
$MBADR  001002       374#
$MFLG   021770      3287*   3293    3328*   3332#
$MNEW   020327      2888    2981#
$MSGAD  001214       438#   3303*   3306
$MSGLG  001216       439#   3308*
$MSGTY  001200       432#   3301    3309*   3321    3325*
$MSWR   020316      2885    2979#
$MTYP1  001231       453#
$MTYP2  001235       461#
$MTYP3  001241       464#
$MTYP4  001245       467#
$MXCNT  017450      2803    2813#
$NULL   001152       411#   3253    3282
$NWTST= 000001       765#    793#    816#    831#    846#    861#    876#    891#    906#    920#    934#    970#    984#
                     999#   1013#   1028#   1043#   1058#   1073#   1089#   1116#   1159#   1212#   1242#   1294#   1304#
                    1324#   1344#   1364#   1386#   1408#   1430#   1472#   1491#   1509#   1535#   1560#   1579#   1597#
                    1615#   1632#   1694#   1766#
$OCNT   021240      3157*   3186*   3199#
$OMODE  021242      3152*   3156*   3161    3164*   3175*   3201#
$OVER   017434      2767    2783    2791    2801    2810#
$PASS   001206       435#    670*   1800*   1801*   1809    1822    2797    2814
$PASTM  001006       376#
$PRIOR  001252       471#
$PWRAD  020736      3110#
$PWRDN  020576       645    3077#   3105
$PWRMG  020732      3108#
$PWRUP  020650      3087    3093#
$QUES   001174       422#   2867    2961    2977    3282
$RDCHR  020104      2934#   3372
$RDDEC= ****** U    3375
$RDLIN  020166      2953#   3373
$RDOCT  020340      2994#   3374
$RDSZ = 000010      2946#
```

```
$REGAD  001156          415#
$REGO   001160          417#
$REG1   001162          418#
$RTNAD  010450         1821#
$R2A  = ****** U        3375
$SAVRE= ****** U        3375
$SAVR6  020746          3066*   3094    3095*   3096*   3114#
$SCOPE  017172           639    2764#
$SETUP= 000037           632#    638     639     641     643     645     647     648     649     651     668    1798    2765
                        2830    2855    2862
$STUP = 177777           632#
$SVLAD  017400          2775    2804#
$SVPC = 000214           349#    354
$SWR  = 167400           190#    200     320     321     322     323     324     325     326     327     419     420     421
                         648     649     651     652     769     797     820     835     850     865     880     895     910
                         924     938     974     988    1003    1017    1032    1047    1062    1077    1093    1120    1163
                        1216    1246    1288    1308    1328    1348    1368    1390    1412    1434    1476    1495    1513
                        1539    1564    1583    1601    1619    1636    1698    1770    1793    1799    1814    1820    1822
                        2756    2757    2758    2759    2760    2766    2778    2780    2781    2784    2785    2786    2793
                        2794    2795    2807    2810    2813    2821    2822    2823    2824    2825    2833    2840    2852
                        2855    2867    3111
$SWREG  001222           443#    673
$SWRMK= 000000           327     328    2760    2761    2782
$TESTN  001204           434#   2805*
$TIMES  001164           419#    648*    769*    797*    820*    835*    850*    865*    880*    895*    910*    924*    938*
                         974*    988*   1003*   1017*   1032*   1047*   1062*   1077*   1093*   1120*   1163*   1216*   1246*
                        1288*   1308*   1328*   1348*   1368*   1390*   1412*   1434*   1476*   1495*   1513*   1539*   1564*
                        1583*   1601*   1619*   1636*   1698*   1770*   1799*   2793*   2800    2803*   2813
$TKB    001144           408#   2871    2879    2938
$TKS    001142           407#   2871    2877    2936
$TN   = 000054           190#    200     765     769*    790     793     797*    816     820*    827     831     835*    842
                         846     850*    857     861     865*    872     876     880*    887     891     895*    902     906
                         910*    917     920     924*    931     934     938*    956     970     974*    981     984     988*
                         996     999    1003*   1010    1013    1017*   1024    1028    1032*   1039    1043    1047*   1054
                        1058    1062*   1069    1073    1077*   1089    1093*   1112    1116    1120*   1155    1159    1163*
                        1199    1209    1212    1216*   1237    1239    1242    1246*   1270    1280    1284    1288*   1300
                        1304    1308*   1320    1324    1328*   1340    1344    1348*   1360    1364    1368*   1382    1386
                        1390*   1404    1408    1412*   1426    1430    1434*   1453    1472    1476*   1488    1491    1495*
                        1507    1509    1513*   1532    1535    1539*   1558    1560    1564*   1576    1579    1583*   1595
                        1597    1601*   1615    1619*   1632    1636*   1694    1698*   1755    1766    1770#
$TPB    001150           410#   3271*   3282
$TPFLG  001155           414#   3220    3282
$TPS    001146           409#   3269    3282
$TRAP   021774           643    3349#
$TRP  = 000011          3357#   3367#   3368#   3369#   3370#   3371#   3372#   3373#   3374#   3375#
$TRPAD  022016          3354    3365#
$TSTM   001004           375#
$TSTNM  001102           389#   1798*   2755    2782    2804*   2805    2810    2814    2832    2867
$TTYIN  020274          2954    2955    2972    2976#
$TYPBN= ****** U        3371
$TYPDS  010472          1841#   3370
$TYPE   021244          3220#   3314    3357    3366
$TYPEC  021456          2916    3250    3257    3264    3269#   3270
$TYPEX  021524          3275    3277    3280#
$TYPOC  021042          3155#   3367
$TYPON  021056          3154    3157#   3369
```

```
$TYPOS  021016          3150#   3368
$UNIT   001212           437#
$UNITM  001010           377#
$USWR   001224           444#
$VECT1  001250           469#    629    1778
$VECT2  001251           470#
$XTSTR  017204          2769#
$$GET4= 000000          1814#
$OFILL  021241          3151*   3155*   3165    3200#
$40CAT= ****** U        2768    2842
.       = 024506         331#    335#    349     350#    352#    354#    355#    362     363#    365#    367#    386#    425
                         636     651     652     695#    699#   1822    1826   1895#   2014    2666#   2813    2814    2867
                        2871    2976#   2977    2983    3070#   3089    3113    3282   3335#   3377#   3379#
.$ASTA= ****** U        3287    3290
.$X     = 001000         362#    367
```

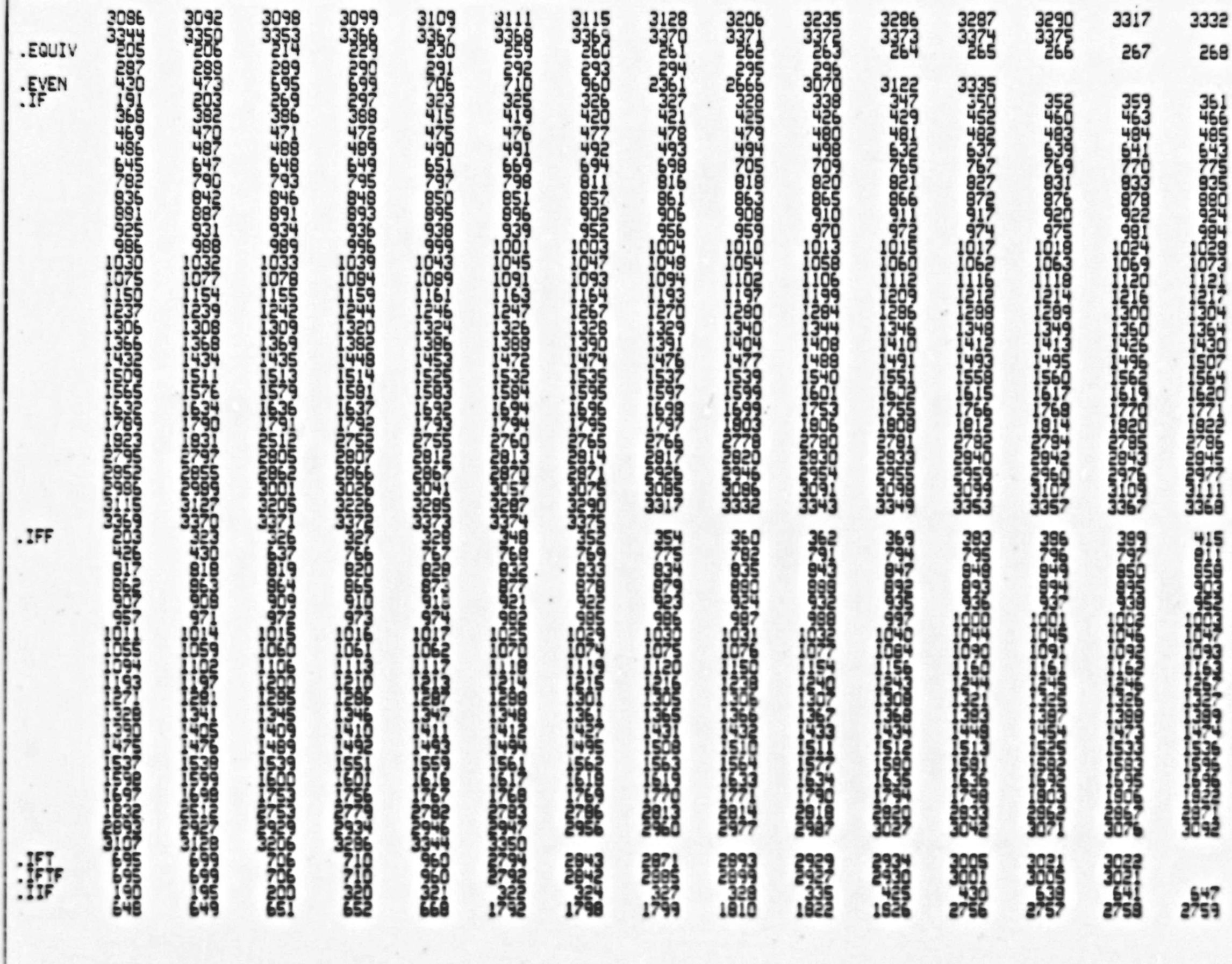
```
COMMEN    311#
ENDCOM    311#
ERROR     205#    776    783    791    812    828    843    858    873    888    903    918    932    953    982
              997   1011   1025   1040   1055   1070   1085   1103   1107   1113   1151   1194   1238   1268   1301
             1321   1341   1361   1383   1405   1427   1449   1489   1508   1526   1533   1552   1559   1577   1596
             1610   1628   1689   1750
ESCAPE    311#
GETPRI    311#
MULT      311#
NEWTST    311#    765    793    816    831    846    861    876    891    906    920    934    970    984    999
             1013   1028   1043   1058   1073   1089   1116   1159   1212   1242   1284   1304   1324   1344   1364
             1386   1408   1430   1472   1491   1509   1535   1560   1579   1597   1615   1632   1694   1766
POP       311#   1882   3017   3098   3099   3329   3330
PUSH      311#   1841   2996   3079   3085   3290   3292   3313
REPORT    311#
SCOPE     206#    768    796    819    834    849    864    879    894    909    923    937    973    987   1002
             1016   1031   1046   1061   1076   1092   1119   1162   1215   1245   1287   1307   1327   1347   1367
             1389   1411   1433   1475   1494   1512   1538   1563   1582   1600   1618   1635   1697   1769   1797
SETPRI    311#
SETTRA   3357#   3367   3368   3369   3370   3371   3372   3373   3374
SETUP     311#    632
SKIP      311#    775    782    790    811    827    842    857    872    887    902    917    931    952    956
              981    996   1010   1024   1039   1054   1069   1084   1102   1106   1112   1150   1154   1155   1193
             1197   1199   1209   1237   1239   1267   1270   1280   1300   1320   1340   1360   1382   1404   1426
             1448   1453   1488   1507   1525   1532   1551   1558   1576   1595   1692   1753   1755   2512
SLASH     311#
SPACE     311#
STARS     311#    347    359    361    368    382    425    429    765    767    793    795    816    819    831
              833    846    848    861    863    876    878    891    893    906    908    920    922    934    936
              970    972    984    986    999   1001   1013   1015   1028   1030   1043   1045   1058   1060   1073
             1075   1089   1091   1116   1118   1159   1161   1212   1214   1242   1244   1284   1286   1304   1306
             1324   1326   1344   1346   1364   1366   1386   1388   1408   1410   1430   1432   1472   1474   1491
             1493   1509   1511   1535   1537   1560   1562   1579   1581   1597   1599   1615   1617   1632   1634
             1694   1696   1766   1768   1789   1831   2752   2817   2870   2926   2946   2986   3026   3075   3091
             3127   3205   3285   3343
SWRSU     311#    653#
TRMTRP   3357#
TYPBIN    311#
TYPDEC    311#   1809
TYPNAM    311#
TYPNUM    311#
TYPOCS    311#
TYPOCT    311#   2886   3038   3062
TYPTXT    311#    691    696    703    707    957
$$CMRE    379#    417    418
$$CMTM    379#
$$ESCA    311#
$$NEWT    311#    765    793    816    831    846    861    876    891    906    920    934    970    984    999
             1013   1028   1043   1058   1073   1089   1116   1159   1212   1242   1284   1304   1324   1344   1364
             1386   1408   1430   1472   1491   1509   1535   1560   1579   1597   1615   1632   1694   1766
$$SET    3357#   3367   3368   3369   3370   3371   3372   3373   3374
$$SETM    669#    670
$$SKIP    311#    790    827    842    857    872    887    902    917    931    956    981    996   1010   1024
             1039   1054   1069   1112   1155   1199   1209   1237   1239   1270   1280   1300   1320   1340   1360
             1382   1404   1426   1453   1488   1507   1532   1558   1576   1595   1755
.EQUAT    190#    200
```

```
.HEADE    190#
.SETUP    190#     632
.SWRHI    190#     315
.SWRLO    328#
.$ACTI    190#     344
.$APTB    190#     426#
.$APTH    190#     356
.$APTY    190#    3282
.$CATC    190#     328
.$CMTA    190#     379
.$EOP     190#    1786
.$ERRO    190#    2814
.$ERRT    190#    3023
.$PARM    190#
.$POWE    190#    3072
.$RDOC    190#    2983
.$READ    190#    2867
.$SAVE    190#
.$SCOP    190#    2749
.$SPAC    190#
.$SWDO    190#
.$TRAP    190#    3340
.$TYPD    190#    1828
.$TYPE    190#    3202
.$TYPO    190#    3124
```

```
INC      617   716   803   946  1800  1859  1921  1950  1955  1973  1999  2005  2051  2501  2503
        2504  2506  2509  2510  2515  2526  2602  2617  2799  2836  2923  3096  3181  3189  3325
INCB    2454  2490  2804  2830  3278
IOT      206
JMP      340   341   342   687   690  1781  1783  1820
JSR      684   822   826   837   841   852   856   867   871   882   886   897   901   912   916
         926   930   951   977   992  1006  1020  1035  1050  1065  1080  1095  1127  1137  1149
        1170  1180  1192  1201  1205  1218  1225  1236  1248  1255  1266  1290  1310  1330  1350
        1371  1393  1415  1437  1455  1459  1463  1467  1477  1487  1496  1506  1514  1524  1531
        1540  1550  1557  1566  1575  1585  1594  1604  1608  1622  1626  1642  1676  1704  1737
        1757  1761  1815  2450  2464  2561  2565  2572  2578  2734  2842  2848  2916  3231  3250
        3257  3264  3314
MOV      607   608   611   612   613   625   626   628   629   630   633   637   639   640   641
         642   643   644   645   646   647   651   652   655   656   657   658   663   664   665
         667   673   676   677   678   680   700   712   713   717   718   769   771   772   773
         778   780   786   788   797   799   801   802   806   809   813   820   821   824   825
         835   836   839   840   850   851   854   855   865   866   869   870   880   881   884
         885   895   896   899   900   910   911   914   915   924   925   928   923   938   939
         941   942   943   949   950   961   965   974   975   976   979   988   990   991   994
        1003  1004  1005  1008  1017  1018  1019  1022  1032  1034  1037  1047  1048  1049  1052
        1062  1063  1064  1067  1077  1078  1079  1082  1093  1101  1105  1109  1110  1120  1121
        1122  1124  1125  1133  1135  1143  1144  1146  1147  1148  1163  1164  1165  1167  1168
        1176  1178  1186  1187  1189  1190  1191  1216  1217  1224  1231  1233  1234  1235  1246
        1247  1254  1261  1263  1264  1265  1272  1276  1288  1296  1297  1298  1308  1316  1317
        1318  1328  1336  1337  1338  1348  1356  1357  1358  1368  1369  1377  1379  1380  1390
        1392  1399  1401  1402  1412  1414  1421  1423  1424  1434  1436  1443  1445  1446  1476
        1483  1484  1485  1486  1495  1502  1503  1504  1505  1513  1520  1521  1522  1523  1530
        1539  1546  1547  1548  1549  1556  1564  1565  1571  1574  1583  1584  1590  1593  1601
        1602  1603  1606  1607  1619  1620  1621  1624  1625  1636  1637  1638  1640  1641  1[illegible]8
        1652  1653  1659  1675  1678  1682  1698  1699  1700  1702  1703  1710  1714  1715  1721
        1736  1739  1743  1770  1777  1778  1779  1782  1805  1809  1812  1842  1843  1844  1845
        1846  1847  1848  1853  1856  1876  1882  1883  1884  1885  1886  1888  1889  1910  1911
        1912  1913  1914  1915  1916  1917  1922  1923  1927  1929  1932  1933  1936  1940  1945
        1953  1959  1969  1972  1976  1978  1985  1986  1987  1988  1994  1995  1996  1997  1998
        2007  2025  2029  2034  2035  2036  2045  2046  2048  2050  2072  2073  2074  2075  2081
        2082  2086  2087  2435  2436  2437  2438  2440  2441  2442  2443  2444  2466  2468  2477
        2478  2479  2517  2522  2532  2539  2542  2546  2550  2554  2558  2564  2570  2609  2612
        2621  2624  2672  2675  2681  2686  2722  2723  2733  2735  2771  2772  2774  2777  2790
        2802  2803  2806  2807  2810  2811  2832  2837  2857  2860  2886  2903  2910  2912  2934
        2936  2953  2954  2969  2970  2971  2972  2994  2995  2996  2997  2998  3000  3015  3016
        3017  3018  3019  3033  3038  3047  3052  3057  3059  3063  3077  3078  3079  3080  3081
        3082  3083  3084  3085  3086  3087  3093  3094  3098  3099  3100  3101  3102  3103  3104
        3105  3106  3109  3150  3158  3159  3160  3166  3173  3191  3192  3193  3194  3195  3225
        3225  3230  3238  3253  3291  3292  3299  3303  3308  3309  3311  3313  3323  3329  3330
        3349  3350  3354
MOVB     650   940  1681  1688  1742  1749  1851  1854  1868  1871  1880  1918  1920  1934  2004
        2456  2487  2573  2574  2575  2577  2579  2580  2581  2582  2595  2676  2677  2678  2679
        2682  2683  2684  2685  2738  2739  2740  2741  2744  2745  2746  2747  2905  2909  2839
        2847  2879  2938  2958  2963  3003  3151  3152  3155  3156  3157  3161  3164  3165  3184
        3235  3263  3271  3286  3287  3289  3352
NEG     1850  1983  2078  3162
NOP     1816  1817  1818
RESET    609   631   675  1780  1814
ROL     3006  3008  3010  3168  3170  3171  3172  3174
RTI      666  1890  2812  2866  2915  2945  2973  3020  3111  3196  3240
RTS     1989  2008  2032  2042  2063  2084  2088  2533  2613  2625  2689  2742  2748  3061  3280
```

DZARCB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
            3331    3355
SUB         1145    1188    1232    1262    1378    1400    1422    1444    1528    1554    1572    1591    1611    1612    1629
            1630    1654    1662    1673    1716    1724    1735    1857    1964    1982    2000    2076    2559    2560    2730
            2732    2838    3306
SWAB        2047
TRAP        3357    3367    3368    3369    3370    3371    3372    3373    3374
TST          615     620     662     681     685     688     720    1649    1664    1690    1711    1726    1751    1771    1862
            1872    1925    1970    1974    2458    2513    2673    2687    2773    2797    2852    2858    2901    2908    2918
            3014    3065    3179    3237    3245    3267    3301    3319    3321    3351
TSTB         804     947    1864    1878    1930    1941    1951    1979    2052    2507    2527    2603    2618    2622    2784
            2877    2936    3220    3269    3293    3304    3317
WAIT        1946    1956
.ASCII       422     423    2148    2211    2212    2213    2219    2224    2232    2248    2261    2288    2293    2657
.ASCIZ       421     424     695     699     706     710     960    1823    2091    2095    2102    2107    2112    2118    2123
            2129    2135    2141    2144    2155    2163    2170    2175    2181    2187    2194    2202    2209    2217    2238
            2240    2242    2255    2258    2272    2276    2280    2284    2305    2309    2313    2319    2322    2326    2330
            2336    2342    2350    2356    2636    2640    2644    2648    2653    2662    2977    2978    2979    2981    3069
            3115
.BLKB       2976
.BLKW       1895    3377    3379
.BYTE        389     390     395     396     411     412     413     414     441     442     452     453     460     461     463
             464     466     467     469     470     471     472     967    1098    1129    1139    1172    1182    1220    1227
            1250    1257    1274    1278    1292    1312    1332    1352    1373    1395    1417    1439    1479    1498    1516
            1542    1568    1587    1644    1706    1822    2020    2021    2022    2023    2027    2031    2038    2152    2153
            2421    2519    2524    2544    2548    2552    2556    2611    2627    2628    2629    2630    2631    2632    2633
            2634    2651    2652    2654    2655    2656    2661    2849    2850    2974    2975    3197    3198    3199    3200
            3332    3333    3334
.ENABL       190
.END        3381
.ENDC        195     205     297     311     324     326     327     328     341     348     352     354     360     362     369
             383     387     389     415     419     420     421     422     426     430     452     460     463     466     469
             470     471     472     475     476     477     478     479     480     481     482     483     484     485     486
             487     488     489     490     491     492     493     494     498     632     637     638     641     643     645
             647     648     649     651     653     675     695     699     706     710     766     767     768     769     770
             776     783     791     794     795     796     797     798     812     817     818     819     820     821     828
             832     833     834     835     836     843     847     848     849     850     851     858     862     863     864
             865     866     873     877     878     879     880     881     888     892     893     894     895     896     903
             907     908     909     910     911     918     921     922     923     924     925     932     935     936     937
             938     939     953     957     960     971     972     973     974     975     982     985     986     987     988
             989     997    1000    1001    1002    1003    1004    1011    1014    1015    1016    1017    1018    1025    1029
            1030    1031    1032    1033    1040    1044    1045    1046    1047    1048    1055    1059    1060    1061    1062
            1063    1070    1074    1075    1076    1077    1078    1085    1090    1091    1092    1093    1094    1103    1107
            1113    1117    1118    1119    1120    1121    1151    1155    1156    1160    1161    1162    1163    1164    1194
            1198    1200    1210    1213    1214    1215    1216    1217    1238    1240    1243    1244    1245    1246    1247
            1268    1271    1281    1285    1286    1287    1288    1289    1301    1305    1306    1307    1308    1309    1321
            1325    1326    1327    1328    1329    1341    1345    1346    1347    1348    1349    1361    1365    1366    1367
            1368    1369    1383    1387    1388    1389    1390    1391    1405    1409    1410    1411    1412    1413    1427
            1431    1432    1433    1434    1435    1449    1454    1473    1474    1475    1476    1477    1488    1492    1493
            1494    1495    1496    1508    1510    1511    1512    1513    1514    1526    1533    1536    1537    1538    1539
            1540    1552    1559    1561    1562    1563    1564    1565    1577    1580    1581    1582    1583    1584    1596
            1598    1599    1600    1601    1602    1616    1617    1618    1619    1620    1633    1634    1635    1636    1637
            1693    1695    1696    1697    1698    1699    1754    1756    1767    1768    1769    1770    1771    1790    1792
            1793    1795    1798    1804    1807    1808    1812    1814    1820    1822    1823    1826    1832    2513    2753
            2756    2761    2766    2768    2779    2782    2783    2784    2786    2788    2795    2799    2804    2806    2810
            2813    2814    2818    2821    2830    2837    2842    2843    2844    2852    2862    2866    2867    2871    2927
            2946    2947    2954    2956    2959    2961    2977    2987    2989    3022    3027    3042    3071    3076    3085
```



DZARCB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

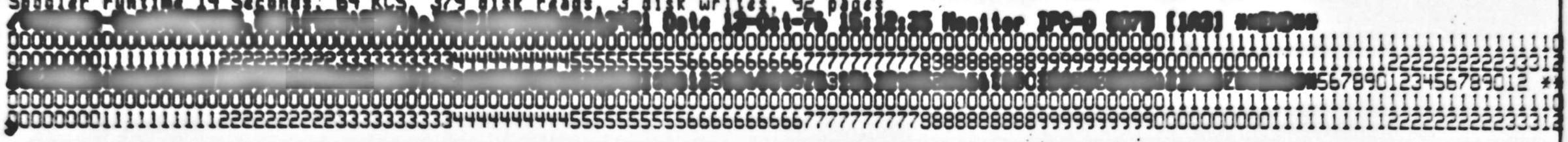
```
            2760    2761    2765    2793    2794    2810    2813    2814    2821    2822    2823    2824    2825    2830    2855
            2858    2867    2871    2887    2969    2977    2983    3039    3064    3282    3366    3367    3368    3369    3370
            3371    3372    3373    3374
.IRP         632     765     793     816     831     846     861     876     891     906     920     934     970     984     999
            1013    1028    1043    1058    1073    1089    1116    1159    1212    1242    1284    1304    1324    1344    1364
            1386    1408    1430    1472    1491    1509    1535    1560    1579    1597    1615    1632    1694    1766    1842
            1882    2765    2996    3017    3079    3085    3098    3099    3291    3292    3313    3329    3330
.LIST        190     311     327     335     415     417     418     419     426     430     632     653     695     699     706
             710     765     769     793     797     816     820     831     835     846     850     861     865     876     880
             891     895     906     910     920     924     934     938     960     970     974     984     988     999    1003
            1013    1017    1028    1032    1043    1047    1058    1062    1073    1077    1089    1093    1116    1120    1159
            1163    1212    1216    1242    1246    1284    1288    1304    1308    1324    1328    1344    1348    1364    1368
            1386    1390    1408    1412    1430    1434    1472    1476    1491    1495    1509    1513    1535    1539    1560
            1564    1579    1583    1597    1601    1615    1619    1632    1636    1694    1698    1766    1770    1798    1814
            2760    2862    2946    3357    3366    3367    3368    3369    3370    3371    3372    3373    3374    3375
.MACRO       338     379     669    3357
.MCALL       190     311     426     653
.NLIST       190     311     327     335     415     417     418     419     426     430     632     653     695     699     706
             710     765     769     793     797     816     820     831     835     846     850     861     865     876     880
             891     895     906     910     920     924     934     938     960     970     974     984     988     999    1003
            1013    1017    1028    1032    1043    1047    1058    1062    1073    1077    1089    1093    1116    1120    1159
            1163    1212    1216    1242    1246    1284    1288    1304    1308    1324    1328    1344    1348    1364    1368
            1386    1390    1408    1412    1430    1434    1472    1476    1491    1495    1509    1513    1535    1539    1560
            1564    1579    1583    1597    1601    1615    1619    1632    1636    1694    1698    1766    1770    1798    1814
            2760    2862    2946    3357    3366    3367    3368    3369    3370    3371    3372    3373    3374    3375
.PAGE        379     425
.REM           1
.REPT        335     417
.SBTTL       201     316     329     339     345     357     380     427     499     601     723     762     763     764     765
             793     816     831     846     861     876     891     906     920     934     970     984     999    1013    1028
            1043    1058    1073    1089    1116    1159    1212    1242    1284    1304    1324    1344    1364    1386    1408
            1430    1472    1491    1509    1535    1560    1579    1597    1615    1632    1694    1766    1784    1797    1829
            1896    1991    2089    2401    2671    2718    2750    2815    2868    2984    3024    3073    3125    3203    3283
            3341    3358
.TITLE       190
.WORD        335     336     337     353     373     374     375     376     377     379     388     391     392     393     394
             397     398     399     400     401     402     403     404     405     406     415     417     418     432     433
             434     435     436     437     438     439     443     444     445     458     462     465     468     474     475
             476     477     478     479     480     481     482     483     484     485     486     487     488     489     490
             491     492     493     702    1803    1806    1821    3021    3050    3055    3108    3110    3201    3232    3279
            3315
```

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

```
*,DZARCB.SEQ/SOL/CRF=DZARCB.P11
RUN-TIME:  65 38 8 SECONDS
RUN-TIME RATIO: 237/113=2.0
CORE USED:  25K  (49 PAGES)
```

Spooler runtime 14 Seconds, 64 KCS, 379 disk reads, 3 disk writes, 92 pages

[redacted]



```
 198                        ...F1    2555   016032  104402    ...F5
 254             000020     ...G1    2611   016330     004    ...G5
 310             000240     ...H1    2667   016640  000000    ...H5
 323                        ...I1    2723   017050  010037    ...I5
 352             000052     ...J1    2758                     ...J5
 388  001100     000000     ...K1    2814   017450  003720    ...K5
 434  001204     000000     ...L1    2870                     ...L5
 490  001314     000000     ...M1    2926                     ...M5
 546  001374     012161     ...N1    2982   020334  036440    ...N5

 560  001420     014650     ...B2    3038   020460  013746    ...B6
 612  001554     012737     ...C2    3080   020614  010146    ...C6
 668  002104     012637     ...D2    3132                     ...D6
 724                        ...E2    3188   021210  002402    ...E6
 774  002600     023737     ...F2    3244   021356  001006    ...F6
 825  003062     012737     ...G2    3300   021610  062766    ...G6
 870  003252     012737     ...H2    3356   022014  000200    ...H6
 915  003442     013737     ...I2                             ...I6
 971                        ...J2  ARBVCT  001466              ...J6
1022  004172     013737     ...K2  CH16  = 000016              ...K6
1067  004366     013737     ...L2  DIFERR  015742              ...L6
1098  004504        001     ...M2  NBEXT   001472              ...M6
1125  004610     010137     ...N2  RSX2    016406              ...N6

1168  004776     010137     ...B3                             ...B7
1221  005232     001771     ...C3  TST13   003530              ...C7
1251  005360     000010     ...D3  VARLT1  017004              ...D7
1293  005544     001000     ...E3  $DDWO   001264              ...E7
1333  005674     000003     ...F3  $ITEMB  001114              ...F7
1373  006030        000     ...G3                             ...G7
1417  006214        000     ...H3  $40CAT= ****** U            ...H7
1473                       ...I3  NEWTST    311#    765       ...I7
1518  006642     000201     ...J3  .$CMTA    190#    379#RE    ...J7
1569  007076     001001     ...K3  BCC      1867    2455       ...K7
1606  007262     013737     ...L3           1540    1550       ...L7
1641  007444     012077     ...M3  TSTB      804     947       ...M7
1697  007712     000004     ...N3            645     647       ...N7

1753  010202     001341     ...B4             891     895      ...B8
1794                       ...C4  **END** USER DAVIES,TOM      ...C8
1836                       ...D4
1892  010700     001750     ...E4
1906                       ...F4
1962  011220     020103     ...G4
2001  011402     100373     ...H4
2052  011576     105777     ...I4
2098  012012     050124     ...J4
2155  012451        105     ...K4
2211  013146     024040     ...L4
2267  013606     020051     ...M4
2323  014250     050111     ...N4
```