LPA/AR11

DIAGNOSTIC TEST 3
MD-11-DRLPE-A

EP-DRLPE-A-DL

FICHE 1 OF 1

MAR 1978
digital
MADE IN USA

EOF1DRLPDRSEQ411 00010000 780223 IDENTIFICATION10 411 HDR1DRLPEASEQ 00010000 780223
SEQ 0001

PRODUCT CODE: MAINDEC-11-DRLPE-A-D

PRODUCT NAME: LPA/AR11 DIAGNOSTIC TEST III

DATE: JAN 1978

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, CHO2 L, AND WRITE-THRU L) ARE USED AS INPUTS, ALLOWING PARAMETRIC TESTS OF THEIR HIGH AND LOW OUTPUT LEVELS; THE CHO2 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.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZARC-B". IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE AR11 OPTION WHEN IT IS ON THE LPA11-KX I/O BUS. NO RECABLING IS NEEDED. SOME TEST DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION TEST, WERE DELETED AS THEY COULD NOT BE CHECKED. IF THIS DIAGNOSTIC DOEN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO RUN "MD-11-DZARC-B". YOU SHOULD RUN "MD-11-DRLPA" BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 11.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 FAMILY COMPUTER WITH 16K 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 'O' POSITION.

```
**************************************************************
THE WRAPAROUND (G5036) MODULE MUST BE CONNECTED AS FOLLOWS:
        BR11 TO BC08R CONNECTION A-A, VV-VV
        BC08R TO G5036 CONNECTION "UPSIDE-DOWN" A-VV, VV-A
**************************************************************
```

8. MISCELLANEOUS

SEQ 0004

8.1 EXECUTION TIME

THE FIRST PASS WILL TAKE APPROXIMATELY 10 MINUTES FOR COMPLETION.
ADDITIONAL PASSES WILL TAKE APPROXIMATELY 40 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 XXDP/ACT/APT NOTES

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

8.4 USER LINK TO I/O DEVICE

A SPECIAL USER LINK HAS BEEN PROVIDED IN ORDER FOR THE OPERATOR TO EXAMINE OR MODIFY LOCATIONS ON THE LPA11-KX I/O BUS. (NOTE: THIS CANNOT BE DONE DIRECTLY.)

PROCEDURE:

1) START THE PROCESSOR AT LOCATION $UTK:

2) THE DIALOG TO EXAMINE A LOCATION IS AS FOLLOWS:

E OR D	"E"
DEVICE ADDRS= XXXXXX	"OCTAL ADDRS"

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

3) THE DIALOG TO MODIFY A LOCATION IS AS FOLLOWS:

E OR D	"D"
DATA=	"DATA TO BE DEPOSITED"

4) THE PROGRAM WILL STAY IN THIS LOOP UNTIL THE OPERATOR IS FINISHED. AT THIS TIME THE PROCESSOR SHOULD BE HALTED.

NOTE: THE OPERATORS RESPONSE IS ENCLOSED IN QUOTES.

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".

SEQ 0005

10. TABLE OF CONTENTS

ATTACHED

11. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY

DIAGNOSTICS FOR THE LPA11 ARE WRITTEN AT THREE LEVELS: (1) TOTAL PDP-11 SYSTEM, (2) LPA11 SYSTEM; AND, (3) LPA11 OPTIONS.

LEVEL 1, IS DESIGNED TO ISOLATE A FAILURE TO THE LPA11 SYSTEM. ALL OPTIONS ON THE PDP-11 ARE EXERCISED.

LEVEL 2 DIAGNOSTICS ISOLATE A FAILURE TO THE INDIVIDUAL OPTION WITHIN THE LPA11. THE LEVEL 2 DIAGNOSTIC IS MD-11-DRLPA. WHEN THE USER RUNS DRLPA HE CAN GENERALLY TELL WHICH OPTION DIAGNOSTIC (LEVEL 3) TO RUN NEXT. M8254 AND M8200-YC ERRORS MAY "LOOK" ALIKE AND DRLPA MAY NOT BE ABLE TO DISTINGUISH BETWEEN THEM. ARBITRATION ERRORS WILL NOT BE DETECTED BY THIS DIAGNOSTIC.

LEVEL THREE DIAGNOSTICS AID IN DETERMINING IF THE ERROR WAS IN FACT ON THE OPTION THE DRLPA SPECIFIED. THE USER MAY "LOOP" ON THE ERROR. WITHIN LEVEL THREE, THERE ARE TWO GROUPS OF DIAGNOSTICS. THE FIRST GROUP REQUIRES NO "EXTRA" WORK BY THE USER IN ORDER TO RUN. GROUP "A" DIAGNOSTICS DO NOT CHECK ARBITRATION, AND REQUIRE EXTRA TIME FOR EXECUTION. THE SECOND GROUP (GROUP "B") REQUIRES THAT THE USER RECONFIGURE THE PDP-11 SYSTEM. THIS RECONFIGURATION INVOLVES CABLING THE UNIBUS TO THE LPA'S I/O BUS.

THE DIAGNOSTIC FOR THE M8254 FALLS INTO THE GROUP "B" CATAGORY.

THE LPA11-KX DIAGNOSTIC KIT WILL INCLUDE:

SEQ 0007

OPTION	GROUP	DIAG. #	DIAG. TITLE
LPA11-KX	LEVEL 2	MD-11-DRLPA	LPA11-K SYSTEM DIAG.
M8254	"B"	MD-11-DRLPN	M8254 (IPBM) DIAG.
AA11-K	A	MD-11-DRLPB	AA11-K DIAG.
	B	MD-11-DZAAC	AA11-K DIAG.
AR11	A	MD-11-DRLPC	LPA/AR11 DIAG. #1
	A	MD-11-DRLPD	LPA/AR11 DIAG. #2
	A	MD-11-DRLPE	LPA/AR11 DIAG. #3
	B	MD-11-DZARA	AR11 DIAG. #1
	B	MD-11-DZARB	AR11 DIAG. #2
	B	MD-11-DZARC	AR11 DIAG. #3
DR11-K	A	MD-11-DRLPF	LPA/DR11-K DIAG.
	B	MD-11-DZDRG	DR11-K DIAG.
KW11-K	A	MD-11-DRLPG	LPA/KW11-K DIAG.
	B	MD-11-DZKWK	KW11-K DIAG.
LPS11	A	MD-11-DRLPH	LPA/LPS11 DIAG. #1
	A	MD-11-DRLPI	LPA/LPS11 DIAG. #2
	A	MD-11-DRLPJ	LPA/LPS11 DIAG. #3
	B	MD-11-DZLPC	LPS11 DIAG. #1
	B	MD-11-DZLPD	LPS11 DIAG. #2
	B	MD-11-DZLPI	LPS11 DIAG. #3
AD11-K	A	MD-11-DRLPK	LPA/AD11-K DIAG.
	B	MD-11-DZADL	AD11-K DIAG.
M8200-YC	B	MD-11-DZLPL	LPA/M8200-YC BASIC MICRO-CPU R/W TEST
	B	MD-11-DZLPM	LPA/M8200-YC JMP+ROM READ TEST

```
                        .REM    [

                                        LPA.MAC

WELCOME, THIS DIAGNOSTIC IS ONE IN A SERIES OF DIAGNOSTIC
DESIGNED IN ORDER TO AID YOU IN TESTING THE LPA-11XX OPTION.
I HOPE THAT YOU HAVE READ THE DOCuMENTATION SECTION OF THIS
DIAGNOSTIC.  IF YOU HAVE,YOU KNOW ABOUT ALL OF THE DIAGNOSTICS
THAT ARE AVAILIBLE FOR TESTING THE LPA SYSTEM.

                                        GOOD LUCK !

[
.GLOBL  DRLPX2
```

```
                        .TITLE  MAINDEC-11-DRLPE-A
                        ;*COPYRIGHT (C) 1978
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY EDWARD C. BADGER
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
                        ;*
                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK=  1100
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

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

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

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

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
```

```
                        020000          SW13=   20000
                        010000          SW12=   10000
                        004000          SW11=   4000
                        002000          SW10=   2000
                        001000          SW09=   1000
                        000400          SW08=   400
                        000200          SW07=   200
                        000100          SW06=   100
                        000040          SW05=   40
                        000020          SW04=   20
                        000010          SW03=   10
                        000004          SW02=   4
                        000002          SW01=   2
                        000001          SW00=   1
                                        .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
```

```
                                ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                                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  001612                  JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001634                  JMP     BEGIN1          ;RESTART ADDRESS
000210  000137  001626                  JMP     BEGIN2          ;STARTING ADDRESS OF OPTION TEST AREA

                                .SBTTL  ACT11 HOOKS

                                ;;**************************************************************
                                ;HOOKS REQUIRED BY ACT11
000214                                  $SVPC=.                 ;SAVE PC
000046                                  .=46
000046  011436                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
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  001202          $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  000                 $AUTOB: .BYTE   0        ;;AUTOMATIC MODE INDICATOR
001135  000                 $INTAG: .BYTE   0        ;;INTERRUPT MODE INDICATOR
001136  000000                      .WORD   0
001140  177570              SWR:    .WORD   DSWR     ;;ADDRESS OF SWITCH REGISTER
001142  177570              DISPLAY: .WORD  DDIsP    ;;ADDRESS OF DISPLAY REGISTER
001144  177560              $TKS:   177560           ;;TTY KBD STATUS
001146  177562              $TKB:   177562           ;;TTY KBD BUFFER
001150  177564              $TPS:   177564           ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566              $TPB:   177566           ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000                 $NULL:  .BYTE   0        ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002                 $FILLS: .BYTE   2        ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012                 $FILLC: .BYTE   12       ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000                 $TPFLG: .BYTE   0        ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000              $REGAD: .WORD   0        ;;CONTAINS THE ADDRESS FROM
                                                     ;;WHICH  ($REG0) WAS OBTAINED
001162  000000              $REG0:  .WORD   0        ;;CONTAINS (($REGAD)+0)
001164  000000              $REG1:  .WORD   0        ;;CONTAINS (($REGAD)+2)
001166  000000              $TIMES: 0                ;;MAX. NUMBER OF ITERATIONS
001170  000000              $ESCAPE:0                ;;ESCAPE ON ERROR ADDRESS
001172  177607  000377        $BELL:  .ASCIZ  <207><377><377>  ;;CODE FOR BELL
001176  077                 $QUES:  .ASCII  /?/      ;;QUESTION MARK
001177  015                 $CRLF:  .ASCII  <15>     ;;CARRIAGE RETURN
001200  000012              $LF:    .ASCIZ  <12>     ;;LINE FEED
                                ;;**********************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;**********************************************************************
                                .EVEN
001202                          $MAIL:                  ;;APT MAILBOX
001202  000000                 $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
```

```
                                ;
                                ;APT MAILBOX-ETABLE
                                ;
001204  000000                $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
001206  000000                $TESTN: .WORD   ATESTN  ;;TEST NUMBER
001210  000000                $PASS:  .WORD   APASS   ;;PASS COUNT
001212  000000                $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
001214  000000                $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
001216  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
001220  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001222                        $ETABLE:                ;;APT ENVIRONMENT TABLE
001222      000               $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001223      000               $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001224  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001226  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001230  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
001232      000               $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001233      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
001234  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001236      000               $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001237      000               $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001240  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001242      000               $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001243      000               $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001244  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001246      000               $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001247      000               $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001250  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001252  000340                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001254  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001256  170400                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001260  000000                $DEVm:  .WORD   ADEVM   ;;DEVICE MAP
001262  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001264  000000                $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001266  000000                $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001270  000000                $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001272  000000                $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001274  000000                $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001276  000000                $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001300  000000                $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001302  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001304  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001306  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001310  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001312  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001314  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
```

```
   321  001316  000000                  $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
   322  001320  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
   323  001322  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
   324  001324  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
   325
   326
   327  001326                          $ETEND:
   328
```

```
                                .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

001326                          $ERRTB:

                                ;ITEM   1
001326  013062                          EM1             ;ERROR ON A/D CHANNEL
001330  013575                          DH1             ;ERRPC  ARADD   CHANNEL NOMINAL TOLERANCE ACTUAL
001332  015716                          DT1             ;$ERRPC ARBADD  CHANL   $GDDAT  SPREAD  $BDDAT
001334  000000                          0

                                ;ITEM   2
001336  013107                          EM2             ;VC LOGIC SIGNAL HIGH OUTPUT TOO LOW
001340  013654                          DH2             ;ERRPC  ARADD   CHANNEL 3V LEV. OUTPUT
001342  015734                          DT2             ;$ERRPC ARBADD  CHANL   $GDDAT  $BDDAT
001344  000000                          0

                                ;ITEM   3
001346  013153                          EM3             ;VC STATUS REGISTER IN ERROR
001350  013723                          DH3             ;ERRPC  ARADD   GOOD    BAD
001352  015750                          DT3             ;$ERRPC ARBADD  $GDDAT  $BDDAT
001354  000000                          0

                                ;ITEM   4
001356  013207                          EM4             ;EXTERNAL AD START FAILED
001360  013723                          DH3             ;ERRPC  ARADD   GOOD    BAD
001362  015750                          DT3             ;$ERRPC ARBADD  $GDDAT  $BDDAT
001364  000000                          0

                                ;ITEM   5
001366  013244                          EM5             ;A/D DIFFERENTIAL LINEARITY ERROR
001370  013760                          DH5             ;ERRPC  ARADD   # OF STATES     ALLOWED
001372  015762                          DT5             ;$ERRPC ARBADD    $BDDAT        $GDDAT
001374  000000                          0

                                ;ITEM   6
001376  013305                          EM6             ;NOISE LEVEL EXCEEDED LIMIT
001400  014020                          DH6             ;ERRPC  ARADD   LIMIT   MEASURED
001402  015750                          DT3             ;$ERRPC ARBADD  $GDDAT  $BDDAT
001404  000000                          0

                                ;ITEM   7
001406  013340                          EM7             ;VC LOGIC SIGNAL LOW OUTPUT TOO HIGH
```

```
001410  014061                  DH7             ;ERRPC  ARADD   A/D CHAN +.4V LEV. OUTPUT
001412  015734                  DT2             ;$ERRPC ARBADD  CHANL   $GDDAT   $BDDAT
001414  000000                  0
                        ;ITEM   10
001416  013404                  EM10            ;D/A DIFFERENTIAL LINEARITY ERROR
001420  014130                  DH10            ;ERRPC  ARADD   STEP    NOMINAL SPREAD  ACTUAL
001422  015774                  DT10            ;$ERrPC ARBADD  EDGE    $GDDAT  SPREAD  $BDDAT
001424  000000                  0
                        ;ITEM   11
001426  013445                  EM11            ;A/D INTER-CHANNEL SETTLING ERROR
001430  014207                  DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001432  016012                  DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001434  000000                  0
                        ;ITEM   12
001436  013506                  EM12            ;OFFSET ERROR
001440  014207                  DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001442  016012                  DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001444  000000                  0
                        ;ITEM   13
001446  013523                  EM13            ;CALIBRATION ERROR
001450  014207                  DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001452  016012                  DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001454  000000                  0
                        ;ITEM   14
001456  013545                  EM14            ;LINEARITY ERROR AT XX00
001460  014207                  DH11            ;ERRPC  ARADD   NOMINAL SPREAD  ACTUAL
001462  016012                  DT11            ;$ERRPC ARBADD  $GDDAT  SPREAD  $BDDAT
001464  000000                  0

                        ;
                        ;ADDRESS OF KMC-11 OF LPA-11    THE ADDR FOR KMADO MAY BE
                        ;                               CHANGED BY THE USER TO REFLECT
                        ;                               A DIFFERENT KMC-11 ADDR. THE
                        ;                               REST OF THE ADDRESSES WILL
                        ;                               BE CHANGED BY THE PROGRAM.
                        ;

001466          LPCI:
001466  170460  KMADO:  .WORD   170460          ;BASE KMC ADDR. MAY BE PATCHED BY USER.

001470          LPMR:
001470  170461  KMAD1:  .WORD   170460+1                ;>DO NOT        <;KMC-CSR ADDR
001472          LPCO:
001472  170462  KMAD2:  .WORD   170460+2                ;>PATCH         <;
001474          LPSO:
001474  170463  KMAD3:  .WORD   170460+3                ;>THIS AREA     <
001476          LPADL:
001476  170464  KMAD4:  .WORD   170460+4                ;
001500          LPADH:
```

```
   437  001500  170465                    KMAD5:  .WORD   170460+5              ;>DO NOT        <
   438  001502                            LPMS1:
   439  001502  170466                    KMAD6:  .WORD   170460+6              ;>PATCH         <
   440  001504                            LPMS2:
   441  001504  170467                    KMAD7:  .WORD   170460+7              ;>THIS AREA     <
   442
   443  001506  000340                    VECTOR: .WORD   AVECT1&777      ;BASE VECTOR OF KMC
   444  001510  000344                    VECTPS: .WORD   4+AVECT1&777    ;VECOTR ADDR.+2
   445
   446  001512  000004                    VERSN:  .WORD   4               ;CURRENT VERSION NUMBER OF MICROCODE.
   447
   448  001514  000000                    .DVLS:  .WORD   0               ;/DEVICE LIST OF I/O ADDR. DEFINED
   449  001516  000020                            .BLKW   16.             ;/BY INIT.
   450
   451  001556  170400                    ARBADD: 170400
   452  001560  000340                    ARBVCT: 340
   453  001562  000000                    NMBEXT: 0
   454  001564  000000                    $TMDAT: 0
   455  001566  000000                    NBEXT:  0
   456
   457  001570  170400                    ADCS:   170400                  ;A TO D STATUS/CONTROL REGISTER
   458  001572  170401                    ADCS1:  170401                  ;A TO D STATUS REGISTER <HIGH BYTE>
   459  001574  170402                    ADDBR:  170402                  ;A TO D CONVERTED VALUE <READ>
   460  001576  170404                    CSR:    170404                  ;CLOCK STATUS REGISTER
   461  001600  170406                    CSB:    170406                  ;CLOCK PRESET BUFFER
   462  001602  170410                    VCSTAT: 170410                  ;DAC STATUS REGISTER
   463  001604  170412                    VCXREG: 170412                  ;X BUFFER
   464  001606  170414                    VCYREG: 170414                  ;Y BUFFER
   465  001610  170416                    CSC:    170416                  ;CLOCK COUNTER
   466                                            .SBTTL  PROGRAM START-UP
   467
   468  001612  000240                    BEGIN:  NOP
   469  001614  005037  001126                    CLR     $BDDAT                  ;CLEAR R0
   470  001620  005037  020264                    CLR     WFTEST                  ;CLEAR TOLERANCE FLAG
   471  001624  000406                            BR      RBEG
   472  001626  012737  000001  020264    BEGIN2: MOV     #1,WFTEST               ;SET TOLERANCE FLAG
   473  001634  012737  177777  001126    BEGIN1: MOV     #-1,$BDDAT                      ;LOAD R0
   474  001642                            RBEG:
   475                                            ;
   476                                            ;THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
   477                                            ;
   478
   479  001642  010046                            MOV     R0,-(SP)
   480  001644  010146                            MOV     R1,-(SP)
   481  001646  013700  001466                    MOV     KMAD0,R0        ;GET KMC-11 ADDRESS.
   482  001652  012701  001470                    MOV     #KMAD1,R1       ;GET ADDR. OF ADDR. LIST.
   483
   484  001656  005200                    64$:    INC     R0              ;UPDATE ADDR.
   485  001660  010021                            MOV     R0,(1)+         ;WRITE ADDR.
   486  001662  020127  001506                    CMP     R1,#KMAD7+2     ;DONE ALL ADDRESSES?
   487  001666  001373                            BNE     64$             ;NO - DO NEXT ADDR.
   488  001670  005037  001514                    CLR     .DVLS           ;CLR ADDR. LIST.
   489  001674  012601                            MOV     (SP)+,R1
   490  001676  012600                            MOV     (SP)+,R0
```

```
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001700  012706  001100                          MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001704  005026                                  CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001706  022706  001140                          CMP     #SWR,R6 ;;DONE?
001712  001374                                  BNE     .-6             ;;LOOP BACK IF NO
001714  012706  001100                          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
001720  012737  020474  000020                  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001726  012737  000340  000022                  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001734  012737  020756  000030                  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001742  012737  000340  000032                  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001750  012737  023366  000034                  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001756  012737  000340  000036                  MOV     #340,@#TRAPVEC+2;LEVEL 7
001764  012737  022170  000024                  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001772  012737  000340  000026                  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
002000  013737  011404  011376                  MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
002006  005037  001166                          CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
002012  005037  001170                          CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
002016  112737  000001  001115                  MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
002024  012737  002024  001106                  MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
002032  012737  002032  001110                  MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
002040  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
002044  012737  002100  000004                  MOV     #65$,@#ERRVEC   ;;SET UP ERROR VECTOR
002052  012737  177570  001140                  MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
002060  012737  177570  001142                  MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
002066  022777  177777  177044                  CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
002074  001012                                  BNE     67$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
002076  000403                                  BR      66$             ;;BRANCH IF NO TIMEOUT
002100  012716  002106          65$:            MOV     #66$,(SP)       ;;SET UP FOR TRAP RETURN
002104  000002                                  RTI
002106  012737  000176  001140  66$:            MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
002114  012737  000174  001142                  MOV     #DISPREG,DISPLAY
002122  012637  000004          67$:            MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

002126  005037  001210                          CLR     $PASS           ;;CLEAR PASS COUNT
002132  132737  000200  001223                  BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
002140  001403                                  BEQ     68$             ;;YES,USE NON-APT SWITCH
002142  012737  001224  001140                  MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
002150                          68$:
                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002150  005227  177777                          INC     #-1             ;;FIRST TIME?
002154  001041                                  BNE     69$             ;;BRANCH IF NO
002156  022737  011436  000042                  CMP     #$ENDAD,@#42    ;;ACT-11?
002164  001435                                  BEQ     69$             ;;BRANCH IF YES
002166  104401  002234                          TYPE    ,70$            ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002172  005737  000042                          TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
002176  001012                                  BNE     71$             ;;BRANCH IF YES
002200  123727  001222  000001                  CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
```

```
                                                BEQ     71$                     ;;BRANCH IF YES
                                                CMP     SWR,#SWREG              ;;SOFTWARE SWITCH REG SELECTED?
                                                BNE     72$                     ;;BRANCH IF NO
                                                GTSWR                           ;;GET SOFT-SWR SETTINGS
                                                BR      72$
                                        71$:    MOVB    #1,$AUTOB               ;;SET AUTO-MODE INDICATOR
                                        72$:
                                                BR      69$                     ;;GET OVER THE ASCIZ
                                        ;;70$:  .ASCIZ  <CRLF><15><12>#MD-11-DRLPE-A#<15><12><CRLF>
                                        69$:
                                                CLR     .DVLS
                                                CLR     PS
                                                MOV     #STACK,SP               ;LOAD STACK
                                                MOV     $BASE,R2                ;LOAD STARTING ADDRESS
                                                CLR     R3                      ;CLEAR COUNT
                                        2$:     MOV     R2,$TMDAT

                                        ;*      MOV     @$TMDAT,$GDDAT  ;/READ DEVICE REG $TMDAT,PUT DATA IN $GDDAT.
                                                TST     $AERR
                                                BNE     1$
                                                ADD     #20,R2                  ;EXIST, UPDATE TEST ADDRESS
                                                INC     R3                      ;UPDATE # OF AR11'S
                                                BR      2$
                                        1$:     TST     R3                      ;TEST IF FIRST DOEs EXIST
                                                BNE     3$                      ;BR
                                                HALT                            ;FIRST AR11 DOES NOT EXIST
                                                JMP     BEGIN
                                        3$:     DEC     R3                      ;ADJUST R3
                                                MOV     R3,NMBEXT               ;SAVE THE NUMBER OF ADDITIONAL AR11'S
                                                MOV     #6,@#4                  ;RESET BUS ERROR
                                                CLR     @#6
                                                MOV     $BASE,ARBADD            ;LOAD FIRST ADDRESS
                                                MOV     $VECT1,ARBVCT           ;LOAD FIRST VECTOR
                                                MOV     NMBEXT,NBEXT            ;LOAD NUMBER OF ADDITIONAL AR11'S
                                                MOV     $BDDAT,R0
                                        RBEG2:  JSR     PC,$RESET
                                                MOV     #232,R2                 ;LOAD R2
                                                MOV     #230,R1                 ;LOAD R1
                                        5$:     MOV     R2,(R1)+                ;LOAD .+2
                                                CLR     (R1)+                   ;LOAD HALT
                                                MOV     R1,R2                   ;LOAD R2
                                                TST     (R2)+                   ;BUMP R2
                                                CMP     R2,#1002                ;TEST FOR LAST
                                                BNE     5$                      ;BR UNTIL DONE
                                                JSR     PC,WFADJ                ;ADJUST SOME TOLERANCES
                                                TST     R0                      ;TEST R0
                                                BEQ     2$                      ;BR IF CLEARED
                                                JMP     4$                      ;INHIBIT TYPOUT
                                        2$:     TST     @#42                    ;TEST ACT-11 OR DDP
                                                BEQ     3$                      ;BR IF CLEARED
                                                JMP     4$                      ;INHIBIT TYPOUT
                                        3$:     MOV     NMBEXT,-(SP)            ;PUSH ON STACK
                                                TYPOS                           ;TYPE OCTAL
                                                .WORD   2
```

```
                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  /(8) ADDITIONAL AR11'S CONNECTED/
                                64$:
                                        TYPE    ,67$            ;;TYPE ASCIZ STRING
                                        BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  <15><12>/ALL AR-11'S MUST HAVE G5036 WRAPAROUND MODULE INSTALLED/
                                66$:

                                4$:     MOV     #ADCS,R0                ;LOAD POINTER
                                10$:    MOV     ARBADD,(R0)+
                                        CMP     #BEGIN,R0               ;TEST FOR END
                                        BNE     10$
                                        INC     ADCS1
                                        MOV     #ADDBR,R0
                                        MOV     #2,R1
                                12$:    ADD     R1,(R0)+
                                        ADD     #2,R1
                                        CMP     #20,R1
                                        BNE     12$
                                        .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
```

```
                    000053                          CH53=   53
                    000054                          CH54=   54
                    000055                          CH55=   55
                    000056                          CH56=   56
                    000057                          CH57=   57
                                                    .SBTTL
                                                    .SBTTL  TEST NUMBER                     ABSTRACT
                                                    .SBTTL
                                                ;;**********************************************************************
                                                ;*TEST 1         TEST THAT "ERASE" AND DONe CAN BE SET AND CLEARED
                                                ;;**********************************************************************
002720  000004                                TST1:   SCOPE
002722  012737  000010  001166                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
002730  012737  000000  001564                        MOV     #0,$TMDAT               ;CLEAR

                                                ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
002746  012737  010000  001124                        MOV     #BIT12,$GDDAT           ;LOAD EXPECTED

                                                ;*      MOV     $GDDAT,@VCSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCSTAT

                                                ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
002774  023737  001124  001126                        CMP     $GDDAT,$BDDAT           ;COMPARE
003002  001401                                       BEQ     1$              ;;BR IF SET
003004  104003                                       ERROR   3                       ;ERASE FAILED TO SET, CHECK G5036-
                                                                                ;BC08R CONNECTION: A-VV, VV-A
003006  012737  001000  001124  1$:                  MOV     #BIT9,$GDDAT            ;LOAD EXPECTED

                                                ;*      MOV     @VCSTAT,$TMDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
003024  052737  001000  001564                        BIS     #BIT9,$TMDAT

                                                ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT

                                                ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
003052  023737  001124  001126                        CMP     $GDDAT,$BDDAT           ;COMPARE
003060  001401                                       BEQ     2$              ;;BR IF CLEARED
003062  104003                                       ERROR   3                       ;ERASE BIT FAILED TO CLEAR BY ERASE
                                                                                ;RETURN (SETTING OF THE CH 2 BIT)

003064  012737  000200  001124  2$:                  MOV     #BIT7,$GDDAT            ;LOAD EXPECTED

                                                ;*      MOV     @VCSTAT,$TMDAT  ;/READ DEViCE REG VCSTAT,PUT DATA IN $TMDAT.
003102  042737  001012  001564                        BIC     #BIT9!12,$TMDAT

                                                ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDaT TO DEVICE REG VCSTAT

                                                ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
003130  023737  001124  001126                        CMP     $GDDAT,$BDDAT           ;COMPARE
003136  001401                                       BEQ     TST2            ;;BR IF EQUAL
003140  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
                                                ;;**********************************************************************
003142  000004                                TST2:   SCOPE
```

```
                                                ;;DO 10 ITERATIONS
                                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
                                                ;LOAD EXPECTED
                                                ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
                                                ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
                                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                                ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
                                                ;SET UP A COUNTER
                                                ;DELAY
                                                ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
                                                ;COMPARE RESULTS
                                                ;;BR IF EQUAL
                                                ;INTENSIFY PULSE FAILED TO sTART
                                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
                ;*********************************************************************
                ;*TEST 3         TEST THAT CH 0 IS A BIPOLAR GROUND
                ;*********************************************************************
TST3:   SCOPE
                                                ;;DO 10 ITERATIONS
                                                ;LOAD EXPECTED VALUE
                                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
                                                ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADDBR
                                                ;CONVERT
                                                ;CH 0
                                                ;LOAD VALUE READ
                                                ;LOAD + OR - COUNT SPREAD
                                                ;COMPARE $GDDAT AND $BDDAT
                                                ;;BR IF WITHIN TOLERANCE
```

```
                                ERROR   1                       ;CH 0 FAILED TO EQUAL EXPECTED
                                                                ; VALUE


                        ;;**********************************************************************
                        ;*TEST 4         TEST THAT CH 40 IS A UNIpOLAR GROUND
                        ;;**********************************************************************
003460  000004          TST4:   SCOPE
003462  012737  000010  001166          MOV     #10,$TImES      ;;DO 10 ITERATIONS
003470  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED VALUE
003476  004537  012610                  JSR     R5,CONVRT               ;CONVERT
003502  000040                          CH40                    ;CH 40
003504  013737  013002  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003512  012737  000001  013010          MOV     #1,SPREAD               ;LOAD + OR - COUNT SPREAD
003520  004737  013012                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003524  000401                          BR      TST5            ;;BR IF WITHIN TOLERANCE
003526  104001                          ERROR   1                       ;CH 40 FAILED TO eQUAL EXPECTED
                                                                ; VALUE


                        ;;**********************************************************************
                        ;*TEST 5         TESt THAT CH 2 IS AT +1 VOLT BIPOLAR
                        ;;**********************************************************************
003530  000004          TST5:   SCOPE
003532  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003540  012737  001315  001124          MOV     #1315,$GDDAT            ;LOAD EXPECTED VALUE
003546  004537  012610                  JSR     R5,CONVRT               ;CONVERT
003552  000002                          CH2                     ;CH 2
003554  013737  013002  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003562  012737  000024  013010          MOV     #24,SPREAD              ;LOAD + OR - COUNT SPREAD
003570  004737  013012                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
003574  000401                          BR      TST6            ;;BR IF WITHIN TOLERANCE
003576  104001                          ERROR   1                       ;CH 2 FAILED TO EQUAL EXPECTED
                                                                ; VALUE


                        ;;**********************************************************************
                        ;*TESt 6         TEST THAT CH 42 IS AT +1 VOLT UNIPOLAR
                        ;;**********************************************************************
003600  000004          TST6:   SCOPE
003602  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003610  012737  000315  001124          MOV     #315,$GDDAT             ;LOAD EXPECTED VALUE
003616  004537  012610                  JSR     R5,CONVRT               ;CONVERT
003622  000042                          CH42                    ;CH 42
003624  013737  013002  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
003632  012737  000024  013010          MOV     #24,SPREAD              ;LOAD + OR - COUNT SPREAD
003640  004737  013012                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAt
003644  000401                          BR      TST7            ;;BR IF WITHIN TOLERANCE
003646  104001                          ERROR   1                       ;CH 42 FAILED TO EQUAL EXPECTED
                                                                ; VALUE


                        ;;**********************************************************************
                        ;*TEST 7         TEST THAT CH 3 IS AT +2.5 BIPOLAR
                        ;;**********************************************************************
003650  000004          TST7:   SCOPE
003652  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003660  013737  020266  001124          MOV     V1754,$GDDAT            ;LOAD EXPECTED VALUE
```

```
                                                JSR     R5,CONVRT               ;CONVERT
                                                CH3                             ;CH 3
                                                MOV     ADEND,$BDDAT            ;LOAD VALUE READ
                                                MOV     V24,SPREAD              ;LOAD + OR - COUNT SPREAD
                                                JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
                                                BR      TST10           ;;BR IF WITHIN TOLERANCE
                                                ERROR   1                       ;CH 3 FAILED TO EQUAL EXPECTED
                                                                                ; VALUE

                                        ;;**************************************************************
                                        ;*TEST 10       TEST THAT CH 43 IS AT +2.5 UNIPOLAR
                                        ;;**************************************************************
                                        TST10:  SCOPE
                                                MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                                MOV     #1000,$GDDAT            ;LOAD EXPECTED VALUE
                                                JSR     R5,CONVRT               ;CONVERT
                                                CH43                            ;CH 43
                                                MOV     ADEND,$BDDAT            ;LOAD VALUE READ
                                                MOV     V50,SPREAD              ;LOAD + OR - COUNT SPREAD
                                                JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
                                                BR      TST11           ;;BR IF WITHIN TOLERANCE
                                                ERROR   1                       ;CH 435FAILED TO EQUAL EXPECTED
                                                                                ; VALUE

                                        ;;**************************************************************
                                        ;*TEST 11       TEST THAT CH 4 IS AT -2.5 BIPOLAR
                                        ;;**************************************************************
                                        TST11:  SCOPE
                                                MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                                MOV     V24,$GDDAT              ;LOAD EXPECTED VALUE
                                                JSR     R5,CONVRT               ;CONVERT
                                                CH4                             ;CH 4
                                                MOV     ADEND,$BDDAT            ;LOAD VALUE READ
                                                MOV     V24,SPREAD              ;LOAD + OR - COUNT SPREAD
                                                JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
                                                BR      TST12           ;;BR IF WITHIN TOLERANCE
                                                ERROR   1                       ;CH 4 FAILED TO EQUAL EXPECTED VALUE

                                        ;;**************************************************************
                                        ;*TEST 12       TEST THAT CH 57 IS AT +4 VOLTS UNIPOLAR
                                        ;;**************************************************************
                                        TST12:  SCOPE
                                                MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                                MOV     #1463,$GDDAT            ;LOAD EXPECTED VALUE
                                                JSR     R5,CONVRT               ;CONVERT
                                                CH57                            ;CH 57
                                                MOV     ADEND,$BDDAT            ;LOAD VALUE READ
                                                MOV     #120,SPREAD             ;LOAD + OR - COUNT SPREAD
                                                JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
                                                BR      TST13           ;;BR IF WITHIN TOLERANCE
                                                ERROR   1                       ;CH 57 FAILED TO EQUAL EXPECTED VALUE

                                        ;;**************************************************************
                                        ;*TEST 13       TEST THAT CH 41 IS LESS THAN +200 MVOLTS UNIPOLAR
```

```
                                        ;;*****************************************************************
004110  000004                  TST13:  SCOPE
004112  012737  000004  001166          MOV     #4,$TIMES       ;;DO 4 ITERATIONS
004120  012737  000041  013006          MOV     #CH41,CHANL             ;LOAD CHANNEL # FOR ERROR TYPOUT
004126  112737  000041  001564          MOVB    #CH41,$TMDAT            ;LOAD MUX
004134  000337  001564                  SWAB    $TMDAT
                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
004150  013737  020274  001124          MOV     VA24,$GDDAT             ;LOAD EXPECTED VALUE
004156  013737  020274  013010          MOV     VA24,SPREAD             ;LOAD + OR - COUNT SPREAD
004164  012737  000200  016342          MOV     #BIT7,TEMp              ;LOAD DELAY
004172  005337  016342          2$:     DEC     TEMP                    ;DELAY
004176  001375                  BNE     2$
                                        ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
004210  005237  001564                  INC     $TMDAT
                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
004224                          3$:
                                        ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
004234  105737  001564                  TSTB    $TMDAT
004240  100371                  BPL     3$
                                        ;*      MOV     @ADDBR,$BDDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
004252  013737  001126  016310          MOV     $BDDAT,BIASC1           ;SAVE RESULTS
004260  004737  013012                  JSR     PC,COMPAR               ;COMPARE $GDDAT AND $BDDAT
004264  000401                  BR      1$              ;;BR IF WITHIN TOLERANCE
004266  104001                  ERROR   1               ;CH 41 FAILED TO EQUAL EXPECTED
                                                        ; VALUE, BIAS CURRENT TOO HIGH
004270  032777  010000  174642  1$:     BIT     #BIT12,@SWR             ;TEST FORCED SW
004276  001432                  BEQ     TST14           ;;BR IF NOT FORCED PRINTOUT
004300  104401  004306                  TYPE    ,65$            ;;TYPE ASCIZ STRING
004304  000412                  BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12><12>/AR-11 ADDRESS = /
004332                          64$:
004332  013746  001556                  MOV     ARBADD,-(SP)            ;SAVE ADDRESS
004336  104402                          TYPOC
004340  104401  020040                  TYPE    ,ACRLF
004344  104401  014256                  TYPE    ,BIASST
004350  013746  016310                  MOV     BIASC1,-(SP)
004354  104403                          TYPOS
004356     003     000                  .BYTE   3,0
004360  104401  014272                  TYPE    ,BIASDN
                                        ;;*****************************************************************
                                        ;*TEST 14        TEST THAT ERASE L (CH 53) IS GREATER THAN +3 VOLTS WHEN HIGH
                                        ;;*****************************************************************
004364  000004                  TST14:  SCOPE
004366  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
004374  012737  005000  001564          MOV     #BIT11!BIT9,$TMDAT      ;LOAD VC STATUS
                                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
004412  012737  001146  001124          MOV     #1146,$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,$TIMES      ;;DO 10 ITERATIONS
                                MOV     #0,$TMDAT               ;CLEAR VC STATUS

                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                MOV     #BIT12!BIT10,$TMDAT     ;SET ERASE

                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                MOV     #120,$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,$TIMES      ;;DO 10 ITERATIONS
                                MOV     #BIT12!BIT9,$TMDAT      ;LOAD VC STATUS

                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                MOV     #1146,$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
                        ;;******************************************************************
                        TST17:  SCOPE
                                MOV     #10,$TImES      ;;DO 10 ITERATIONS
                                MOV     #BIT11!BIT10,$TMDAT     ;SET WRITE-THRU

                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                MOV     #120,$GDDAT             ;LOAD EXPECTED VALUE
                                JSR     R5,CONVRT               ;CONVERT
                                CH54                            ;CH 54
                                MOV     ADEND,$BDDAT            ;LOAD VALUE READ
```

```
  977  004700  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
  978  004706  002001                          BGE     TST20           ;;BR IF GREATER THAN OR EQUAL
  979  004710  104007                          ERROR   7                       ;CH 54 FAILED TO EQUAL EXPECTED
  980                                                                          ; VALUE
  981
  982                                  ;;*****************************************************************
  983                                  ;*TEST 20        TEST THAT NON-STORE (CH 55) IS LESS THAN +400 MVOLTS WHEN LOW
  984                                  ;;*****************************************************************
  985  004712  000004                  TST20:  SCOPE
  986  004714  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
  987  004722  012737  000000  001564          MOV     #0,$TMDAT               ;CLEAR STORE MODE
  988
  989                                  ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
  990  004740  012737  000120  001124          MOV     #120,$GDDAT             ;LOAD EXPECTED VALUE
  991  004746  004537  012610                  JSR     R5,CONVRT               ;CONVERT
  992  004752  000055                          CH55                            ;CH 55
  993  004754  013737  013002  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
  994  004762  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
  995  004770  002001                          BGE     TST21           ;;BR IF GREATER THAN OR EQUAL
  996  004772  104007                          ERROR   7                       ;CH 54 FAILED TO EQUAL EXPECTED
  997                                                                          ; VALUE
  998
  999                                  ;;*****************************************************************
 1000                                  ;*TEST 21        TEST THAT NON-STORE L (CH 55) IS GREATER THAN +3 VOLTS WHEN HIGH
 1001                                  ;;*****************************************************************
 1002  004774  000004                  TST21:  SCOPE
 1003  004776  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 1004  005004  012737  017000  001564          MOV     #BIT12!BIT11!BIT10!BIT9,$TMDAT          ;SET STORE MODE
 1005
 1006                                  ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
 1007  005022  012737  001146  001124          MOV     #1146,$GDDAT            ;LOAD EXPECTED VALUE
 1008  005030  004537  012610                  JSR     R5,CONVRT               ;CONVERT
 1009  005034  000055                          CH55                            ;CH 55
 1010  005036  013737  013002  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
 1011  005044  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1012  005052  003401                          BLE     TST22           ;;BR IF LESS THAN OR EQUAL
 1013  005054  104002                          ERROR   2                       ;CH 55 FAILED TO EQUAL EXPECTED
 1014                                                                          ; VALUE
 1015
 1016                                  ;;*****************************************************************
 1017                                  ;*TEST 22        TEST THAT CHANNEL 02 L (CH 56) IS GREATER THAN +3 VOLTS WHEN HIGH
 1018                                  ;;*****************************************************************
 1019  005056  000004                  TST22:  SCOPE
 1020  005060  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 1021  005066  012737  014000  001564          MOV     #BIT12!BIT11,$TMDAT     ;LOAD VC STATUS
 1022
 1023                                  ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
 1024  005104  012737  001146  001124          MOV     #1146,$GDDAT            ;LOAD EXPECTED VALUE
 1025  005112  004537  012610                  JSR     R5,CONVRT               ;CONVERT
 1026  005116  000056                          CH56                            ;CH 56
 1027  005120  013737  013002  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
 1028  005126  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1029  005134  003401                          BLE     TST23           ;;BR IF LESS THAN OR EQUAL
 1030  005136  104002                          ERROR   2                       ;CH 56 FAILED TO EQUAL EXPECTED
```

```
                                                                ; VALUE
                                ;;*****************************************************************
                                ;*TEST 23        TEST THAT CHANNEL 02 L (CH 56) IS LESS THAN +400 MVOLTS WHEN LOW
                                ;;*****************************************************************
005140  000004                  TST23:  SCOPE
005142  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
005150  012737  003000  001564          MOV     #BIT9!BIT10,$TMDAT      ;SET CHANNEL 2
                                ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
005166  012737  000120  001124          MOV     #120,$GDDAT             ;LOAD EXPECTED VALUE
005174  004537  012610                  JSR     R5,CONVRT               ;CONVERT
005200  000056                          CH56                            ;CH 56
005202  013737  013002  001126          MOV     ADEND,$BDDAT            ;LOAD VALUE READ
005210  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005216  002001                          BGE     1$              ;;BR IF GREATER THAN
005220  104007                          ERROR   7                       ;CH 56 FAILED TO EQUAL EXPECTED
                                                                        ; VALUE
005222  012737  000000  001564  1$:     MOV     #0,$TMDAT               ;CLEAR BIT 9
                                ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                ;;*****************************************************************
                                ;*TEST 24        A TO D DIFFERENTIAL LINEARITY TEST
                                ;;*****************************************************************
005240  000004                  TST24:  SCOPE
005242  012737  000001  001166          MOV     #1,$TIMES       ;;DO 1 ITERATION

005250  004537  016344                  JSR     R5,DIFLIN               ;START TEST
005254  001604                          VCXREG                          ;X COARSE
005256  001606                          VCYREG                          ;Y FINE
005260      001     005                 .BYTE   1,5                     ;GO AND CHANNEL 5

005262  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
005266  013737  017170  001126          MOV     SKIPST,$BDDAT           ;LOAD # OF SKIPPED STATES
005274  001401                          BEQ     1$              ;;BR IF NO SKIPPED STATES
005276  104005                          ERROR   5                       ;SKIPPED STATE(S) DETECTED

005300  013737  017174  001126  1$:     MOV     EXCESS,$BDDAT           ;LOAD # OF >2LSB STATES
005306  001401                          BEQ     2$              ;;BR IF NO GREATER THAN 2LSB WIDE STATES
005310  104005                          ERROR   5                       ;> 2LSB WIDE STATE(S) DETECTED

005312  013737  017172  001126  2$:     MOV     DIFERR,$BDDAT           ;LOAD # OUTSIDE +- 5/2 LSB
005320  012737  000064  001124          MOV     #52.,$GDDAT             ;LOAD MAX # OF ALLOWABLE STATES OUTSIDE +- 1/2 L
005326  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;ANY ERRORS
005334  002001                          BGE     TST25           ;;BR IF NO ERROR
005336  104005                          ERROR   5                       ;DIFFERENTIAL LINEARITY ERROR, >5% OF STATE
                                                                        ;WIDTHS OUTSIDE +- 1/2 LSB

                                ;;*****************************************************************
                                ;*TEST 25        X D/A DIFFERENTIAL LINEARITY USING CH 5
                                ;;*****************************************************************
005340  000004                  TST25:  SCOPE
005342  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
```

```
1085  005350  012701  000001            MOV     #1,R1                   ;LOAD EDGE VALUE
1086  005354  012702  016146            MOV     #RSLT2,R2               ;LOAD RESULT POINTER
1087
1088  005360  010137  001564      1$:   MOV     R1,$TMDAT               ;LOAD X DAC WITH PRESET
1089
1090                              ;*    MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1091  005374  010137  005410            MOV     R1,10$                  ;LOAD EDGE VALUE
1092
1093  005400  004437  011716            JSR     R4,SAR                  ;SOFTWARE SUCCESSIVE APPROX. ROUTINE
1094  005404  001606                    VCYREG
1095  005406     000     005            .BYTE   0,CH5
1096  005410  000000      10$:  0                                       ;EDGE VALUE
1097  005412  000200            BIT7                                    ;50-50
1098
1099  005414  013737  016246  005552      MOV     DACSAV,12$              ;SAVE VALUE
1100
1101                              ;*    MOV     @VCXREG,$TMDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $TMDAT.
1102  005432  005337  001564            DEC     $TMDAT
1103
1104                              ;*    MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1105  005446  010137  005462            MOV     R1,11$                  ;LOAD SAME EDGE
1106
1107  005452  004437  011716            JSR     R4,SAR                  ;SOFTWARE S.A.R.
1108  005456  001606                    VCYREG
1109  005460     000     005            .BYTE   0,CH5
1110  005462  000000      11$:  0
1111  005464  000200            BIT7
1112
1113  005466  013737  016246  005554      MOV     DACSAV,13$              ;SAVE VALUE
1114  005474  013737  016246  001126      MOV     DACSAV,$BDDAT           ;LOAD VALUE INTO $BDDAT
1115  005502  163737  005552  001126      SUB     12$,$BDDAT              ;STEP HEIGHT IN $BDDAT
1116  005510  013722  001126            MOV     $BDDAT,(R2)+            ;SAVE RESULT
1117  005514  012737  000062  001124      MOV     #62,$GDDAT              ;LOAD EXPECTED VALUE FOR 1 LSB STEP
1118  005522  013737  020276  013010      MOV     V62,SPREAD              ;LOAD TOLERANCE
1119  005530  004737  013012            JSR     PC,COMPAR               ;TEST LIMITS
1120  005534  000401            BR      2$                      ;;BR IF WITHIN LIMITS
1121  005536  104010            ERROR   10                              ;D/A DIFFERENTIAL LINEARITY ERROR
1122  005540  006301      2$:   ASL     R1                              ;MOVE LEFT
1123  005542  020127  002000            CMP     R1,#2000                ;DONE
1124  005546  001304            BNE     1$                      ;;BR OF NOT DONE
1125  005550  000402            BR      TST26                   ;;
1126
1127  005552  000000      12$:  0
1128  005554  000000      13$:  0
1129                              ;;*****************************************************************
1130                              ;*TEST 26       Y D/A DIFFERENTIAL LINEARITY USING CH 6
1131                              ;;*****************************************************************
1132  005556  000004      TST26:  SCOPE
1133  005560  012737  000010  001166      MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1134  005566  012701  000001            MOV     #1,R1                   ;LOAD EDGE VALUE
1135  005572  012702  016204            MOV     #RSLT3,R2               ;LOAD RESULT POINTER
1136
1137  005576  010137  001564      1$:   MOV     R1,$TMDAT               ;LOAD Y DAC WITH PRESET
1138
```

```
1139                                    ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMdAT TO DEVICE REG VCYREG
1140  005612  010137  005626                    MOV     R1,10$                  ;LOAD EDGE VALUE
1141
1142  005616  004437  011716                    JSR     R4,SAR                  ;SOFTWARE S.A.R.
1143  005622  001604                            VCXREG
1144  005624     000     006                    .BYTE   0,CH6
1145  005626  000000                    10$:    0                               ;EDGE VALUE
1146  005630  000200                            BIT7
1147
1148  005632  013737  016246  006034            MOV     DACSAV,12$              ;SAVE VALUE
1149
1150                                    ;*      MOV     @VCYREG,$TMDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
1151  005650  005337  001564                    DEC     $TMDAT
1152
1153                                    ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1154  005664  010137  005700                    MOV     R1,11$                  ;LOAD EDGE
1155
1156  005670  004437  011716                    JSR     R4,SAR                  ;SOFTWARE S.A.R.
1157  005674  001604                            VCXREG
1158  005676     000     006                    .BYTE   0,CH6
1159  005700  000000                    11$:    0
1160  005702  000200                            BIT7
1161
1162  005704  013737  016246  006036            MOV     DACSAV,13$              ;SAVE RESULTS
1163  005712  013737  016246  001126            MOV     DACSAV,$BDDAT           ;LOAD $BDDAT
1164  005720  163737  006034  001126            SUB     12$,$BDDAT              ;STEP HEIGHT IN $BDDAT
1165  005726  013722  001126                    MOV     $BDDAT,(R2)+            ;SAVE RESULT
1166  005732  012737  000062  001124            MOV     #62,$GDDAT              ;LOAD EXPECTED
1167  005740  013737  020276  013010            MOV     V62,SPREAD              ;LOAD TOLERANCE
1168  005746  004737  013012                    JSR     PC,COMPAR               ;TEST IF WITHIN LIMITS
1169  005752  000401                            BR      2$              ;;BR IF WITHIN LIMITS
1170  005754  104010                            ERROR   10                      ;D/A DIFFERENTIAL LINEARITY ERROR
1171  005756  006301                    2$:     ASL     R1
1172  005760  020127  002000                    CMP     R1,#2000                ;DONE ?
1173  005764  001304                            BNE     1$              ;;BR IF NOT DONE
1174  005766  032777  010000  173144            BIT     #BIT12,@SWR             ;TEST FORCED PRINTOUT
1175  005774  001421                            BEQ     TST27           ;;BR IF NOT FORCED PRINTOUT
1176  005776  104401  014327                    TYPE    ,DIFMSG
1177  006002  004537  012564                    JSR     R5,TYPSTG               ;TYPE A STRING OF OCTAL #
1178  006006  016146                            RSLT2                           ;STARTING
1179  006010  000012                            10.                             ; # OF LOCATIONS
1180  006012  104401  014513                    TYPE    ,DIYMSG
1181  006016  004537  012564                    JSR     R5,TYPSTG               ;TYPE A STRING OF OCTAL #
1182  006022  016204                            RSLT3
1183  006024  000012                            10.                             ; # OF LOCATIONS
1184  006026  104401  014523                    TYPE    ,DIEMSG
1185  006032  000402                            BR      TST27           ;;
1186  006034  000000                    12$:    0
1187  006036  000000                    13$:    0
1188                                    ;;**************************************************************
1189                                    ;*TEST 27        A TO D POSITIVE MULTIPLEXER SETTLING TEST
1190                                    ;;**************************************************************
1191  006040  000004                    TST27:  SCOPE
1192  006042  012737  000010  001166            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
```

```
1193  006050  012737  001770  001564          MOV     #1770,$TMDAT            ;LOAD Y DAC
1194
1195                                  ;*        MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1196  006066  004437  011716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
1197  006072  001604                          VCXREG
1198  006074     000     006                  .BYTE   0,6
1199  006076  001771                          1771
1200  006100  000200                          BIT7
1201
1202  006102  013737  016246  006174          MOV     DACSAV,10$              ;SAVE RESULTS
1203  006110  004437  011716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
1204  006114  001604                          VCXREG
1205  006116     004     006                  .BYTE   CH4,CH6                 ;-2.5V  CH AND CH 6
1206  006120  001771                          1771                            ;EDGE
1207  006122  000200                          BIT7
1208
1209  006124  013737  016246  001126          MOV     DACSAV,$BDDAT           ;SAVE RESULT IN $BDDAT
1210  006132  163737  006174  001126          SUB     10$,$BDDAT              ;SUB FIRST VALUE
1211  006140  013737  001126  016304          MOV     $BDDAT,ADPMUX           ;SAVE RESULT
1212  006146  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED SETTLING ERROR
1213  006154  012737  000077  013010          MOV     #77,SPREAD              ;LOAD TOLERANCE
1214  006162  004737  013012                  JSR     PC,COMPAR               ;TEST IF WITHIN LIMITS
1215  006166  000403                          BR      TST30           ;;BR IF WITHIN
1216  006170  104011                          ERROR   11                      ;A/D POSITIVE SETTLING ERROR
1217  006172  000401                          BR      TST30           ;;
1218  006174  000000                  10$:    0
1219
1220                                  ;;**************************************************************
1221                                  ;*TESt 30        A TO D NEGATIVE MULTIPLEXER SETTLING TESt
1222                                  ;;**************************************************************
1223  006176  000004                  TST30:  SCOPE
1224  006200  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1225  006206  012737  000010  001564          MOV     #10,$TMDAT              ;LOAD Y DAC
1226
1227                                  ;*        MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1228  006224  004437  011716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
1229  006230  001604                          VCXREG
1230  006232     000     006                  .BYTE   0,6
1231  006234  000010                          10
1232  006236  000200                          BIT7
1233
1234  006240  013737  016246  006376          MOV     DACSAV,10$              ;SAVE RESULTS
1235  006246  004437  011716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
1236  006252  001604                          VCXREG
1237  006254     003     006                  .BYTE   CH3,CH6                 ;+2.5V  CH AND CH 6
1238  006256  000010                          10                              ;EDGE
1239  006260  000200                          BIT7
1240
1241  006262  013737  016246  001126          MOV     DACSAV,$BDDAT           ;SAVE RESULT IN $BDDAT
1242  006270  163737  006376  001126          SUB     10$,$BDDAT              ;SUB FIRST VALUE
1243  006276  013737  001126  016306          MOV     $BDDAT,ADNMUX           ;SAVE RESULTS
1244  006304  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED ZERO SETTLING ERROR
1245  006312  012737  000077  013010          MOV     #77,SPREAD              ;LOAD TOLERANCE
1246  006320  004737  013012                  JSR     PC,COMPAR               ;TEST IF WITHIN LIMITS
```

```
1247  006324  000401                          BR      1$                ;;BR IF WITHIN
1248  006326  104011                          ERROR   11                ;A/D NEGATIVE SETTLING ERROR
1249  006330  032777  010000  172602  1$:     BIT     #BIT12,@SWR       ;TEST FOR FORCED TYPEOUT
1250  006336  001420                          BEQ     TST31             ;;BR IF NOT FORCED
1251  006340  104401  014562                  TYPE    ,ADSMSG
1252  006344  013746  016304                  MOV     ADPMUX,-(SP)      ;SAVE MUX SETTLING
1253  006350  104403                          TYPOS
1254  006352     003     000                  .BYTE   3,0
1255  006354  104401  014651                  TYPE    ,ADEMSG
1256  006360  013746  016306                  MOV     ADNMUX,-(SP)      ;SAVE MUX SETTLING
1257  006364  104403                          TYPOS
1258  006366     003     000                  .BYTE   3,0
1259  006370  104401  020040                  TYPE    ,ACRLF
1260  006374  000401                          BR      TST31             ;;
1261
1262  006376  000000                  10$:    0
1263
1264                                          ;;*****************************************************************
1265                                          ;*TEST 31       A/D RMS NOISE TEST ON CHANNEL 7 - BIPOLAR
1266                                          ;;*****************************************************************
1267  006400  000004                  TST31:  SCOPE
1268  006402  012737  000010  001166          MOV     #10,$TIMES        ;;DO 10 ITERATIONS
1269
1270  006410  004437  011716                  JSR     R4,SAR                    ;S A R  ROUTINE
1271  006414  001606                          VCYREG                    ;Y AXIS
1272  006416     000     007                  .BYTE   0,7               ;CHANNEL 7, FINE Y
1273  006420  001000                          1000                      ;EDGE VALUE
1274  006422  000000                          0                         ;RMS RESULTS
1275
1276  006424  010537  001126                  MOV     R5,$BDDAT                 ;SAVE RESULTS (2X RMS NOISE, 1=1/50 LSB)
1277  006430  010537  016254                  MOV     R5,CH7RMS                 ;SAVE RESULTS (RMS NOISE, 1 = 1/100 LSB)
1278  006434  013737  016250  001124          MOV     RMSMAX,$GDDAT             ;LOAD EXPECTED
1279  006442  023737  001124  001126          CMP     $GDDAT,$BDDAT             ;COMPARE RESULTS
1280  006450  002001                          BGE     TST32             ;;BR IF NOISE WITHIN SPEC
1281  006452  104006                          ERROR   6                 ;A/D RMS NOISE (BIPOLAR) IS GREATER
1282                                                                    ;  THAN SPEC
1283
1284                                          ;;*****************************************************************
1285                                          ;*TEST 32       A/D PEAK NOISE TEST ON CHANNEL 7 - BIPOLAR
1286                                          ;;*****************************************************************
1287  006454  000004                  TST32:  SCOPE
1288  006456  012737  000010  001166          MOV     #10,$TIMES        ;;DO 10 ITERATIONS
1289
1290  006464  004437  011716                  JSR     R4,SAR                    ;S A R  ROUTINE
1291  006470  001606                          VCYREG                    ;Y AXIS
1292  006472     000     007                  .BYTE   0,7               ;CHANNEL 7, FINE Y
1293  006474  001000                          1000                      ;EDGE VALUE
1294  006476  100000                          BIT15                     ;PEAK RESULTS
1295
1296  006500  010537  001126                  MOV     R5,$BDDAT                 ;SAVE RESULTS (PEAK NOISE, 1 = .01 LSB)
1297  006504  010537  016256                  MOV     R5,CH7PEK                 ;SAVE RESULTS
1298  006510  013737  016252  001124          MOV     PEKMAX,$GDDAT             ;LOAD EXPECTED
1299  006516  023737  001124  001126          CMP     $GDDAT,$BDDAT             ;COMPARE RESULTS
1300  006524  002001                          BGE     TST33             ;;BR IF NOISE WITHIN SPEC
```

```
1301  006526  104006                          ERROR   6                       ;A/D PEAK NOISE (BIPOLAR) IS GREATER
1302                                                                          ; THAN SPEC
1303
1304                                  ;;*****************************************************************
1305                                  ;*TEST 33        A/D RMS NOISE TEST ON CHANNEL 47 - UNIPOLAR
1306                                  ;;*****************************************************************
1307  006530  000004                 TST33:  SCOPE
1308  006532  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1309
1310  006540  004437  011716                  JSR     R4,SAR                  ;S A R  ROUTINE
1311  006544  001606                          VCYREG                          ;Y AXIS
1312  006546     000     047                  .BYTE   0,CH47                  ;CHANNEL 47, FINE Y
1313  006550  000003                          3                               ;EDGE VALUE
1314  006552  000000                          0                               ;RMS RESULTS
1315
1316  006554  010537  001126                  MOV     R5,$BDDAT               ;SAVE RESULTS (RMS NOISE, 1=.01 LSB)
1317  006560  010537  016260                  MOV     R5,CH47RM               ;SAVE RESULT
1318  006564  013737  016250  001124          MOV     RMSMAX,$GDDAT           ;LOAD EXPECTED
1319  006572  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1320  006600  002001                          BGE     TST34           ;;BR IF NOISE WITHIN SPEC
1321  006602  104006                          ERROR   6                       ;A/D RMS NOISE (UNIPOLAR) IS GREATER
1322                                                                          ; THAN SPEC
1323
1324                                  ;;*****************************************************************
1325                                  ;*TEST 34        A/D PEAK NOISE TEST ON CHANNEL 47 - UNIPOLAR
1326                                  ;;*****************************************************************
1327  006604  000004                 TST34:  SCOPE
1328  006606  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1329
1330  006614  004437  011716                  JSR     R4,SAR                  ;S A R  ROUTINE
1331  006620  001606                          VCYREG                          ;Y AXIS
1332  006622     000     047                  .BYTE   0,CH47                  ;CHANNEL 47, FINE Y
1333  006624  000003                          3                               ;EDGE VALUE
1334  006626  100000                          BIT15                           ;PEAK RESULTS
1335
1336  006630  010537  001126                  MOV     R5,$BDDAT               ;SAVE RESULTS (PEAK NOISE, 1 = .01 LSB)
1337  006634  010537  016262                  MOV     R5,CH47PK               ;SAVE RESULT
1338  006640  013737  016252  001124          MOV     PEKMAX,$GDDAT           ;LOAD EXPECTED
1339  006646  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1340  006654  002001                          BGE     TST35           ;;BR IF NOISE WITHIN SPEC
1341  006656  104006                          ERROR   6                       ;A/D PEAK NOISE (UNIPOLAR) IS GREATER
1342                                                                          ; THAN SPEC
1343
1344                                  ;;*****************************************************************
1345                                  ;*TEST 35        X DAC RMS NOISE TEST ON CHANNEL 5
1346                                  ;;*****************************************************************
1347  006660  000004                 TST35:  SCOPE
1348  006662  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1349  006670  012737  001000  001564          MOV     #1000,$TMDAT
1350
1351                                  ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1352
1353  006706  004437  011716                  JSR     R4,SAR                  ;S A R  ROUTINE
1354  006712  001606                          VCYREG                          ;Y AXIS
```

```
1355  006714  000     005                     .BYTE   0,5                     ;CHANNFL 5, COARSE X AND FINE Y
1356  006716  001000                          1000                            ;EDGE VALUE
1357  006720  000000                          0                               ;RMS RESULTS
1358
1359  006722  010537  001126                  MOV     R5,$BDDAT               ;SAVE RESULTS (X DAC + A/D RMS NOISE, 1=.01 LSB)
1360  006726  163737  016254  001126          SUB     CH7RMS,$BDDAT           ;XDAC RMS NOISE ONLY
1361  006734  013737  001126  016264          MOV     $BDDAT,XDACRM           ;SAVE RESULT
1362  006742  013737  016250  001124          MOV     RMSMAX,$GDDAT           ;LOAD EXPECTED
1363  006750  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1364  006756  002001                          BGE     TST36           ;;BR IF NOISE WITHIN SPEC
1365  006760  104006                          ERROR   6                       ;X D/A RMS NOISE  IS GREATER
1366                                                                          ; THAN SPEC
1367
1368                                  ;;**********************************************************************
1369                                  ;*TEST 36        X DAC PEAK NOISE TEST ON CHANNEL 5
1370                                  ;;**********************************************************************
1371  006762  000004                  TST36:  SCOPE
1372  006764  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1373
1374
1375                                  ;*      MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1376  007002  004437  011716                  JSR     R4,SAR                  ;S A R  ROUTINE
1377  007006  001606                          VCYREG                          ;Y AXIS
1378  007010     000     005                  .BYTE   0,5                     ;CHANNEL 5, COARSE X AND FINE Y
1379  007012  001000                          1000                            ;EDGE VALUE
1380  007014  100000                          BIT15                           ;PEAK RESULTS
1381
1382  007016  010537  001126                  MOV     R5,$BDDAT               ;SAVE RESULTS (X DAC + A/D PEAK NOISE, 1=.01 LSB
1383  007022  163737  016256  001126          SUB     CH7PEK,$BDDAT           ;X DAC PEAK NOISE ONLY
1384  007030  013737  001126  016266          MOV     $BDDAT,XDACPK           ;SAVE RESULT
1385  007036  013737  016252  001124          MOV     PEKMAX,$GDDAT           ;LOAD EXPECTED
1386  007044  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1387  007052  002001                          BGE     TST37           ;;BR IF NOISE WITHIN SPEC
1388  007054  104006                          ERROR   6                       ;X D/A PEAK NOISE  IS GREATER
1389                                                                          ; THAN SPEC
1390
1391                                  ;;**********************************************************************
1392                                  ;*TEST 37        Y DAC RMS NOISE TEST ON CHANNEL 6
1393                                  ;;**********************************************************************
1394  007056  000004                  TST37:  SCOPE
1395  007060  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1396
1397
1398                                  ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1399  007076  004437  011716                  JSR     R4,SAR                  ;S A R  ROUTINE
1400  007102  001604                          VCXREG                          ;X AXIS
1401  007104     000     006                  .BYTE   0,6                     ;CHANNEL 6 ,COARSE Y AND FINE X
1402  007106  001000                          1000                            ;EDGE VALUE
1403  007110  000000                          0                               ;RMS RESULTS
1404
1405  007112  010537  001126                  MOV     R5,$BDDAT               ;SAVE RESULTS (Y DAC + A/D RMS NOISE, 1=.01 LSB)
1406  007116  163737  016254  001126          SUB     CH7RMS,$BDDAT           ;Y DAC RMS NOISE ONLY
1407  007124  013737  001126  016270          MOV     $BDDAT,YDACRM           ;SAVE RESULT
1408  007132  013737  016250  001124          MOV     RMSMAX,$GDDAT           ;LOAD EXPECTED
```

```
1409  007140  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1410  007146  002001                          BGE     TST40           ;;BR IF NOISE WITHIN SPEC
1411  007150  104006                          ERROR   6                       ;Y D/A RMS NOISE  IS GREATER
1412                                                                          ; THAN SPEC
1413
1414                                  ;;*****************************************************************
1415                                  ;*TEST 40        Y DAC PEAK NOISE TEST ON CHANNEL 6
1416                                  ;;*****************************************************************
1417  007152  000004                  TST40:  SCOPE
1418  007154  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1419
1420
1421                                  ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1422  007172  004437  011716                  JSR     R4,SAR                  ;S A R  ROUTINE
1423  007176  001604                          VCXREG                  ;X AXIS
1424  007200      000     006                 .BYTE   0,6                     ;CHANNEL 6, COARSE Y AND FINE X
1425  007202  001000                          1000                    ;EDGE VALUE
1426  007204  100000                          BIT15                   ;PEAK RESULTS
1427
1428  007206  010537  001126                  MOV     R5,$BDDAT               ;SAVE RESULTS (Y DAC + A/D PEAK NOISE, 1=.01 LSB
1429  007212  163737  016256  001126          SUB     CH7PEK,$BDDAT           ;Y DAC PEAK NOISE ONLY
1430  007220  013737  001126  016272          MOV     $BDDAT,YDACPK           ;SAVE RESULT
1431  007226  013737  016252  001124          MOV     PEKMAX,$GDDAT           ;LOAD EXPECTED
1432  007234  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE RESULTS
1433  007242  002001                          BGE     1$              ;;BR IF NOISE WITHIN SPEC
1434  007244  104006                          ERROR   6                       ;Y DAC PEAK NOISE  IS GREATER
1435                                                                          ; THAN SPEC
1436
1437  007246  032777  010000  171664  1$:     BIT     #BIT12,@SWR             ;TEST FORCED PRINTOUT
1438  007254  001432                          BEQ     TST41           ;;BR IF NOT FORCED
1439  007256  104401  014667                  TYPE    ,NOIMSG
1440  007262  004537  012542                  JSR     R5,BYTWO                ;TYPE 2 OCTAL COL.
1441  007266  016254                          CH7RMS
1442  007270  016256                          CH7PEK
1443  007272  104401  015010                  TYPE    ,NOIMG1
1444  007276  004537  012542                  JSR     R5,BYTWO                ;TYPE 2 OCTAL COL.
1445  007302  016260                          CH47RM
1446  007304  016262                          CH47PK
1447  007306  104401  015033                  TYPE    ,NOIMG2
1448  007312  004537  012542                  JSR     R5,BYTWO                ;TYPE 2 OCTAL COL.
1449  007316  016264                          XDACRM
1450  007320  016266                          XDACPK
1451  007322  104401  015056                  TYPE    ,NOIMG3
1452  007326  004537  012542                  JSR     R5,BYTWO
1453  007332  016270                          YDACRM
1454  007334  016272                          YDACPK
1455  007336  104401  020040                  TYPE    ,ACRLF
1456
1457                                  ;;*****************************************************************
1458                                  ;*TEST 41        BIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
1459                                  ;;*****************************************************************
1460  007342  000004                  TST41:  SCOPE
1461  007344  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1462  007352  004437  011716                  JSR     R4,SAR                  ;SOFTWARE S.A.R.
```

```
                                        VCYREG
                        007             .BYTE   0,CH7                   ;CHANNEL 7, FINE Y
                                        1001                            ;EDGE, 1000/1001 TRANSITION
                                        BIT7                            ;50-50

007366  013737  016246  016274           MOV     DACSAV,OFSTBX          ;SAVE A/D OFFSET BIPOLAR
007374  013737  016246  001126           MOV     DACSAV,$BDDAT          ;LOAD VALUE READ
007402  012737  001014  001124           MOV     #1014,$GDDAT           ;LOAD EXPECTED
007410  013737  020276  013010           MOV     V62,SPREAD             ;LOAD +- VARIABLE, 1LSB
007416  004737  013012                   JSR     PC,COMPAR              ;TEST IF WITHIN +- VALUE
007422  000401                           BR      TST42          ;;BR IF WITHIN LIMITS
007424  104012                           ERROR   12                     ;A/D BIPOLAR OFFSET ERROR

                                ;;*****************************************************************
                                ;*TEST 42        UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT INTERRUPT
                                ;;*****************************************************************
007426  000004                  TST42:  SCOPE
007430  012737  000010  001166           MOV     #10,$TIMES     ;;DO 10 ITERATIONS
007436  004437  011716                   JSR     R4,SAR                 ;SOFTWARE S.A.R.
007442  001606                           VCYREG
007444      000     047                 .BYTE   0,CH47                  ;CHANNEL 7, UNIPOLAR FINE Y
007446  000001                           1                              ;EDGE, 0/1 TRANSITION
007450  000200                           BIT7                           ;50-50

007452  013737  016246  016276           MOV     DACSAV,OFSTUX          ;SAVE A/D OFFSET UNIPOLAR
007460  013737  016246  001126           MOV     DACSAV,$BDDAT          ;LOAD VALUE READ
007466  012737  001014  001124           MOV     #1014,$GDDAT           ;LOAD EXPECTED
007474  013737  020276  013010           MOV     V62,SPREAD             ;LOAD +- VARIABLE, 1 LSB
007502  004737  013012                   JSR     PC,COMPAR              ;TEST IF WITHIN +- VALUE
007506  000401                           BR      TST43          ;;BR IF WITHIN LIMITS
007510  104012                           ERROR   12                     ;A/D UNIPOLAR OFFSET ERROR
                                ;;*****************************************************************
                                ;*TEST 43        BIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
                                ;;*****************************************************************
007512  000004                  TST43:  SCOPE
007514  012737  000010  001166           MOV     #10,$TIMES     ;;DO 10 ITERATIONS
007522  004437  011716                   JSR     R4,SAR                 ;SOFTWARE S.A.R.
007526  001606                           VCYREG
007530      000     007                 .BYTE   0,CH7                           ;CHANNEL 7, FINE Y
007532  001001                           1001                           ;EDGE, 1000/1001 TRANSITION
007534  000201                           BIT7!BIT0                      ;50-50, WAIT LOOP

007536  013737  016246  016300           MOV     DACSAV,OFSLBX          ;SAVE A/D OFFSET BIPOLAR
007544  013737  016246  001126           MOV     DACSAV,$BDDAT          ;LOAD VALUE READ
007552  012737  001014  001124           MOV     #1014,$GDDAT           ;LOAD EXPECTED
007560  013737  020276  013010           MOV     V62,SPREAD             ;LOAD +- VARIABLE, 1 LSB
007566  004737  013012                   JSR     PC,COMPAR              ;TEST IF WITHIN +- VALUE
007572  000401                           BR      1$             ;;BR IF WITHIN LIMITS
007574  104012                           ERROR   12                     ;A/D BIPOLAR OFFSET ERROR

007576  163737  016274  001126  1$:      SUB     OFSTBX,$BDDAT          ;OFFSET DUE TO WAIT LOOP
007604  005037  001124                   CLR     $GDDAT                 ;
007610  012737  000031  013010           MOV     #31,SPREAD             ;1/2 LSB ALLOWED
007616  004737  013012                   JSR     PC,COMPAR              ;WITHIN LIMITS
```

```
                                    BR      TST44           ;;BR IF WITHIN LIMITS
                                    ERROR   12              ;A/D BIPOLAR OFFSET ERROR DUE TO WAIT LOOP

                            ;;**************************************************************
                            ;*TEST 44        UNIPOLAR A/D OFFSET USING CH 7 WITH WAIT LOOP
                            ;;**************************************************************
                            TST44:  SCOPE
                                    MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                    JSR     R4,SAR          ;SOFTWARE S.A.R.
                                    VCYREG
                                    .BYTE   0,CH47          ;CHANNEL 7, UNIPOLAR, FINE Y
                                    1                       ;EDGE, 0/1 TRANSITION
                                    BIT7!BIT0               ;50-50, WAIT LOOP

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

                            1$:     SUB     OFSTUX,$BDDAT   ;OFFSET DUE TO WAIT LOOP
                                    CLR     $GDDAT          ;
                                    MOV     #31,SPREAD      ;1/2 LSB ALLOWED
                                    JSR     PC,COMPAR       ;WITHIN LIMITS
                                    BR      TST45           ;;BR IF WITHIN LIMITS
                                    ERROR   12              ;A/D UNIPOLAR OFFSET ERROR DUE TO WAIT LOOP
                            ;;**************************************************************
                            ;*TEST 45        X D/A OFFSET USING CH 5
                            ;;**************************************************************
                            TST45:  SCOPE
                                    MOV     #10,$TIMES      ;;DO 10 ITERATIONS

                            ;*      MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
                                    JSR     R4,SAR          ;SOFTWARE S.A.R.
                                    VCYREG                  ;Y AXIS
                                    .BYTE   0,CH5           ;CHANNEL 5, COARSE X AND FINE Y
                                    1001                    ;EDGE VALUE, 1000/1001 TRANSITION
                                    BIT7                    ;50-50
                                    MOV     DACSAV,$BDDAT   ;SAVE RESULT (X DAC + A/D OFFSET)
                                    SUB     OFSTBX,$BDDAT   ;OFFSET DUE TO x DAC
                                    CLR     $GDDAT          ;CLEAR EXPECTED
                                    MOV     V77,SPREAD      ;TOLERANCE DUE TO X DAC
                                    JSR     PC,COMPAR       ;TEST IF WITHIN +- LIMITS
                                    BR      TST46           ;;BR IF WITHIN LIMITS
                                    ERROR   12              ;X DAC OFFSET ERROR

                            ;;**************************************************************
                            ;*TEST 46        Y D/A OFFSET USING CH 6
                            ;;**************************************************************
                            TST46:  SCOPE
                                    MOV     #10,$TIMES      ;;DO 10 ITERATIONS
```

```
1571                                    ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1572  010054  004437  011716                      JSR     R4,SAR                  ;SOFTWARE S.A.R.
1573  010060  001604                              VCXREG                  ;X AXIS
1574  010062     000     006                      .BYTE   0,CH6                   ;CHANNEL 6, COARSE Y AND FINE X
1575  010064  001001                              1001                    ;EDGE VALUE, 1000/1001 TRANSITION
1576  010066  000200                              BIT7                    ;50-50
1577  010070  013737  016246  001126              MOV     DACSAV,$BDDAT           ;SAVE RESULT (Y DAC + A/D OFFSET)
1578  010076  163737  016274  001126              SUB     OFSTBX,$BDDAT           ;OFFSET DUE TO y DAC
1579  010104  005037  001124                      CLR     $GDDAT                  ;CLEAR EXPECTED
1580  010110  013737  020300  013010              MOV     V77,SPREAD              ;TOLERANCE DUE TO Y DAC
1581  010116  004737  013012                      JSR     PC,COMPAR               ;TEST IF WITHIN +- LIMITS
1582  010122  000401                              BR      TST47           ;;BR IF WITHIN LIMITS
1583  010124  104012                              ERROR   12                      ;Y DAC OFFSET ERROR
1584                                  ;;**********************************************************************
1585                                  ;*TEST 47        CALIBRATION USING CHANNEL 11 - X DAC VS. A/D
1586                                  ;;**********************************************************************
1587  010126  000004                  TST47:  SCOPE
1588  010130  012737  000010  001166              MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1589  010136  012737  001300  001564              MOV     #1300,$TMDAT            ;LOAD X DAC
1590
1591                                    ;*      MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1592  010154  012737  001600  001124              MOV     #1600,$GDDAT            ;LOAD EXPECTED
1593  010162  004537  012610          2$:         JSR     R5,CONVRT               ;CONVERT 32X AND AVERAGE
1594  010166  000011                              CH11
1595  010170  013737  013002  001126              MOV     ADEND,$BDDAT            ;READ VALUE OBTAINED
1596  010176  012737  000003  013010              MOV     #3,SPREAD               ;LOAD DEVIATION
1597  010204  004737  013012                      JSR     PC,COMPAR               ;TEST IF WITHIN
1598  010210  000401                              BR      1$                      ;BR IF WITHIN
1599  010212  104013                              ERROR   13                      ;CALIBRATION ERROR - X DAC VS. A/D
1600  010214                          1$:
1601
1602                                    ;*      MOV     @VCXREG,$TMDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $TMDAT.
1603  010224  162737  000100  001564              SUB     #100,$TMDAT
1604
1605                                    ;*      MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1606  010242  162737  000200  001124              SUB     #200,$GDDAT             ;DEC. EXPECTED
1607  010250  100344                              BPL     2$                      ;BR UNTIL DONE
1608
1609                                  ;;**********************************************************************
1610                                  ;*TEST 50        CALIBRATION USING CHANNEL 12 - Y DAC VS. A/D
1611                                  ;;**********************************************************************
1612  010252  000004                  TST50:  SCOPE
1613  010254  012737  000010  001166              MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1614  010262  012737  001300  001564              MOV     #1300,$TMDAT            ;LOAD Y DAC
1615
1616                                    ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1617  010300  012737  001600  001124              MOV     #1600,$GDDAT            ;LOAD EXPECTED
1618  010306  004537  012610          2$:         JSR     R5,CONVRT               ;CONVERT 32X AND AVERAGE
1619  010312  000012                              CH12
1620  010314  013737  013002  001126              MOV     ADEND,$BDDAT            ;READ VALUE OBTAINED
1621  010322  012737  000003  013010              MOV     #3,SPREAD               ;LOAD DEVIATION
1622  010330  004737  013012                      JSR     PC,COMPAR               ;TEST IF WITHIN
1623  010334  000401                              BR      1$                      ;BR IF WITHIN
1624  010336  104013                              ERROR   13                      ;CALIBRATION ERROR - Y DAC VS. A/D
```

```
1625  010340                          1$:
1626
1627                                  ;*      MOV     @VCYREG,$TMDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $TMDAT.
1628  010350  162737  000100  001564         SUB     #100,$TMDAT
1629
1630                                  ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1631  010366  162737  000200  001124         SUB     #200,$GDDAT             ;DEC. EXPECTED
1632  010374  100344                         BPL     2$                      ;BR UNTIL DONE
1633                                  ;;**********************************************************************
1634                                  ;*TEST 51        LINEARITY TEST USING CH 6 - Y DAC VS. A/D
1635                                  ;;**********************************************************************
1636  010376  000004                  TST51:  SCOPE
1637  010400  012737  000010  001166         MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1638  010406  012700  016026                 MOV     #NUMBF2,R0              ;LOAD EDGE VALUE POINTER
1639  010412  012702  016052                 MOV     #RSLTO,R2               ;LOAD RESULT POINTER
1640
1641  010416  011037  010446          1$:    MOV     (R0),10$                ;LOAD EDGE EXPECTED
1642  010422  012037  001564                 MOV     (R0)+,$TMDAT            ;LOAD DAC
1643
1644                                  ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1645  010436  004437  011716                 JSR     R4,SAR                  ;SOFTWARE S.A.R.
1646  010442  001604                         VCXREG
1647  010444     000     006                 .BYTE   0,CH6                   ;CHANNEL 6, COARSE Y AND FINE X
1648  010446  000000                  10$:   0                               ;EDGE VALUE
1649  010450  000200                         BIT7                            ;50-50
1650
1651  010452  013722  016246                 MOV     DACSAV,(R2)+            ;SAVE RESULTS
1652  010456  005710                         TST     (R0)                    ;LAST RESULT ?
1653  010460  100356                         BPL     1$                      ;BR UNTIL DONE
1654  010462  005037  001124                 CLR     $GDDAT                  ;ZERO NOMINAL LINEARITY ERROR
1655  010466  013737  020300  013010         MOV     V77,SPREAD              ;TOLERANCE FOR A/D + DAC
1656  010474  013700  016072                 MOV     RSLTO+20,R0             ;GET LAST VALUE
1657  010500  163700  016052                 SUB     RSLTO,R0                ;SUBTRACT FIRST VALUE
1658  010504  006200                         ASR     R0
1659  010506  006200                         ASR     R0
1660  010510  006200                         ASR     R0
1661  010512  005500                         ADC     R0                      ;AVERAGE DIFFERENCE IN R0
1662  010514  012701  000002                 MOV     #2,R1
1663  010520  005037  016072                 CLR     RSLTO+20                ;START RUNNING SUM OF ERRORS
1664  010524  060037  016052          2$:    ADD     R0,RSLTO                ;COMPUTE NOM. VALUE ON LINE BETW. END PTS.
1665  010530  163761  016052  016052         SUB     RSLTO,RSLTO(R1)         ;COMPUTE ERROR WITH RESP. TO END PT. LINE
1666  010536  066137  016052  016072         ADD     RSLTO(R1),RSLTO+20      ;KEEP RUNNING SUM
1667  010544  005721                         TST     (R1)+                   ;BUMP R1
1668  010546  020127  000020                 CMP     R1,#20                  ;DONE ?
1669  010552  001364                         BNE     2$              ;BR IF NOT
1670  010554  006237  016072                 ASR     RSLTO+20
1671  010560  006237  016072                 ASR     RSLTO+20
1672  010564  006237  016072                 ASR     RSLTO+20
1673  010570  005537  016072                 ADC     RSLTO+20                ;AVERAGE ERROR WITH RESP. TO END PT. LINE
1674  010574  005037  016052                 CLR     RSLTO
1675  010600  005001                         CLR     R1
1676  010602  163761  016072  016052  3$:    SUB     RSLTO+20,RSLTO(R1)      ;ERROR WITH RESP. TO "BEST STRAIGHT LINE"
1677
1678  010610  016137  016052  001126         MOV     RSLTO(R1),$BDDAT
```

```
1679  010616  004737  013012              JSR     PC,COMPAR           ;COMPARE LINEARITY ERROR WITH + OR - TOLERANCE
1680  010622  000422                      BR      4$
1681  010624  010102                      MOV     R1,R2
1682  010626  042702  177770              BIC     #177770,R2          ;MASK BITS 0-2
1683  010632  062702  000060              ADD     #60,R2              ;CONVERT TO ASCII
1684  010636  110237  013571              MOVB    R2,EM14A
1685  010642  010102                      MOV     R1,R2               ;LOAD R2
1686  010644  006202                      ASR     R2
1687  010646  006202                      ASR     R2
1688  010650  006202                      ASR     R2
1689  010652  042702  177770              BIC     #177770,R2          ;MASK
1690  010656  062702  000060              ADD     #60,R2              ;MAKE ASCII
1691  010662  110237  013570              MOVB    R2,EM14B            ;SAVE #
1692  010666  104014                      ERROR   14                  ;LINEARITY ERROR USING CH 6 - YDAC VS. A/D
1693  010670  005721              4$:     TST     (R1)+               ;BUMP R1
1694  010672  020127  000020              CMP     R1,#20              ;DONE ?
1695  010676  001341                      BNE     3$                  ;;BR IF NOT DONE
1696
1697                                      ;;*****************************************************************
1698                                      ;*TEST 52        LINEARITY TEST USING CH 5 - X DAC VS. A/D
1699                                      ;;*****************************************************************
1700  010700  000004              TST52:  SCOPE
1701  010702  012737  000010  001166      MOV     #10,$TIMES          ;;DO 10 ITERATIONS
1702  010710  012700  016026              MOV     #NUMBF2,R0          ;LOAD EDGE VALUE POINTER
1703  010714  012702  016110              MOV     #RSLT1,R2           ;LOAD RESULT POINTER
1704
1705  010720  011037  010750      1$:     MOV     (R0),10$            ;LOAD EDGE EXPECTED
1706  010724  012037  001564              MOV     (R0)+,$TMDAT        ;LOAD DAC
1707
1708                               ;*     MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1709  010740  004437  011716              JSR     R4,SAR              ;SOFTWARE S.A.R.
1710  010744  001606                      VCYREG
1711  010746     000     005              .BYTE   0,CH5               ;CHANNEL 5, COARSE X AND FINE Y
1712  010750  000000              10$:    0                           ;EDGE VALUE
1713  010752  000200                      BIT7                                ;50-50
1714
1715  010754  013722  016246              MOV     DACSAV,(R2)+        ;SAVE RESULTS
1716  010760  005710                      TST     (R0)                ;LAST RESULT ?
1717  010762  100356                      BPL     1$                  ;BR UNTIL DONE
1718  010764  005037  001124              CLR     $GDDAT              ;ZERO NOMINAL LINEARITY ERROR
1719  010770  013737  020300  013010      MOV     V77,SPREAD          ;TOLERANCE FOR A/D + DAC
1720  010776  013700  016130              MOV     RSLT1+20,R0         ;GET LAST VALUE
1721  011002  163700  016110              SUB     RSLT1,R0            ;SUBTRACT FIRST VALUE
1722  011006  006200                      ASR     R0
1723  011010  006200                      ASR     R0
1724  011012  006200                      ASR     R0
1725  011014  005500                      ADC     R0                  ;AVERAGE DIFFERENCE IN R0
1726  011016  012701  000002              MOV     #2,R1
1727  011022  005037  016130              CLR     RSLT1+20            ;START RUNNING SUM OF ERRORS
1728  011026  060037  016110      2$:     ADD     R0,RSLT1            ;COMPUTE NOM. VALUE ON LINE BETW. END PTS.
1729  011032  163761  016110  016110      SUB     RSLT1,RSLT1(R1)     ;COMPUTE ERROR WITH RESP. TO END PT. LINE
1730  011040  066137  016110  016130      ADD     RSLT1(R1),RSLT1+20  ;KEEP RUNNING SUM
1731  011046  005721                      TST     (R1)+               ;BUMP R1
1732  011050  020127  000020              CMP     R1,#20              ;DONE ?
```

```
1733  011054  001364                        BNE     2$              ;BR IF NOT
1734  011056  006237  016130                ASR     RSLT1+20
1735  011062  006237  016130                ASR     RSLT1+20
1736  011066  006237  016130                ASR     RSLT1+20
1737  011072  005537  016130                ADC     RSLT1+20                ;AVERAGE ERROR WITH RESP. TO END PT. LINE
1738  011076  005037  016110                CLR     RSLT1
1739  011102  005001                        CLR     R1
1740  011104  163761  016130  016110  3$:   SUB     RSLT1+20,RSLT1(R1)      ;ERROR WITH RESP. TO "BEST STRAIGHT LINE"
1741  011112  016137  016110  001126        MOV     RSLT1(R1),$BDDAT
1742  011120  004737  013012                JSR     PC,COMPAR               ;COMPARE LINEARITY ERROR WITH + OR - TOLERANCE
1743  011124  000422                        BR      4$
1744  011126  010102                        MOV     R1,R2
1745  011130  042702  177770                BIC     #177770,R2              ;MASK BITS 0-2
1746  011134  062702  000060                ADD     #60,R2                  ;CONVERT TO ASCII
1747  011140  110237  013571                MOVB    R2,EM14A
1748  011144  010102                        MOV     R1,R2                   ;LOAD R2
1749  011146  006202                        ASR     R2
1750  011150  006202                        ASR     R2
1751  011152  006202                        ASR     R2
1752  011154  042702  177770                BIC     #177770,R2              ;MASK
1753  011160  062702  000060                ADD     #60,R2                  ;MAKE ASCII
1754  011164  110237  013570                MOVB    R2,EM14B                ;SAVE #
1755  011170  104014                        ERROR   14                      ;LINEARITY ERROR USING CH 5 - XDAC VS. A/D
1756  011172  005721                4$:     TST     (R1)+                   ;BUMP R1
1757  011174  020127  000020                CMP     R1,#20                  ;DONE ?
1758  011200  001341                        BNE     3$              ;;BR IF NOT DONE
1759  011202  032777  010000  167730        BIT     #BIT12,@SWR             ;TEST IF FORCED TYPEOUT
1760  011210  001416                        BEQ     TST53           ;;BR IF NOT FORCED
1761  011212  104401  015101                TYPE    ,LINMSG
1762  011216  004537  012564                JSR     R5,TYPSTG               ;TYPE OCTAL STRING
1763  011222  016052                        RSLT0                           ;STARTING AT
1764  011224  000010                        8.                              ; # OF LOCATIONS
1765  011226  104401  015264                TYPE    ,LINMG1
1766  011232  004537  012564                JSR     R5,TYPSTG               ;TYPE OCTAL STRING
1767  011236  016110                        RSLT1                           ;STARTING AT
1768  011240  000010                        8.                              ; # OF LOCATIONS
1769  011242  104401  020040                TYPE    ,ACRLF
1770
1771                                ;;*****************************************************************
1772                                ;*TEST 53       DETERMINE IF MORE AR11'S ARE TO BE TESTED
1773                                ;;*****************************************************************
1774  011246  000004                TST53:  SCOPE
1775  011250  012737  000001  001166        MOV     #1,$TIMES       ;;DO 1 ITERATION
1776  011256  005737  001566                TST     NBEXT                   ;TEST IF ANY
1777  011262  001411                        BEQ     1$                      ;BR IF NONE
1778  011264  062737  000020  001556        ADD     #20,ARBADD              ;UPDATE DEVICE ADDRESS
1779  011272  062737  000020  001560        ADD     #20,ARBVCT              ;UPDATE DEVICE VECTOR
1780  011300  005337  001566                DEC     NBEXT                   ;ANOTHER ONE ?
1781  011304  000415                        BR      BYPAS1                  ;BR IF ANOTHER
1782  011306  013737  001256  001556  1$:   MOV     $BASE,ARBADD            ;RELOAD ADDRESS
1783  011314  013737  001252  001560        MOV     $VECT1,ARBVCT           ;RELOAD VECTOR
1784  011322  013737  001562  001566        MOV     NMBEXT,NBEXT            ;RELOAD NUMBER
1785  011330  004737  025300                JSR     PC,$RESET
1786  011334  000137  011350                JMP     $EOP                    ;DONE
```

```
                                        BYPAS1: MOV     #-1,R0
                                                JMP     RBEG2                   ;TEST ANOTHER UNIT
                                                .SBTTL
                                        .SBTTL  END OF PASS ROUTINE
                                        ;;**********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO BYPAS1
011350                                  $EOP:
                                                SCOPE
000004
005037  001102                                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
005037  001166                                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
005237  001210                                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
042737  100000  001210                        BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
005327                                          DEC     (PC)+           ;;LOOP?
000001                                  $EOpCT: .WORD   1
003022                                          BGT     $DOAGN          ;;YES
012737                                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
000001                                  $ENDCT: .WORD   1
011376                                          $EOPCT
104401  011455                                  TYPE    ,$ENDMG         ;;TYPE "END PASS #"
013746  001210                                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
104405                                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
104401  011452                                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
013700  000042                          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
001405                                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
000005                                          RESET                   ;;CLEAR THE WORLD
004710                                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
000240                                          NOP                     ;;SAVE ROOM
000240                                          NOP                     ;;FOR
000240                                          NOP                     ;;ACT11
011446                                  $DOAGN:
000137                                          JMP     @(PC)+          ;;RETURN
011340                                  $RTNAD: .WORD   BYPAS1
   377      377      000                $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
    15   042412   042116                $ENDMG: .ASCIZ  <15><12>/END PASS #/
050040   051501   020123
000043
                                        .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
                                        $TYPDS:
011472                                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
011472  010046                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
011474  010146                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
011476  010246                                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
011500  010346                                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
011502  010546                                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
011504  012746  020200                         MOV     20(SP),R5       ;;GET THE INPUT NUMBER
011510  016605  000020                         BPL     1$              ;;BR IF INPUT IS POS.
011514  100004                                 NEG     R5              ;;MAKE THE BINARY NUMBER POS.
011516  005405                                 MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
011520  112766  000055  000001                 CLR     R0              ;;ZERO THE CONSTANTS INDEX
                                        1$:     MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
011526  005000                                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
011530  012703  011706                         CLR     R2              ;;CLEAR THE BCD NUMBER
011534  112723  000040                         MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
011540  005002                          2$:    SUB     R1,R5           ;;FORM THIS BCD DIGIT
011542  016001  011676                 3$:    BLT     4$              ;;BR IF DONE
011546  160105                                 INC     R2              ;;INCREASE THE BCD DIGIT BY 1
011550  002402                                 BR      3$
011552  005202                          4$:    ADD     R1,R5           ;;ADD BACK THE CONSTANT
011554  000774                                 TST     R2              ;;CHECK IF BCD DIGIT=0
011556  060105                                 BNE     5$              ;;FALL THROUGH IF 0
011560  005702                                 TSTB    (SP)            ;;STILL DOING LEADING 0'S?
011562  001002                                 BMI     7$              ;;BR IF YES
011564  105716                                 ASLB    (SP)            ;;MSD?
011566  100407                          5$:    BCC     6$              ;;BR IF NO
011570  106316                                 MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
011572  103003                                 BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
011574  116663  000001  177777         6$:    BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
011602  052702  000060                 7$:    MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
011606  052702  000040                         TST     (R0)+           ;;JUST INCREMENTING
011612  110223                                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
011614  005720                                 BLT     2$              ;;GO DO THE NEXT DIGIT
011616  020027  000010                         BGT     8$              ;;GO TO EXIT
011622  002746                                 MOV     R5,R2           ;;GET THE LSD
011624  003002                                 BR      6$              ;;GO CHANGE TO ASCII
011626  010502                          8$:    TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
011630  000764                                 BPL     9$              ;;BR IF NO
011632  105726                                 MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
011634  100003                          9$:    CLRB    (R3)            ;;SET THE TERMINATOR
011636  116663  177777  177776                 MOV     (SP)+,R5        ;;POP STACK INTO R5
011644  105013                                 MOV     (SP)+,R3        ;;POP STACK INTO R3
011646  012605                                 MOV     (SP)+,R2        ;;POP STACK INTO R2
011650  012603                                 MOV     (SP)+,R1        ;;POP STACK INTO R1
011652  012602                                 MOV     (SP)+,R0        ;;POP STACK INTO R0
011654  012601                                 TYPE    $DBLK           ;;NOW TYPE THE NUMBER
011656  012600                                 MOV     2(SP),4(SP)     ;;ADJUST THE STACK
011660  104401  011706                         MOV     (SP)+,(SP)
011664  016666  000002  000004                 RTI                     ;;RETURN TO USER
011672  012616
011674  000002
011676  023420                          $DTBL:  10000.
```

```
                                        1000.
                                        100.
                                        10.
                        $DBLK:  .BLKW   4
                                .SBTTL  SUCCESSIVE APPROX. SUBROUTINE
                        ;S A R SUBROUTINE
                        ;       JSR     R4,SAR
                        ;       AXIS POINTER
                        ;       CH. IN HIGH BYTE / DUMMY CH. IN LOW BYTE
                        ;       EDGE VALUE
                        ;       RESULT/MODE
                        ;       RETURN WITH R5 = DIFFERENCE
                        ;
                        ;       BIT15 = 0       RMS
                        ;       BIT15 = 1       PEAK
                        ;       BIT7 = 1        50-50
                        SAR:    MOV     (R4)+,ADAC              ;SAVE DAC ADDRESS POINTER
                        MOV     (R4)+,SARCHN            ;SAVE CHANNEL
                        MOV     (R4)+,EDGE              ;SAVE VALUE EDGE
                        MOV     (R4)+,CONS              ;SAVE RMS/PEAK SW
                        MOV     R0,-(SP)                ;SAVE GPR
                        MOV     R1,-(SP)
                        MOV     R2,-(SP)
                        MOV     R3,-(SP)
                        MOVB    SARCHN+1,CHANL
                        CLR     FCHANL
                        MOVB    SARCHN+1,FCHANL+1
                        INC     FCHANL                  ;SET START BIT
                        MOV     @ADAC,ADAC              ;GET BUS ADDRESS OF ACTIVE DAC
                        CLR     FINS                    ;CLEAR 2 PASS COUNTER FLAG
                        TST     CONS                    ;TEST IF RMS
                        BPL     10$                     ;BR IF RMS
                        MOV     #2,R3                   ;LOAD A = .4% OF 512
                        BR      15$
                10$:    MOV     #121,R3                 ;LOAD A = 16% OF 512
                        TSTB    CONS                    ;TEST FOR 50-50
                        BPL     15$
                        MOV     #400,R3                 ;LOAD 50-50, A = 50% COUNT
                15$:    MOV     #1000,R2                ;SET UP MSB
                        CLR     $TMDAT
                ;*      MOV     $TMDAT,@ADAC    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADAC
                14$:    CLR     R1                      ;SET HIGH = 0
                        MOV     #1000,R0                ;SET UP 512 CONVERSIONS
                ;*      MOV     @ADAC,$TMDAT    ;/READ DEVICE REG ADAC,PUT DATA IN $TMDAT.
                        ADD     R2,$TMDAT               ;NEXT BIT OF ACTIVE DAC
                ;*      MOV     $TMDAT,@ADAC    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADAC
                6$:     MOV     FCHANL,$TMDAT           ;START CONVERSION
                ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
                12$:
```

```
1949
1950                                        ;*      MOV     @ADDBR,$TMDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
1951  012136  023737  001564  016314                CMP     $TMDAT,EDGE
1952  012144  002401                               BLT     1$                      ;YES, BR
1953  012146  005201                               INC     R1                      ;NO, INC HIGH FOR RESULT > OR = EDGE VALUE
1954  012150  105737  016312              1$:     TSTB    SARCHN                  ;TEST IF DUMMY CH. IS SET
1955  012154  001417                               BEQ     3$                      ;BR IF NOT
1956  012156  005037  001564                       CLR     $TMDAT
1957  012162  113737  016312  001565               MOVB    SARCHN,$TMDAT+1         ;LOAD MUX
1958  012170  005237  001564                       INC     $TMDAT
1959
1960                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
1961  012204                                4$:
1962  012204                                13$:
1963
1964                                        ;*      MOV     @ADDBR,$TMDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
1965  012214  005300                        3$:     DEC     R0                      ;DONE 512 TIMES ?
1966  012216  001334                               BNE     6$                      ;NO
1967  012220  020103                               CMP     R1,R3                   ;IS HIGH > OR = A ?
1968  012222  002412                               BLT     2$                      ;NO, LEAVE BIT ON
1969
1970                                        ;*      MOV     @ADAC,$TMDAT    ;/READ DEVICE REG ADAC,PUT DATA IN $TMDAT.
1971  012234  160237  001564                       SUB     R2,$TMDAT               ;YES, TAKE IT OUT
1972
1973                                        ;*      MOV     $TMDAT,@ADAC    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADAC
1974  012250  022702  000001                2$:     CMP     #1,R2                   ;TEST FOR Z = 1, S.A.R. DONE ?
1975  012254  001402                               BEQ     21$                     ;BR IF DONE
1976  012256  006202                               ASR     R2                      ;NO SET Z /2
1977  012260  000676                               BR      14$                     ;TRY AGAIN
1978  012262                                21$:
1979
1980                                        ;*      MOV     @ADAC,DACSAV    ;/READ DEVICE REG ADAC,PUT DATA IN DACSAV.
1981  012272  005737  016320                       TST     FINS                    ;FIRST OR SECOND TIME ?
1982  012276  001025                               BNE     45$                     ;SECOND, BR
1983
1984                                        ;*      MOV     @ADAC,$TMDAT    ;/READ DEVICE REG ADAC,PUT DATA IN $TMDAT.
1985  012310  013705  001564                       MOV     $TMDAT,R5               ;READ VALUE
1986  012314  005237  016320                       INC     FINS                    ;SET FLAG
1987  012320  005737  016316                       TST     CONS                    ;TEST IF PEAK/RMS
1988  012324  100003                               BPL     41$                     ;BR IF RMS
1989  012326  012703  000776                       MOV     #776,R3                 ;LOAD 99.6% OF 512
1990  012332  000405                               BR      42$
1991  012334  012703  000657              41$:     MOV     #657,R3                 ;SET A = 84% OF 512
1992  012340  105737  016316              43$:     TSTB    CONS
1993  012344  100411                               BMI     5$
1994  012346  000137  012036              42$:     JMP     15$                     ;DO MORE TESTING
1995  012352                                45$:
1996
1997                                        ;*      MOV     @ADAC,$TMDAT    ;/READ DEVICE REG ADAC,PUT DATA IN $TMDAT.
1998  012362  163705  001564                       SUB     $TMDAT,R5               ;SUBTRACT DIFF
1999  012366  005405                               NEG     R5
2000  012370  005037  001564              5$:      CLR     $TMDAT                  ;RETURN WITH 2 X NOISE IN R5
2001
2002                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
```

```
                                        MOV     (SP)+,R3
                                        MOV     (SP)+,R2
                                        MOV     (SP)+,R1
                                        MOV     (SP)+,R0
                                        RTS     R4                      ;EXIT

                                        .SBTTL  MISC. SUBROUTINES
                                ;CONVERT R2 INTO DECIMAL DIGITS

                        DECpRT: MOV     #-4,DIGCNT              ;LOAD COUNT
                                MOV     R1,-(SP)                ;SAVE R1
                                MOV     #DIGT1-1,R1             ;LOAD POINTER
                                MOV     #DECPNT+2,DECPNT                ;LOAD POINTER
                        1$:     MOV     #-1,DIGIT               ;LOAD #
                        2$:     INC     DIGIT                   ;UPDATE IT
                                SUB     @DECPNT,R2              ;SUB MAGNITUDE
                                BPL     2$                      ;BR IF +
                                ADD     @DECPNT,R2              ;RESTORE
                                BIS     #60,DIGIT               ;MAKE A #
                                MOVB    DIGIT,(R1)+             ;LOAD INTO STRING
                                INC     DIGCNT                  ;UPDATE COUNT
                                BNE     3$                      ;BR IF NOT DONE
                                MOV     (SP)+,R1                ;RESTORE R1
                                RTS     PC                      ;EXIT
                        3$:     ADD     #2,DECpNT               ;UPDATE POINTER
                                BR      1$                      ;BR BACK

                        DIGIT:  0
                        DIGCNT: 0
                        DECPNT: .+2
                                1000.
                                100.
                                10.
                                1.

                        DIGT0:  .BYTE   0
                        DIGT1:  .BYTE   0
                        DIGT2:  .BYTE   0
                        DIGT3:  .BYTE   0

                        BYTWO:  MOV     @(R5)+,-(SP)            ;SAVE ON STACK
                                TYPOS
                                .BYTE   4,1
                                TYPE    ,MULTSP
                                MOV     @(R5)+,-(SP)            ;SAVE ON STACK
                                TYPOS
                                .BYTE   4,1
                                RTS     R5                      ;EXIT

                        TYPSTG: MOV     (R5)+,R0                ;GET ADDRESS POINTER
                                MOV     (R5)+,R1                ;LOAD # OF LOC.
                        1$:     MOV     (R0)+,-(SP)             ;SAVE ON STACK
                                TYPOS
                                .BYTE   4,1
```

```
2057  012576  104401  020045            TYPE    ,SP3MSG
2058  012602  005301                    DEC     R1                      ;DONE ?
2059  012604  001371                    BNE     1$                      ;
2060  012606  000205              2$:   RTS     R5                      ;EXIT
2061                               ;SUBROUTINE TO GET AVERAGE OF 32 CONVERSIONS
2062
2063  012610  012537  012776      CONVRT: MOV     (R5)+,10$
2064  012614  013737  012776  013006      MOV     10$,CHANL
2065  012622  000337  012776              SWAB    10$
2066  012626  012737  000040  013000      MOV     #32.,11$
2067  012634  005037  013002              CLR     ADEND
2068  012640  013737  012776  001564      MOV     10$,$TMDAT
2069
2070                               ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2071  012656                       2$:
2072
2073                               ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2074  012666  005237  001564              INC     $TMDAT
2075
2076                               ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2077  012702                       1$:
2078
2079                               ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2080  012712  105737  001564              TSTB    $TMDAT
2081  012716  100371                      BPL     1$
2082
2083                               ;*      MOV     @ADDBR,$TMDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2084  012730  063737  001564  013002      ADD     $TMDAT,ADEND
2085  012736  005337  013000              DEC     11$
2086  012742  001345                      BNE     2$
2087  012744  006237  013002              ASR     ADEND
2088  012750  006237  013002              ASR     ADEND
2089  012754  006237  013002              ASR     ADEND
2090  012760  006237  013002              ASR     ADEND
2091  012764  006237  013002              ASR     ADEND
2092  012770  005537  013002              ADC     ADEND                   ;ROUND UP
2093  012774  000205                      RTS     R5
2094
2095  012776  000000              10$:    0
2096  013000  000000              11$:    0
2097  013002  000000              ADEND:  0
2098  013004  000000              FCHANL: 0
2099  013006  000000              CHANL:  0
2100  013010  000000              SPREAD: 0
2101
2102  013012  010046              COMPAR: MOV     R0,-(SP)
2103  013014  010146                      MOV     R1,-(SP)
2104  013016  013700  001124              MOV     $GDDAT,R0               ;LOAD R0
2105  013022  013701  001126              MOV     $BDDAT,R1               ;LOAD R1
2106  013026  160100                      SUB     R1,R0                   ;SUBTRACT
2107  013030  100001                      BPL     8$                      ;
2108  013032  005400                      NEG     R0
2109  013034  020037  013010      8$:     CMP     R0,SPREAD               ;MAGNITUDE OF DIFF. IN R0
2110  013040  003405                      BLE     10$
```

```
                                        9$:     MOV     (SP)+,R1
                                                MOV     (SP)+,R0
                                                ADD     #2,(SP)
                                                RTS     PC

                                        10$:    MOV     (SP)+,R1
                                                MOV     (SP)+,R0
                                                RTS     PC
                                                .SBTTL  ASCII MESSAGES AND ERROR POINTERS

                                        EM1:    .ASCIZ  \ERROR ON A/D CHANNEL\

                                        EM2:    .ASCIZ  /VC LOGIC SIGNAL HIGH OUTPUT TOO LOW/

                                        EM3:    .ASCIZ  /VC STATUS REGISTER IN ERROR/

                                        EM4:    .ASCIZ  /EXTERNAL A TO D START FAILED/

                                        EM5:    .ASCIZ  \A/D DIFFERENTIAL LINEARITY ERROR\

                                        EM6:    .ASCIZ  /NOISE LEVEL EXCEEDED LIMIT/

                                        EM7:    .ASCIZ  /VC LOGIC SIGNAL LOW OUTPUT TOO HIGH/

                                        EM10:   .ASCIZ  \D/A DIFFERENTIAL LINEARITY ERROR\
```

```
2165  013445     101  042057  044440  EM11:   .ASCIZ  \A/D INTER-CHANNEL SETTLING ERROR\
2166  013452  052116  051105  041455
2167  013460  040510  047116  046105
2168  013466  051440  052105  046124
2169  013474  047111  020107  051105
2170  013502  047522  000122
2171  013506  043117  051506  052105  EM12:   .ASCIZ  /OFFSET ERROR/
2172  013514  042440  051122  051117
2173  013522     000
2174  013523     103  046101  041111  EM13:   .ASCIZ  /CALIBRATION ERROR/
2175  013530  040522  044524  047117
2176  013536  042440  051122  051117
2177  013544     000
2178  013545     114  047111  040505  EM14:   .ASCII  /LINEARITY ERROR AT /
2179  013552  044522  054524  042440
2180  013560  051122  051117  040440
2181  013566  020124
2182  013570     060                  EM14B:  .BYTE   60
2183  013571     060     060     060  EM14A:  .BYTE   60,60,60,0
2184  013574     000
2185  013575     105  051122  041520  DH1:    .ASCIZ  /ERRPC   ARADD   CHANL   NOMINAL SPREAD  ACTUAL/
2186  013602  020040  040440  040522
2187  013610  042104  020040  041440
2188  013616  040510  046116  020040
2189  013624  047040  046517  047111
2190  013632  046101  051440  051120
2191  013640  040505  020104  040440
2192  013646  052103  040525  000114
2193  013654  051105  050122  020103  DH2:    .ASCIZ  /ERRPC   ARADD    CHANL  3V LEV. OUTPUT/
2194  013662  020040  051101  042101
2195  013670  020104  020040  041440
2196  013676  040510  046116  020040
2197  013704  053063  046040  053105
2198  013712  020056  052517  050124
2199  013720  052125     000
2200  013723     105  051122  041520  DH3:    .ASCIZ  /ERRPC   ARADD    GOOD    BAD/
2201  013730  020040  040440  040522
2202  013736  042104  020040  020040
2203  013744  047507  042117  020040
2204  013752  020040  040502  000104
2205  013760  051105  050122  020103  DH5:    .ASCIZ  /ERRPC   ARADD  # OF ST. ALLOWED/
2206  013766  020040  051101  042101
2207  013774  020104  021440  047440
2208  014002  020106  052123  020056
2209  014010  046101  047514  042527
2210  014016  000104
2211  014020  051105  050122  020103  DH6:    .ASCIZ  /ERRPC   ARADD   LIMIT   MEASURED/
2212  014026  020040  051101  042101
2213  014034  020104  020040  044514
2214  014042  044515  020124  020040
2215  014050  042515  051501  051125
2216  014056  042105     000
2217  014061     105  051122  041520  DH7:    .ASCIZ  \ERRPC   ARADD   A/D CH. .4 LEV  OUTPUT\
2218  014066  020040  040440  040522
```

```
 2219  014074  042104  020040  040440
 2220  014102  042057  041440  027110
 2221  014110  027040  020064  042514
 2222  014116  020126  047440  052125
 2223  014124  052520  000124
 2224  014130  051105  050122  020103  DH10:   .ASCIZ  \ERRPC   ARADD    STEP   NOMINAL SPREAD  ACTUAL\
 2225  014136  020040  051101  042101
 2226  014144  020104  020040  051440
 2227  014152  042524  020120  020040
 2228  014160  047516  044515  040516
 2229  014166  020114  050123  042522
 2230  014174  042101  020040  041501
 2231  014202  052524  046101     000
 2232  014207     105  051122  041520  DH11:   .ASCIZ  /ERRPC   ARADD   NOMINAL SPREAD  ACTUAL/
 2233  014214  020040  040440  040522
 2234  014222  042104  020040  047040
 2235  014230  046517  047111  046101
 2236  014236  051440  051120  040505
 2237  014244  020104  040440  052103
 2238  014252  040525  000114
 2239  014256  005015  020111  044502  BIASST: .ASCIZ  <15><12>/I BIAS = /
 2240  014264  051501  036440  000040
 2241  014272  024040                  BIASDN: .ASCII  / (/
 2242  014274  027062                  BASBT1: .ASCII  /2./
 2243  014276  020060  044515  051103  BASBT2: .ASCII  /0 MICROAMP LIMIT = /
 2244  014304  040517  050115  046040
 2245  014312  046511  052111  036440
 2246  014320     040
 2247  014321     065  024460  005015  BASBT3: .ASCIZ  /50)/<15><12>
 2248  014326     000
 2249  014327     015  005012  027504  DIFMSG: .ASCII  <15><12><12>\D/A DIFFERENTIAL LINEARITY\
 2250  014334  020101  044504  043106
 2251  014342  051105  047105  044524
 2252  014350  046101  046040  047111
 2253  014356  040505  044522  054524
 2254  014364  005015  052123  050105          .ASCII  <15><12>\STEP 0/1    1/2    3/4    7/10   17/20  37/40\
 2255  014372  030040  030457  020040
 2256  014400  020040  027461  020062
 2257  014406  020040  031440  032057
 2258  014414  020040  020040  027467
 2259  014422  030061  020040  030440
 2260  014430  027467  030062  020040
 2261  014436  033463  032057     060
 2262  014443     040  033467  030457          .ASCII  \ 77/100 177/200 377/400 777/1000\
 2263  014450  030060  030440  033467
 2264  014456  031057  030060  031440
 2265  014464  033467  032057  030060
 2266  014472  033440  033467  030457
 2267  014500  030060     060
 2268  014503     015  020012  020130          .ASCIZ  <15><12>\ X   \
 2269  014510  020040     000
 2270  014513     015  020012  020131  DIYMSG: .ASCIZ  <15><12>/ Y   /
 2271  014520  020040     000
 2272  014523     015  024012  047516  DIEMSG: .ASCIZ  <15><12>/(NOMINAL STEP HEIGHT = 62)/<15><12>
```

```
2273  014530  044515  040516  020114
2274  014536  052123  050105  044040
2275  014544  044505  044107  020124
2276  014552  020075  031066  006451
2277  014560  000012
2278  014562  005015  027501  020104  ADSMSG: .ASCII  <15><12>\A/D INTER-CHANNEL SETTLING (1 LSB = 62)\
2279  014570  047111  042524  026522
2280  014576  044103  047101  042516
2281  014604  020114  042523  052124
2282  014612  044514  043516  024040
2283  014620  020061  051514  020102
2284  014626  020075  031066     051
2285  014633     015  050012  051517          .ASCIZ  <15><12>/POSITIVE = /
2286  014640  052111  053111  020105
2287  014646  020075     000
2288  014651     040  047040  043505  ADEMSG: .ASCIZ  /  NEGATIVE = /
2289  014656  052101  053111  020105
2290  014664  020075     000
2291  014667     015  047012  044517  NOIMSG: .ASCII  <15><12>\NOISE:    RMS (1/4 LSB LIMIT = 31)  PEAK (2 LSB LIMIT = 144)\
2292  014674  042523  020072  020040
2293  014702  051040  051515  024040
2294  014710  027461  020064  051514
2295  014716  020102  044514  044515
2296  014724  020124  020075  030463
2297  014732  020051  050040  040505
2298  014740  020113  031050  046040
2299  014746  041123  046040  046511
2300  014754  052111  036440  030440
2301  014762  032064     051
2302  014765     015  040412  042057          .ASCIZ  <15><12>\A/D BIPOLAR     \
2303  014772  041040  050111  046117
2304  015000  051101  020040  020040
2305  015006  000040
2306  015010  005015  027501  020104  NOIMG1: .ASCIZ  <15><12>\A/D UNIPOLAR     \
2307  015016  047125  050111  046117
2308  015024  051101  020040  020040
2309  015032     000
2310  015033     015  054012  042040  NOIMG2: .ASCIZ  <15><12>\X D/A            \
2311  015040  040457  020040  020040
2312  015046  020040  020040  020040
2313  015054  000040
2314  015056  005015  020131  027504  NOIMG3: .ASCIZ  <15><12>\Y D/A            \
2315  015064  020101  020040  020040
2316  015072  020040  020040  020040
2317  015100     000
2318  015101     015  005012  044514  LINMSG: .ASCII  <15><12><12>/LINEARITY (1 LSB = 62)/<15><12>
2319  015106  042516  051101  052111
2320  015114  020131  030450  046040
2321  015122  041123  036440  033040
2322  015130  024462  005015
2323  015134  020040  020040  020040          .ASCII  /                0      200     400     600     1000    1200    1400    1600/
2324  015142  020040  020040  020040
2325  015150  020040  020040  020060
2326  015156  020040  020040  031040
```

```
2327  015164  030060  020040  020040
2328  015172  030064  020060  020040
2329  015200  033040  030060  020040
2330  015206  020040  030061  030060
2331  015214  020040  030440  030062
2332  015222  020060  020040  032061
2333  015230  030060  020040  030440
2334  015236  030066    060
2335  015241     015  054412  042040          .ASCIZ  <15><12>\Y D/A VS. A/D   \
2336  015246  040457  053040  027123
2337  015254  040440  042057  020040
2338  015262  000040
2339  015264  005015  020130  027504  LINMG1: .ASCIZ  <15><12>\X D/A VS. A/D   \
2340  015272  020101  051526  020056
2341  015300  027501  020104  020040
2342  015306     000
2343  015307     015  005012  027501  ADDIFM: .ASCIZ  <15><12><12>\A/D DIFFERENTIAL LINEARITY:\<15><12>
2344  015314  020104  044504  043106
2345  015322  051105  047105  044524
2346  015330  046101  046040  047111
2347  015336  040505  044522  054524
2348  015344  006472  000012
2349  015350  052123  052101  026505  ADDIF:  .ASCIZ  /STATE-WIDTH/<15><12>
2350  015356  044527  052104  006510
2351  015364  000012
2352  015366  034050  020051  045523  SKPMSG: .ASCIZ  /(8) SKIPPED STATE(S)/<15><12>
2353  015374  050111  042520  020104
2354  015402  052123  052101  024105
2355  015410  024523  005015     000
2356  015415     050  024470  037040  GRT2MG: .ASCIZ  /(8) > 2 LSB STATE(S)/<15><12>
2357  015422  031040  046040  041123
2358  015430  051440  040524  042524
2359  015436  051450  006451  000012
2360  015444  034050  020051  020074  NARMSG: .ASCIZ  \(8) < 1/2 LSB NARROW STATE(S)\<15><12>
2361  015452  027461  020062  051514
2362  015460  020102  040516  051122
2363  015466  053517  051440  040524
2364  015474  042524  051450  006451
2365  015502  000012
2366  015504  034050  020051  020076  WIDMSG: .ASCIZ  \(8) > 1 1/2 LSB WIDE STATE(S)\<15><12>
2367  015512  020061  027461  020062
2368  015520  051514  020102  044527
2369  015526  042504  051440  040524
2370  015534  042524  051450  006451
2371  015542  000012
2372  015544  054130  054056  020045  PERHLF: .ASCIZ  \XX.X% OF STATES WITHIN 1/2 LSB  (SPEC = 95%)\<15><12>
2373  015552  043117  051440  040524
2374  015560  042524  020123  044527
2375  015566  044124  047111  030440
2376  015574  031057  046040  041123
2377  015602  020040  051450  042520
2378  015610  020103  020075  032471
2379  015616  024445  005015     000
2380  015623     130  027130  022530  PERQRT: .ASCIZ  \XX.X% OF STATES WITHIN 1/4 LSB \<15><12>
```

```
2381  015630  047440  020106  052123
2382  015636  052101  051505  053440
2383  015644  052111  044510  020116
2384  015652  027461  020064  051514
2385  015660  020102  005015  000
2386  015665  040     020040  020040  MULTSP: .ASCIZ  /                    /
2387  015672  020040  020040  020040
2388  015700  020040  020040  020040
2389  015706  020040  020040  020040
2390  015714  000040
2391                                  .EVEN
2392  015716  001116  001556  013006  DT1:    $ERRPC,ARBADD,CHANL,$GDDAT,SPREAD,$BDDAT,0
2393  015724  001124  013010  001126
2394  015732  000000
2395  015734  001116  001556  013006  DT2:    $ERRPC,ARBADD,CHANL,$GDDAT,$BDDAT,0
2396  015742  001124  001126  000000
2397  015750  001116  001556  001124  DT3:    $ERRPC,ARBADD,$GDDAT,$BDDAT,0
2398  015756  001126  000000
2399  015762  001116  001556  001126  DT5:    $ERRPC,ARBADD,$BDDAT,$GDDAT,0
2400  015770  001124  000000
2401  015774  001116  001556  016314  DT10:   $ERRPC,ARBADD,EDGE,$GDDAT,SPREAD,$BDDAT,0
2402  016002  001124  013010  001126
2403  016010  000000
2404  016012  001116  001556  001124  DT11:   $ERRPC,ARBADD,$GDDAT,SPREAD,$BDDAT,0
2405  016020  013010  001126  000000
2406  016026  000004  000203  000402  NUMBF2: 4,203,402,601,1000,1177,1376,1575,1774,BIT15
2407  016034  000601  001000  001177
2408  016042  001376  001575  001774
2409  016050  100000
2410  016052  000000  000000  000000  RSLT0:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2411  016060  000000  000000  000000
2412  016066  000000  000000  000000
2413  016074  000000  000000  000000
2414  016102  000000  000000  000000
2415  016110  000000  000000  000000  RSLT1:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2416  016116  000000  000000  000000
2417  016124  000000  000000  000000
2418  016132  000000  000000  000000
2419  016140  000000  000000  000000
2420  016146  000000  000000  000000  RSLT2:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2421  016154  000000  000000  000000
2422  016162  000000  000000  000000
2423  016170  000000  000000  000000
2424  016176  000000  000000  000000
2425  016204  000000  000000  000000  RSLT3:  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2426  016212  000000  000000  000000
2427  016220  000000  000000  000000
2428  016226  000000  000000  000000
2429  016234  000000  000000  000000
2430  016242  000000
2431                                  .SBTTL  DIFFERENTIAL LINEARITY SUBROUTINE
2432  016244  170412                  ADAC:   170412                  ;BUS ADDRESS OF DAC
2433  016246  000000                  DACSAV: 0
2434  016250  000031                  RMSMAX: 25.                     ;1/4 LSB
```

```
2435  016252  000310                PEKMAX: 200.                    ;2 LSB
2436  016254  000000                CH7RMS: 0
2437  016256  000000                CH7PEK: 0
2438  016260  000000                CH47RM: 0
2439  016262  000000                CH47PK: 0
2440  016264  000000                XDACRM: 0
2441  016266  000000                XDACPK: 0
2442  016270  000000                YDACRM: 0
2443  016272  000000                YDACPK: 0
2444  016274  000000                OFSTBX: 0
2445  016276  000000                OFSTUX: 0
2446  016300  000000                OFSLBX: 0
2447  016302  000000                OFSLUX: 0
2448  016304  000000                ADPMUX: 0
2449  016306  000000                ADNMUX: 0
2450  016310  000000                BIASC1: 0
2451  016312     000      006       SARCHN: .BYTE   0,6             ;CHANNEL # IN HIGH BYTE
2452  016314  001000                EDGE:   1000                    ;CONVERTED VALUE EDGE
2453  016316  000000                CONS:   0                       ;0 = RMS BIT 15 = 1 PEAK
2454  016320  000000                FINS:   0                       ;2 PASS COUNTER
2455  016322  000031                LOLIM1: 31
2456  016324  000046                LOLIM2: 46
2457  016326  000113                HILIM1: 113
2458  016330  000076                HILIM2: 76
2459  016332  000000                QRTOK:  0
2460  016334  000000                NARROW: 0
2461  016336  000000                WIDE:   0
2462  016340  000062                AMEAN:  62
2463  016342  000000                TEMP:   0
2464
2465  016344  012537  020142        DIFLIN: MOV     (R5)+,VCREG1            ;LOAD COARSE REGISTER ADDRESS
2466  016350  012537  020144                MOV     (R5)+,VCREG2            ;LOAD FINE ADDRESS
2467  016354  012537  020146                MOV     (R5)+,DIFCHN            ;LOAD CHANNEL
2468  016360  010546                        MOV     R5,-(SP)
2469  016362  005037  001564                CLR     $TMDAT
2470
2471                            ;*          MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
2472  016376  017737  001540  020142        MOV     @VCREG1,VCREG1
2473  016404  017737  001534  020144        MOV     @VCREG2,VCREG2
2474
2475                            ;*          MOV     $TMDAT,@VCREG1  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG1
2476
2477                            ;*          MOV     $TMDAT,@VCREG2  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG2
2478  016432  012701  025726                MOV     #ADBUFF,%1
2479  016436  005000                        CLR     %0
2480  016440  005021                1$:     CLR     (1)+                    ;CLEAR MEMORY
2481  016442  022701  032322                CMP     #LAST,%1
2482  016446  001374                        BNE     1$
2483
2484  016450  004737  017570        CONVR:  JSR     PC,CONVT
2485  016454  001410                        BEQ     2$                      ;BRANCH IF RESULT = 0
2486  016456  020500                        CMP     R5,R0                   ;COMPARE RESULT AND POINT
2487  016460  100406                        BMI     2$                      ;BRANCH FOR RESULT < POINT
2488  016462  105265  025726                INCB    ADBUFF(%5)              ;INCREMENT CONTENTS OF Z
```

```
2489  016466  103003                        BCC     2$
2490  016470  112765  000377  025726        MOVB    #377,ADBUFF(R5)
2491  016476                          2$:
2492
2493                          ;*      MOV     @VCREG2,$TMDAT  ;/READ DEVICE REG VCREG2,PUT DATA IN $TMDAT.
2494  016506  062737  000001  001564        ADD     #1,$TMDAT               ;UPDATE FINE REG.
2495
2496                          ;*      MOV     $TMDAT,@VCREG2  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG2
2497
2498                          ;*      MOV     @VCREG2,$BDDAT  ;/READ DEVICE REG VCREG2,PUT DATA IN $BDDAT.
2499  016534  005737  001126                TST     $BDDAT
2500  016540  001343                        BNE     CONVR                   ;BRANCH UNTIL FINISH
2501
2502                          ;*      MOV     @VCREG1,$TMDAT  ;/READ DEVICE REG VCREG1,PUT DATA IN $TMDAT.
2503  016552  022737  001774  001564        CMP     #1774,$TMDAT            ;TEST COARSE REG.
2504  016560  001436                        BEQ     READ
2505
2506                          ;*      MOV     @VCREG1,$TMDAT  ;/READ DEVICE REG VCREG1,PUT DATA IN $TMDAT.
2507  016572  062737  000014  001564        ADD     #14,$TMDAT              ;UPDATE COARSE REG.
2508
2509                          ;*      MOV     $TMDAT,@VCREG1  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG1
2510  016610  005037  001564                CLR     $TMDAT                  ;CLEAR FINE REG.
2511
2512                          ;*      MOV     $TMDAT,@VCREG2  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCREG2
2513  016624  004737  017570                JSR     PC,CONVT
2514  016630  062705  000004                ADD     #4,R5                   ;(4+RESULT) IN R5
2515  016634  010500                        MOV     %5,%0                   ;LOAD POINT
2516  016636  062705  025726                ADD     #ADBUFF,%5              ;
2517  016642  012704  000010                MOV     #10,%4
2518  016646  105025                  1$:   CLRB    (R5)+                   ;CLEAR 8. MEM BYTE LOCATIONS
2519  016650  005304                        DEC     %4
2520  016652  001375                        BNE     1$
2521  016654  000675                        BR      CONVR
2522  016656  032777  010000  162254  READ: BIT     #BIT12,@SWR
2523  016664  001402                        BEQ     13$
2524  016666  104401  015307                TYPE    ,ADDIFM
2525  016672  005000                  13$:  CLR     R0
2526  016674  012702  001776                MOV     #1776,%2
2527  016700  012703  025727                MOV     #ADBUFF+1,%3
2528  016704  012737  000001  016342        MOV     #1,TEMP
2529  016712  005037  016332                CLR     QRTOK
2530  016716  005037  017172                CLR     DIFERR
2531  016722  005037  016334                CLR     NARROW
2532  016726  005037  016336                CLR     WIDE
2533  016732  005037  017166                CLR     20$
2534  016736  005037  017170                CLR     SKIPST
2535  016742  005037  017174                CLR     EXCESS
2536  016746  111300                  2$:   MOVB    (R3),R0
2537  016750  020027  000144                CMP     R0,#144                 ;TEST IF STATE WIDTH > 2LSB
2538  016754  003024                        BGT     1$                      ;YES, BR
2539  016756  105260  031720                INCB    ABUFF4(R0)              ;UPDATE STATE-WIDTH BUFFER
2540  016762  020037  016326                CMP     R0,HILIM1               ;TEST IF STATE WIDTH > 1 1/2 LSB
2541  016766  003021                        BGT     3$                      ;YES, BR
2542  016770  020037  016330                CMP     R0,HILIM2               ;TEST IF STATE WIDTH > 1 1/4 LSB
```

```
2543  016774  003061                        BGT     4$                      ;YES, BR
2544  016776  020037  020302                CMP     R0,V1                   ;IS IT A SKIPPED STATE?
2545  017002  103416                        BLO     10$                     ;YES, BR
2546  017004  020037  016322                CMP     R0,LOLIM1               ;TEST IF STATE WIDTH < 1/2 LSB
2547  017010  002415                        BLT     11$                     ;YES BR
2548  017012  020037  016324                CMP     R0,LOLIM2               ;TEST IF STATE WIDTH < 3/4 LSB
2549  017016  002450                        BLT     4$                      ;YES, BR
2550  017020  005237  016332                INC     QRTOK                   ;STATE WIDTH BETWEEN 3/4 AND 1 1/4 LSB
2551  017024  000445                        BR      4$
2552  017026  005237  017174        1$:     INC     EXCESS                  ;UPDATE > 2 LSB WIDE STATE COUNT
2553  017032  005237  016336        3$:     INC     WIDE                    ;UPDATE > 1 1/2 LSB WIDE STATE COUNT
2554  017036  000406                        BR      12$
2555  017040  005237  017170        10$:    INC     SKIPST                  ;UPDATE SKIPPED STATE COUNT
2556  017044  105700                11$:    TSTB    R0
2557  017046  100767                        BMI     1$
2558  017050  005237  016334                INC     NARROW                  ;UPDATE < 1/2 LSB NARROW STATE COUNT
2559  017054  005237  017172        12$:    INC     DIFERR                  ;UPDATE OUTSIDE + OR - 1/2 LSB COUNT
2560  017060  032777  010000  162052  5$:    BIT     #BIT12,@SWR             ;TEST FOR FORCED TYPEOUT
2561  017066  001424                        BEQ     4$              ;;BR IF NOT
2562  017070  005737  017166                TST     20$                     ;TEST IF FIRST TIME ?
2563  017074  001004                        BNE     6$                      ;BR IF YES
2564  017076  005237  017166                INC     20$
2565  017102  104401  015350                TYPE    ,ADDIF
2566  017106  013746  016342        6$:     MOV     TEMP,-(SP)              ;SAVE CURRENT STATE
2567  017112  104403                        TYPOS
2568  017114     004     001                .BYTE   4,1
2569  017116  104401  020054                TYPE    ,TYANXX
2570  017122  042700  177400                BIC     #177400,R0
2571  017126  010046                        MOV     R0,-(SP)
2572  017130  104403                        TYPOS
2573  017132     003     001                .BYTE   3,1
2574  017134  104401  020040                TYPE    ,ACRLF
2575  017140  005237  016342        4$:     INC     TEMP
2576  017144  105723                        TSTB    (R3)+
2577  017146  005302                        DEC     %2
2578  017150  001276                        BNE     2$
2579  017152  032777  010000  161760        BIT     #BIT12,@SWR             ;TEST BIT 12
2580  017160  001006                        BNE     SWDIST                  ;BR IF FORCED PRINTOUT
2581  017162  012605                        MOV     (SP)+,R5
2582  017164  000205                        RTS     R5
2583  017166  000000                20$:    0
2584  017170  000000                SKIPST: 0                               ;SKIPPED STATE COUNT
2585  017172  000000                DIFERR: 0                               ;OUTSIDE + OR - 1/2 LSB STATE COUNT
2586  017174  000000                EXCESS: 0                               ;> 2 LSB STATE COUNT
2587
2588  017176  012703  031720        SWDIST: MOV     #ABUFF4,%3
2589  017202  005037  016342                CLR     TEMP
2590  017206  104401  020040                TYPE    ,ACRLF
2591  017212  013746  017170                MOV     SKIPST,-(SP)            ;SAVE SKIPPED STATES
2592  017216  104403                        TYPOS
2593  017220     003     000                .BYTE   3,0
2594  017222  104401  015366                TYPE    ,SKPMSG
2595  017226  013746  017174                MOV     EXCESS,-(SP)            ;SAVE EXCESSIVE WIDE STATES
2596  017232  104403                        TYPOS
```

```
2597  017234  003  000                          .BYTE   3,0
2598  017236  104401  015415                    TYPE    ,QRT2MG
2599  017242  013746  016334                    MOV     NARROW,-(SP)
2600  017246  104403                            TYPOS
2601  017250  003  000                          .BYTE   3,0
2602  017252  104401  015444                    TYPE    ,NARMSG
2603  017256  013746  016336                    MOV     WIDE,-(SP)              ;SAVE WIDE STATES
2604  017262  104403                            TYPOS
2605  017264  003  000                          .BYTE   3,0
2606  017266  104401  015504                    TYPE    ,WIDMSG
2607  017272  012700  001776                    MOV     #1022.,R0
2608  017276  163700  016336                    SUB     WIDE,R0
2609  017302  163700  016334                    SUB     NARROW,R0
2610  017306  004537  020346                    JSR     R5,PRCNT                ;CONVERT TO PERCENT
2611  017312  015544                            PERHLF
2612  017314  104401  015544                    TYPE    ,PERHLF
2613  017320  013700  016332                    MOV     QRTOK,R0
2614  017324  004537  020346                    JSR     R5,PRCNT
2615  017330  015623                            PERQRT
2616  017332  104401  015623                    TYPE    ,PERQRT
2617  017336  104401                            TYPE
2618  017340  020056                            STDMSG
2619  017342  013702  016342          6$:       MOV     TEMP,R2
2620  017346  006302                            ASL     R2
2621  017350  004737  012416                    JSR     PC,DECPRT               ;CONVERT R2 TO DEC.
2622  017354  113737  012537  017702            MOVB    DIGT1,RSV1
2623  017362  113737  012540  017704            MOVB    DIGT2,RSV2
2624  017370  113737  012541  017705            MOVB    DIGT3,RSV3
2625
2626  017376  111302                            MOVB    (R3),R2
2627  017400  004737  012416                    JSR     PC,DECPRT               ;CONVERT R2 TO DEC.
2628  017404  113737  012536  017707            MOVB    DIGT0,RSX1
2629  017412  113737  012537  017710            MOVB    DIGT1,RSX2
2630  017420  113737  012540  017711            MOVB    DIGT2,RSX3
2631  017426  113737  012541  017712            MOVB    DIGT3,RSX4
2632  017434  104401                            TYPE
2633  017436  017700                            RSVMSG
2634  017440  111305                            MOVB    (%3),R5
2635  017442  005305                  2$:       DEC     R5
2636  017444  100403                            BMI     1$
2637  017446  104401                            TYPE
2638  017450  020043                            TYANX
2639  017452  000773                            BR      2$
2640  017454  023737  016322  016342  1$:       CMP     LOLIM1,TEMp
2641  017462  001413                            BEQ     4$
2642  017464  023737  016326  016342            CMP     HILIM1,TEMp
2643  017472  001412                            BEQ     5$
2644  017474  023737  016340  016342            CMP     AMEAN,TEMP
2645  017502  001010                            BNE     3$
2646  017504  104401  017774                    TYPE    ,TYMEAN
2647  017510  000405                            BR      3$
2648  017512  104401  017716                4$: TYPE    ,TYLOW
2649  017516  000402                            BR      3$
2650  017520  104401  017744                5$: TYPE    ,TYHIGH
```

```
2651  017524  005237  016342          3$:     INC     TEMP
2652  017530  105723                          TSTB    (3)+
2653  017532  022737  000145  016342          CMP     #145,TEMP
2654  017540  001300                          BNE     6$
2655  017542  104401                          TYPE
2656  017544  020040                          ACRLF
2657  017546  104401  020020                  TYPE    ,OVERNG
2658  017552  013746  017174                  MOV     EXCESS,-(SP)
2659  017556  104403                          TYPOS
2660  017560     004     001                  .BYTE   4,1
2661  017562  012605                          MOV     (SP)+,R5
2662  017564  000205                          RTS     R5
2663
2664  017566  000000                  12$:    0
2665
2666  017570                          CONVT:
2667
2668                                  ;*      MOV     @VCSTAT,$TMDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
2669  017600  005237  001564                  INC     $TMDAT
2670
2671                                  ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
2672  017614                          2$:
2673
2674                                  ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
2675  017624  105737  001126                  TSTB    $BDDAT
2676  017630  100371                          BPL     2$
2677  017632  005005                          CLR     %5
2678
2679                                  ;*      MOV     DIFCHN,@ADCS    ;/ PUT DATA FROM DIFCHN TO DEVICE REG ADCS
2680  017644                          1$:
2681
2682                                  ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2683  017654  105737  001126                  TSTB    $BDDAT          ;WAIT FOR DONE
2684  017660  100371                          BPL     1$
2685
2686                                  ;*      MOV     @ADDBR,$TMDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2687  017672  013705  001564                  MOV     $TMDAT,R5
2688  017676  000207                          RTS     PC
2689
2690  017700     015     012          RSVMSG: .BYTE   15,12
2691  017702     060     056          RSV1:   .BYTE   60,56
2692  017704     060                  RSV2:   .BYTE   60
2693  017705     060     055          RSV3:   .BYTE   60,55
2694  017707     060                  RSX1:   .BYTE   60
2695  017710     060                  RSX2:   .BYTE   60
2696  017711     060                  RSX3:   .BYTE   60
2697  017712     060     040     111  RSX4:   .BYTE   60,40,111,0
2698  017715     000
2699  017716  026455  026455  026455  TYLOW:  .ASCIZ  "----------  (1/2 LSB)"
2700  017724  026455  026455  020040
2701  017732  030450  031057  046040
2702  017740  041123  000051
2703  017744  026455  026455  026455  TYHIGH: .ASCIZ  "----------  (1 1/2 LSB)"
2704  017752  026455  026455  020040
```

```
 2705  017760  030450  030440  031057
 2706  017766  046040  041123  000051
 2707  017774  026455  026455  026455  TYMEAN: .ASCIZ  "----------  (1 LSB)"
 2708  020002  026455  026455  020040
 2709  020010  030450  046040  041123
 2710  020016  000051
 2711  020020  052517  020124  043117  OVERNG: .ASCIZ  /OUT OF RANGE - /
 2712  020026  051040  047101  042507
 2713  020034  026440  000040
 2714  020040     015     012     000  ACRLF:  .BYTE   15,12,0
 2715  020043     052     000          TYANX:  .BYTE   52,0
 2716  020045     040  020040     000  SP3MSG: .ASCIZ  /   /
 2717  020051     000     000     000          .BYTE   0,0,0
 2718  020054     055     000          TYANXX: .BYTE   55,0
 2719  020056     015     012          STDMSG: .BYTE   15,12
 2720  020060  052123  052101  026505          .ASCII  /STATE-WIDTH DISTRIBUTION/
 2721  020066  044527  052104  020110
 2722  020074  044504  052123  044522
 2723  020102  052502  044524  047117
 2724  020110     015     012          .BYTE   15,12
 2725  020112  044527  052104  026510          .ASCIZ  /WIDTH-NUMBER OF STATES/
 2726  020120  052516  041115  051105
 2727  020126  047440  020106  052123
 2728  020134  052101  051505     000
 2729  020142                       .EVEN
 2730  020142  000000          VCREG1: 0
 2731  020144  000000          VCREG2: 0
 2732  020146  000000          DIFCHN: 0
 2733
 2734                          .SBTTL  PARAMETER ADJUSTMENT ROUTINE
 2735  020150  012701  020266  WFADJ:  MOV     #V1754,R1               ;LOAD PARM. POINTER
 2736  020154  005737  020264          TST     WFTEST                  ;TEST IF OPTION TEST AREA
 2737  020160  001017                  BNE     1$                      ;BR IF IT WAS
 2738  020162  012702  020306          MOV     #VARLT1,R2              ;LOAD "STANDARD" PARM. VALUES
 2739  020166  112737  000062  014274  MOVB    #'2,BASBT1              ;LOAD BIAS MESSAGE
 2740  020174  112737  000060  014276  MOVB    #'0,BASBT2              ;
 2741  020202  112737  000065  014321  MOVB    #'5,BASBT3              ;LOAD BIAS LIMIT
 2742  020210  112737  000060  014322  MOVB    #'0,BASBT3+1
 2743  020216  000416                  BR      2$
 2744  020220  012702  020324  1$:     MOV     #VARLT2,R2              ;LOAD "OPTION TEST AREA" PARM. VALUES
 2745  020224  112737  000061  014274  MOVB    #'1,BASBT1              ;LOAD BIAS MESSAGE
 2746  020232  112737  000070  014276  MOVB    #'8,BASBT2
 2747  020240  112737  000064  014321  MOVB    #'4,BASBT3              ;LOAD BIAS LIMIT
 2748  020246  112737  000064  014322  MOVB    #'4,BASBT3+1
 2749  020254  012221          2$:     MOV     (R2)+,(R1)+             ;LOAD INTO PARM. LIST
 2750  020256  005711                  TST     (R1)                    ;TEST FOR LAST
 2751  020260  100375                  BPL     2$                      ;BR IF NOT
 2752  020262  000207                  RTS     PC                      ;EXIT
 2753
 2754  020264  000000          WFTEST: 0
 2755
 2756  020266  001754          V1754:  1754            ;1760 IF OPTION TEST AREA SELECTED
 2757  020270  000024          V24:    24              ;20     "       "       "       "
 2758  020272  000050          V50:    50              ;40     "       "       "       "
```

```
2759  020274  000024                  VA24:   24              ;22     "       "       "       "
2760  020276  000062                  V62:    62              ;45     "       "       "       "
2761  020300  000077                  V77:    77              ;45     "       "       "       "
2762  020302  000001                  V1:     1               ;15     "       "       "       "
2763  020304  100000                          BIT15           ;TERM.
2764
2765  020306  001754                  VARLT1: 1754
2766  020310  000024                          24
2767  020312  000050                          50
2768  020314  000024                          24
2769  020316  000062                          62
2770  020320  000077                          77
2771  020322  000001                          1
2772
2773  020324  001760                  VARLT2: 1760
2774  020326  000020                          20
2775  020330  000040                          40
2776  020332  000022                          22
2777  020334  000045                          45
2778  020336  000045                          45
2779  020340  000015                          15
2780
2781                                          .SBTTL  SUBROUTINE TO CONVERT R0 TO DECIMAL PERCENTAGE OF 1022.
2782
2783  020342  000000                  MSGPNT: 0
2784  020344  000000                  PCT:    0
2785  020346  012537  020342          PRCNT:  MOV     (R5)+,MSGPNT            ;GET MESSAGE POINTER
2786  020352  010037  020344                  MOV     R0,PCT                  ;
2787  020356  006200                          ASR     R0
2788  020360  006200                          ASR     R0
2789  020362  006200                          ASR     R0
2790  020364  006200                          ASR     R0
2791  020366  006200                          ASR     R0
2792  020370  006200                          ASR     R0
2793  020372  160037  020344                  SUB     R0,PCT
2794  020376  006200                          ASR     R0
2795  020400  160037  020344                  SUB     R0,PCT                  ;PCT CONTAINS OCTAL %
2796  020404  013702  020344                  MOV     PCT,R2                  ;LOAD R2 WITH PERCENTAGE
2797  020410  004737  012416                  JSR     PC,DECPRT               ;CONVERT TO DEC.
2798  020414  013700  020342                  MOV     MSGPNT,R0               ;LOAD MSG. POINTER
2799  020420  022737  001750  020344          CMP     #1000.,PCT              ;TEST FOR 100 %
2800  020426  001411                          BEQ     1$                      ;BR IF 100 %
2801  020430  113720  012537                  MOVB    DIGT1,(R0)+             ;LOAD A DIGIT
2802  020434  113720  012540                  MOVB    DIGT2,(R0)+
2803  020440  112720  000056                  MOVB    #56,(R0)+               ;LOAD "."
2804  020444  113720  012541                  MOVB    DIGT3,(R0)+             ;LOAD 1/10 % DIGIT
2805  020450  000205                          RTS     R5                      ;EXIT
2806
2807  020452  112720  000040          1$:     MOVB    #40,(R0)+               ;LOAD "SPACE"
2808  020456  112720  000061                  MOVB    #'1,(R0)+               ;LOAD '1' 100%
2809  020462  112720  000060                  MOVB    #'0,(R0)+
2810  020466  112720  000060                  MOVB    #'0,(R0)+
2811  020472  000205                          RTS     R5                      ;EXIT
2812                                  .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

020474                          $SCOPE:
020474  104407                         CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
020476  104407                         CKSWR
020500  032777  040000  160432  1$:    BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
020506  001114                         BNE     $OVER           ;;YES IF SW14=1
                                ;#####START OF CODE FOR THE XOR TESTER#####
020510  000416                 $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
020512  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
020516  012737  020536  000004         MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
020524  005737  177060                 TST     @#177060        ;;TIME OUT ON XOR?
020530  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
020534  000463                         BR      $SVLAD          ;;GO TO THE NEXT TEST
020536  022626                 5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
020540  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
020544  000423                         BR      7$              ;;LOOP ON THE PRESENT TEST
020546                          6$:;#####END OF CODE FOR THE XOR TESTER#####
020546  032777  000400  160364         BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
020554  001404                         BEQ     2$              ;;BR IF NO
020556  127737  160356  001102         CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
020564  001465                         BEQ     $OVER           ;;BR IF YES
020566  105737  001103         2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
020572  001421                         BEQ     3$              ;;BR IF NO
020574  123737  001115  001103         CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
020602  101015                         BHI     3$              ;;BR IF NO
020604  032777  001000  160326         BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
020612  001404                         BEQ     4$              ;;BR IF NO
020614  013737  001110  001106  7$:    MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
020622  000446                         BR      $OVER
020624  105037  001103          4$:    CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
020630  005037  001166                 CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
020634  000415                         BR      1$              ;;ESCAPE TO THE NEXT TEST
020636  032777  004000  160274  3$:    BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
020644  001011                         BNE     1$              ;;BR IF YES
020646  005737  001210                 TST     $PASS           ;;IF FIRST PASS OF PROGRAM
020652  001406                         BEQ     1$              ;;      INHIBIT ITERATIONS
020654  005237  001104                 INC     $ICNT           ;;INCREMENT ITERATION COUNT
020660  023737  001166  001104         CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
020666  002024                         BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
020670  012737  000001  001104  1$:    MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
020676  013737  020754  001166         MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
```

```
 2867  020704  105237  001102            $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 2868  020710  113737  001102  001206            MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 2869  020716  011637  001106                    MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 2870  020722  011637  001110                    MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 2871  020726  005037  001170                    CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 2872  020732  112737  000001  001115            MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 2873  020740  013777  001102  160174    $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 2874  020746  013716  001106                    MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 2875  020752  000002                            RTI                     ;;FIXES PS
 2876  020754  000012            $MXCNT: 10.                             ;;MAX. NUMBER OF ITERATIONS
 2877                            .SBTTL  ERROR HANDLER ROUTINE
 2878
 2879                            ;;**********************************************************************
 2880                            ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 2881                            ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 2882                            ;*AND GO TO $ERRTYP ON ERROR
 2883                            ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 2884                            ;*SW15=1         HALT ON ERROR
 2885                            ;*SW13=1         INHIBIT ERROR TYPEOUTS
 2886                            ;*SW10=1         BELL ON ERROR
 2887                            ;*SW09=1         LOOP ON ERROR
 2888                            ;*CALL
 2889                            ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 2890
 2891  020756                    $ERROR:
 2892  020756  104407                    CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
 2893  020760  105237  001103    7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
 2894  020764  001775                    BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
 2895  020766  013777  001102  160146    MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
 2896  020774  032777  002000  160136    BIT     #BIT10,@SWR             ;;BELL ON ERROR?
 2897  021002  001402                    BEQ     1$                      ;;NO - SKIP
 2898  021004  104401  001172            TYPE    ,$BELL                  ;;RING BELL
 2899  021010  005237  001112    1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
 2900  021014  011637  001116            MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
 2901  021020  162737  000002  001116    SUB     #2,$ERRPC
 2902  021026  117737  160064  001114    MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
 2903  021034  032777  020000  160076    BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
 2904  021042  001004                    BNE     20$                     ;;SKIP TYPEOUTS
 2905  021044  004737  022034            JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
 2906  021050  104401  001177            TYPE    ,$CRLF
 2907  021054                    20$:
 2908  021054  122737  000001  001222            CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 2909  021062  001007                    BNE     2$                      ;;NO,SKIP APT ERROR REPORT
 2910  021064  113737  001114  021076            MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
 2911  021072  004737  023136            JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
 2912  021076     000            21$:    .BYTE   0
 2913  021077     000                    .BYTE   0
 2914  021100  000777            22$:    BR      22$                     ;;APT ERROR LOOP
 2915  021102  005777  160032    2$:     TST     @SWR                    ;;HALT ON ERROR
 2916  021106  100002                    BPL     3$                      ;;SKIP IF CONTINUE
 2917  021110  000000                    HALT                            ;;HALT ON ERROR!
 2918  021112  104407                    CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
 2919  021114  032777  001000  160016    BIT     #BIT09,@SWR             ;;LOOP ON ERROR SWITCH SET?
 2920  021122  001402                    BEQ     4$                      ;;BR IF NO
```

```
2921  021124  013716  001110            MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
2922  021130  005737  001170    4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
2923  021134  001402                    BEQ     5$              ;;BR IF NONE
2924  021136  013716  001170            MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
2925  021142                    5$:
2926  021142  022737  011436  000042    CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
2927  021150  001001                    BNE     6$              ;;BRANCH IF NO
2928  021152  000000                    HALT                    ;;YES
2929  021154                    6$:
2930  021154  000002                    RTI                     ;;RETURN
2931                            .SBTTL  TTY INPUT ROUTINE
2932
2933                            ;;**************************************************************
2934                            .ENABL  LSB
2935
2936                            ;;**************************************************************
2937                            ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2938                            ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2939                            ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2940                            ;*WHEN OPERATING IN TTY FLAG MODE.
2941  021156  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
2942  021164  001074                    BNE     15$             ;;BRANCH IF NO
2943  021166  105777  157752            TSTB    @$TKS           ;;CHAR THERE?
2944  021172  100071                    BPL     15$             ;;IF NO, DON'T WAIT AROUND
2945  021174  117746  157746            MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
2946  021200  042716  177600            BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
2947  021204  022726  000007            CMP     #7,(SP)+        ;;IS IT A CONTROL G?
2948  021210  001062                    BNE     15$             ;;NO, RETURN TO USER
2949  021212  123727  001134  000001    CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
2950  021220  001456                    BEQ     15$             ;;BRANCH IF YES
2951
2952  021222  104401  021703            TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
2953  021226  104401  021710    $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
2954  021232  013746  000176            MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
2955  021236  104402                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2956  021240  104401  021721            TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
2957  021244  005046            19$:    CLR     -(SP)           ;;CLEAR COUNTER
2958  021246  005046                    CLR     -(SP)           ;;THE NEW SWR
2959  021250  105777  157670    7$:     TSTB    @$TKS           ;;CHAR THERE?
2960  021254  100375                    BPL     7$              ;;IF NOT TRY AGAIN
2961
2962  021256  117746  157664            MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
2963  021262  042716  177600            BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2964
2965
2966
2967  021266  021627  000025    9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
2968  021272  001005                    BNE     10$             ;;BRANCH IF NOT
2969  021274  104401  021676            TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
2970  021300  062706  000006    20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
2971  021304  000757                    BR      19$             ;;LET'S TRY IT AGAIN
2972
2973
2974  021306  021627  000015    10$:    CMP     (SP),#15        ;;IS IT A <CR>?
```

```
2975   021312  001022                          BNE     16$             ;;BRANCH IF NO
2976   021314  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
2977   021320  001403                          BEQ     11$             ;;BRANCH IF YES
2978   021322  016677  000002  157610          MOV     2(SP),@SWR      ;;SAVE NEW SWR
2979   021330  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
2980   021334  104401  001177          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
2981   021340  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
2982   021346  001003                          BNE     15$             ;;BRANCH IF NOT
2983   021350  012777  000100  157566          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
2984   021356  000002          15$:    RTI                             ;;RETURN
2985   021360  004737  023050          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
2986   021364  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
2987   021370  002420                          BLT     18$             ;;BRANCH IF YES
2988   021372  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
2989   021376  003015                          BGT     18$             ;;BRANCH IF YES
2990   021400  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2991   021404  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
2992   021410  001403                          BEQ     17$             ;;BRANCH IF YES
2993   021412  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
2994   021414  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
2995   021416  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
2996   021420  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
2997   021424  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2998   021430  000707                          BR      7$              ;;GET THE NEXT ONE
2999   021432  104401  001176          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
3000   021436  000720                          BR      20$             ;;SIMULATE CONTROL-U
3001                                   .DSABL  LSB
3002
3003
3004                                   ;;*****************************************************************
3005                                   ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3006                                   ;*CALL:
3007                                   ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
3008                                   ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
3009                                   ;*                              ;;WITH PARITY BIT STRIPPED OFF
3010                                   ;
3011
3012   021440  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
3013   021442  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
3014   021450  105777  157470          1$:     TSTB    @$TKS           ;;WAIT FOR
3015   021454  100375                          BPL     1$              ;;A CHARACTER
3016   021456  117766  157464  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
3017   021464  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
3018   021472  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
3019   021500  001013                          BNE     3$              ;;BRANCH IF NO
3020   021502  105777  157436          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
3021   021506  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
3022   021510  117746  157432                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
3023   021514  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3024   021520  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
3025   021524  001366                          BNE     2$              ;;IF NOT DISCARD IT
3026   021526  000750                          BR      1$              ;;YES, RESUME
3027   021530  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
3028   021536  002407                          BLT     4$              ;;BRANCH IF YES
```

```
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                        ;*CALL:
                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

021560  010346          $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
021562  012703  021666  1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
021566  022703  021676  2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
021572  101405                  BLOS    4$              ;;BR IF YES
021574  104410                  RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
021576  112613                  MOVB    (SP)+,(R3)      ;;GET CHARACTER
021600  122713  000177  10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
021604  001003                  BNE     3$              ;;SKIP IF NOT
021606  104401  001176  4$:     TYPE    ,$QUES          ;;TYPE A '?'
021612  000763                  BR      1$              ;;CLEAR THE BUFFER AND LOOP
021614  111337  021664  3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
021620  104401  021664          TYPE    ,9$
021624  122723  000015          CMPB    #15,(R3)+       ;;CHECK FOR RETURN
021630  001356                  BNE     2$              ;;LOOP IF NOT RETURN
021632  105063  177777          CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
021636  104401  001200          TYPE    ,$LF            ;;TYPE A LINE FEED
021642  012603                  MOV     (SP)+,R3        ;;RESTORE R3
021644  011646                  MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
021646  016666  000004  000002  MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
021654  012766  021666  000004  MOV     #$TTYIN,4(SP)
021662  000002                  RTI                     ;;RETURN
021664     000          9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
021665     000                  .BYTE   0               ;;TERMINATOR
021666  000010          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
021676  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
021703     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
021710  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
021716  020075     000
021721     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
021726  036440  000040

                                        .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

021732  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
021734  016666  000004  000002  MOV     4(SP),2(SP)     ;;INPUT NUMBER
021742  010046                  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.

                                $ERrTYP:
                                        TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
                                        MOV     R0,-(SP)        ;;SAVE R0
                                        CLR     R0              ;;PICKUP THE ITEM INDEX
                                        BISB    @#$ITEMB,R0
                                        BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
                                        MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
                                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                        BR      6$              ;;GET OUT
                                1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
                                        ASL     R0              ;;      WORK FOR THE ERROR TABLE
                                        ASL     R0
                                        ASL     R0
                                        ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
                                        MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
                                        BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
                                        TYPE                    ;;TYPE THE "ERROR MESSAGE"
                                2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
                                        TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
```

```
3137  022114  012037  022124          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
3138  022120  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
3139  022122  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
3140  022124  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
3141  022126  104401  001177                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
3142  022132  011000                  5$:     MOV     @(R0),R0        ;;PICKUP "DATA TABLE" POINTER
3143  022134  001004                          BNE     7$              ;;GO TYPE THE DATA
3144  022136  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
3145  022140  104401  001177                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
3146  022144  000207                          RTS     PC              ;;RETURN
3147  022146                          7$:
3148  022146  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
3149  022150  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3150  022152  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
3151  022154  001770                          BEQ     6$              ;;BR IF NO
3152  022156  104401  022164                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
3153  022162  000771                          BR      7$              ;;LOOP
3154  022164  020040     000          8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
3155  022170                                  .EVEN
3156
3157                                  .SBTTL  POWER DOWN AND UP ROUTINES
3158
3159                                  ;;**************************************************************
3160                                  ;POWER DOWN ROUTINE
3161  022170  012737  022334  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
3162  022176  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
3163  022204  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3164  022206  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3165  022210  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3166  022212  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3167  022214  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3168  022216  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
3169  022220  017746  156714                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
3170  022224  010637  022340                  MOV     SP,$SAVR6       ;;SAVE SP
3171  022230  012737  022242  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
3172  022236  000000                          HALT
3173  022240  000776                          BR      .-2             ;;HANG UP
3174
3175                                  ;;**************************************************************
3176                                  ;POWER UP ROUTINE
3177  022242  012737  022334  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
3178  022250  013706  022340                  MOV     $SAVR6,SP       ;;GET SP
3179  022254  005037  022340                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
3180  022260  005237  022340          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
3181  022264  001375                          BNE     1$              ;;OF   WORD
3182  022266  012677  156646                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
3183  022272  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
3184  022274  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
3185  022276  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3186  022300  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
3187  022302  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3188  022304  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3189  022306  012737  022170  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
3190  022314  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
```

```
                                            TYPE                    ;;REPORT THE POWER FAILURE
                                $PWRMG:     .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
                                            MOV     (PC)+,(SP)      ;;RESTART AT BEGIN
                                $PWRAD:     .WORD   BEGIN           ;;RESTART ADDRESS
                                            RTI
                                $ILLUP:     HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
                                            BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
                                $SAVR6:     0                       ;;PUT THE SP HERE
                                PWRMSG:     .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>

                                            .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

022410  017646  000000          $TYPOS:     MOV     @(SP),-(SP)     ;;PICKUP THE MODE
022414  116637  000001  022633              MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
022422  112637  022635                      MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
022426  062716  000002                      ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
022432  000406                              BR      $TYPON
022434  112737  000001  022633  $TYPOC:     MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
022442  112737  000006  022635              MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
022450  112737  000005  022632  $TYpON:     MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
022456  010346                              MOV     R3,-(SP)        ;;SAVE R3
022460  010446                              MOV     R4,-(SP)        ;;SAVE R4
022462  010546                              MOV     R5,-(SP)        ;;SAVE R5
022464  113704  022635                      MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
```

```
3245  022470  005404                        NEG     R4
3246  022472  062704  000006                ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
3247  022476  110437  022634                MOVB    R4,$OMODE       ;;SAVE IT FOR USE
3248  022502  113704  022633                MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
3249  022506  016605  000012                MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
3250  022512  005003                        CLR     R3              ;;CLEAR THE OUTPUT WORD
3251  022514  006105                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
3252  022516  000404                        BR      3$              ;;GO DO MSB
3253  022520  006105                2$:     ROL     R5              ;;FORM THIS DIGIT
3254  022522  006105                        ROL     R5
3255  022524  006105                        ROL     R5
3256  022526  010503                        MOV     R5,R3
3257  022530  006103                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
3258  022532  105337  022634                DECB    $OMODE          ;;TYPE THIS DIGIT?
3259  022536  100016                        BPL     7$              ;;BR IF NO
3260  022540  042703  177770                BIC     #177770,R3      ;;GET RID OF JUNK
3261  022544  001002                        BNE     4$              ;;TEST FOR 0
3262  022546  005704                        TST     R4              ;;SUPPRESS THIS 0?
3263  022550  001403                        BEQ     5$              ;;BR IF YES
3264  022552  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
3265  022554  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
3266  022560  052703  000040          5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3267  022564  110337  022630                MOVB    R3,8$           ;;SAVE FOR TYPING
3268  022570  104401  022630                TYPE    ,8$             ;;GO TYPE THIS DIGIT
3269  022574  105337  022632          7$:   DECB    $OCNT           ;;COUNT BY 1
3270  022600  003347                        BGT     2$              ;;BR IF MORE TO dO
3271  022602  002402                        BLT     6$              ;;BR IF DONE
3272  022604  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3273  022606  000744                        BR      2$              ;;GO DO THE LAST DIGIT
3274  022610  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
3275  022612  012604                        MOV     (SP)+,R4        ;;RESTORE R4
3276  022614  012603                        MOV     (SP)+,R3        ;;RESTORE R3
3277  022616  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3278  022624  012616                        MOV     (SP)+,(SP)
3279  022626  000002                        RTI                     ;;RETURN
3280  022630     000                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3281  022631     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3282  022632     000                $OCNT:   .BYTE   0               ;;OCTAL DIGIT COUNTER
3283  022633     000                $OFILL:  .BYTE   0               ;;ZERO FILL SWITCH
3284  022634  000000                $OMODE:  .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3285                                .SBTTL  TYPE ROUTINE
3286
3287                                ;;**************************************************************
3288                                ;*ROUTINE TO tYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3289                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3290                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3291                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3292                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3293                                ;*
3294                                ;*CALL:
3295                                ;*1) USING A TRAP INSTRUCTION
3296                                ;*       TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3297                                ;*OR
3298                                ;*       TYPE
```

```
                                    ;*      MESADR
                                    ;*
                                    
022636  105737  001157      $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERmINAL?
022642  100002                      BPL     1$              ;;BR IF YES
022644  000000                      HALT                    ;;HALT HERE IF NO TERmINAL
022646  000430                      BR      3$              ;;LEAVE
022650  010046              1$:     MOV     R0,-(SP)        ;;SAVE R0
022652  017600  000002              MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
022656  122737  000001  001222      CMPB    #APTENV,$ENv    ;;RUNNING IN APT MODE
022664  001011                      BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
022666  132737  000100  001223      BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
022674  001405                      BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
022676  010037  022706              MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
023702  004737  023126              JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
022706  000000              61$:    .WORD   0               ;;MESSAGE ADDRESS
022710  132737  000040  001223  62$:  BITB    #APTCSUP,$ENvM  ;;APT CONSOLE SUPPRESSED
022716  001003                      BNE     60$             ;;YES,SKIP TYPE OUT
022720  112046              2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
022722  001005                      BNE     4$              ;;BR IF IT ISN'T THE TERmINATOR
022724  005726                      TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
022726  012600              60$:    MOV     (SP)+,R0        ;;RESTORE R0
022730  062716  000002      3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
022734  000002                      RTI                     ;;RETURN
022736  122716  000011      4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
022742  001430                      BEQ     8$
022744  122716  000200              CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
022750  001006                      BNE     5$
022752  005726                      TST     (SP)+           ;;POP  <CR><LF> EQUIV
022754  104401                      TYPE                    ;;TYPE A CR AND LF
022756  001177                      $CRLF
022760  105037  023114              CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
022764  000755                      BR      2$              ;;GET NEXT CHARACTER
022766  004737  023050      5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
022772  123726  001156      6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
022776  001350                      BNE     2$              ;;IF NO GO GET NEXT CHAR.
023000  013746  001154              MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                            ;;AND THE NULL CHAR.
023004  105366  000001      7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
023010  002770                      BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
023012  004737  023050              JSR     PC,$TYPEC       ;;GO TYPE A NULL
023016  105337  023114              DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
023022  000770                      BR      7$              ;;LOOP

                                    ;HORIZONTAL TAB PROCESSOR

023024  112716  000040      8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
023030  004737  023050      9$:     JSR     PC,$TYpEC       ;;TYPE A SPACE
023034  132737  000007  023114      BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
023042  001372                      BNE     9$              ;;TAB STOP
023044  005726                      TST     (SP)+           ;;POP SPACE OFF STACK
023046  000724                      BR      2$              ;;GET NEXT CHARACTER
023050  105777  156074      $TYpEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
023054  100375                      BPL     $TYPEC
```

```
 3353  023056  116677  000002  156066          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 3354  023064  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 3355  023072  001003                          BNE     1$              ;;BRANCH IF NO
 3356  023074  105037  023114                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 3357  023100  000406                          BR      $TYPEX          ;;EXIT
 3358  023102  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 3359  023110  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
 3360  023112  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
 3361  023114  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
 3362  023116  000207                  $TYPEX: RTS     PC
 3363
 3364                                  .SBTTL  APT COMMUNICATIONS ROUTINE
 3365
 3366                                  ;;**************************************************************
 3367  023120  112737  000001  023364  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
 3368  023126  112737  000001  023362  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
 3369  023134  000403                          BR      $ATYC
 3370  023136  112737  000001  023364  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
 3371  023144                          $ATYC:
 3372  023144  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 3373  023146  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 3374  023150  105737  023362                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
 3375  023154  001450                          BEQ     5$              ;;IF NOT:  BR
 3376  023156  122737  000001  001222          CMPB    #APTENV,$ENv    ;;OPERATING UNDER APT?
 3377  023164  001031                          BNE     3$              ;;IF NOT:  BR
 3378  023166  132737  000100  001223          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
 3379  023174  001425                          BEQ     3$              ;;IF NOT:  BR
 3380  023176  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
 3381  023202  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 3382  023210  005737  001202          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
 3383  023214  001375                          BNE     1$              ;;IF NOT:  WAIT
 3384  023216  010037  001216                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
 3385  023222  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
 3386  023224  001376                          BNE     2$
 3387  023226  163700  001216                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
 3388  023232  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
 3389  023234  010037  001220                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
 3390  023240  012737  000004  001202          MOV     #4,$MSgTYPE     ;;TELL APT TO tAKE MSG.
 3391  023246  000413                          BR      5$
 3392  023250  017637  000004  023274  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
 3393  023256  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
 3394  023264  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
 3395  023270  004737  022636                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
 3396  023274  000000                  4$:     .WORD   0
 3397  023276                          5$:
 3398  023276  105737  023364          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
 3399  023302  001416                          BEQ     12$             ;;IF NOT:  BR
 3400  023304  005737  001222                  TST     $ENV            ;;RUNNING UNDER APT?
 3401  023310  001413                          BEQ     12$             ;;IF NOT:  BR
 3402  023312  005737  001202          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
 3403  023316  001375                          BNE     11$             ;;IF NOT:  WAIT
 3404  023320  017637  000004  001204          MOV     @4(SP),$FATAL   ;;GET ERROR #
 3405  023326  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 3406  023334  005237  001202                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
```

```
3407  023340  105037  023364          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
3408  023344  105037  023363                  CLRB    $LFLG           ;;CLEAR LOG FLAG
3409  023350  105037  023362                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
3410  023354  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3411  023356  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3412  023360  000207                          RTS     PC              ;;RETURN
3413  023362     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
3414  023363     000          $LFLG:  .BYTE   0               ;;LOG FLAG
3415  023364     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
3416        023366                  .EVEN
3417        000200          APTSIZE=200
3418        000001          APTENV=001
3419        000100          APTSPOOL=100
3420        000040          APTCSUP=040
3421                        .SBTTL  TRAP DECODER
3422
3423                        ;;**************************************************************
3424                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3425                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3426                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3427                        ;*GO TO THAT ROUTINE.
3428
3429  023366  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
3430  023370  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
3431  023374  005740                  TST     -(R0)           ;;BACKUP BY 2
3432  023376  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
3433  023400  006300                  ASL     R0              ;;POSITION FOR INDEXING
3434  023402  016000  023422          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
3435  023406  000200                  RTS     R0              ;;GO TO ROUTINE
3436
3437
3438                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3439
3440  023410  011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWn
3441  023412  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
3442  023420  000002                  RTI                     ;;RESTORE THE PSW
3443
3444                        .SBTTL  TRAP TABLE
3445
3446                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3447                        ;*BY THE "TRAP" INSTRUCTION.
3448
3449                        ;       ROUTINE
3450                        ;       -------
3451  023422  023410  $TRPAD: .WORD   $TRAP2
3452  023424  022636          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
3453  023426  022434          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3454  023430  022410          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3455  023432  022450          $TYpON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3456  023434  011472          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
3457
3458  023436  021226          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
3459
3460  023440  021156          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
```

```
3461  023442  021440                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
3462  023444  021560                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
3463  023446  021732                          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3464
3465                                          ;*
3466                                          ;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
3467                                          ;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
3468                                          ;*NEXT WE WILL INIT BOTH UPROCESSORS
3469                                          ;*THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
3470                                          ;*THE ORDER OF LOAD IS DETERMINED BY THE USER.
3471                                          ;*
3472                                          ;*      CALL=   JSR     R5,$LPAI
3473                                          ;*              .WORD   0               ;ADDR. OF DEVICE ADDRESS.
3474                                          ;* ROUTINES REQUIRED:   .LOADLP
3475                                          ;* PROGRAMS REQUIRED:   DRLPX2
3476                                          ;*
3477
3478                                          ;*              ;RETURNS WITH $AERR=1 IF SLAVE
3479                                          ;*              ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
3480                                          ;*
3481  023450                          $LPAI:
3482  023450  013746  000004                  MOV     4,-(SP)
3483
3484  023454  000413                          BR      31$             ;FIELD DOEs NOT HAVE A BUS SWITCH TO
3485                                                                  ;WORRY ABOUT,SO WE WILL UNCONDITIONALLY
3486                                                                  ;BRANCH ARROUD THE NEXT CODE THAT
3487                                                                  ;WORKS BASED ON A BUS SWITCH.
3488                                                                  ;CODE LEFT IN HERE FOR IN HOUSE
3489                                                                  ;PERSONAL WHO MAY PATCH THIS BRANCH
3490                                                                  ;INSTRUCTION TO A <NOP> OCTAL <240>
3491                                                                  ;IN ORDER TO RUN PROGRAM WITH A SWITCH.
3492
3493                                                                  ;NOTE THIS "SWITCH" IS A PIECE OF INHOUSE
3494                                                                  ;TEST EQUIPMENT ONLY IT CONNECTS
3495                                                                  ;THE UNIBUS TO THE I/O BUS FOR
3496                                                                  ;CERTAIN TESTING.
3497  023456  012737  023502  000004          MOV     #30$,4
3498  023464  005237  170000                  INC     170000
3499  023470  104401  023476                  TYPE    ,65$            ;;TYPE ASCIZ STRING
3500  023474  000401                          BR      64$             ;;GET OVER THE ASCIZ
3501                                  ;;65$:  .ASCIZ  <7>##
3502  023500                          64$:
3503  023500  000401                          BR      31$
3504  023502  022626                  30$:    CMP     (SP)+,(SP)+
3505  023504  012637  000004          31$:    MOV     (SP)+,4         ;ALL THIS JUNK MUST BE REMOVED!!
3506  023510  005037  024326                  CLR     $AERR
3507  023514  004537  024330                  JSR     R5,$LOAD        ;LOAD MICRO-CODE.
3508  023520  000000G                         .WORD   DRLPX2          ;FILE "DRLPX2.OBJ"
3509
3510  023522  052777  040000  155736          BIS     #BIT14,@KMADO   ;ISSUE KMC+DMC INIT.
3511
3512  023530                          1$:
3513                                                                  ;"HANGS" HERE THEN KMC-11 ERROR.
3514  023530  010146                          MOV     R1,-(SP)
```

```
3515  023532  005001                        CLR     R1
3516  023534  005201                2$:     INC     R1                ;STALL FOR DMC-UP
3517  023536  001376                        BNE     2$
3518  023540  012777  104000  155720        MOV     #BIT15!BIT11,@KMADO     ;SET RUN, AND ENABLE ARBITRATION.
3519  023546  105201                25$:    INCB    R1
3520  023550  001376                        BNE     25$
3521
3522  023552  032777  000040  155706        BIT     #BIT5,@KMADO      ;SLAVE READY? (READING IPBM SR)
3523  023560  001401                        BEQ     3$
3524                                                                  ;FATAL LPA-11 ERROR SLAVE NOT READY.
3525  023562  104000                        ERROR
3526
3527  023564  012777  000004  155700  3$:   MOV     #4,@KMAD2         ;READ FAST PATH
3528  023572                        4$:
3529  023572  004537  025240                JSR     R5, $TOUT         ;-TOUT-CHECK FOR TIMEOUT
3530
3531  023576  104000                        ERROR                     ;/TIME-OUT ERROR
3532                                                                  ;/WE FAILED TO COMPLETE
3533                                                                  ;/CURRENT OPERATION.
3534                                                                  ;/CONTINUES IN THIS LOOP
3535                                                                  ;/WOULD MAKE US "HANG" HERE
3536
3537  023600  000774                        BR              4$
3538
3539                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3540  023602  122777  000377  155662        CMPB    #377,@KMAD2       ;WAIT TILL KMC DONE COMMAND.
3541  023610  001370                        BNE     4$
3542  023612  122777  000377  155656        CMPB    #377,@KMAD4       ;IF FAST PATH=377 THEN ERROR.
3543  023620  001001                        BNE     35$
3544  023622  104000                        ERROR                     ;IPBM ERROR (SLAVE SIDE)
3545                                                                  ;YOU MUST RUN IPBM DIAGNOSTIC.
3546
3547  023624  122777  000004  155644  35$:   CMPB    #4,@KMAD4         ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
3548  023632  001543                        BEQ     5$                ;YES-CONTINUE.
3549  023634  005227  177777                INC     #-1
3550  023640  001140                        BNE     5$
3551  023642  005227  177777                INC     #-1
3552  023646  001135                        BNE     5$
3553  023650  104401  023656                TYPE    ,67$              ;;TYPE ASCIZ STRING
3554  023654  000440                        BR      66$               ;;GET OVER THE ASCIZ
3555                                ;;67$:  .ASCIZ  <200>"W A R N I N G  THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
3556  023756                        66$:
3557  023756  104401  023764                TYPE    ,69$              ;;TYPE ASCIZ STRING
3558  023762  000430                        BR      68$               ;;GET OVER THE ASCIZ
3559                                ;;69$:  .ASCIZ  <200>"MICRO-CODE ANOTHER VERSION CODE WAS DETECTED."
3560  024044                        68$:
3561  024044  104401  024052                TYPE    ,71$              ;;TYPE ASCIZ STRING
3562  024050  000434                        BR      70$               ;;GET OVER THE ASCIZ
3563                                ;;71$:  .ASCIZ  <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
3564  024142                        70$:
3565
3566  024142  112737  177777  024274  5$:    MOVB    #0-1,11$          ;DAC CODE FOR SLAVE.
3567  024150  012501                        MOV     (5)+,R1           ;GET NEXT DEVICE ADDR.
3568  024152  021127  000000                6$:     CMP     (R1),#0           ;TERM REACHED?
```

```
3569  024156  001444                        BEQ     10$
3570  024160  105237  024274                INCB    11$
3571  024164  113777  024274  155304        MOVB    11$,@KMAD4      ;FIFO DATA
3572  024172  004737  024276                JSR     PC,20$          ;ISSUE SEND
3573  024176  112177  155274                MOVB    (R1)+,@KMAD4    ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
3574  024202  004737  024276                JSR     PC,20$          ;ISSUE SEND
3575  024206  112177  155264                MOVB    (R1)+,@KMAD4    ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
3576  024212  004737  024276                JSR     PC,20$
3577
3578  024216  032777  000002  155242  7$:   BIT     #BIT1,@KMAD0    ;WAIT FOR FIFO DATA
3579  024224  001374                        BNE     7$              ;=1 NO DATA. =0 DATA.
3580  024226  112777  000002  155236        MOVB    #2,@KMAD2       ;READ FIFO.
3581
3582  024234                          8$:
3583  024234  004537  025240                JSR     R5, $TOUt       ;-TOUT-CHECK FOR TIMEOUT
3584
3585  024240  104000                        ERROR                   ;/TIME-OUT ERROR
3586                                                                ;/WE FAILED TO COMPLETE
3587                                                                ;/CURRENT OPERATION.
3588                                                                ;/CONTINUES IN THIS LOOP
3589                                                                ;/WOULD MAKE US "HANG" HERE
3590
3591  024242  000774                        BR              8$
3592
3593                                                                ;/RETURNS HERE-FROM-TIMED OUT.
3594  024244  122777  000377  155220        CMPB    #377,@KMAD2     ;WAIT FOR READ.
3595  024252  001370                        BNE     8$
3596  024254  105777  155216                TSTB    @KMAD4          ;WAS A ZERO RETURNED?
3597  024260  001734                        BEQ     6$              ;YES GET NEXT ADDR.
3598                                                                ;SLAVE WILL RETURN CODE 0 IF
3599  024262  005237  024326                INC     $AERR           ;DEV PRESENT.   ELSE
3600                                                                ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
3601  024266  005041                        CLR     -(1)            ;GET RID OF REFERENCE TO BAD ADDR.
3602  024270  012601                  10$:  MOV     (SP)+,R1
3603  024272  000205                        RTS     R5              ;RETURN ALL ADDR. CHECKED.
3604
3605  024274  000000                  11$:  .WORD   0               ;HOLDS DAC CODE PLUS OFFSET
3606                                                                ;TO SLAVES ADDR. TABLE.
3607
3608  024276  112777  000003  155166  20$:  MOVB    #3,@KMAD2       ;ISSUE FIFO WRITE
3609  024304                          21$:
3610  024304  004537  025240                JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
3611
3612  024310  104000                        ERROR                   ;/TIME-OUT ERROR
3613                                                                ;/WE FAILED TO COMPLETE
3614                                                                ;/CURRENT OPERATION.
3615                                                                ;/CONTINUES IN THIS LOOP
3616                                                                ;/WOULD MAKE US "HANG" HERE
3617
3618  024312  000774                        BR              21$
3619
3620                                                                ;/RETURNS HERE-FROM-TIMED OUT.
3621  024314  122777  000377  155150        CMPB    #377,@KMAD2     ;KMC CODE WILL RETURN A "377"
3622  024322  001370                        BNE     21$             ;WHEN DONE COMMAND.
```

```
3623  024324  000207                          RTS     PC
3624
3625  024326  000000                  $AERR:  .WORD   0               ;=0 IF ADDR. LIST OK,=1 IF BAD.
3626
3627                                          ;*
3628                                          ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
3629                                          ;*      CALL =  JSR     R5,$LOAD
3630                                          ;*              .WORD   XX              ;ADDR. OF MICRO CODE.
3631                                          ;*              ;RETURNS HERE
3632                                          ;*      NOTE:   MICRO CODE FILE MUST END IN -1 DATA.
3633                                          ;*
3634
3635  024330  010446                  $LOAD:  MOV     R4,-(SP)        ;SAVE R4.
3636  024332  010046                          MOV     R0,-(SP)        ;SAVE R0.
3637  024334  012500                  1$:     MOV     (5)+,R0         ;GET PROG. ADDR.
3638  024336  005077  155124                  CLR     @KMAD0          ;CLEAR CSR
3639  024342  005077  155130                  CLR     @KMAD4          ;CLEAR CRAM ADDR.
3640  024346  052777  002000  155112  2$:     BIS     #2000,@KMAD0    ;SELECT CRAM.
3641  024354  012077  155122                  MOV     (0)+,@KMAD6     ;WRITE DATA.
3642  024360  052777  020000  155100          BIS     #20000,@KMAD0   ;SET CRAM WRITE
3643  024366  005077  155074                  CLR     @KMAD0          ;DISABLE CRAM.
3644  024372  005277  155100                  INC     @KMAD4          ;UPDATE CRAM ADDR.
3645  024376  021027  177777                  CMP     (0),#-1         ;ALL DONe?
3646  024402  001361                          BNE     2$              ;NO LOOP.
3647  024404  005077  155066                  CLR     @KMAD4          ;CLEAR CRAM ADDR.
3648  024410  016500  177776                  MOV     -2(5),R0        ;GET MICRO CODE ADDR.
3649
3650  024414  052777  002000  155044  3$:     BIS     #2000,@KMAD0    ;SELECT CRAM
3651  024422  022077  155054                  CMP     (R0)+,@KMAD6    ;DATA OK?
3652  024426  001013                          BNE     5$              ;NO - REPORT AN ERROR.
3653  024430  021027  177777                  CMP     (0),#-1         ;ALL DONE?
3654  024434  001405                          BEQ     4$              ;YES - EXIT
3655  024436  005077  155024                  CLR     @KMAD0          ;NO - DESELECT CRAM.
3656  024442  005277  155030                  INC     @KMAD4          ;UPDATE CRAM ADDR.
3657  024446  000762                          BR      3$
3658
3659  024450  012600                  4$:     MOV     (SP)+,R0        ;RESTORE R0
3660  024452  012604                          MOV     (SP)+,R4        ;RESTORE R4
3661  024454  000205                          RTS     R5              ;EXIT
3662
3663  024456                          5$:                             ;COME HERE ON LOAD ERROR
3664  024456  005745                          TST     -(5)
3665  024460  105204                          INCB    R4              ;UPDATE ERROR COUNTER.
3666  024462  100324                          BPL     1$              ;IF NOT TOO MANY, TRY AGAIN.
3667  024464  000000                          HALT                    ;MICRO CODE LOAD ERROR.
3668                                                                  ;KMC-11 FAULT. YOU COULD TRY
3669  024466  000722                          BR      1$              ;TO PRESS CONTINUE TO GIVE IT
3670                                                                  ;ANOTHER CHANCE, BUT I DOUBT
3671                                                                  ;THAT THAT WOULD WORK. SINCE I'VE
3672                                                                  ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
3673                                                                  ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
3674
3675
3676                                          ;
```

```
3677                                            ;*THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
3678                                            ;*
3679                                            ;*      CALL =  JSR     R5,$TLKW
3680                                            ;*              .WORD   0               ;OFFSET OF DEVICE ADDR.
3681                                            ;*              .WORD   0               ;DATA TO BE WRITTEN
3682                                            ;*
3683  024470  010046                   $TLKW:  MOV     R0,-(SP)        ;SAVE R0
3684  024472  012500                           MOV     (5)+,R0         ;GET DEVICE OFFSET
3685  024474  052700  000340                   BIS     #340,R0         ;ADD WRITE CODE.
3686  024500  004737  024752                   JSR     PC,$LPW         ;WAIT FOR FAST PATH READY
3687  024504  010037  024576                   MOV     R0,W1
3688  024510  010077  154762                   MOV     R0,@KMAD4
3689  024514  112777  000005  154750         MOVB    #5,@KMAD2       ;ISSUE FAST PATH WRITE
3690  024522  004737  024752                   JSR     PC,$LPW         ;WAIT FOR RDY
3691  024526  011537  024600                   MOV     (5),W2
3692  024532  112577  154740                   MOVB    (5)+,@KMAD4     ;WRITE LOW BYTE DATA.
3693
3694  024536  112777  000005  154726         MOVB    #5,@KMAD2       ;FP WRITE
3695  024544  004737  024752                   JSR     PC,$LPW
3696  024550  111537  024602                   MOVB    (5),W3
3697  024554  112577  154716                   MOVB    (5)+,@KMAD4     ;WRITE HIGH BYTE
3698  024560  112777  000005  154704         MOVB    #5,@KMAD2
3699  024566  004737  024752                   JSR     PC,$LPW
3700  024572  012600                           MOV     (SP)+,R0
3701  024574  000205                           RTS     R5              ;EXIT DONE.
3702  024576  000000                   W1:     0
3703  024600  000000                   W2:     0
3704  024602  000000                   W3:     0
3705
3706                                            ;*
3707                                            ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
3708                                            ;*
3709                                            ;*      CALL =  JSR     R5,$TLKR
3710                                            ;*              .WORD   0               ;OFFSET OF DEVICE
3711                                            ;*              ;RETURNS HERE
3712                                            ;*DATA IN WORD $DATR
3713                                            ;*
3714
3715  024604  010046                   $TLKR:  MOV     R0,-(SP)        ;SAVE R0
3716  024606  012500                           MOV     (5)+,R0         ;GET OFFSET
3717  024610  052700  000300                   BIS     #300,R0         ;ADD READ CODE
3718  024614  004737  024752                   JSR     PC,$LPW         ;WAIT TILL READY
3719  024620  110077  154652                   MOVB    R0,@KMAD4
3720  024624  112777  000005  154640         MOVB    #5,@KMAD2       ;ISSUE WRITE FP
3721  024632  004737  024752                   JSR     PC,$LPW
3722  024636  010037  024746                   MOV     R0,RD1
3723  024642                           1$:
3724  024642  004537  025240                   JSR     R5, $TOuT       ;-TOUT-CHECK FOR TIMEOUT
3725
3726  024646  104000                           ERROR                   ;/TIME-OUT ERROR
3727                                            ;/WE FAILED TO COMPLETE
3728                                            ;/CURRENT OPERATION.
3729                                            ;/CONTINUES IN THIS LOOP
3730                                            ;/WOULD MAKE US "HANG" HERE
```

```
3731
3732  024650  000774                          BR          1$
3733
3734                                                      ;/RETURNS HERE-FROM-TIMED OUT.
3735  024652  032777  000040  154606          BIT     #BIT5,@KMADO    ;FAST PATH GOT DATA?
3736  024660  001370                          BNE     1$
3737  024662  112777  000004  154602          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
3738  024670  004737  024752                  JSR     PC,$LPW
3739  024674  117737  154576  024750          MOVB    @KMAD4,$DATR    ;GET LOW BYTE
3740  024702                          2$:
3741  024702  004537  025240                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
3742
3743  024706  104000                          ERROR                   ;/TIME-OUT ERROR
3744                                                                  ;/WE FAILED TO COMPLETE
3745                                                                  ;/CURRENT OPERATION.
3746                                                                  ;/CONTINUES IN THIS LOOP
3747                                                                  ;/WOULD MAKE US "HANG" HERE
3748
3749  024710  000774                          BR          2$
3750
3751                                                      ;/RETURNS HERE-FROM-TIMED OUT.
3752  024712  032777  000040  154546          BIT     #BIT5,@KMADO    ;FAST PATH READY?
3753  024720  001370                          BNE     2$
3754  024722  112777  000004  154542          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
3755  024730  004737  024752                  JSR     PC,$LPW
3756  024734  117737  154536  024751          MOVB    @KMAD4,$DATR+1  ;SAVE HIGH BYTE
3757  024742  012600                          MOV     (SP)+,R0
3758  024744  000205                          RTS     R5
3759  024746  000000                  RD1:    0
3760  024750  000000                  $DATR:  .WORD   0
3761                                          ;
3762                                          ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
3763                                          ;AS FAST PATH TO BE READ.
3764                                          ;
3765                                          ;       CALL =  JSR     PC,$LPw
3766                                          ;
3767                                          ;IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
3768                                          ;THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
3769                                          ;
3770
3771  024752  010146                  $LPW:   MOV     R1,-(SP)        ;SAVE R1
3772  024754  005001                          CLR     R1
3773  024756  122777  000377  154506  1$:     CMPB    #377,@KMAD2     ;FINISHED INSTRUCTION?
3774  024764  001403                          BEQ     2$
3775  024766  005201                          INC     R1              ;TIME OUT?
3776  024770  001372                          BNE     1$
3777  024772  000411                          BR      10$
3778
3779  024774  032777  000020  154464  2$:     BIT     #BIT4,@KMADO    ;FAST PATH READ?
3780  025002  001403                          BEQ     3$
3781  025004  005201                          INC     R1              ;NO - TIME OUT?
3782  025006  001372                          BNE     2$
3783  025010  000402                          BR      10$             ;YES - REPORT AN ERROR
3784
```

```
3785  025012  012601                  3$:     MOV     (SP)+,R1        ;RESTORE R1
3786  025014  000207                          RTS     PC              ;EXIT
3787
3788  025016                          10$:
3789  025016  104401  025024                  TYPE    ,65$            ;;TYPE ASCIZ STRING
3790  025022  000407                          BR      64$             ;;GET OVER THE ASCIZ
3791                                  ;;65$:  .ASCIZ  <200>#LPA-11 FAULT#
3792  025042                          64$:
3793
3794  025042  000000                  11$:    HALT                    ;LPA-11 FAULT RUN LPA-11
3795  025044  000776                          BR      11$             ;DIAGNOSTICS.
3796
3797
3798
3799                                          ;*
3800                                          ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
3801                                          ;*A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.
3802                                          ;*
3803                                          ;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED
3804                                          ;* BEFORE. IF NOT WE HAVE TO INITIALIZE THE LPA WITH
3805                                          ;* THAT ADDRESS.
3806                                          ;* WHEN THE ADDR. IS KNOWN BY THE LPA, DO THE OUTPUT BY
3807                                          ;* $TLKW
3808                                          ;*
3809
3810  025046  010046                  $OUTLP: MOV     R0,-(SP)        ;SAVE R0
3811  025050  010146                          MOV     R1,-(SP)        ;SAVE R1
3812
3813  025052  012700  001514                  MOV     #.DVLS,R0       ;PROGRAM DEFINED LIST.
3814  025056  005001                          CLR     R1
3815  025060  005710                  1$:     TST     (0)             ;TERMINATOR REACHED?
3816  025062  001421                          BEQ     10$             ;YES NEXT STEP.
3817  025064  027520  000000                  CMP     @(5),(0)+       ;MATCH WITH ADDR IN LIST?
3818  025070  001402                          BEQ     2$
3819  025072  005201                          INC     R1
3820  025074  000771                          BR      1$
3821
3822  025076  010137  025114          2$:     MOV     R1,3$           ;SAVE OFFSET, DEVICE KNOWN.
3823  025102  005725                          TST     (5)+
3824  025104  013537  025116                  MOV     @(5)+,4$        ;GET DATA TO bE WRITTEN
3825  025110  004537  024470                  JSR     R5,$TLKW        ;DO WRITE
3826  025114  000000                  3$:     .WORD   0               ;DEVICE OFFSET
3827  025116  000000                  4$:     .WORD   0               ;DATA TO BE WRITTEN.
3828  025120  012601                          MOV     (SP)+,R1
3829  025122  012600                          MOV     (SP)+,R0
3830  025124  000205                          RTS     R5
3831  025126  017520  000000          10$:    MOV     @(5),(0)+       ;SAVE ADDR.
3832  025132  005010                          CLR     (0)
3833  025134  004537  023450                  JSR     R5,$LPAI
3834  025140  001514                          .WORD   .DVLS
3835  025142  000755                          BR      2$
3836
3837                                          ;*
3838                                          ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE
```

```
                                        ;*TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
                                        ;*
                                        ;*FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN
                                        ;*USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
                                        ;*WITH THE NEW ADDR.
                                        ;*WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH
                                        ;*$TLKR
                                        ;*      CALL THROUGH    MOVEI   DATA,ADDR.
                                        ;*              WHICH EQUALS:
                                        ;*                      JSR     R5,$INLP
                                        ;*                      .WORD   XX      ADDR OF DEVICE
                                        ;*                      .WORD   YY      ADDR TO STORE READ DATA.

025144  010046          $INLP:  MOV     R0,-(SP)        ;SAVE R0
025146  010146                  MOV     R1,-(SP)        ;SAVE R1

025150  012700  001514          MOV     #.DVLS,R0       ;PROG DEFINED ADDR. LIST.
025154  005001                  CLR     R1
025156  005710          1$:     TST     (0)             ;EOL REACHED?
025160  001420                  BEQ     10$             ;YES - DEFINE NEW ADDR.

025162  027520  000000          CMP     @(5),(0)+       ;ADDR. MATCH?
025166  001402                  BEQ     2$
025170  005201                  INC     R1
025172  000771                  BR      1$

025174  010137  025206  2$:     MOV     R1,3$           ;SAVE LIST OFFSET
025200  005725                  TST     (5)+
025202  004537  024604          JSR     R5,$TLKr        ;GO READ DEVICE
        025206  $OFS=.
025206  000000          3$:     .WORD   0               ;OFFSET OF DEVICE

025210  013735  024750          MOV     $DATR,@(5)+     ;STORE DATA.
025214  012601                  MOV     (SP)+,R1        ;RESTORE R1
025216  012600                  MOV     (SP)+,R0        ;RESTORE R2
025220  000205                  RTS     R5              ;EXIT

025222  017520  000000  10$:     MOV     @(5),(0)+
025226  005010                  CLR     (0)
025230  004537  023450          JSR     R5,$LPAI
025234  001514                  .WORD   .DVLS
025236  000756                  BR      2$
                                        ;*
                                        ;*$TOUT ROUTINE USED TO WATCH IF
                                        ;*      WE'RE IN A LOOP TOO-LONG
                                        ;*      CALL=   JSR R5, $TOUT
                                        ;*              ERROR X     ;RETURNS HERE ON TIMEOUT
                                        ;*              BR
                                        ;*              ;RETURNS HERE NO ERROR
                                        ;*

025240  020537  025274  $TOUt:   CMP     R5,$SAD         ;SAME ADDR?
025244  001405                  BEQ     1$
025246  010537  025274          MOV     R5,$SAD         ;NO-SAVE THIS ADDR.
```

```
3893  025252  005037  025276          CLR     $CNT            ;CLR CNT AT ADDR.
3894  025256  000403                  BR      2$
3895  025260  005237  025276    1$:   INC     $CNT            ;OVERFLOW?
3896  025264  100402                  BMI     3$              ;YES-ERROR RETURN
3897  025266  062705  000004    2$:   ADD     #4,R5           ;NO-NON ERROR RETURN
3898  025272  000205              3$:   RTS     R5              ;RETURN.
3899
3900  025274  000000          $SAD:   .WORD   0               ;CONTAINS LOOP ADDR.
3901  025276  000000          $CNT:   .WORD   0               ;# OF TIMES AT ADDR.
3902
3903                                  ;*
3904                                  ;* THIS ROUTINE REPLACES WHAT THE USER WOULD ORDINARILY
3905                                  ;*USE FOR A RESET.  FIRST WE DO A RESET INSTRUCTION.
3906                                  ;*THEN WE CLR ".DVLST" WHICH FORCES US TO RESET BOTH THE
3907                                  ;*KMC AND DMC AS SOON AS A DEVICE IS REFERENCED.
3908                                  ;*
3909                                  ;*      CALL=JSR        PC,$RESET       ;REPLACES "RESET INSTRUCTION
3910                                  ;*              ;RETURNS HERE.
3911                                  ;*
3912  025300  000005          $RESET: RESET                   ;RESET THE WORLD.
3913
3914                          ;*      MOV     @2$,1$  ;/READ DEVICE REG 2$,PUT DATA IN 1$.
3915  025312  005737  024326          TST     $AERR           ;IF NO ERROR,LOOP
3916  025316  001004                  BNE     10$             ;THERE WAS AN ERROR.
3917  025320  062737  000002  025334  ADD     #2,2$           ;UPDATE DEVICE ADDR.
3918                                                          ;YOU SEE ,WE HAVE TO PROTECT OUR SELF!
3919                                                          ;IF 2$ CONTAINED A VALID ADDR,WE
3920                                                          ;MUST KEEP TRYING UNTIL WE GENERATE
3921                                                          ;AN INVALID ADDR.
3922  025326  000764                  BR      $RESET
3923  025330                  10$:
3924  025330  000207                  RTS     PC
3925  025332  000000          1$:     .WORD   0               ;JUNK LOC.
3926  025334  160000          2$:     .WORD   160000          ;DUMB ADDR. FORCES INIT OF DMC/KMC.
3927
3928
3929                                  ;
3930                                  ;SDELAY- ROUTINE TO GIVE A MINOR DELAY.
3931                                  ;        IS NOT TIME DEPENDENT CODE SENCE
3932                                  ;        NOT USED TO GET SPECIFIC TIME BUT
3933                                  ;        JUST A LITTLE DELAY.
3934                                  ;
3935                                  ;        THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
3936                                  ;        THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
3937                                  ;
3938                                  ;
3939                                  ;        CALL=   JSR PC,  SDELAY
3940                                  ;
3941  025336                  SDELAY:
3942  025336  005737  025420          TST     RTCCSR          ;CLOCK PRESENT?
3943  025342  100016                  BPL     10$
3944  025344  012737  000002  025410  MOV     #2,TIME
3945  025352  052777  000115  000040  BIS     #115,@RTCCSR    ;START CLOCK
3946  025360  005037  177776          CLR     PS
```

```
3947  025364  005737  025410          1$:     TST     TIME
3948  025370  001375                          BNE     1$
3949  025372  005077  000022                  CLR     @RTCCSR         ;STOP CLOCK
3950
3951  025376  000207                          RTS PC
3952  025400  105237  025410          10$:    INCB    TIME
3953  025404  001375                          BNE     10$
3954  025406  000207                          RTS     PC
3955
3956  025410  000000          TIME:   .WORD   0
3957
3958  025412  005337  025410          CLKINT: DEC     TIME
3959  025416  000002                          RTI
3960  025420  000000          RTCCSR: .WORD   0               ;CLOCK CSR IF USED.
3961                                  ;*
3962                                  ;*THIS MACRO ALLOWS THE OPERATOR TO TALK TO
3963                                  ;*ANY DEVICE ON THE I/O BUS
3964                                  ;*USER MUST START AT THIS ADDR.
3965                                  ;*HE MUST SAY EITHER "E" FOR EXAMINE, OR "D" FOR DEPOSIT.
3966                                  ;*"E" IS DEFAULT.
3967                                  ;*NEXT, HE MUST SUPPLY AN ADDR.
3968                                  ;*NOTE IF ADDR. IS NOT FOUND ON I/O BUS, A HALT
3969                                  ;*WILL OCCUR.
3970
3971
3972  025422                  $UTK:
3973  025422  005037  001514          CLR     .DVLS
3974  025426                  21$:
3975  025426  104401  025434          TYPE    ,65$            ;;TYPE ASCIZ STRING
3976  025432  000405                  BR      64$             ;;GET OVER THE ASCIZ
3977                          ;;65$:  .ASCIZ  <200>#E OR D?#
3978  025446                  64$:
3979  025446  105777  153472  1$:     TSTB    @$TKS
3980  025452  100375                  BPL     1$
3981  025454  117737  153466  025576  MOVB    @$TKB,20$       ;GET INPUT
3982  025462  104401  025576          TYPE,   20$             ;ECHO, NEXT MESSAGE.
3983  025466  142737  000240  025576  BICB    #240,20$        ;STRIP PARITY, LC
3984  025474  104412                  RDOCT                   ;GET ADDR.
3985  025476  012637  025574          MOV     (SP)+,14$
3986  025502  123727  025576  000104  CMPB    20$,#'D         ;DEPOSIT?
3987  025510  001411                  BEQ     10$
3988
3989  025512  004537  025144          JSR     R5,$INLP        ;GET DATA
3990  025516  025574          2$:     .WORD   14$
3991  025520  025532                  .WORD   5$
3992
3993  025522  013746  025532          MOV     5$,-(SP)        ;;SAVE 5$ FOR TYPEOUT
3994  025526  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3995  025530  000736                  BR      21$             ;LOOP.
3996  025532  000000          5$:     .WORD   0
3997
3998  025534                  10$:
3999  025534  104401  025542          TYPE    ,67$            ;;TYPE ASCIZ STRING
4000  025540  000404                  BR      66$             ;;GET OVER THE ASCIZ
```

```
                                        ;;67$:  .ASCIZ  <200>#DATA= #
                                        66$:
025552                                          RDOCT
025552  104412                                  MOV     (SP)+,13$
025554  012637  025572
                                        11$:    JSR     R5,$OUTLP       ;OUTPUT ROUTINE.
025560  004537  025046                  12$:    .WORD   14$             ;DEVICE ADDR.
025564  025574                                  .WORD   13$             ;DATA
025566  025572                                  BR      21$
025570  000716
                                        13$:    .WORD   0
025572  000000                          14$:    .WORD   0
025574  000000                          20$:    .ASCIZ  <1><200>#DEVICE ADDR= #
025576  100001  042504  044526
025604  042503  040440  042104
025612  036522  000040
                                                .EVEN
                                        ;
                                        ;THIS ROUTINE LOOKS THROUGH CURENT .DVLS FOR A/D ADDR.
                                        ;IF UNFOUND,GENERATES IT. THIS ROUTINE'S WHOLE PURPOSE IS
                                        ;TO SET UP THE USER PROGRAM TO LINK TO FILE "DRLPX2" FOR
                                        ;SAMPLE TAKEING PURPOSES.
                                        ;       TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
                                        ;       A/D CSR IN BSEL 4,AND 5.
                                        ;       (2) HE MUST CALL THIS ROUTINE:
                                        ;               JSR     R5,$PUTS        ;CALL SET UP ROUTINE.
                                        ;               .WORD   ADCSR           ;ADDR. OF A/D CSR.
                                        ;               ;RETURNS HERE ;KMC BSEL 3,6,7 PERMINENTLY SET UP
                                        ;                               ;(UNTILL ONE DOES A RESET)
                                        ;
                                        ;       (3)THE USER MUST PUT CODE 006 INTO KMC REG 2 TO
                                        ;          START CONVERSION  CAUTION*DO WITH MOVB INSTR.!
                                        ;       (4)MONITOR KMC REG 2 FOR CODE 377 (DRLPX2 IS DONE)
                                        ;       (5)READ KMC REG 4,5 FOR A/D RESULT.
                                        ;       (6) TO TAKE MORE SAMPLES,SIMPLY PUT A/D CSR INTO
                                        ;          BSEL 4,5 AND CODE 6 INTO BSEL 2.
                                        ;
025616  012537  025626          $PUTS:  MOV     (5)+,1$                 ;GET ADDR OF ADDR. OF A/D
025622  004537  025144                  JSR     R5,$INLP
025626  000000                  1$:     .WORD   0
025630  025724                          .WORD   10$
025632  113777  025206  153642          MOVB    $OFS,@KMAD6
025640  113777  025206  153636          MOVB    $OFS,@KMAD7
025646  013737  025626  025666          MOV     1$,2$
025654  062737  000002  025666          ADD     #2,2$
025662  004537  025144                  JSR     R5,$INLP
025666  000000                  2$:     .WORD   0
025670  025724                          .WORD   10$
025672  113777  025206  153574          MOVB    $OFS,@KMAD3
025700  152777  000340  153574          BISB    #340,@KMAD6
025706  152777  000300  153570          BISB    #300,@KMAD7
025714  152777  000300  153552          BISB    #300,@KMAD3
```

```
 4055  025722  000205                        RTS     R5
 4056  025724  000000              10$:      .WORD   0
 4057
 4058
 4059  025726  000000              ADBUFF:   0
 4060  025730  001774                        .BLKW   1020.
 4061  031720  000000              ABUFF4:   0
 4062  031722  000200                        .BLKW   200
 4063  032322  000000              LAST:     0
 4064          000001                        .END
```

```
ABASE = 170400          150#   264    305
ABUFF4  031720         2539*  2588   4061#
ACDW1 = 000000          264    307
ACDW2 = 000000          264    308
ACPUOP= 000000          264    279
ACRLF   020040          907   1259   1455   1769   2574   2590   2656   2714#
ADAC    016244         1912*  1924*  1938   1942   1945   1971   1974   1981   1985   1998   2432#
ADBUFF  025726         2478   2488*  2490*  2516   2527   4059#
ADCS    001570          457#   608    715    718    733    745    753    877    884    887    890   1948   1961
                       2003   2071   2074   2077   2080   2680   2683
ADCS1   001572          458#   612*
ADDBR   001574          459#   613    755    894   1951   1965   2084   2687
ADDIF   015350         2349#  2565
ADDIFM  015307         2343#  2524
ADDW0 = 000000          264    309
ADDW1 = 000000          264    310
ADDW10= 000000          264    319
ADDW11= 000000          264    320
ADDW12= 000000          264    321
ADDW13= 000000          264    322
ADDW14= 000000          264    323
ADDW15= 000000          264    324
ADDW2 = 000000          264    311
ADDW3 = 000000          264    312
ADDW4 = 000000          264    313
ADDW5 = 000000          264    314
ADDW6 = 000000          264    315
ADDW7 = 000000          264    316
ADDW8 = 000000          264    317
ADDW9 = 000000          264    318
ADEMSG  014651         1255   2288#
ADEND   013002          757    772    787    802    817    832    847    861    925    944    960    976    993
                       1010   1027   1044   1595   1620   2067*  2084*  2087*  2088*  2089*  2090*  2091*  2092*
                       2097#
ADEVCT= 000000          264    270
ADEVM = 000000          264    306
ADNMUX  016306         1243*  1256   2449#
ADPMUX  016304         1211*  1252   2448#
ADSMSG  014562         1251   2278#
AENV  = 000000          264    275
AENVM = 000000          264    276
AFATAL= 000000          264    267
AMADR1= 000000          264    292
AMADR2= 000000          264    296
AMADR3= 000000          264    299
AMADR4= 000000          264    302
AMAMS1= 000000          264    286
AMAMS2= 000000          264    294
AMAMS3= 000000          264    297
AMAMS4= 000000          264    300
AMEAN   016340         2462#  2644
AMSGAD= 000000          264    272
AMSGLG= 000000          264    273
AMSGTY= 000000          264    266
```

```
AMTYP1= 000000   264    287
AMTYP2= 000000   264    295
AMTYP3= 000000   264    298
AMTYP4= 000000   264    301
APASS = 000000   264    269
APRIOR= 000200   152#   264
APTCSU= 000040  3315    3420#
APTENV= 000001  2908    3308    3376    3418#
APTSIZ= 000200   530    3417#
APTSPD= 000100  3310    3378    3419#
ARBADD  001556   451#    576*    609     905     1778*   1782*   2392    2395    2397    2399    2401    2404
ARBVCT  001560   452#    577*   1779*   1783*
ASWREG= 000000   264    277
ATESTN= 000000   264    268
AUNIT = 000000   264    271
AUSWR = 000000   264    278
AVECT1= 000340   151#   264     303     443     444
AVECT2= 000000   264    304
BASBT1  014274  2242#   2739*   2745*
BASBT2  014276  2243#   2740*   2746*
BASBT3  014321  2247#   2741*   2742*   2747*   2748*
BEGIN   001612   176     468#    571     610    3194
BEGIN1  001634   177     473#
BEGIN2  001626   178     472#
BIASC1  016310   894*    909    2450#
BIASDN  014272   912    2241#
BIASST  014256   908    2239#
BIT0  = 000001   135#   1503    1529
BIT00 = 000001   125#    135
BIT01 = 000002   124#    134
BIT02 = 000004   123#    133
BIT03 = 000010   122#    132
BIT04 = 000020   121#    131
BIT05 = 000040   120#    130
BIT06 = 000100   119#    129
BIT07 = 000200   118#    128
BIT08 = 000400   117#    127    2843
BIT09 = 001000   116#    126    2851    2919
BIT1  = 000002   134#   3578
BIT10 = 002000   115#    938     970    1004    1038    2896
BIT11 = 004000   114#    919     970    1004    1021    2858    3518
BIT12 = 010000   113#    669     899     938     954    1004    1021    1174    1249    1437    1759    2522    2560
                2579
BIT13 = 020000   112#   2903
BIT14 = 040000   111#   2829    3510
BIT15 = 100000   110#   1294    1334    1380    1426    2406    2763    3518
BIT2  = 000004   133#
BIT3  = 000010   132#
BIT4  = 000020   131#    715    3779
BIT5  = 000040   130#   3522    3735    3752
BIT6  = 000100   129#
BIT7  = 000200   128#    691     715     733     734     879    1097    1111    1146    1160    1200    1207    1232
                1239   1466    1485    1503    1529    1556    1576    1649    1713
BIT8  = 000400   127#    728
```

```
BIT9   = 001000   126#  678  681  694  919  954  1004  1038
BPTVEC= 000014   142#
BYPAS1  011340  1781  1787#  1825
BYTWO   012542  1440  1444  1448  1452  2043#
CHANL   013006   872*  1920*  2064*  2099#  2392  2395
CH0    = 000000   623#  756
CH1    = 000001   624#
CH10   = 000010   631#
CH11   = 000011   632#  1594
CH12   = 000012   633#  1619
CH13   = 000013   634#
CH14   = 000014   635#
CH15   = 000015   636#
CH16   = 000016   637#
CH17   = 000017   638#
CH2    = 000002   625#  786
CH3    = 000003   626#  816  1237
CH4    = 000004   627#  846  1205
CH40   = 000040   642#  771
CH41   = 000041   643#  872  873
CH42   = 000042   644#  801
CH43   = 000043   645#  831
CH44   = 000044   646#
CH45   = 000045   647#
CH46   = 000046   648#
CH47   = 000047   649#  1312  1332  1483  1527
CH47PK  016262  1337*  1446  2439#
CH47RM  016260  1317*  1445  2438#
CH5    = 000005   628#  1095  1109  1554  1711
CH50   = 000050   650#
CH51   = 000051   651#
CH52   = 000052   652#
CH53   = 000053   653#  924  943
CH54   = 000054   654#  959  975
CH55   = 000055   655#  992  1009
CH56   = 000056   656#  1026  1043
CH57   = 000057   657#  860
CH6    = 000006   629#  1144  1158  1205  1237  1574  1647
CH7    = 000007   630#  1464  1501
CH7PEK  016256  1297*  1383  1429  1442  2437#
CH7RMS  016254  1277*  1360  1406  1441  2436#
CKSWR  = 104407  2827  2828  2892  2918  3460#
CLKINT  025412  3958#
COMPAR  013012   759  774  789  804  819  834  849  863  895  1119  1168  1214  1246
                1472  1491  1509  1516  1535  1542  1561  1581  1597  1622  1679  1742  2102#
CONS    016316  1915*  1926  1931  1987  1992  2453#
CONVR   016450  2484#  2500  2521
CONVRT  012610   755  770  785  800  815  830  845  859  923  942  958  974  991
                1008  1025  1042  1593  1618  2063#
CONVT   017570  2484  2513  2666#
CR     = 000015   50#  3354  3364
CRLF   = 000200   51#  554  3325  3364
CSB     001600   461#
CSC     001610   465#
```

```
CSR     001576          460#
DACSAV  016246         1099*  1113   1114   1148   1162   1163   1202   1209   1234   1241   1468   1469   1487
                       1488   1505   1506   1531   1532   1557   1577   1651   1715   1981   2433#
DDIsP = 177570           57#   241    518
DECPNT  012524         2015*  2018   2020   2027*  2032#
DECPRT  012416         2012#  2621   2627   2797
DH1     013575          347   2185#
DH10    014130          389   2224#
DH11    014207          395    401    407    413   2232#
DH2     013654          353   2193#
DH3     013723          359    365   2200#
DH5     013760          371   2205#
DH6     014020          377   2211#
DH7     014061          383   2217#
DIEMSG  014523         1184   2272#
DIFCHN  020146         2467*  2680   2732#
DIFERR  017172         1073   2530*  2559*  2585#
DIFLIN  016344         1059   2465#
DIFMSG  014327         1176   2249#
DIGCNT  012522         2012*  2023*  2031#
DIGIT   012520         2016*  2017*  2021*  2022   2030#
DIGT0   012536         2038#  2628
DIGT1   012537         2014   2039#  2622   2629   2801
DIGT2   012540         2040#  2623   2630   2802
DIGT3   012541         2041#  2624   2631   2804
DISPLA  001142          241#   518*   526*  2873*  2895*
DISPRE  000174          173#   526
DIYMSG  014513         1180   2270#
DRLPX2= ******  G        14#  3508
DSWR  = 177570           56#   240    517
DT1     015716          348   2392#
DT10    015774          390   2401#
DT11    016012          396    402    408    414   2404#
DT2     015734          354    384   2395#
DT3     015750          360    366    378  2397#
DT5     015762          372   2399#
EDGE    016314         1914*  1951   2401   2452#
EMTVEC= 000030          145#   501*   502*
EM1     013062          346   2121#
EM10    013404          388   2159#
EM11    013445          394   2165#
EM12    013506          400   2171#
EM13    013523          406   2174#
EM14    013545          412   2178#
EM14A   013571         1684*  1747*  2183#
EM14B   013570         1691*  1754*  2182#
EM2     013107          352   2125#
EM3     013153          358   2132#
EM4     013207          364   2137#
EM5     013244          370   2142#
EM6     013305          376   2148#
EM7     013340          382   2153#
ERRVEC= 000004          138#   515    516*   527*  2834   2835*  2837*  2840*
EXCESS  017174         1069   2535*  2552*  2586#  2595   2658
```

```
FCHANL  013004          1921*  1922*  1923*  1945   2098#
FINS    016320          1925*  1981   1986*  2454#
GNS   = ****** U         172    553    601    605    903   3452  3453  3454  3455  3456  3458  3460  3461
                        3462   3463   3501   3555   3559  3563  3791  3977  4001
GRT2MG  015415          2356#  2598
GTSWR = 104406           548   3458#
HILIM1  016326          2457#  2540   2642
HILIM2  016330          2458#  2542
HT    = 000011            48#  3323   3364
IOTVEC= 000020           143#   499*   500*
KMADO   001466           426#   481   3510*  3518*  3522  3578  3638*  3640*  3642*  3643*  3650*  3655*  3735
                        3752   3779
KMAD1   001470           429#   482
KMAD2   001472           431#  3527*  3540   3580*  3594  3608*  3621  3689*  3694*  3698*  3720*  3737*  3754*
                        3773
KMAD3   001474           433#  4051*  4054*
KMAD4   001476           435#  3542   3547   3571*  3573*  3575*  3596  3639*  3644*  3647*  3656*  3688*  3692*
                        3697*  3719*  3739   3756
KMAD5   001500           437#
KMAD6   001502           439#  3641*  3651   4044*  4052*
KMAD7   001504           441#   486   4045*  4053*
LAST    032322          2481   4063#
LF    = 000012            49#  3358   3364
LINMG1  015264          1765   2339#
LINMSG  015101          1761   2318#
LOLIM1  016322          2455#  2546   2640
LOLIM2  016324          2456#  2548
LPADH   001500           436#
LPADL   001476           434#
LPCI    001466           425#
LPCO    001472           430#
LPMR    001470           428#
LPMS1   001502           438#
LPMS2   001504           440#
LPSO    001474           432#
MSGPNT  020342          2783#  2785*  2798
MULTSP  015665          2046   2386#
NARMSG  015444          2360#  2602
NARROW  016334          2460#  2531*  2558*  2599   2609
NBEXT   001566           455#   578*  1776   1780*  1784*
NMBEXT  001562           453#   573*   578    596   1784
NOIMG1  015010          1443   2306#
NOIMG2  015033          1447   2310#
NOIMG3  015056          1451   2314#
NOIMSG  014667          1439   2291#
NUMBF2  016026          1638   1702   2406#
OFSLBX  016300          1505*  2446#
OFSLUX  016302          1531*  2447#
OFSTBX  016274          1468*  1513   1558   1578   2444#
OFSTUX  016276          1487*  1539   2445#
OVERNG  020020          2657   2711#
PC    =%000007            69#   580*   589*   759*   774*  789*  804*  819*  834*  849*  863*  895*  1119*
                       1168*  1214*  1246*  1472*  1491*  1509*  1516*  1535*  1542*  1561*  1581*  1597*  1622*
                       1679*  1742*  1785*  1806*  1809*  1819*  1824   2026*  2114*  2118*  2484*  2513*  2621*
```

```
                    2627*  2688*  2752*  2797*  2905*  2911*  2985*  3146*  3193   3313*  3332*  3339*  3346*
                    3360*  3362*  3395*  3412*  3572*  3574*  3576*  3623*  3686*  3690*  3695*  3699*  3718*
                    3721*  3738*  3755*  3786*  3924*  3951*  3954*
PCT      020344     2784#  2786*  2793*  2795*  2796   2799
PEKMAX   016252     1298   1338   1385   1431   2435#
PERHLF   015544     2372#  2611   2612
PERQRT   015623     2380#  2615   2616
PIRQ   = 177772       55#
PIRQVE= 000240       149#
PRCNT    020346     2610   2614   2785#
PR0    = 000000       72#
PR1    = 000040       73#
PR2    = 000100       74#
PR3    = 000140       75#
PR4    = 000200       76#
PR5    = 000240       77#
PR6    = 000300       78#
PR7    = 000340       79#
PS     = 177776       52#     53    556*   3946*
PSW    = 177776       53#
PWRMSG   022342     3192   3199#
PWRVEC= 000024       144#    505*   506*   3161*  3162*  3171*  3177*  3189*  3190*
QRTOK    016332     2459#  2529*  2550*  2613
RBEG     001642      471    474#
RBEG2    002414      580#  1788
RDCHR  = 104410     3044   3461#
RDLIN  = 104411     3085   3462#
RDOCT  = 104412     3463#  3984   4003
RD1      024746     3722*  3759#
READ     016656     2504   2522#
RESVEC= 000010       139#
RMSMAX   016250     1278   1318   1362   1408   2434#
RSLTO    016052     1639   1656   1657   1663*  1664*  1665*  1666*  1670*  1671*  1672*  1673*  1674*  1676*
                    1678   1763   2410#
RSLT1    016110     1703   1720   1721   1727*  1728*  1729*  1730*  1734*  1735*  1736*  1737*  1738*  1740*
                    1741   1767   2415#
RSLT2    016146     1086   1178   2420#
RSLT3    016204     1135   1182   2425#
RSVMSG   017700     2633   2690#
RSV1     017702     2622*  2691#
RSV2     017704     2623*  2692#
RSV3     017705     2624*  2693#
RSX1     017707     2628*  2694#
RSX2     017710     2629*  2695#
RSX3     017711     2630*  2696#
RSX4     017712     2631*  2697#
RTCCSR   025420     3942   3945*  3949*  3960#
R0     =%000000      60#    479    481*   484*   485    490*   579*   590    608*   609*   610    613*   615*
                    1638*  1641   1642   1652   1656*  1657*  1658*  1659*  1660*  1661*  1664   1702*  1705
                    1706   1716   1720*  1721*  1722*  1723*  1724*  1725*  1728   1787*  1816*  1819   1845
                    1855*  1859   1875   1876   1889*  1916   1939*  1965*  2006*  2052*  2054   2102   2104*
                    2106*  2108*  2109   2112*  2117*  2486   2525*  2536*  2537   2539*  2540   2542   2544
                    2546   2548   2556   2570*  2571   2607*  2608*  2609*  2613*  2786   2787*  2788*  2789*
                    2790*  2791*  2792*  2793   2794*  2795   2798*  2801*  2802*  2803*  2804*  2807*  2808*
```

```
                2809*  2810*  3082   3086*  3089   3105*  3118   3119*  3120*  3127*  3128*  3129*  3130*
                3131*  3132   3137   3142*  3144*  3148   3150   3163   3188*  3306   3307*  3312   3317
                3320*  3372   3380*  3384   3385   3387*  3388*  3389   3411*  3429   3430*  3431   3432*
                3433*  3434*  3435*  3636   3637*  3648*  3651   3659*  3683   3684*  3685*  3687   3688
                3700*  3715   3716*  3717*  3719   3722   3757*  3810   3813*  3829*  3852   3855*  3873*
R1    =%000001    61#   480    482*   486    489*   582*   583*   584*   585    614*   615    616*   617
                1085*  1088   1091   1105   1122*  1123   1134*  1137   1140   1154   1171*  1172   1662*
                1665*  1666   1667   1668   1675*  1676*  1678   1681   1685   1693   1694   1726*  1729*
                1730   1731   1732   1739*  1740*  1741   1744   1748   1756   1757   1846   1859*  1860
                1864   1888*  1917   1938*  1953*  1967   2005*  2013   2014*  2022*  2025*  2053*  2058*
                2103   2105*  2106   2111*  2116*  2735*  2749*  2750   3083   3087*  3091*  3093*  3095*
                3098*  3101   3104*  3164   3187*  3373   3410*  3514   3515*  3516*  3519*  3567*  3568
                3573   3575   3602*  3771   3772*  3775*  3781*  3785*  3811   3814*  3819*  3822   3828*
                3853   3856*  3862*  3865   3872*
R2    =%000002    62#   558*   560    565*   581*   583    585*   586    587    1086*  1116*  1135*  1165*
                1639*  1651*  1681*  1682*  1683*  1684   1685*  1686*  1687*  1688*  1689*  1690*  1691
                1703*  1715*  1744*  1745*  1746*  1747   1748*  1749*  1750*  1751*  1752*  1753*  1754
                1847   1858*  1862*  1865   1872*  1873*  1874   1879*  1887*  1918   1934*  1942   1971
                1974   1976*  2004*  2018*  2020*  2619*  2620*  2626*  2738*  2744*  2749   2796*  3084
                3088*  3092*  3094*  3096*  3102   3103*  3165   3186*
R3    =%000003    63#   559*   566*   568    572*   573    1848   1856*  1857*  1871*  1874*  1883*  1884*
                1886*  1919   1928*  1930*  1933*  1967   1989*  1991*  2003*  2536   2576   2626   3040
                3041*  3042   3045*  3046   3050   3052   3054*  3056*  3166   3185*  3241   3250*  3256*
                3257*  3260*  3265*  3266*  3267   3276*
R4    =%000004    64#   1093*  1107*  1142*  1156*  1196*  1203*  1228*  1235*  1270*  1290*  1310*  1330*
                1353*  1376*  1399*  1422*  1462*  1481*  1499*  1525*  1552*  1572*  1645*  1709*  1912
                1913   1914   1915   2007*  3167   3184*  3242   3244*  3245*  3246*  3247   3248*  3262
                3264*  3272*  3275*  3635   3660*  3665*
R5    =%000005    65#   563*   669*   672*   674*   681*   684*   686*   694*   697*   699*   711*   715*
                718*   720*   723*   726*   733*   743*   745*   753*   755*   770*   785*   800*   815*
                830*   845*   859*   877*   884*   887*   890*   894*   922*   923*   938*   941*   942*
                957*   958*   973*   974*   990*   991*   1007*  1008*  1024*  1025*  1041*  1042*  1052*
                1059*  1091*  1102*  1105*  1140*  1151*  1154*  1177*  1181*  1196*  1228*  1276   1277
                1296   1297   1316   1317   1336   1337   1352*  1359   1376*  1382   1399*  1405   1422*
                1428   1440*  1444*  1448*  1452*  1552*  1572*  1592*  1593*  1603*  1606*  1617*  1618*
                1628*  1631*  1645*  1709*  1762*  1766*  1849   1851*  1853*  1860*  1864*  1879   1885*
                1938*  1942*  1945*  1948*  1951*  1961*  1965*  1971*  1974*  1981*  1985*  1998*  1999*
                2003*  2043   2047   2050*  2052   2053   2060*  2063   2071*  2074*  2077*  2080*  2084*
                2093*  2465   2466   2467   2468   2472*  2476*  2478*  2486   2490*  2494*  2497*  2499*
                2503*  2507*  2510*  2513*  2514*  2518*  2581*  2582*  2610*  2614*  2634*  2635*  2661*
                2662*  2669*  2672*  2675*  2680*  2683*  2687*  2785   2805*  2811*  3168   3183*  3243
                3249*  3251*  3253*  3254*  3255*  3256   3274*  3507*  3529*  3583*  3603*  3610*  3661*
                3701*  3724*  3741*  3758*  3825*  3830*  3833*  3867*  3874*  3878*  3890   3892   3897*
                3898*  3915*  3989*  4006*  4041*  4048*  4055*
R6    =%000006    66#   493*   494*   495
R7    =%000007    67#
SAR     011716  1093   1107   1142   1156   1196   1203   1228   1235   1270   1290   1310   1330   1353
                1376   1399   1422   1462   1481   1499   1525   1552   1572   1645   1709   1912#
SARCHN  016312  1913*  1920   1922   1954   1957   2451#
SDELAY  025336  3941#
SKIPST  017170  1065   2534*  2555*  2584#  2591
SKPMSG  015366  2352#  2594
$P    =%000006    68#   479*   480*   489    490    497*   515*   523*   527    557*   596*   905*   909*
                1252*  1256*  1813*  1845*  1846*  1847*  1848*  1849*  1850*  1851   1854*  1867   1869*
```

```
TBITVE= 000014            140#
TEMP    016342            728*    729*    879*    880*   2463#   2528*   2566    2575*   2589*   2619    2640    2642    2644
                         2651*   2653
TIME    025410           3944*   3947    3952*   3956#   3958*
TKVEC = 000060            147#
TPVEC = 000064            148#
TRAPVE= 000034            146#    503*    504*
TRTVEC= 000014            141#
TST1    002720            664#
TST10   003720            820     827#
TST11   003770            835     842#
TST12   004040            850     856#
TST13   004110            864     870#
TST14   004364            900     917#
TST15   004446            927     933#
TST16   004546            946     952#
TST17   004630            962     968#
TST2    003142            700     706#
TST20   004712            978     985#
TST21   004774            995    1002#
TST22   005056           1012    1019#
TST23   005140           1029    1036#
TST24   005240           1056#
TST25   005340           1076    1083#
TST26   005556           1125    1132#
TST27   006040           1175    1185    1191#
TST3    003370            748#
TST30   006176           1215    1217    1223#
TST31   006400           1250    1260    1267#
TST32   006454           1280    1287#
TST33   006530           1300    1307#
TST34   006604           1320    1327#
TST35   006660           1340    1347#
TST36   006762           1364    1371#
TST37   007056           1387    1394#
TST4    003460            760     767#
TST40   007152           1410    1417#
TST41   007342           1438    1460#
TST42   007426           1473    1479#
TST43   007512           1492    1497#
TST44   007626           1517    1523#
TST45   007742           1543    1548#
TST46   010034           1562    1568#
TST47   010126           1582    1587#
TST5    003530            775     782#
TST50   010252           1612#
TST51   010376           1636#
TST52   010700           1700#
TST53   011246           1760    1774#
TST6    003600            790     797#
TST7    003650            805     812#
TYANX   020043           2638    2715#
TYANXX  020054           2569    2718#
TYHIGH  017744           2650    2703#
```

```
TYLOW   017716          2648    2699#
TYMEAN  017774          2646    2707#
TYPDS = 104405          1814    3456#
TYPE  = 104401           540     599    603    901    907    908    912   1176   1180   1184   1251   1255   1259
                        1439    1443   1447   1451   1455   1761   1765   1769   1812   1815   1890   2046   2057
                        2524    2565   2569   2574   2590   2594   2598   2602   2606   2612   2616   2617   2632
                        2637    2646   2648   2650   2655   2657   2898   2906   2952   2953   2956   2969   2980
                        2999    3048   3051   3055   3117   3134   3136   3139   3141   3145   3152   3191   3268
                        3328    3452#  3499   3553   3557   3561   3789   3975   3982   3999
TYPOC = 104402           906    2955   3125   3149   3453#  3994
TYPON = 104404          3455#
TYPOS = 104403           597     910   1253   1257   2044   2048   2055   2567   2572   2592   2596   2600   2604
                        2659    3454#
TYPSTG  012564          1177    1181   1762   1766   2052#
VARLT1  020306          2738    2765#
VARLT2  020324          2744    2773#
VA24    020274           877     878   2759#
VCREG1  020142          2465*   2472*  2476   2503   2507   2510   2730#
VCREG2  020144          2466*   2473*  2478   2494   2497   2499   2513   2731#
VCSTAT  001602           462#    669    672    674    681    684    686    694    697    699    711    720    723
                         726     743    922    938    941    957    973    990   1007   1024   1041   1052   2472
                        2669    2672   2675
VCXREG  001604           463#   1060   1091   1102   1105   1143   1157   1197   1204   1229   1236   1376   1400
                        1423    1552   1573   1592   1603   1606   1646   1709
VCYREG  001606           464#   1061   1094   1108   1140   1151   1154   1196   1228   1271   1291   1311   1331
                        1352    1354   1377   1399   1422   1463   1482   1500   1526   1553   1572   1617   1628
                        1631    1645   1710
VECTOR  001506           443#
VECTPS  001510           444#
VERSN   001512           446#
V1      020302          2544    2762#
V1754   020266           814    2735   2756#
V24     020270           818     844    848   2757#
V50     020272           833    2758#
V62     020276          1118    1167   1471   1490   1508   1534   2760#
V77     020300          1560    1580   1655   1719   2761#
WFADJ   020150           589    2735#
WFTEST  020264           470*    472*  2736   2754#
WIDE    016336          2461#   2532*  2553*  2603   2608
WIDMSG  015504          2366#   2606
WI      024576          3687*   3702#
W2      024600          3691*   3703#
W3      024602          3696*   3704#
XDACPK  016266          1384*   1450   2441#
XDACRM  016264          1361*   1449   2440#
YDACPK  016272          1430*   1454   2443#
YDACRM  016270          1407*   1453   2442#
$AERR   024326           563    3506*  3599*  3625#  3915
$APTHD  001000           200     206#
$ASTAT= ****** U        3398    3413
$ATYC   023144          3369    3371#
$ATY1   023120          3367#
$ATY3   023126          3313    3368#
$ATY4   023136          2911    3370#
```

```
$AUTOB  001134  237#  550*  2949  3070
$BASE   001256  305#  558   576   1782
$BDADR  001122  232#
$BDDAT  001126  234#  469*  473*  579   674   686   699   733   734*  735   757*  772*  787*
                802*  817*  832*  847*  861*  894   925*  926   944*  945   960*  961   976*
                977   993*  994   1010*  1011  1027*  1028  1044*  1045  1065*  1069*  1073*  1075
                1114*  1115*  1116  1163*  1164*  1165  1209*  1210*  1211  1241*  1242*  1243  1276*
                1279  1296*  1299  1316*  1319  1336*  1339  1359*  1360*  1361  1363  1382*  1383*
                1384  1386  1405*  1406*  1407  1409  1428*  1429*  1430  1432  1469*  1488*  1506*
                1513*  1532*  1539*  1557*  1558*  1577*  1578*  1595*  1620*  1678*  1741*  2105  2392
                2395  2397  2399  2401  2404  2499  2675  2683
$BELL   001172  256#  2898  2931
$CDW1   001262  307#
$CDW2   001264  308#
$CHARC  023114  3330*  3340*  3347  3356*  3361#
$CKSWR  021156  2941#  3460
$CMTAG  001100  220#  492  493  501  507  508  509
$CM1  =  000002  252#  253#  254#
$CM2  =  000004  252#  253#  254#
$CM3  =  000002  250#  252
$CNT    025276  3893*  3895*  3901#
$CNTLG  021703  2952  3065#
$CNTLU  021676  2969  3064#
$CPUOP  001230  279#
$CRLF   001177  258#  2906  2931  2980  3064  3117  3136  3141  3145  3329  3364
$DATR   024750  3739*  3756*  3760#  3371
$DBLK   011706  1856  1890  1898#
$DDW0   001266  309#
$DDW1   001270  310#
$DDW10  001312  319#
$DDW11  001314  320#
$DDW12  001316  321#
$DDW13  001320  322#
$DDW14  001322  323#
$DDW15  001324  324#
$DDW2   001272  311#
$DDW3   001274  312#
$DDW4   001276  313#
$DDW5   001300  314#
$DDW6   001302  315#
$DDW7   001304  316#
$DDW8   001306  317#
$DDW9   001310  318#
$DEVCT  001212  270#
$DEVM   001260  306#
$DOaGN  011446  1808  1817  1823#
$DTBI   011676  1859  1894#
$ENDAD  011436  186  538  1819#  2926
$ENDCT  011404  507  1810#
$ENdMG  011455  1812  1827#
$ENULL  011452  1815  1826#
$ENV    001222  275#  544  2908  3308  3376  3400
$ENVM   001223  276#  530  3310  3315  3378
$EOP    011350  1786  1800#
```

```
$EOPCT  011376          507*   1807#  1811
$ERFLG  001103          223#   2817   2847   2849   2855*  2877   2893*  2931
$ERMAX  001115          229#    510*  2849   2872*  2877
$ERROR  020756          501    2891#
$ERRPC  001116          230#   2392   2395   2397   2399   2401   2404   2900*  2901*  2902   2931   3123
$ERRTB  001326          343#   3131
$ERRTY  022034         2905    3116#
$ERTTL  001112          227#   2899*  2931
$ESCAP  001170          255#    509*  2871*  2922   2924   2931
$ETABL  001222          274#
$ETEND  001326          212     327#
$FATAL  001204          267#   3404#
$FFLG   023364         3367*   3370*  3398   3407*  3415#
$FILLC  001156          248#   3333   3364
$FILLS  001155          247#   3364
$GDADR  001120          231#
$GDDAT  001124          233#    563    669*   672    674    678*   686    691*   699    715*   718    733*   735
                        750*    769*   784*   799*   814*   829*   844*   858*   877*   922*   926    941*   945
                        957*    961    973*   977    990*   994   1007*  1011   1024*  1028   1041*  1045   1064*
                       1074*   1075   1117*  1166*  1212*  1244*  1278*  1279   1298*  1299   1318*  1319   1338*
                       1339    1362*  1363   1385*  1386   1408*  1409   1431*  1432   1470*  1489*  1507*  1514*
                       1533*   1540*  1559*  1579*  1592*  1606*  1617*  1631*  1654*  1718*  2104   2392   2395
                       2397    2399   2401   2404
$GET42  011426         1816#
$GTSWR  021226         2953#   3458
$HD     =  000000        40
$HIBTS  001000          207#
$HIOCT  022032         3102*   3107#
$ICNT   001104          224#   2862*  2863   2865*  2876
$ILLUP  022334         3161    3177   3196#
$INLP   025144          563     674    681    686    694    699    720    726    733    884    890    894   1102
                       1151    1603   1628   1942   1951   1965   1971   1981   1985   1998   2074   2080   2084
                       2494    2499   2503   2507   2669   2675   2683   2687   3852#  3915   3989   4041   4048
$INTAG  001135          238#   2981   3070
$ITEMB  001114          228#   2902*  2910   2931   3120
$LF     001200          259#   2931   3055   3064   3364
$LFLG   023363         3408*   3414#
$LOAD   024330         3507    3635#
$LPADR  001106          225#    511*  2853*  2869*  2874   2876
$LPAI   023450         3481#   3833   3878
$LPERR  001110          226#    512*  2853   2870*  2876   2921
$LPW    024752         3686    3690   3695   3699   3718   3721   3738   3755   3771#
$MADR1  001234          292#
$MADR2  001240          296#
$MADR3  001244          299#
$MADR4  001250          302#
$MAIL   001202          208     212    265#   529    544   2868   2908   3308
$MAMS1  001232          286#
$MAMS2  001236          294#
$MAMS3  001242          297#
$MAMS4  001246          300#
$MBADR  001002          208#
$MFLG   023362         3368*   3374   3409*  3413#
$MNEW   021721         2956    3068#
```

```
$MSGAD  001216            272#   3384*  3387
$MSGLG  001220            273#   3389*
$MSGTY  001202            266#   3382   3390*   3402   3406*
$MSWR   021710           2953    3066#
$MTYP1  001233            287#
$MTYP2  001237            295#
$MTYP3  001243            298#
$MTYP4  001247            301#
$MXCNT  020754           2866    2876#
$NULL   001154            246#   3335   3364
$NWTST= 000001            661#    703#   745#   764#   779#   794#   809#   824#   839#   853#   867#   914#   930#
                          949#    965#   982#   999#  1016#  1033#  1053#  1080#  1129#  1188#  1220#  1264#  1284#
                         1304#   1324#  1344#  1368#  1391#  1414#  1457#  1476#  1494#  1520#  1545#  1565#  1584#
                         1609#   1633#  1697#  1771#
$OCNT   022632           3240*   3269*  3282#
$OFS  = 025206           3868#   4044   4045   4051
$OMODE  022634           3235*   3239*  3244   3247*  3258*  3284#
$OUTLP  025046            669     672    684    697    711    715    718    723    743    745    753    755    877
                          887     922    938    941    957    973    990   1007   1024   1041   1052   1091   1105
                         1140    1154   1196   1228   1352   1376   1399   1422   1552   1572   1592   1606   1617
                         1631    1645   1709   1938   1945   1948   1961   1974   2003   2071   2077   2472   2476
                         2478    2497   2510   2513   2672   2680   3810#  4006
$OVER   020740           2830    2846   2854   2864   2873#
$PASS   001210            269#    529*  1804*  1805*  1813   1826   2860   2877
$PASTM  001006            210#
$PUTS   025616           4040#
$PWRAD  022330           3194#
$PWRDN  022170            505    3161#  3189
$PWRMG  022324           3192#
$PWRUP  022242           3171    3177#
$QUES   001176            257#   2931   2999   3048   3064   3364
$RDCHR  021440           3012#   3461
$RDDEC= ****** U         3464
$RDLIN  021560           3040#   3462
$RDOCT  021732           3080#   3463
$RDSZ = 000010           3033#
$REGAD  001160            250#
$REGO   001162            252#
$REG1   001164            253#
$RESET  025300            580    1785   3912#  3922
$RTNAD  011450           1825#
$R2A  = ****** U         3464
$SAD    025274           3890    3892*  3900#
$SAVRE= ****** U         3464
$SAVR6  022340           3170*   3178   3179*  3180*  3198#
$SCOPE  020474            499    2826#
$SETUP= 000137            474#    498    499    501    503    505    507    508    509    511    538    541   1802
                         2827    2892   2918   2926   2936   3070
$STUP = 177777            474#
$SVLAD  020704           2838    2867#
$SVPC = 000214            184#    189
$SWR  = 167400             30#     40    158    159    160    161    162    163    164    165    254    255    256
                          508     509    511    512    665    707    749    768    783    798    813    828    843
                          857     871    918    934    953    969    986   1003   1020   1037   1057   1084   1133
```

```
                    1192    1224    1268    1288    1308    1328    1348    1372    1395    1418    1461    1480    1498
                    1524    1549    1569    1588    1613    1637    1701    1775    1797    1803    1818    1824    1826
                    2818    2819    2820    2821    2822    2829    2841    2843    2844    2847    2848    2849    2856
                    2857    2858    2870    2873    2876    2883    2884    2885    2886    2887    2896    2903    2915
                    2919    2931    3195
$SWREG  001224       277#     532
$SWRMK= 000000       165      166    2822    2823    2845
$TESTN  001206       268#    2868*
$TIMES  001166       254#     508*    665*    707*    749*    768*    783*    798*    813*    828*    843*    857*    871*
                     918*     934*    953*    969*    986*   1003*   1020*   1037*   1057*   1084*   1133*   1192*   1224*
                    1268*    1288*   1308*   1328*   1348*   1372*   1395*   1418*   1461*   1480*   1498*   1524*   1549*
                    1569*    1588*   1613*   1637*   1701*   1775*   1803*   2856*   2863    2866*   2876
$TKB    001146       243#    2934    2945    2962    3016    3022    3981
$TKS    001144       242#    2934    2943    2959    2983*   3014    3020    3979
$TLKR   024604      3715#    3867
$TLKW   024470      3683#    3825
$TMDAT  001564       454#     560*    563     666*    669     681*    684     694*    697     708*    711     715     720*
                     723      726     740*    743     745     753     755     873*    874*    877     884*    887     890
                     919*     922     935*    938*    941     954*    957     970*    973     987*    990    1004*   1007
                    1021*    1024    1038*   1041    1049*   1052    1088*   1091    1102*   1105    1137*   1140    1151*
                    1154     1193*   1196    1225*   1228    1349*   1352    1376    1399    1422    1552    1572    1589*
                    1592     1603*   1606    1614*   1617    1628*   1631    1642*   1645    1706*   1709    1935*   1938
                    1942*    1945*   1948    1951    1956*   1957*   1958*   1961    1965    1971*   1974    1985    1998
                    2000*    2003    2068*   2071    2074*   2077    2080    2084    2469*   2472    2476    2478    2494*
                    2497     2503    2507*   2510*   2513    2669*   2672    2687
$TN     000054        30#      40     661     665#    700     703     707#    745     749#    760     764     768#    775
                     779      783#    790     794     798#    805     809     813#    820     824     828#    835     839
                     843#     850     853     857#    864     867     871#    900     914     918#    927     930     934#
                     946      949     953#    962     965     969#    978     982     986#    995     999    1003#   1012
                    1016    1020#   1029    1033    1037#   1053    1057#   1076    1080    1084#   1125    1129    1133#
                    1175    1185    1188    1192#   1215    1217    1220    1224#   1250    1260    1264    1268#   1280
                    1284    1288#   1300    1304    1308#   1320    1324    1328#   1340    1344    1348#   1364    1368
                    1372#   1387    1391    1395#   1410    1414    1418#   1438    1457    1461#   1473    1476    1480#
                    1492    1494    1498#   1517    1520    1524#   1543    1545    1549#   1562    1565    1569#   1582
                    1584    1588#   1609    1613#   1633    1637#   1697    1701#   1760    1771    1775#
$TOUT   025240      3529    3583    3610    3724    3741    3890#
$TPB    001152       245#   3353*   3364
$TPFLG  001157       249#   3302    3364
$TPS    001150       244#   3351    3364
$TRAP   023366        503    3429#
$TRAP2  023410      3440#    3451
$TRP    000013      3444#   3453#   3454#   3455#   3456#   3457#   3458    3459#   3460    3461#   3462#   3463#   3464#
$TRPAD  023422      3434     3451#
$TSTM   001004       209#
$TSTNM  001102       222#   1802*   2817    2845    2867*   2868    2873    2877    2895    2931
$TTYIN  021666      3041    3042    3059    3063#
$TYPBN= ****** U    3457
$TYPDS  011472      1844#   3456
$TYPE   022636      3302#   3395    3444    3452
$TYPEC  023050      2985    3332    3339    3346    3351#   3352
$TYPEX  023116      3357    3359    3362#
$TYPOC  022434      3238#   3453
$TYPON  022450      3237    3240#   3455
$TYPOS  022410      3233#   3454
```

```
$UNIT   001214            271#
$UNITM  001010            211#
$USWR   001226            278#
$UTK    025422           3972#
$VECT1  001252            303#    577    1783
$VECT2  001254            304#
$XTSTR  020510           2832#
$$GET4= 000000           1818#
$OFILL  022633           3234*   3238*   3248    3283#
$40CAT= ****** U         2829    2905
.       = 032324          168#    172#    184     185#    187#    189#    190#    196     197#    199#    201#    219#    260
                          449#    496     511     512    1826    1830    1898#   2032    2729#   2876    2877    2931    2934
                         3063#   3064    3070    3155#   3173    3197    3364    3416#   3868    3978#   4060#   4062#
.DVLS   001514            448#    488*    555*    3813    3834    3855    3879    3973*
.$ASTA= ****** U         3368    3371
.$X     = 001000          196#    201
```

```
COMMEN    150#
ENDCOM    150#
ERROR      44#    676    688    701    737    761    776    791    806    821    836    851    865    897    928
              947    963    979    996   1013   1030   1047   1067   1071   1077   1121   1170   1216   1248   1281
             1301   1321   1341   1365   1388   1411   1434   1474   1493   1511   1518   1537   1544   1563   1583
             1599   1624   1692   1755   3525   3531   3544   3585   3612   3726   3743
ESCAPE    150#
GETPRI    150#
GETSWR    150#    541#
MOVEI      20#    561    672    679    684    692    697    718    723    731    882    887    892   1100   1149
             1600   1625   1940   1948   1962   1969   1978   1983   1995   2071   2077   2082   2491   2497   2501
             2505   2666   2672   2680   2685   3913
MOVEM      19#    667    670    682    695    709    713    716    721    741    743    751    753    875    885
              920    936    939    955    971    988   1005   1022   1039   1050   1089   1103   1138   1152   1194
             1226   1350   1374   1397   1420   1550   1570   1590   1604   1615   1629   1643   1707   1936   1943
             1946   1959   1972   2001   2069   2075   2470   2474   2476   2495   2508   2511   2670   2678
MULT      150#
NEWTST    150#    661    703    745    764    779    794    809    824    839    853    867    914    930    949
              965    982    999   1016   1033   1053   1080   1129   1188   1220   1264   1284   1304   1324   1344
             1368   1391   1414   1457   1476   1494   1520   1545   1565   1584   1609   1633   1697   1771
POP       150#   1885   3103   3182   3183   3410   3411
PUSH      150#   1844   3082   3163   3169   3371   3373   3394
REPORT    150#
SCOPE      45#    664    706    748    767    782    797    812    827    842    856    870    917    933    952
              968    985   1002   1019   1036   1056   1083   1132   1191   1223   1267   1287   1307   1327   1347
             1371   1394   1417   1460   1479   1497   1523   1548   1568   1587   1612   1636   1700   1774   1801
SETPRI    150#
SETTRA   3444#   3453   3454   3455   3456   3458   3460   3461   3462   3463
SETUP     150#    491
SKIP      150#    675    687    700    736    760    775    790    805    820    835    850    864    896    900
              927    946    962    978    995   1012   1029   1046   1066   1070   1076   1120   1124   1125   1169
             1173   1175   1185   1215   1217   1247   1250   1260   1280   1300   1320   1340   1364   1387   1410
             1433   1438   1473   1492   1510   1517   1536   1543   1562   1582   1695   1758   1760   2561
SLASH     150#
SPACE     150#
STARS     150#    182    193    195    202    215    260    263    661    663    703    705    745    747    764
              766    779    781    794    796    809    811    824    826    839    841    853    855    867    869
              914    916    930    932    949    951    965    967    982    984    999   1001   1016   1018   1033
             1035   1053   1055   1080   1082   1129   1131   1188   1190   1220   1222   1264   1266   1284   1286
             1304   1306   1324   1326   1344   1346   1368   1370   1391   1393   1414   1416   1457   1459   1476
             1478   1494   1496   1520   1522   1545   1547   1565   1567   1584   1586   1609   1611   1633   1635
             1697   1699   1771   1773   1793   1834   2814   2879   2933   2936   3004   3033   3072   3111   3159
             3175   3210   3287   3366   3423
SWRSU     150#    513#
TOUT      451#   3528   3582   3609   3723   3740
TRMTRP   3444#
TYPBIN    150#
TYPDEC    150#   1813
TYPNAM    150#    534
TYPNUM    150#
TYPOCS    150#
TYPOCT    150#   2954   3123   3147   3993
TYPTXT    150#    599    603    901   3499   3553   3557   3561   3788   3974   3999
$CAL.      30#   3464
```

```
$DMAST      30#
$DMDT       30#
$MMAST      30#
$$CMRE     213#     252      253
$$CMTM     213#
$$ESCA     150#
$$NEWT     150#     661      703      745      764      779      794      809      824      839      853      867      914      930      949
           965      982      999     1016     1033     1053     1080     1129     1188     1220     1264     1284     1304     1324     1344
          1368     1391     1414     1457     1476     1494     1520     1545     1565     1584     1609     1633     1697     1771
$$SET     3444#    3453     3454     3455     3456     3458     3460     3461     3462     3463
$$SETM     529#
$$SKIP     150#     700      760      775      790      805      820      835      850      864      900      927      946      962      978
           995     1012     1029     1076     1125     1175     1185     1215     1217     1250     1260     1280     1300     1320     1340
          1364     1387     1410     1438     1473     1492     1517     1543     1562     1582     1760
.EQUAT      30#      40
.HEADE      30#
.KMADR      17#     417
.KSIS       22#     474
.LOADL      25#    3627
.LPAIN      22#    3465
.PUTCS      24#    4020
.RESET      22#    3902
.SETUP      30#     474
.SWRHI      30#     154
.SWRLO     166#
.UTK        29#    3962
.$ACT1      30#     180
.$APTB      30#     261#
.$APTH      30#     191
.$APTY      30#    3364
.$CATC      30#     166
.$CMTA      30#     213
.$EOp       30#    1791
.$ERRO      30#    2877
.$ERRT      30#    3109
.$INLP      28#    3837
.$MMAC      18#
.$OUTL      27#    3798
.$PARM      30#
.$POWE      30#    3157
.$RDOC      30#    3070
.$READ      30#    2931
.$SAVE      30#
.$SCOP      30#    2812
.$SPAC      30#
.$SWDO      30#
.$TLKW      26#    3676
.$TOUt     451#    3881
.$TRAP      30#    3421
.$TYPD      30#    1832
.$TYPE      30#    3285
.$TYPO      30#    3208
```

```
          2467    2468    2472    2473    2478    2515    2517    2526    2527    2528    2566    2571    2581    2588    2591
          2595    2599    2603    2607    2613    2619    2658    2661    2687    2735    2738    2744    2749    2785    2786
          2796    2798    2834    2835    2837    2840    2853    2865    2866    2869    2870    2873    2874    2895    2900
          2921    2924    2954    2978    2983    3012    3013    3040    3041    3056    3057    3058    3059    3080    3081
          3082    3083    3084    3086    3101    3102    3103    3104    3105    3118    3123    3132    3137    3142    3144
          3148    3161    3162    3163    3164    3165    3166    3167    3168    3169    3170    3171    3177    3178    3182
          3183    3184    3185    3186    3187    3188    3189    3190    3193    3233    3241    3242    3243    3249    3256
          3274    3275    3276    3277    3278    3306    3307    3312    3320    3335    3372    3373    3380    3384    3389
          3390    3392    3394    3404    3410    3411    3429    3430    3434    3440    3441    3482    3497    3505    3514
          3518    3527    3567    3602    3635    3636    3637    3641    3648    3659    3660    3683    3684    3687    3688
          3691    3700    3715    3716    3722    3757    3771    3785    3810    3811    3813    3822    3824    3828    3829
          3831    3852    3853    3855    3865    3871    3872    3873    3876    3892    3944    3985    3993    4004    4040
          4046
MOVB       510     550     873    1684    1691    1747    1754    1854    1857    1871    1874    1883    1920    1922    1957
          2022    2490    2536    2622    2623    2624    2626    2628    2629    2630    2631    2634    2739    2740    2741
          2742    2745    2746    2747    2748    2801    2802    2803    2804    2807    2808    2809    2810    2868    2872
          2902    2910    2945    2962    3016    3022    3045    3050    3089    3234    3235    3238    3239    3240    3244
          3247    3248    3267    3317    3345    3353    3367    3368    3370    3432    3566    3571    3573    3575    3580
          3608    3689    3692    3694    3696    3697    3698    3719    3720    3737    3739    3754    3756    3981    4044
          4045    4051
NEG       1853    1999    2108    3245
NOP        468     711     712     738     739    1820    1821    1822
RESET     1818    3912
ROL       3092    3094    3096    3251    3253    3254    3255    3257
RTI        524    1893    2875    2930    2984    3032    3060    3106    3195    3279    3322    3442    3959
RTS       2007    2026    2050    2060    2093    2114    2118    2582    2662    2688    2752    2805    2811    3146    3362
          3412    3435    3603    3623    3661    3701    3758    3786    3830    3874    3898    3924    3951    3954    4055
SUB       1115    1164    1210    1242    1360    1383    1406    1429    1513    1539    1558    1578    1603    1606    1628
          1631    1657    1665    1676    1721    1729    1740    1860    1971    1998    2018    2106    2608    2609    2793
          2795    2901    3387
SWAB       874    2065
TRAP      3444    3453    3454    3455    3456    3458    3460    3461    3462    3463
TST        542     563     568     586     590     593    1652    1667    1693    1716    1731    1756    1776    1865    1875
          1926    1981    1987    2499    2562    2736    2750    2836    2860    2915    2922    2976    2991    3100    3150
          3262    3319    3327    3349    3382    3400    3402    3431    3664    3815    3823    3857    3866    3915    3942
          3947
TSTB       726     890    1867    1881    1931    1954    1992    2080    2556    2576    2652    2675    2683    2847    2943
          2959    3014    3020    3302    3351    3374    3385    3398    3596    3979
.ASCII     257     258    2178    2241    2242    2243    2249    2254    2262    2278    2291    2318    2323    2720
.ASCIZ     256     259     554     602     606     904    1827    2121    2125    2132    2137    2142    2148    2153    2159
          2165    2171    2174    2185    2193    2200    2205    2211    2217    2224    2232    2239    2247    2268    2270
          2272    2285    2288    2302    2306    2310    2314    2335    2339    2343    2349    2352    2356    2360    2366
          2372    2380    2386    2699    2703    2707    2711    2716    2725    3064    3065    3066    3068    3154    3199
          3502    3556    3560    3564    3792    3978    4002    4013
.ASECT      14
.BLKB     3063
.BLKW      449    1898    4060    4062
.BYTE      222     223     228     229     237     238     246     247     248     249     275     276     286     287     294
           295     297     298     300     301     911    1062    1095    1109    1144    1158    1198    1205    1230    1237
          1254    1258    1272    1292    1312    1332    1355    1378    1401    1424    1464    1483    1501    1527    1554
          1574    1647    1711    1826    2038    2039    2040    2041    2045    2049    2056    2182    2183    2451    2568
          2573    2593    2597    2601    2605    2660    2690    2691    2692    2693    2694    2695    2696    2697    2714
          2715    2717    2718    2719    2724    2912    2913    3061    3062    3280    3281    3282    3283    3413    3414
          3415
```

```
                1324  1326  1328  1329  1340  1344  1346  1348  1349  1364  1368  1370  1372  1373  1387
                1391  1393  1395  1396  1410  1414  1416  1418  1419  1433  1438  1457  1459  1461  1462
                1473  1476  1478  1480  1481  1492  1494  1496  1498  1499  1510  1517  1520  1522  1524
                1525  1536  1543  1545  1547  1549  1550  1562  1565  1567  1569  1570  1582  1584  1586
                1588  1589  1609  1611  1613  1614  1633  1635  1637  1638  1695  1697  1699  1701  1702
                1758  1760  1771  1773  1775  1776  1793  1794  1795  1796  1797  1798  1799  1801  1807
                1810  1812  1816  1818  1824  1826  1827  1834  2561  2814  2817  2822  2828  2829  2841
                2843  2844  2845  2847  2848  2849  2858  2860  2868  2870  2875  2876  2877  2879  2882
                2893  2896  2903  2905  2906  2908  2915  2919  2926  2930  2931  2933  2935  2936  2937
                2965  3004  3005  3033  3041  3042  3046  3047  3063  3064  3070  3072  3075  3087  3111
                3126  3142  3159  3169  3170  3175  3182  3183  3191  3193  3195  3199  3210  3287  3308
                3366  3368  3371  3398  3413  3423  3429  3433  3444  3453  3454  3455  3456  3457  3458
                3460  3461  3462  3463  3464  3501  3531  3537  3555  3559  3563  3585  3591  3612  3618
                3726  3732  3743  3749  3791  3962  3977  4001
.IFF              42   161   164   165   166   183   187   189   194   196   203   216   219   222   250
                 261   264   497   537   539   662   663   664   665   675   687   701   704   705   706
                 707   736   746   747   748   749   761   765   766   767   768   776   780   781   782
                 783   791   795   796   797   798   806   810   811   812   813   821   825   826   827
                 828   836   840   841   842   843   851   854   855   856   857   865   868   869   870
                 871   896   901   915   916   917   918   928   931   932   933   934   947   950   951
                 952   953   963   966   967   968   969   979   983   984   985   986   996  1000  1001
                1002  1003  1013  1017  1018  1019  1020  1030  1034  1035  1036  1037  1046  1054  1055
                1056  1057  1058  1066  1070  1077  1081  1082  1083  1084  1120  1124  1126  1130  1131
                1132  1133  1169  1173  1176  1186  1189  1190  1191  1192  1216  1218  1221  1222  1223
                1224  1247  1251  1261  1265  1266  1267  1268  1281  1285  1286  1287  1288  1301  1305
                1306  1307  1308  1321  1325  1326  1327  1328  1341  1345  1346  1347  1348  1365  1369
                1370  1371  1372  1388  1392  1393  1394  1395  1411  1415  1416  1417  1418  1433  1439
                1458  1459  1460  1461  1474  1477  1478  1479  1480  1493  1495  1496  1497  1498  1510
                1518  1521  1522  1523  1524  1536  1544  1546  1547  1548  1549  1563  1566  1567  1568
                1569  1583  1585  1586  1587  1588  1610  1611  1612  1613  1634  1635  1636  1637  1695
                1698  1699  1700  1701  1758  1761  1772  1773  1774  1775  1776  1794  1798  1802  1807
                1810  1826  1835  2561  2815  2842  2845  2846  2849  2876  2880  2882  2896  2926  2931
                2934  2937  3005  3007  3012  3033  3034  3043  3047  3064  3073  3112  3127  3156  3160
                3176  3191  3211  3288  3367  3424  3430  3531  3537  3585  3591  3612  3618  3726  3732
                3743  3749
.IFT             554   602   606   904  2857  2906  3007  3012  3091  3107  3108  3502  3556  3560  3564
                3792  3978  4002
.IFTF            554   602   606   904  2855  2905  2952  3005  3008  3087  3091  3107  3502  3556  3560
                3564  3792  3978  4002
.IIF              30    35    40   158   159   160   162   165   166   172   260   264   498   501   507
                 508   509   511   512   538  1796  1802  1803  1814  1826  1830  2818  2819  2820  2821
                2822  2823  2827  2856  2857  2873  2876  2877  2883  2884  2885  2886  2887  2892  2918
                2926  2931  2934  2955  3056  3064  3070  3124  3149  3364  3452  3453  3454  3455  3456
                3458  3460  3461  3462  3463  3994
.IRP             474   661   703   745   764   779   794   809   824   839   853   867   914   930   949
                 965   982   999  1016  1033  1053  1080  1129  1188  1220  1264  1284  1304  1324  1344
                1368  1391  1414  1457  1476  1494  1520  1545  1565  1584  1609  1633  1697  1771  1845
                1885  2828  3082  3103  3163  3169  3182  3183  3372  3373  3394  3410  3411
.LIST              14    30   150   165   172   250   252   253   254   261   264   474   513   538   541
                 554   563   602   606   661   665   669   672   674   681   684   686   694   697   699
                 703   707   711   715   718   720   723   726   733   743   745   749   753   755   764
                 768   779   783   794   798   809   813   824   828   839   843   853   857   867   871
                 877.  884   887   890   894   904   914   918   922   930   934   938   941   949   953
                 957   965   969   973   982   986   990   999  1003  1007  1016  1020  1024  1033  1037
```

```
                1041    1052    1053    1057    1080    1084    1091    1102    1105    1129    1133    1140    1151    1154    1188
                1192    1196    1220    1224    1228    1264    1268    1284    1288    1304    1308    1324    1328    1344    1348
                1352    1368    1372    1376    1391    1395    1399    1414    1418    1422    1457    1461    1476    1480    1494
                1498    1520    1524    1545    1549    1552    1565    1569    1572    1584    1588    1592    1603    1606    1609
                1613    1617    1628    1631    1633    1637    1645    1697    1701    1709    1771    1775    1802    1818    1938
                1942    1945    1948    1951    1961    1965    1971    1974    1981    1985    1998    2003    2071    2074    2077
                2080    2084    2472    2476    2478    2494    2497    2499    2503    2507    2510    2513    2669    2672    2675
                2680    2683    2687    2822    2926    3033    3444    3452    3453    3454    3455    3456    3457    3458    3459
                3460    3461    3462    3463    3464    3502    3556    3560    3564    3792    3915    3978    4002
.MACRO            17      18      19      20      22      24      25      26      27      28      29      30     166     213     451
                 529    3444
.MCALL            30     150     261     513     541
.NLIST            14      30     150     165     172     250     252     253     254     261     264     474     513     538     541
                 554     563     602     606     661     665     669     672     674     681     684     686     694     697     699
                 703     707     711     715     718     720     723     726     733     743     745     749     753     755     764
                 768     779     783     794     798     809     813     824     828     839     843     853     857     867     871
                 877     884     887     890     894     904     914     918     922     930     934     938     941     949     953
                 957     965     969     973     982     986     990     999    1003    1007    1016    1020    1024    1033    1037
                1041    1052    1053    1057    1080    1084    1091    1102    1105    1129    1133    1140    1151    1154    1188
                1192    1196    1220    1224    1228    1264    1268    1284    1288    1304    1308    1324    1328    1344    1348
                1352    1368    1372    1376    1391    1395    1399    1414    1418    1422    1457    1461    1476    1480    1494
                1498    1520    1524    1545    1549    1552    1565    1569    1572    1584    1588    1592    1603    1606    1609
                1613    1617    1628    1631    1633    1637    1645    1697    1701    1709    1771    1775    1802    1818    1938
                1942    1945    1948    1951    1961    1965    1971    1974    1981    1985    1998    2003    2071    2074    2077
                2080    2084    2472    2476    2478    2494    2497    2499    2503    2507    2510    2513    2669    2672    2675
                2680    2683    2687    2822    2926    3033    3444    3452    3453    3454    3455    3456    3457    3458    3459
                3460    3461    3462    3463    3464    3502    3556    3560    3564    3792    3915    3978    4002
.PAGE            213     329
.PSECT            14
.REM               1      14
.REPT            172     252
.SBTTL            40     154     166     175     180     191     213     261     329     466     491     534     541     619     658
                 659     660     661     703     745     764     779     794     809     824     839     853     867     914     930
                 949     965     982     999    1016    1033    1053    1080    1129    1188    1220    1264    1284    1304    1324
                1344    1368    1391    1414    1457    1476    1494    1520    1545    1565    1584    1609    1633    1697    1771
                1789    1791    1832    1899    2009    2119    2431    2734    2781    2812    2877    2931    3070    3109    3157
                3208    3285    3364    3421    3444
.TITLE            30
.WORD            172     173     174     188     207     208     209     210     211     212     221     224     225     226     227
                 230     231     232     233     234     235     236     239     240     241     250     252     253     266     267
                 268     269     270     271     272     273     277     278     279     292     296     299     302     303     304
                 305     306     307     308     309     310     311     312     313     314     315     316     317     318     319
                 320     321     322     323     324     426     429     431     433     435     437     439     441     443     444
                 446     448     563     598     669     672     674     681     684     686     694     697     699     711     715
                 718     720     723     726     733     743     745     753     755     877     884     887     890     894     922
                 938     941     957     973     990    1007    1024    1041    1052    1091    1102    1105    1140    1151    1154
                1196    1228    1352    1376    1399    1422    1552    1572    1592    1603    1606    1617    1628    1631    1645
                1709    1807    1810    1825    1938    1942    1945    1948    1951    1961    1965    1971    1974    1981    1985
                1998    2003    2071    2074    2077    2080    2084    2472    2476    2478    2494    2497    2499    2503    2507
                2510    2513    2669    2672    2675    2680    2683    2687    3107    3135    3140    3192    3194    3284    3314
                3361    3396    3451    3508    3605    3625    3760    3826    3827    3834    3869    3879    3900    3901    3915
                3925    3926    3956    3960    3990    3991    3996    4007    4008    4011    4012    4042    4043    4049    4050
                4056
```

000000

ERRORS DETECTED: 0

```
MAINDEC-11-DRLPE-A      MACY11 27(654)  14-DEC-77  20:24  PAGE 103
DRLPE.P11                                                          SEQ 0112

 *DRLPE,DRLPE/SOL/CRF=DRLPA.MAC,DRLPE
 RUN-TIME:  26 17 2 SECONDS
 CORE USED:  40K
EOF1DRLPEASEQ              00010000      780223            PDP10 411        2
```