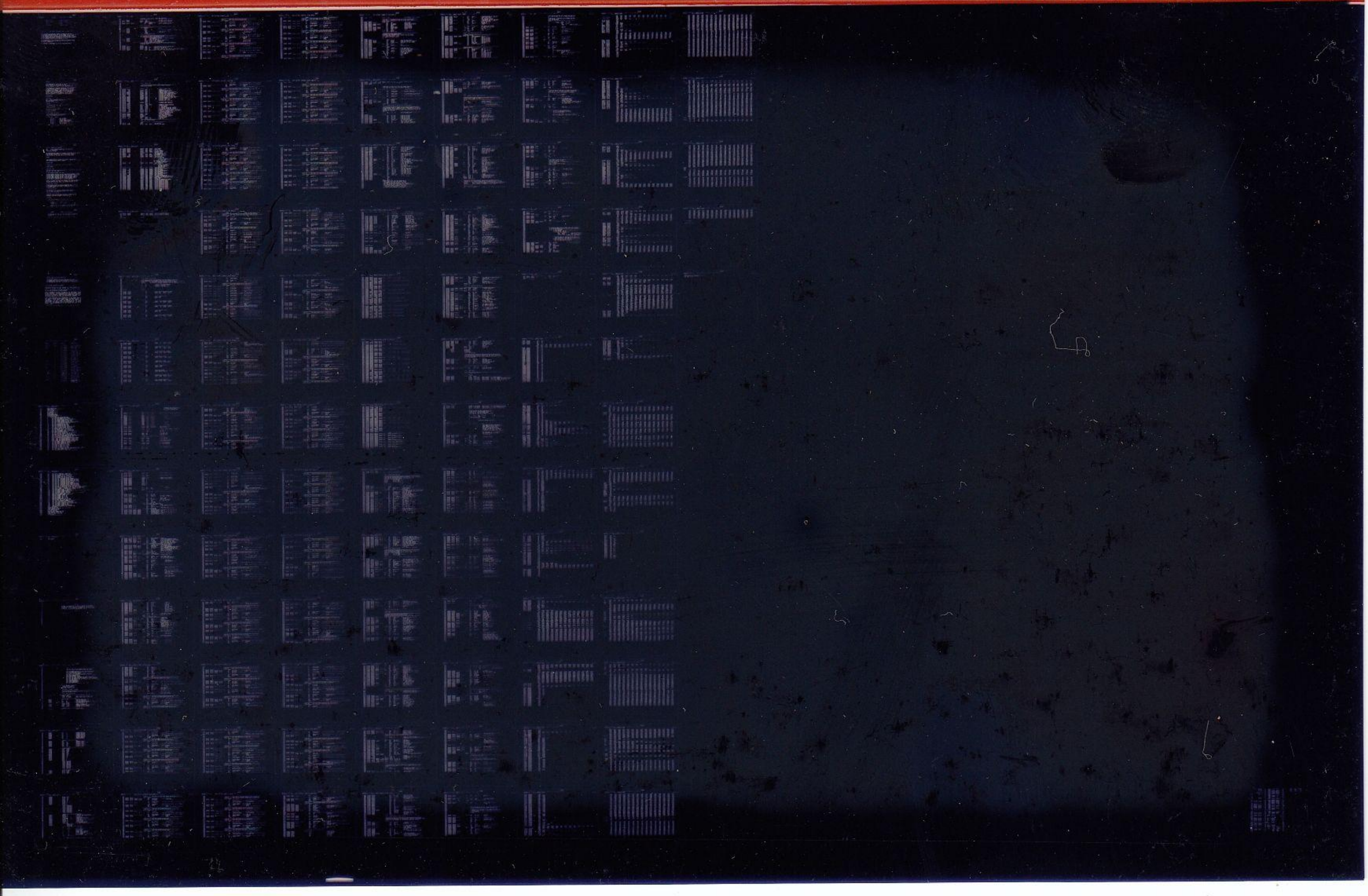
LPA/AR11

DIAGNOSTIC TEST I
MD-11-DRLPC-A

EP-DRLPC-A-DL

FICHE 1 OF 1

MAR 1978
digital
MADE IN USA



BO1

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

PRODUCT NAME: LPA/AR11 DIAGNOSTIC TEST I

DATE: JANUARY 1976

MAINTAINER: DIAGNOSTIC GROUP

1. ABSTRACT

THIS PROGRAM IS A LOGIC TEST OF THE AR11 LAB SYSTEM OPTION.
MOST FUNCTIONS OF THE OPTION WILL BE
TESTED. DUE TO THE FLEXIBILITY OF THE OPTION, THE OPERATOR
MAY BE REQUIRED TO SUPPLY OPTION CHARACTERISTICS.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZARA-[REV]".
IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT
THE AR11 OPTION WHEN IT IS ON THE LPA-11XX I/O BUS.
NO RECABLING IS NEEDED. SOME TESt DONs IN THE ORIGINAL
DIAGNOSTIC SUCH AS ARBITRATION TESiS, WERE DELETED
AS THEY COULD NOT BE CHECKED. IF THIS DIAGNOSTIC
DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO
RUN "MD-11-DZARA", YOU SHOULD RUN "MD-11-DRLPA"
BEFORE RUNNING THIs DIAGNOSTIC. PLEASE READ SECTION 10.

2. REQUIREMENTS

2.1 EQUIPTMENT

PDP-11 FAMILY COMPUTER WITH 16K WORDS OF MEMORY
AR11 HEX OPTION MODULE INSTALLED
LPA-11X SYSTEM
TELETYPE (OR EQUIVALENT)

2.2 STORAGE

THIS PROGRAM USES LESS THAN 16K 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 TYPOUTS
SW 12 = 1	STORAGE SCOPE CONNECTED
SW 11 = 1	INHIBIT INTERATIONS
SW 10 = 1	ALL 1'S A TO D TESt JUMPER INSTALLED
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

SEQ 0003

200 IS THE STARTING ADDRESS OF THE LOGIC TEST.
204 IS THE RESTART ADDRESS OF THE LOGIC TEST.

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

EXTERNAL A TO D START INPUT MUST NOT BE CONNECTED.

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 4 MINUTES FOR COMPLETION AND WILL TYPE 'END PASS'.

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 ADBRL CONTAINS THE AR11 A TO D BR LEVEL <300><6>
LOCATION CKBRL CONTAINS THE AR11 CLOCK BR LEVEL <300><6>
LOCATION VCBRL CONTAINS THE AR11 SCOPE BR LEVEL <200><4>
LOCATION $FILLS CONTAINS THE TTY FILLER CHARACTER COUNT
LOCATION $NUIL CONTAINS THE TTY FILLER CHARACTER

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

8.3 MULTIPLE AR11 TESTING

A PROVISION IS MADE FOR TESTING SEQUENTIAL AR11. STARTING AT BUS ADDRESS/VECTOR DEFINED BY $BASE AND $VECT1. THE HEADER TYPEOUT WILL INFORM THE OPERATOR OF THE NUMBER OF AR11'S FOUND.

8.4 XXDP/ACT/APT NOTES

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

8.5 USER LINK TO I/O DEVICE

SEQ 0004

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=   "OCTAL ADDRS"
XXXXXX
```

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

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

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

E01

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".

10. 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 BMC-11 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 LPA-11KX DIAGNOSTIC KIT WILL INCLUDE:

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/DMC BASIC MICRO-CPU R/W TEST
	B	MD-11-DZLPM	LPA/DMC 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
```

```
                        .REM    !

                        THIS IS A LIST OF TEST DELETED FROM THIS DIAGNOSTIC.
                        THESE TESt COULD NOT BE DONE THROUGH THE LPA-11.

                        TEST FOR PROPER SELECTION OF THE LOW BYTE OPERATION
                        TEST FOR PROPER SELECTION OF HIGH BYTE OPERATIOIN
                        A TO D PRE-INTERRUPT SETUP
                        TEST THAT A TO D INTERRUPTS AT LEVEL INDICATED
                        TEST THAT A TO D DOES NOT INTERRUPT AT LEVEL INDICATED
                        CLOCK PRE-INTERRIPT SETUP
                        SCOPE PRE-INTERRUPT SETUP
                        TEST THAT THE CLOCK INTERRUPTS AT LEVEL INDICATED -1
                        TEST THAT THE CLOCK DOES NOT INTERRUPT AT LEVEL INDICATED
                        TEST THAT THE DISPLAY DOES INTERRUPT AT LEVEL INDICATED
                        TEST THAT THE DISPLAY DOES NOT INTERRUPT AT LEVEL INDICATED
                        TEST 1MHZ REPEATABILTY
                        TEST 100KHZ REPEATABILTY
                        TEST 10KHZ REPEATABILTY
                        TEST 1KHZ REPEATABILTY
                        TEST AD GO (BIT 0) CAN BE SET AND CLEARED

                        !
                .TITLE  MAINDEC-11-DRLPC-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 Or 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
```

```
                        R1=     %1              ;;GENERAL REGISTER
                        R2=     %2              ;;GENERAL REGISTER
                        R3=     %3              ;;GENERAL REGISTER
                        R4=     %4              ;;GENERAL REGISTER
                        R5=     %5              ;;GENERAL REGISTER
                        R6=     %6              ;;GENERAL REGISTER
                        R7=     %7              ;;GENERAL REGISTER
                        SP=     %6              ;;STACK POINTER
                        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
                        000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                        000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        000014          TBITVEC=14              ;;"T" BIT
                        000014          TRTVEC= 14              ;;TRACE TRAP
                        000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                        000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                        000024          PWRVEC= 24              ;;POWER FAIL
                        000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                        000034          TRAPVEC=34              ;;"TRAP" TRAP
                        000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
                        000064          TPVEC=  64              ;;TTY PRINTER VECTOR
                        000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
                        170400                  ABASE=170400
                        000340                  AVECT1=340
                        000200                  APRIOR=200
                                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                                        ;*
                                        ;*      SWITCH                  USE
                                        ;*      ------          --------------------
                                        ;*        15            HALT ON ERROR
                                        ;*        14            LOOP ON TEST
                                        ;*        13            INHIBIT ERROR TYPEOUTS
                                        ;*        12            STORAGE SCOPE CONNECTED
                                        ;*        11            INHIBIT ITERATIONS
                                        ;*        10            ALL 1'S JUMPER IS INSTALLED
                                        ;*         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  001672                 JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001676                 JMP     @#BEGIN1        ;JUMP TO rESTART ADDRESS

                                .SBTTL  ACT11 HOOKS

                                ;;*****************************************************************
                                ;HOOKS REQUIRED BY ACT11
                 000210                 $SVPC=.                 ;SAVE PC
                 000046                 .=46
000046  013746                         $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                 000052                 .=52
000052  000000                         .WORD   0               ;;2)SET LOC.52 TO zERO
                 000210                 .=$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  001176                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000010                  $TSTM:  .WORD   10      ;;RUN TIM OF LONGEST TEST
001006  000010                  $PASTM: .WORD   10      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000200                  $UNITM: .WORD   200     ;;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                  $REGAD: .WORD   0       ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REGO) WAS OBTAINED
001162  000000                  $REGO:  .WORD   0       ;;CONTAINS (($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  077                     $QUES:  .ASCII  /?/     ;;QUESTION MARK
001173  015                     $CRLF:  .ASCII  <15>    ;;CARRIAGE RETURN
001174  000012                  $LF:    .ASCIZ  <12>    ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*****************************************************************
                                .EVEN
001176                  $MAIL:                          ;;APT MAILBOX
001176  000000                 $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
001200  000000                 $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
```

```
001202  000000          $TESTN:  .WORD   ATESTN   ;;TEST NUMBER
001204  000000          $PASS:   .WORD   APASS    ;;PASS COUNT
001206  000000          $DEVCT:  .WORD   ADEVCT   ;;DEVICE COUNT
001210  000000          $UNIT:   .WORD   AUNIT    ;;I/O UNIT NUMBER
001212  000000          $MSGAD:  .WORD   AMSGAD   ;;MESSAGE ADDRESS
001214  000000          $MSGLG:  .WORD   AMSGLG   ;;MESSAGE LENGTH
001216                  $ETABLE:                  ;;APT ENVIRONMENT TABLE
001216     000          $ENV:    .BYTE   AENV     ;;ENVIRONMENT BYTE
001217     000          $ENVM:   .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
001220  000000          $SWREG:  .WORD   ASWREG   ;;APT SWITCH REGISTER
001222  000000          $USWR:   .WORD   AUSWR    ;;USER SWITCHES
001224  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
001226     000          $MAMS1:  .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
001227     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
001230  000000          $MADR1:  .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
                        ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001232     000          $MAMS2:  .BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
001233     000          $MTYP2:  .BYTE   AMTYP2   ;;MEM.TYPE,BLK#2
001234  000000          $MADR2:  .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
001236     000          $MAMS3:  .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
001237     000          $MTYP3:  .BYTE   AMTYP3   ;;MEM.TYPE,BLK#3
001240  000000          $MADR3:  .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
001242     000          $MAMS4:  .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
001243     000          $MTYP4:  .BYTE   AMTYP4   ;;MEM.TYPE,BLK#4
001244  000000          $MADR4:  .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
001246  000340          $VECT1:  .WORD   AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001250  000000          $VECT2:  .WORD   AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001252  170400          $BASE:   .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001254  000000          $DEVM:   .WORD   ADEVM    ;;DEVICE MAP
001256  000000          $CDW1:   .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
001260  000000          $CDW2:   .WORD   ACDW2    ;;CONTROLLER DESCRIPTION WORD#2
001262  000000          $DDW0:   .WORD   ADDW0    ;;DEVICE DESCRIPTOR WORD#0
001264  000000          $DDW1:   .WORD   ADDW1    ;;DEVICE DESCRIPTOR WORD#1
001266  000000          $DDW2:   .WORD   ADDW2    ;;DEVICE DESCRIPTOR WORD#2
001270  000000          $DDW3:   .WORD   ADDW3    ;;DEVICE DESCRIPTOR WORD#3
001272  000000          $DDW4:   .WORD   ADDW4    ;;DEVICE DESCRIPTOR WORD#4
001274  000000          $DDW5:   .WORD   ADDW5    ;;DEVICE DESCRIPTOR WORD#5
001276  000000          $DDW6:   .WORD   ADDW6    ;;DEVICE DESCRIPTOR WORD#6
001300  000000          $DDW7:   .WORD   ADDW7    ;;DEVICE DESCRIPTOR WORD#7
001302  000000          $DDW8:   .WORD   ADDW8    ;;DEVICE DESCRIPTOR WORD#8
001304  000000          $DDW9:   .WORD   ADDW9    ;;DEVICE DESCRIPTOR WORD#9
001306  000000          $DDW10:  .WORD   ADDW10   ;;DEVICE DESCRIPTOR WORD#10
001310  000000          $DDW11:  .WORD   ADDW11   ;;DEVICE DESCRIPTOR WORD#11
001312  000000          $DDW12:  .WORD   ADDW12   ;;DEVICE DESCRIPTOR WORD#12
```

```
   342  001314  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
   343  001316  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
   344  001320  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
   345
   346
   347  001322                          $ETEND:
   348
```

```
                                .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


001322                          $ERRTB:

                                ;ITEM   1
001322  014620                          EM1             ;A TO D STATUS REGISTER IN ERROR
001324  015414                          DH1             ;ERRPC  ADCS    ADSTAT  EXPECTED
001326  016220                          DT1             ;$ERRPC ADCS    $BDDAT  $GDDAT
001330  000000                          0

                                ;ITEM   2
001332  014660                          EM2             ;A TO d INTERRUPT ERROR
001334  015454                          DH2             ;ERRPC  ADCS
001336  016232                          DT2             ;$ERRPC ADCS
001340  000000                          0

                                ;ITEM   3
001342  014707                          EM3             ;CLOCK STATUS REGISTER IN ERROR
001344  015472                          DH3             ;ERRPC    CSR   CKSTAT  EXPECTED
001346  016240                          DT3             ;$ERRPC   CSR   $BDDAT  $GDDAT
001350  000000                          0

                                ;ITEM   4
001352  014746                          EM4             ;CLOCK PRESET REGISTER IN ERROR
001354  015532                          DH4             ;ERRPC    CSB   CKBUFF  EXPECTED
001356  016252                          DT4             ;$ERRPC   CSB   $BDDAT  $GDDAT
001360  000000                          0

                                ;ITEM   5
001362  015005                          EM5             ;CLOCK INTERRUPT ERROR
001364  015572                          DH5             ;ERRPC  CSR
001366  016264                          DT5             ;$ERRPC CSR
001370  000000                          0

                                ;ITEM   6
001372  015033                          EM6             ;CLOCK COUNTER REGISTER IN ERROR
001374  015607                          DH6             ;ERRPC  CSC  CKCNTR    EXPECTED
001376  016272                          DT6             ;$ERRPC   CSC    $BDDAT   $GDDAT
001400  000000                          0

                                ;ITEM   7
001402  015073                          EM7             ;CLOCK COUNTED IN ERROR
```

```
 403  001404  015607                    DH6             ;ERRPC    CSC    CKCNTR  EXPECTED
 404  001406  016272                    DT6             ;$ERRPC   CSC    $BDDAT  $GDDAT
 405  001410  000000                    0
 406
 407                              ;ITEM  10
 408  001412  015122                    EM10            ;CLOCK REPEATABILITY FAILED
 409  001414  015647                    DH10            ;ERRPC  CSC     TIME1   TIME2
 410  001416  016272                    DT6             ;$ERRPC  CSC    $BDDAT  $GDDAT
 411  001420  000000                    0
 412
 413                              ;ITEM  11
 414  001422  015155                    EM11            ;VC STATUS REGISTER IN ERROR
 415  001424  015705                    DH11            ;ERRPC  VCADR   VCSTAT EXPECTED
 416  001426  016304                    DT11            ;$ERRPC VCSTAT  $BDDAT  $GDDAT
 417  001430  000000                    0
 418
 419                              ;ITEM  12
 420  001432  015211                    EM12            ;X REGISTER IN ERROR
 421  001434  015745                    DH12            ;ERRPC  VCXPOS   X POS.  EXPECTED
 422  001436  016316                    DT12            ;$ERRPC VCXPOS   $BDDAT  $GDDAT
 423  001440  000000                    0
 424
 425                              ;ITEM  13
 426  001442  015235                    EM13            ;Y REGISTER IN ERROR
 427  001444  016006                    DH13            ;ERRPC VCYPOS   Y POS. EXPECTED
 428  001446  016330                    DT13            ;$ERRPC VCYPOS $BDDAT $GDDAT
 429  001450  000000                    0
 430
 431                              ;ITEM  14
 432  001452  015261                    EM14            ;SCOPE INTERRUPT ERROR
 433  001454  016047                    DH14            ;ERRPC  VCADR
 434  001456  016342                    DT14            ;$ERRPC VCSTAT
 435  001460  000000                    0
 436
 437                              ;ITEM  15
 438  001462  015307                    EM15            ;DEVICE BUS ERROR
 439  001464  016066                    DH15            ;ERRPC BASE ADDRESS ACTUAL ADDRESS
 440  001466  016350                    DT15            ;$ERRPC ARBADD  $BDDAT
 441  001470  000000                    0
 442
 443                              ;ITEM  16
 444  001472  015330                    EM16            ;INCORRECT A/D BUFFER DATA
 445  001474  016116                    DH16            ;ERRPC  ADDBR   BUFFER  EXPECTED
 446  001476  016360                    DT16            ;$ERRPC ADDBR   $BDDAT  $GDDAT
 447  001500  000000                    0
 448
 449                              ;ITEM  17
 450  001502  015362                    EM17            ;DUAL REGISTER ADDRESSING
 451  001504  016157                    DH17            ;ERRPC  BUFADR  READ    EXPECTED
 452  001506  016372                    DT17            ;$ERRPC BUFADR  $BDDAT  $GDDAT
 453  001510  000000                    0
 454
 455                                     ;
 456                                     ;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.
                                                ;

001512                          LPCI:
001512  170460                  KMAD0:  .WORD   170460          ;BASE KMC ADDR. MAY BE PATCHED BY USER.

001514                          LPMR:
001514  170461                  KMAD1:  .WORD   170460+1                ;>DO NOT        <;KMC-CSR ADDR
001516                          LPC0:
001516  170462                  KMAD2:  .WORD   170460+2                ;>PATCH         <;
001520                          LPS0:
001520  170463                  KMAD3:  .WORD   170460+3                ;>THIS AREA     <
001522                          LPADL:
001522  170464                  KMAD4:  .WORD   170460+4                ;
001524                          LPADH:
001524  170465                  KMAD5:  .WORD   170460+5                ;>DO NOT        <
001526                          LPMS1:
001526  170466                  KMAD6:  .WORD   170460+6                ;>PATCH         <
001530                          LPMS2:
001530  170467                  KMAD7:  .WORD   170460+7                ;>THIS AREA     <

001532  000340                  VECTOR: .WORD   AVECT1&777      ;BASE VECTOR OF KMC
001534  000344                  VECTPS: .WORD   4+AVECT1&777    ;VECOTR ADDR.+2

001536  000004                  VERSN:  .WORD   4               ;CURRENT VERSION NUMBER OF MICROCODE.

001540  000000                  .DVLS:  .WORD   0               ;/DEVICE LIST OF I/O ADDR. DEFINED
001542  000020                          .BLKW   16.             ;/BY INIT.

001602  000300                  ADBRL:  300                     ;A TO D BR LEVEL
001604  000300                  CKBRL:  300                     ;CLOCK BR LEVEL
001606  000200                  VCBRL:  200                     ;SCOPE BR LEVEL

                                ;PROGRAM WILL CHANGE THE NEXT 16 LOCATIONS

001610  170400                  ADCS:   170400                  ;A TO D STATUS REG
001612  170402                  ADDBR:  170402                  ;A TO D BUFFER

001614  170404                  CSR:    170404                  ;CLOCK STATUS REGISTER
001616  170406                  CSB:    170406                  ;CLOCK PRESET BUFFER

001620  170410                  VCSTAT: 170410                  ;VC STATUS REGISTER
001622  170412                  VCXREG: 170412                  ;VC X AXIS REGISTER
001624  170414                  VCYREG: 170414                  ;VC Y AXIS REGISTER

001626  170416                  CSC:    170416                  ;CLOCK COUNTER REGISTER

001630  000340                  ADINT:  340                     ;A TO D INTERRUPT VECTOR
001632  000342                  ADINT1: 342
```

```
                                KWIV:   344                     ;CLOCK INTERRUPT VECTOR
                                KWIVS:  346
                                VCIV:   350                     ;SCOPE INTERRUPT VECTOR
                                VCIVS:  352
                                ARBADD: 170400                  ;CURRENT DEVICE ADDRESS
                                ARBVCT: 340                     ;CURRENT DEVICE VECTOR
                                NMBEXT: 0                       ;NUMBER OF ADDITIONAL AR11'S
                                NBEXT:  0
                                COUNT:  0
                                DELAY:  0
                                TEMP:   0
                                SWITCH: 0
                                BRLEV1: 0
                                BRLEV2: 0
                                $TMDAT: 0

                                BEGIN:  CLR     R0                      ;CLEAR R0
                                        BR      RBEG
                                BEGIN1: MOV     #-1,R0                  ;LOAD R0
                                RBEG:
                                        ;
                                        ;THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
                                        ;
                                        MOV     R0,-(SP)
                                        MOV     R1,-(SP)
                                        MOV     KMADD,R0        ;GET KMC-11 ADDRESS.
                                        MOV     #KMAD1,R1       ;GET ADDR. OF ADDR. LIST.

                                64$:    INC     R0              ;UPDATE ADDR.
                                        MOV     R0,(1)+         ;WRITE ADDR.
                                        CMP     R1,#KMAD7+2     ;DONE ALL ADDRESSES?
                                        BNE     64$             ;NO - DO NEXT ADDR.
                                        CLR     .DVLS           ;CLR ADDR. LIST.
                                        MOV     (SP)+,R1
                                        MOV     (SP)+,R0
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
                                        MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
                                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
                                        CMP     #SWR,R6 ;;DONE?
                                        BNE     .-6             ;;LOOP BACK IF NO
                                        MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
                                        MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
                                        MOV     #340,@#IOTVEC+2 ;;LEVEL 7
                                        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
                                        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
                                        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
                                        MOV     #340,@#TRAPVEC+2;LEVEL 7
                                        MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
```

```
                                                ;;LEVEL 7
                                                ;;SETUP END-OF-PROGRAM COUNTER
                                                ;;INITIALIZE NUMBER OF ITERATIONS
                                                ;;CLEAR THE ESCAPE ON ERROR ADDRESS
                                                ;;ALLOW ONE ERROR PER TEST
                                                ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
                                                ;;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.
                                                ;;SAVE ERROR VECTOR
                                                ;;SET UP ERROR VECTOR
                                                ;;SETUP FOR A HARDWARE SWICH REGISTER
                                                ;;AND A HARDWARE DISPLAY REGISTER
                                                ;;TRY TO REFERENCE HARDWARE SWR
                                                ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                ;;AND  THE HARDWARE SWR IS NOT = -1
                                                ;;BRANCH IF NO TIMEOUT
                                                ;;SET UP FOR TRAP RETURN
                                                ;;POINT TO SOFTWARE SWR
                                                ;;RESTORE ERROR VECTOR
                                                ;;CLEAR PASS COUNT
                                                ;;TEST USER SIZE UNDER APT
                                                ;;YES,USE NON-APT SWITCH
                                                ;;NO,USE APT SWITCH REGISTER
                                                ;LOAD STACK
                                                ;LOAD BUS ERROR
                                                ;LOAD STARTING ADDRESS
                                                ;CLEAR COUNT
                                                ;/READ DEVICE REG $TMDAT,PUT DATA IN $GDDAT.
                                                ;EXIST, UPDATE TEST ADDRESS
                                                ;UPDATE # OF AR11'S
                                                ;TEST IF FIRST DOES EXIST
                                                ;BR
                                                ;FIRST AR11 DOES NOT EXIST
                                                ;CHECK THE PROGRAM DEVICE ADDRESS
                                                ;ADJUST R3
                                                ;SAVE THE NUMBER OF ADDITIONAL AR11
                                                ;RESET BUS ERROR
                                                ;LOAD FIRST ADDRESS
                                                ;LOAD FIRST VECTOR
                                                ;LOAD NUMBER OF AR11'S
                                                ;LOAD R2
```

```
                                                MOV     #230,R1                 ;LOAD R1
                                                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
                                                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$:
                                                TYPE    ,65$                    ;;TYPE ASCIZ STRING
                                                BR      64$                     ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <15><12>/AR-11 DIAGNOSTIC TEST I/
                                        64$:
                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
                                                INC     #-1                     ;;FIRST TIME?
                                                BNE     66$                     ;;BRANCH IF NO
                                                CMP     #$ENDAD,@#42            ;;ACT-11?
                                                BEQ     66$                     ;;BRANCH IF YES
                                                TYPE    ,67$                    ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
                                                TST     @#42                    ;;ARE WE RUNNING UNDER XXDP/ACT?
                                                BNE     68$                     ;;BRANCH IF YES
                                                CMPB    $ENV,#1                 ;;ARE WE RUNNING UNDER APT?
                                                BEQ     68$                     ;;BRANCH IF YES
                                                CMP     SWR,#SWREG              ;;SOFTWARE SWITCH REG SELECTED?
                                                BNE     69$                     ;;BRANCH IF NO
                                                GTSWR                           ;;GET SOFT-SWR SETTINGS
                                                BR      69$
                                        68$:    MOVB    #1,$AUTOB               ;;SET AUTO-MODE INDICATOR
                                        69$:
                                                BR      66$                     ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  <CRLF><15><12>/MAINDEC-11-DRLPC-A/<15><12><CRLF>
                                        66$:
                                                MOV     NMBEXT,-(SP)            ;PUSH ON STACK
                                                TYPOS                           ;TYPE OCTAL
                                                .WORD   2
                                                TYPE    ,71$                    ;;TYPE ASCIZ STRING
                                                BR      70$                     ;;GET OVER THE ASCIZ
                                        ;;71$:  .ASCIZ  /(8) ADDITIONAL AR11'S CONNECTED/<15><12>
                                        70$:

                                        4$:     NOP
                                                MOV     #ADCS,R0                ;LOAD POINTER
                                        10$:    MOV     ARBADD,(R0)+
                                                CMP     #ADINT,R0               ;TEST FOR END
                                                BNE     10$
                                        11$:    MOV     ARBVCT,(R0)+            ;LOAD VECTOR
                                                CMP     #ARBADD,R0
```

```
                001373                  BNE     11$
                012700  001612          MOV     #ADDBR,R0
                012701  000002          MOV     #2,R1
                060120          12$:    ADD     R1,(R0)+
                005721                  TST     (R1)+
                022701  000020          CMP     #20,R1
                001373                  BNE     12$
                005720                  TST     (R0)+
                012701  000002          MOV     #2,R1
                060120          13$:    ADD     R1,(R0)+
                005721                  TST     (R1)+
                022701  000014          CMP     #14,R1
                001373                  BNE     13$
                                ;;**************************************************************
                                ;*TEST 1         TEST EACH BUS ADDRESS FOR TIMEOUT
                                ;;**************************************************************
                000004          TST1:   SCOPE

                005037  001540          CLR     .DVLS
                013737  001610  001126  MOV     ADCS,$BDDAT     ;LOAD WITH BUS ADDRESS TO bE TESTED
                                2$:

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

                005737  022222          TST     $AERR           ;TEST ADDRESS
                001011                  BNE     1$
                023737  001126  001626  CMP     $BDDAT,CSC      ;TEST FOR LAST
                001404                  BEQ     3$              ;BR IF DONe
                062737  000002  001126  ADD     #2,$BDDAT       ;MAKE NEXT ADDRESS
                000761                  BR      2$              ;TRY MORE
                                3$:
                000403                  BR      TST2            ;;BR TO NEXT TEST

                104015          1$:     ERROR   15              ;DEVICE BUS ERROR
                000137  013572          JMP     BYPASS          ;DONT TEST ANY MORE


                                ;;**************************************************************
                                ;*TEST 2         TEST THAT THE PRESET BUFFER CAN HOLD #0
                                ;;**************************************************************
                000004          TST2:   SCOPE
                012737  000000  001124  MOV     #0,$GDDAT               ;LOAD EXPECTED

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

                                ;*      MOV     @CSB,$BDDAT     ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
                023737  001124  001126  CMP     $GDDAT,$BDDAT           ;COMPARE
                001401                  BEQ     TST3            ;;BR IF EQUAL
                104004                  ERROR   4                       ;ERROR, COUNTER PRESET FAILED TO CLEAR


                                ;;**************************************************************
                                ;*TEST 3         TEST THE COUNTER PRESET BUFFER CAN HOLD #377
                                ;;**************************************************************
                000004          TST3:   SCOPE
```

```
                                        MOV     #377,$GDDAT             ;LOAD EXPECTED

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

                        ;*      MOV     @CSB,$BDDAT     ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
003112  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003120  001401                         BEQ     TST4            ;;BR IF EQUAL
003122  104004                         ERROR   4               ;ERROR, COUNTER PRESET FAILED TO LOAD

                        ;;*************************************************************
                        ;*TEST 4        TEST THAT PRESET BUFFER CAN HOLD #125
                        ;*************************************************************
003124  000004          TST4:   SCOPE
003126  012737  000125  001124          MOV     #125,$GDDAT             ;LOAD EXPECTED

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

                        ;*      MOV     @CSB,$BDDAT     ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
003154  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003162  001401                         BEQ     TST5            ;;BR IF EQUAL
003164  104004                         ERROR   4               ;ERROR, COUNTER PRESET FAILED TO LOAD

                        ;;*************************************************************
                        ;*TEST 5        TEST THAT PRESET BUFFER CAN HOLD #252
                        ;*************************************************************
003166  000004          TST5:   SCOPE
003170  012737  000252  001124          MOV     #252,$GDDAT             ;LOAD EXPECTED

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

                        ;*      MOV     @CSB,$BDDAT     ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
003216  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003224  001401                         BEQ     TST6            ;;BR IF EQUAL
003226  104004                         ERROR   4               ;ERROR, COUNTER PRESET FAILED TO LOAD

                        ;;*************************************************************
                        ;*TEST 6        TEST THAT PRESET BUFFER CAN HOLD A COUNT PATTERN
                        ;*************************************************************
003230  000004          TST6:   SCOPE
003232  012737  003244  001110          MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
003240  005037  001124                 CLR     $GDDAT                  ;CLEAR PATTERN
003244                  1$:

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

                        ;*      MOV     @CSB,$BDDAT     ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
003264  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003272  001401                         BEQ     2$              ;;BR IF EQUAL
003274  104004                         ERROR   4               ;PRESET BUFFER FAILED TO HOLD A COUNT PATTERN

003276  105237  001124          2$:     INCB    $GDDAT                  ;UPDATE PATTERN
003302  001360                         BNE     1$                      ;BR UNTIL DONE

                        ;;*************************************************************
```

```
                                        ;*TEST 7        TEST INIT TO CLEAR COUNT PRESET BUFFER WHEN IT IS =-1
                                        ;;*************************************************************
003304  000004                          TST7:   SCOPE
003306  012737  000020  001166                  MOV     #20,$TIMES      ;;DO 20 ITERATIONS
003314  012737  000000  001124                  MOV     #0,$GDDAT               ;LOAD EXPECTED
003322  012737  177777  001670                  MOV     #-1,$TMDAT

                                        ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
003340  004737  023174                          JSR     PC,$RESET

                                        ;*      MOV     @CSB,$BDDAT     ;/READ DEVICE REG CSB,PUT DATA IN $BDDAT.
003354  005737  001126                          TST     $BDDAT
003360  001401                          ;;BR IF EQUAL
003362  104004                          ;ERROR, INIT FAILED TO cLEAR CSB

                                        ;;*************************************************************
                                        ;*TEST 10        TEST THAT THE COUNTER CAN HOLD #0
                                        ;;*************************************************************
003364  000004                          TST10:  SCOPE
003366  012737  000000  001124                  MOV     #0,$GDDAT               ;LOAD EXPECTED

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

                                        ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
003414  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
003422  001401                          ;;BR IF EQUAL
003424  104006                          ;ERROR, COUNTER FAILED TO CLEAR

                                        ;;*************************************************************
                                        ;*TEST 11        TEST THE COUNTER CAN HOLD #377
                                        ;;*************************************************************
003426  000004                          TST11:  SCOPE
003430  012737  000377  001124                  MOV     #377,$GDDAT             ;LOAD EXPECTED

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

                                        ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
003456  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
003464  001401                          ;;BR IF EQUAL
003466  104006                          ;ERROR, COUNTER FAILED TO LOAD


                                        ;;*************************************************************
                                        ;*TEST 12        TEST THAT COUNTER CAN HOLD #125
                                        ;;*************************************************************
003470  000004                          TST12:  SCOPE
003472  012737  000125  001124                  MOV     #125,$GDDAT             ;LOAD EXPECTED

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

                                        ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
003520  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
003526  001401                          ;;BR IF EQUAL
003530  104006                          ;ERROR, COUNTER FAILED TO LOAD
```

```
                                ;;*****************************************************************
                                ;*TEST 13        TEST THAT COUNTER CAN HOLD #252
                                ;;*****************************************************************
003532  000004                  TST13:  SCOPE
003534  012737  000252  001124          MOV     #252,$GDDAT             ;LOAD EXPECTED

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

                                ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
003562  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003570  001401                        BEQ     TST14           ;;BR IF EQUAL
003572  104006                        ERROR   6                       ;ERROR, COUNTER FAILED TO LOAD

                                ;;*****************************************************************
                                ;*TEST 14        TEST THAT COUNTER CAN HOLD A COUNT PATTERN
                                ;;*****************************************************************
003574  000004                  TST14:  SCOPE
003576  012737  003610  001110          MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
003604  005037  001124                  CLR     $GDDAT                  ;CLEAR PATTERN
003610                          1$:

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

                                ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
003630  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003636  001401                        BEQ     2$              ;;BR IF EQUAL
003640  104006                        ERROR   6                       ;COUNTER FAILED TO HOLD A COUNT PATTERN

003642  105237  001124          2$:     INCB    $GDDAT                  ;UPDATE PATTERN
003646  001360                        BNE     1$                      ;BR UNTIL DONE

                                ;;*****************************************************************
                                ;*TEST 15        TEST INIT TO CLEAR COUNTER WHEN IT IS =-1
                                ;;*****************************************************************
003650  000004                  TST15:  SCOPE
003652  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED
003660  012737  177777  001670          MOV     #-1,$TMDAT

                                ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
003676  004737  023174                  JSR     PC,$RESET

                                ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
003712  005737  001126                  TST     $BDDAT
003716  001401                        BEQ     TST16           ;;BR IF EQUAL
003720  104006                        ERROR   6                       ;ERROR, INIT FAILED TO CLEAR COUNTER

                                ;;*****************************************************************
                                ;*TEST 16        TEST ENABLE COUNTER (BIT 0) CAN BE SET
                                ;;*****************************************************************
003722  000004                  TST16:  SCOPE
003724  012737  000021  001124          MOV     #BIT4!BIT0,$GDDAT       ;LOAD EXPECTED
```

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

                                        ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
003752  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
003760  001401                                  BEQ     TST17           ;;BR IF EQUAL
003762  104003                                  ERROR   3                       ;ERROR COUNTER ENABLE FAILED TO SET

                                        ;;**************************************************************
                                        ;*TEST 17        TEST RATE SELECT (BIT 1) MAY BE SET
                                        ;;**************************************************************
003764  000004                          TST17:  SCOPE
003766  012737  000022  001124                  MOV     #BIT4!BIT1,$GDDAT       ;LOAD EXPECTED

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

                                        ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004014  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
004022  001401                                  BEQ     TST20           ;;BR IF EQUAL
004024  104003                                  ERROR   3                       ;ERROR, RATE BIT 1 FAILED TO SET

                                        ;;**************************************************************
                                        ;*TEST 20        TEST THAT RATE SELECT (BIT 2) MAY BE SET
                                        ;;**************************************************************
004026  000004                          TST20:  SCOPE
004030  012737  000024  001124                  MOV     #BIT4!BIT2,$GDDAT       ;LOAD EXPECTED

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

                                        ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004056  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
004064  001401                                  BEQ     TST21           ;;BR IF EQUAL
004066  104003                                  ERROR   3                       ;ERROR, RATE BIT 2 FAILED TO SET

                                        ;;**************************************************************
                                        ;*TEST 21        TEST THAT RATE SELECT (BIT 3) MAY BE SET
                                        ;;**************************************************************
004070  000004                          TST21:  SCOPE
004072  012737  000030  001124                  MOV     #BIT4!BIT3,$GDDAT       ;LOAD EXPECTED

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

                                        ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004120  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
004126  001401                                  BEQ     TST22           ;;BR IF EQUAL
004130  104003                                  ERROR   3                       ;ERROR, RATE BIT 3 FAILED TO SET

                                        ;;**************************************************************
                                        ;*TEST 22        TEST CLOCK INTERRUPT ENABLE (BIT 6) CAN BE SET
                                        ;;**************************************************************
004132  000004                          TST22:  SCOPE
004134  012737  000120  001124                  MOV     #BIT6!BIT4,$GDDAT       ;LOAD EXPECTED

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

```
                                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004162  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004170  001401                         BEQ     TST23           ;;BR IF EQUAL
004172  104003                         ERROR   3                       ;ERROR, CLOCK INTERRUPT ENABLE FAILED TO SET

                                ;;**************************************************************
                                ;*TEST 23        TEST MODE (BIT 8) CAN BE SET
                                ;;**************************************************************
004174  000004                  TST23:  SCOPE
004176  012737  000420  001124          MOV     #BIT8!BIT4,$GDDAT       ;LOAD EXPECTED

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

                                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004224  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004232  001401                         BEQ     TST24           ;;BR IF EQUAL
004234  104003                         ERROR   3                       ;ERROR, CSR NOT = 420

                                ;;**************************************************************
                                ;*TEST 24        TEST EXT INTERRUPT ENABLE (BIT 14) CAN BE SET
                                ;;**************************************************************
004236  000004                  TST24:  SCOPE
004240  012737  040020  001124          MOV     #BIT14!BIT4,$GDDAT      ;LOAD EXPECTED

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

                                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004266  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004274  001401                         BEQ     TST25           ;;BR IF EQUAL
004276  104003                         ERROR   3                       ;ERROR, EXT INTERRUPT ENABLE FAILED TO SET

                                ;;**************************************************************
                                ;*TEST 25        TEST THAT CLK DONE (BIT 7) CAN BE SET
                                ;;**************************************************************
004300  000004                  TST25:  SCOPE
004302  012737  000220  001124          MOV     #BIT7!BIT4,$GDDAT       ;LOAD EXPECTED

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

                                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004330  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004336  001401                         BEQ     TST26           ;;BR IF EQUAL
004340  104003                         ERROR   3                       ;CLOCK DONE FAILED TO sET

                                ;;**************************************************************
                                ;*TEST 26        TEST THAT CLK EXT INPUT (BIT 15) CAN BE SET
                                ;;**************************************************************
004342  000004                  TST26:  SCOPE
004344  012737  100020  001124          MOV     #BIT15!BIT4,$GDDAT      ;LOAD EXPECTED

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

                                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004372  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
004400  001401                         BEQ     TST27           ;;BR IF EQUAL
```

```
                                        ERROR   3                       ;CLOCK EXT INPUT FAILED TO SET
                                ;;*****************************************************************
                                ;*TEST 27       TEST THAT THE EXT FLAG DOES NOT SET FROM THE OUTSIDE SOURCE
                                ;;*****************************************************************
                                TST27:  SCOPE
004404  000004
004406  012737  000000  001670          MOV     #0,$TMDAT
                                ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
004424  012737  000020  001124          MOV     #BIT4,$GDDAT            ;LOAD EXPECTED
004432  005037  001660                  CLR     TEMP
004436                          1$:
                                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004446  005737  001126                  TST     $BDDAT                  ;TEST BIT15
004452  100001                          BPL     2$                      ;;BR IF CLEARED
004454  104003                          ERROR   3                       ;ERROR EXT FLG SET IN ERROR
004456  005237  001660          2$:     INC     TEMP
004462  001365                          BNE     1$
                                ;;*****************************************************************
                                ;*TEST 30       MAINT. COUNT THE COUNTER REGISTER AT RATE 1MHZ
                                ;;*****************************************************************
                                TST30:  SCOPE
004464  000004
004466  012737  000004  001166          MOV     #4,$TIMES       ;;DO 4 ITERATIONS
004474  005037  001670                  CLR     $TMDAT
                                ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
                                ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
004520  012737  000002  001670          MOV     #2,$TMDAT               ;SET UP 1MHZ RATE
                                ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
004536  005037  001660                  CLR     TEMP                    ;CLEAR PATTERN
004542  013737  001660  001124  1$:     MOV     TEMP,$GDDAT             ;LOAD EXPECTED
                                ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
004560  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMP
004566  001401                          BEQ     2$              ;;BR IF EQUAL
004570  104007                          ERROR   7                       ;ERROR, CLOCK COUNTER
                                                                        ;BUFFER COUNTED IN ERROR
004572                          2$:
                                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
004602  105737  001126                  TSTB    $BDDAT
004606  100005                          BPL     3$              ;;BR IF CLEAR
004610  012737  000022  001124          MOV     #22,$GDDAT              ;LOAD EXPECTED STATUS
004616  104003                          ERROR   3                       ;CLK DONE SET IN ERROR
004620  000431                          BR      TST31           ;;BR TO SCOPE
004622                          3$:
                                ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
004632  052737  004000  001670          BIS     #BIT11,$TMDAT
                                ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
```

```
                                        ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
004660  042737  004000  001670                  BIC     #BIT11,$TMDAT
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
004676  105237  00.660                          INCB    TEMP
004702  001317                                  BNE     1$                      ;BRANCH IF NOT FULL COUNT
                                        ;;*********************************************************************
                                        ;*TEST 31        TEST THAT OVERFLOW SET CLK DONE (BIT 7)
                                        ;;*********************************************************************
004704  000004                          TST31:  SCOPE
004706  012737  000000  001670                  MOV     #0,$TMDAT               ;CLEAR STATUS
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
004724  012737  177777  001670                  MOV     #-1,$TMDAT              ;LOAD PRESET
                                        ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
004742  012737  000002  001670                  MOV     #BIT1,$TMDAT            ;LOAD 1 MHZ. RATE
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMdAT TO DEVICE REG CSR
004760  012737  000222  001124                  MOV     #BIT7!BIT4!BIT1,$GDDAT  ;LOAD EXPECTED
                                        ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
004776  052737  004000  001670                  BIS     #BIT11,$TMDAT           ;MAINT COUNT
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
                                        ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
005024  042737  004000  001670                  BIC     #BIT11,$TMDAT           ;CLEAR BIT
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
                                        ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
005052  105737  001126                          TSTB    $BDDAT                  ;TEST BIT 7
005056  100401                                  BMI     TST32           ;;BR IF MINUS
005060  104003                                  ERROR   3                       ;ERROR, OVERFLOW FAILED TO SET BIT 7
                                        ;;*********************************************************************
                                        ;*TEST 32        MAINT. COUNT THE COUNTER REGISTER AT RATE #100KHZ
                                        ;;*********************************************************************
005062  000004                          TST32:  SCOPE
005064  012737  000004  001166                  MOV     #4,$TIMES               ;;DO 4 ITERATIONS
005072  012737  000000  001670                  MOV     #0,$TMDAT               ;CLEAR CLOCK STATUS
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
                                        ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
005120  005037  001124                          CLR     $GDDAT
005124  012737  000004  001670                  MOV     #4,$TMDAT               ;LOAD STATUS, 100KHZ RATE
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
005142  005037  001124                          CLR     $GDDAT
005146                                  1$:
                                        ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
```

```
1105  005156  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1106  005164  001401                          BEQ     2$              ;;BR IF EQUAL
1107  005166  104007                          ERROR   7               ;ERROR, CLOCK COUNTER BUFFER
1108                                                                  ;COUNTED IN ERROR, FAULT IS PROBABLY IN THE
1109                                                                  ;CLOCK UP COUNT OR RATE SELECTION LOGIC
1110  005170  012737  000012  001654  2$:     MOV     #10.,COUNT              ;LOAD COUNT
1111  005176                          3$:
1112
1113                                  ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1114  005206  052737  004000  001670          BIS     #BIT11,$TMDAT
1115
1116                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1117
1118                                  ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1119  005234  042737  004000  001670          BIC     #BIT11,$TMDAT
1120
1121                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1122  005252  005337  001654                  DEC     COUNT                   ;DONE
1123  005256  001347                          BNE     3$                      ;BR IF NOT
1124  005260  105237  001124                  INCB    $GDDAT                  ;INCREMENT  EXPECTED VALUE
1125  005264  001330                          BNE     1$
1126
1127                                  ;;**************************************************************
1128                                  ;*TEST 33        MAINT. COUNT THE COUNTER REGISTER AT RATE #10KHZ
1129                                  ;;**************************************************************
1130  005266  000004                  1ST33:  SCOPE
1131  005270  012737  000004  001166          MOV     #4,$TIMES       ;;DO 4 ITERATIONS
1132  005276  012737  000000  001670          MOV     #0,$TMDAT               ;CLEAR CLOCK STATUS
1133
1134                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1135
1136                                  ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
1137  005324  012737  000006  001670          MOV     #6,$TMDAT               ;LOAD STATUS. 10KHZ RATE
1138
1139                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1140  005342  005037  001124                  CLR     $GDDAT
1141  005346                          1$:
1142
1143                                  ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1144  005356  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1145  005364  001401                          BEQ     2$              ;;BR IF EQUAL
1146  005366  104007                          ERROR   7               ;ERROR, CLOCK COUNTER BUFFER
1147                                                                  ;COUNTED IN ERROR, FAULT IS PROBABLY IN THE
1148                                                                  ;CLOCK UP COUNT OR RATE SELECTION LOGIC
1149  005370  012737  000144  001654  2$:     MOV     #100.,COUNT             ;LOAD COUNT
1150  005376                          3$:
1151
1152                                  ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1153  005406  052737  004000  001670          BIS     #BIT11,$TMDAT
1154
1155                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1156
1157                                  ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1158  005434  042737  004000  001670          BIC     #BIT11,$TMDAT
```

```
                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
005452  005337  001654                          DEC     COUNT                   ;DONE
005456  001347                                  BNE     3$                      ;BR IF NOT
005460  105237  001124                          INCB    $GDDAT                  ;UPDATE PATTERN
005464  001330                                  BNE     1$                      ;BR IF NOT
                                        ;;*************************************************************
                                        ;*TEST 34       MAINT. COUNT THE COUNTER REGISTER AT RATE #1KHZ
                                        ;;*************************************************************
005466  000004                          TST34:  SCOPE
005470  012737  000004  001166                  MOV     #4,$TIMES       ;;DO 4 ITERATIONS
005476  012737  000000  001670                  MOV     #0,$TMDAT               ;CLEAR CLOCK STATUS

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

                                        ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
005524  012737  000001  001124                  MOV     #1,$GDDAT               ;LOAD EXPECTED
005532  012737  000010  001670                  MOV     #10,$TMDAT              ;LOAD STATUS, 1 KHZ RATE

                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
005550  012737  001750  001654                  MOV     #1000.,COUNT            ;SET UP A COUNTER
005556                                  1$:

                                        ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
005566  052737  004000  001670                  BIS     #BIT11,$TMDAT

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

                                        ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
005614  042737  004000  001670                  BIC     #BIT11,$TMDAT

                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
005632  005337  001654                          DEC     COUNT
005636  001347                                  BNE     1$                      ;BR

                                        ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
005650  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
005656  001401                                  BEQ     TST35           ;;BR IF EQUAL
005660  104007                                  ERROR   7                       ;ERROR, CLOCK COUNTER BUFFER
                                                                                ;COUNTED IN ERROR, FAULT IS PROBABLY IN THE
                                                                                ;CLOCK UP COUNT OR RATE SELECTION LOGIC

                                        ;;*************************************************************
                                        ;*TEST 35       MAINT. COUNT THE COUNTER REGISTER AT RATE #100HZ
                                        ;;*************************************************************
005662  000004                          TST35:  SCOPE
005664  012737  000004  001166                  MOV     #4,$TIMES       ;;DO 4 ITERATIONS
005672  012737  000000  001670                  MOV     #0,$TMDAT               ;CLEAR CLOCK STATUS, 100HZ RATE

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

                                        ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
005720  012737  000001  001124                  MOV     #1,$GDDAT               ;LOAD EXPECTED
005726  012737  000012  001670                  MOV     #12,$TMDAT              ;LOAD STATUS, 100 HZ
```

```
1213
1214                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1215  005744  012737  023420  001654                MOV     #10000.,COUNT           ;SET UP A COUNTER
1216  005752                              1$:
1217
1218                                        ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1219  005762  052737  004000  001670                BIS     #BIT11,$TMDAT
1220
1221                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1222
1223                                        ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
1224  006010  042737  004000  001670                BIC     #BIT11,$TMDAT
1225
1226                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1227  006026  005337  001654                        DEC     COUNT
1228  006032  001347                                BNE     1$                      ;BR
1229
1230                                        ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1231  006044  023737  001124  001126                CMP     $GDDAT,$BDDAT           ;COMPARE
1232  006052  001401                                BEQ     TST36           ;.BR IF EQUAL
1233  006054  104007                                ERROR   7                       ;ERROR, CLOCK COUNTER BUFFER
1234                                                                            ;COUNTED IN ERROR, FAULT IS PROBABLY IN THE
1235                                                                            ;CLOCK UP COUNT OR RATE SELECTION LOGIC
1236
1237                                        ;;*****************************************************************
1238                                        ;*TEST 36        TEST THAT RESET CLEARS RATE SELECT AND MODE BITS
1239                                        ;;*****************************************************************
1240  006056  000004                        TST36:  SCOPE
1241  006060  012737  000002  001166                MOV     #2,$TIMES       ;;DO 2 ITERATIONS
1242  006066  012737  000416  001670                MOV     #BIT8!BIT3!BIT2!BIT1,$TMDAT     ;SET MODE BITS
1243
1244                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1245  006104  012737  000020  001124                MOV     #20,$GDDAT
1246  006112  004737  023174                        JSR     PC,$RESET
1247
1248                                        ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1249  006126  023737  001124  001126                CMP     $GDDAT,$BDDAT           ;COMPARE
1250  006134  001401                                BEQ     TST37           ;;BR IF EQUAL
1251  006136  104003                                ERROR   3                       ;ERROR, RESET FAILED TO CLEAR RATE OR MODE BITS
1252
1253                                        ;;*****************************************************************
1254                                        ;*TEST 37        TEST THAT RESET CLEARS CLK INTERRUPT ENABLE
1255                                        ;;*****************************************************************
1256  006140  000004                        TST37:  SCOPE
1257  006142  012737  000002  001166                MOV     #2,$TIMES       ;;DO 2 ITERATIONS
1258  006150  012737  040100  001670                MOV     #BIT6!BIT14,$TMDAT      ;SET CLK INT ENABLE
1259
1260                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
1261  006166  012737  000020  001124                MOV     #20,$GDDAT
1262  006174  004737  023174                        JSR     PC,$RESET
1263
1264                                        ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1265  006210  023737  001124  001126                CMP     $GDDAT,$BDDAT
1266  006216  001401                                BEQ     TST40           ;;BR IF EQUAL
```

```
                        ERROR   3                       ;ERROR, RESET FAILED TO CLEAR CLK INT ENABLE

                ;;*****************************************************************
                ;*TEST 40        TEST THAT RESET CLEARS CLK FLAGS
                ;;*****************************************************************
                TST40:  SCOPE
                        MOV     #2,$TIMES       ;;DO 2 ITERATIONS
                        MOV     #BIT7!BIT15,$TMDAT      ;SET CLK FLAGS

                ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
                        MOV     #20,$GDDAT
                        JSR     PC,$RESET

                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
                        CMP     $GDDAT,$BDDAT
                        BEQ     TST41           ;;BR IF EQUAL
                        ERROR   3                       ;ERROR, RESET FAILED TO CLEAR CLK FLAGS

                ;;*****************************************************************
                ;*TEST 41        TEST THAT RESET CLEARS COUNTER ENABLE
                ;;*****************************************************************
                TST41:  SCOPE
                        MOV     #2,$TIMES       ;;DO 2 ITERATIONS
                        MOV     #0,$TMDAT                       ;CLEAR COUNTER

                ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
                        MOV     #BIT0,$TMDAT            ;LOAD COUNTER ENABLE

                ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
                        MOV     #20,$BDDAT
                        JSR     PC,$RESET

                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
                        CMP     $GDDAT,$BDDAT
                        BEQ     TST42           ;;BR IF EQUAL
                        ERROR   3                       ;ERROR, RESET FAILED TO CLEAR COUNTER ENABLE


                ;;*****************************************************************
                ;*TEST 42        TEST CLOCK TO COUNT UP AT 1 MHZ
                ;;*****************************************************************
                TST42:  SCOPE
                        MOV     #20,$TIMES      ;;DO 20 ITERATIONS
                        MOV     #3,RATE                 ;SELECT MODE 0, 1MHZ.. GO
                        MOV     #BIT7!BIT4!BIT1,$GDDAT
                        JSR     PC,UPCNT

                ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
                        CMP     $GDDAT,$BDDAT           ;COMPARE
                        BEQ     1$              ;;BR IF EQUAL
                        ERROR   3                       ;ERROR, 1MHZ RATE FAILED TO SET DONE
                1$:     MOV     #0,$GDDAT

                ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
```

```
1321  006474  023737  001124  001126          CMP     $GDDAT,$BDDAT
1322  006502  001401                          BEQ     TST43                   ;;BR IF EQUAL
1323  006504  104006                          ERROR   6                               ;IN MODE 0 CLOCK COUNTER WAS
1324                                                                                  ;LOADED ON CLK OVERFLOW
1325
1326                                  ;;**************************************************************
1327                                  ;*TEST 43        TEST CLOCK TO COUNT UP AT 100KHZ
1328                                  ;;**************************************************************
1329  006506  000004                  TST43:  SCOPE
1330  006510  012737  000020  001166          MOV     #20,$TIMES              ;;DO 20 ITERATIONS
1331  006516  012737  000005  014610          MOV     #5,RATE                         ;SELECT MODE 0, 100KHZ., GO
1332  006524  012737  000224  001124          MOV     #BIT7!BIT4!BIT2,$GDDAT
1333  006532  004737  014002                  JSR     PC,UPCNT
1334
1335                                  ;*      MOV     @CSR,$BDDAT             ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1336  006546  023737  001124  001126          CMP     $GDDAT,$BDDAT                   ;COMPARE
1337  006554  001401                          BEQ     1$                      ;;BR IF EQUAL
1338  006556  104003                          ERROR   3                               ;ERROR, 100KHZ. RATE FAILED TO SET DONE
1339  006560  012737  000000  001124  1$:     MOV     #0,$GDDAT
1340
1341                                  ;*      MOV     @CSC,$BDDAT             ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1342  006576  023737  001124  001126          CMP     $GDDAT,$BDDAT
1343  006604  001401                          BEQ     TST44                   ;;BR IF EQUAL
1344  006606  104006                          ERROR   6                               ;IN MODE 0 CLOCK COUNTER WAS
1345                                                                                  ;LOADED ON CLK OVERFLOW
1346
1347                                  ;;**************************************************************
1348                                  ;*TEST 44        TEST CLOCK TO COUNT UP AT 10 KHZ
1349                                  ;;**************************************************************
1350  006610  000004                  TST44:  SCOPE
1351  006612  012737  000020  001166          MOV     #20,$TIMES              ;;DO 20 ITERATIONS
1352  006620  012737  000007  014610          MOV     #7,RATE                         ;SELECT MODE 0, 10KHZ., GO
1353  006626  012737  000226  001124          MOV     #BIT7!BIT4!BIT2!BIT1,$GDDAT
1354  006634  004737  014002                  JSR     PC,UPCNT
1355
1356                                  ;*      MOV     @CSR,$BDDAT             ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1357  006650  023737  001124  001126          CMP     $GDDAT,$BDDAT                   ;COMPARE
1358  006656  001401                          BEQ     TST45                   ;;BR IF EQUAL
1359  006660  104007                          ERROR   7                               ;ERROR, 10 KHZ. FAILED TO sET DONE
1360
1361
1362                                  ;;**************************************************************
1363                                  ;*TEST 45        TEST CLOCK TO COUNT UP AT 1KHZ
1364                                  ;;**************************************************************
1365  006662  000004                  TST45:  SCOPE
1366  006664  012737  000005  001166          MOV     #5,$TIMES               ;;DO 5 ITERATIONS
1367  006672  012737  000011  014610          MOV     #11,RATE                        ;SELECT MODE 0, 1 KHZ, GO
1368  006700  012737  000230  001124          MOV     #BIT7!BIT4!BIT3,$GDDAT
1369  006706  004737  014002                  JSR     PC,UPCNT
1370
1371                                  ;*      MOV     @CSR,$BDDAT             ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1372  006722  023737  001124  001126          CMP     $GDDAT,$BDDAT                   ;COMPARE
1373  006730  001401                          BEQ     TST46                   ;;BR IF EQUAL
1374  006732  104007                          ERROR   7                               ;ERROR, 1KHZ FAILED TO SET DONE
```

```
1375
1376                                    ;;*****************************************************************
1377                                    ;*TEST 46        TEST CLOCK TO COUNT UP AT 100HZ
1378                                    ;*****************************************************************
1379  006734  000004                    TST46:  SCOPE
1380  006736  012737  000005  001166            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1381  006744  012737  000013  014610            MOV     #13,RATE                ;SELECT MODE 0, 100 HZ, GO
1382  006752  012737  000232  001124            MOV     #BIT7!BIT4!BIT3!BIT1,$GDDAT     ;LOAD EXPECTED
1383  006760  004737  014002                    JSR     PC,UPCNT
1384
1385                                    ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1386  006774  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1387  007002  001401                            BEQ     TST47           ;;BR IF EQUAL
1388  007004  104007                            ERROR   7                       ;ERROR, 100HZ FAILED TO SET DONE
1389
1390                                    ;;*****************************************************************
1391                                    ;*TEST 47        TEST THAT CLOCK ENABLE DOES NOT CLEAR ON DONE (MODE 1) 1 KHZ
1392                                    ;*****************************************************************
1393  007006  000004                    TST47:  SCOPE
1394  007010  012737  000005  001166            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1395  007016  012737  000411  014610            MOV     #411,RATE               ;SELECT MODE 1, 1KHZ., GO
1396  007024  012737  000631  001124            MOV     #631,$GDDAT             ;LOAD EXP
1397  007032  004737  014002                    JSR     PC,UPCNT
1398
1399                                    ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT.
1400  007046  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1401  007054  001401                            BEQ     1$              ;;BR IF EQUAL
1402  007056  104007                            ERROR   7                       ;ERROR, 1 KHZ. FAILED TO OVERFLOW
1403                                                                            ; AND CONTINUE COUNTING
1404  007060  012737  000376  001124  1$:       MOV     #376,$GDDAT
1405
1406                                    ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1407  007076  023737  001124  001126            CMP     $GDDAT,$BDDAT
1408  007104  001401                            BEQ     TST50           ;;BR IF EQUAL
1409  007106  104006                            ERROR   6                       ;IN MODE 1, CLOCK COUNTER FAILED
1410                                                                            ;TO BE RE-LOADED ON CLOCK OVERFLOW
1411
1412
1413                                    ;;*****************************************************************
1414                                    ;*TEST 50        TEST THAT CLOCK ENABLE DOES NOT CLEAR DONE (MODE 1 100HZ)
1415                                    ;*****************************************************************
1416  007110  000004                    TST50:  SCOPE
1417  007112  012737  000020  001166            MOV     #20,$TIMES      ;;DO 20 ITERATIONS
1418  007120  012737  000413  014610            MOV     #413,RATE               ;MODE 1 100HZ GO
1419  007126  012737  000633  001124            MOV     #633,$GDDAT             ;LOAD EXPECTED
1420  007134  004737  014002                    JSR     PC,UPCNT
1421
1422                                    ;*      MOV     @CSR,$BDDAT     ;/READ DEVICE REG CSR,PUT DATA IN $BDDAT
1423  007150  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1424  007156  001401                            BEQ     1$              ;;BR IF EQUAL
1425  007160  104007                            ERROR   7                       ;ERROR, 100HZ FAILED TO
1426                                                                            ; OVERFLOW AND CONTINUE COUNTING
1427  007162  012737  000376  001124  1$:       MOV     #376,$GDDAT
1428
```

```
1429                                    ;*      MOV     @CSC,$BDDAT     ;/READ DEVICE REG CSC,PUT DATA IN $BDDAT.
1430  007200  023737  001124  001126            CMP     $GDDAT,$BDDAT
1431  007206  001401                            BEQ     TST51           ;;BR IF EQUAL
1432  007210  104006                            ERROR   6                       ;IN MODE 1, CLOCK COUNTER FAILED
1433                                                                            ;TO BE RE-LOADED ON CLOCK OVERFLOW
1434
1435                                    ;;**************************************************************
1436                                    ;*TEST 51        TEST THAT RESET SETS VC READY BIT
1437                                    ;;**************************************************************
1438  007212  000004                    TST51:  SCOPE
1439  007214  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1440  007222  004737  023174                    JSR     PC,$RESET
1441
1442                                    ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1443  007236  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1444  007244  001401                            BEQ     TST52           ;;BR IF SET
1445  007246  104011                            ERROR   11                      ;RESET FAILED TO SET READY
1446
1447                                    ;;**************************************************************
1448                                    ;*TEST 52        TEST THAT VC MODE BIT 2 CAN BE SET AND CLEARED
1449                                    ;;**************************************************************
1450  007250  000004                    TST52:  SCOPE
1451  007252  012737  000004  001670            MOV     #BIT2,$TMDAT            ;LOAD DISPLAY STATUS
1452
1453                                    ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1454  007270  012737  000204  001124            MOV     #BIT7!BIT2,$GDDAT       ;LOAD EXPECTED
1455
1456                                    ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1457  007306  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1458  007314  001401                            BEQ     1$              ;;BR IF EQUAL
1459  007316  104011                            ERROR   11                      ;ERROR, VC STATUS NOT = 204
1460  007320  012737  000000  001670    1$:     MOV     #0,$TMDAT               ;CLEAR STATUS
1461
1462                                    ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1463  007336  012737  000200  001124            MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1464
1465                                    ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1466  007354  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1467  007362  001401                            BEQ     TST53           ;;BR IF CLEARED
1468  007364  104011                            ERROR   11                      ;MODE FAILED TO CLEAR
1469
1470                                    ;;**************************************************************
1471                                    ;*TEST 53        TEST THAT VC MODE BIT 3 CAN BE SET
1472                                    ;;**************************************************************
1473  007366  000004                    TST53:  SCOPE
1474  007370  012737  000010  001670            MOV     #BIT3,$TMDAT            ;LOAD
1475
1476                                    ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1477  007406  012737  000210  001124            MOV     #BIT7!BIT3,$GDDAT       ;LOAD EXPECTED
1478
1479                                    ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1480  007424  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1481  007432  001401                            BEQ     TST54           ;;BR IF EQUAL
1482  007434  104011                            ERROR   11                      ;ERROR, VC STATUS NOT = 210
```

```
1483
1484                                    ;;*****************************************************************
1485                                    ;*TEST 54        TEST THAT VC INTERRUPT ENABLE (BIT 6) CAN BE SET
1486                                    ;;*****************************************************************
1487  007436  000004                    TST54:  SCOPE
1488  007440  012737  000100  001670            MOV     #BIT6,$TMDAT            ;LOAD DISPLAY STATUS
1489
1490                                    ;*          MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1491  007456  012737  000300  001124            MOV     #BIT7!BIT6,$GDDAT       ;LOAD EXPECTED
1492
1493                                    ;*          MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1494  007474  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1495  007502  001401                            BEQ     TST55           ;;BR IF EQUAL
1496  007504  104011                            ERROR   11              ;ERROR, VC STATUS NOT = 300
1497
1498                                    ;;*****************************************************************
1499                                    ;*TEST 55        TEST THAT CHANNEL (BIT 9) CAN BE SET
1500                                    ;;*****************************************************************
1501  007506  000004                    TST55:  SCOPE
1502  007510  012737  001000  001670            MOV     #BIT9,$TMDAT            ;LOAD DISPLAY STATUS
1503
1504                                    ;*          MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1505  007526  012737  001200  001124            MOV     #BIT9!BIT7,$GDDAT       ;LOAD EXPECTED
1506
1507                                    ;*          MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1508  007544  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1509  007552  001401                            BEQ     TST56           ;;BR IF EQUAL
1510  007554  104011                            ERROR   11              ;ERROR, VC STATUS NOT = 1200
1511
1512                                    ;;*****************************************************************
1513                                    ;*TEST 56        TEST THAT STORE (BIT 10) CAN BE SET
1514                                    ;;*****************************************************************
1515  007556  000004                    TST56:  SCOPE
1516  007560  012737  002000  001670            MOV     #BIT10,$TMDAT           ;LOAD DISPLAY STATUS
1517
1518                                    ;*          MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1519  007576  012737  002200  001124            MOV     #BIT10!BIT7,$GDDAT      ;LOAD EXPECTED
1520
1521                                    ;*          MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1522  007614  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1523  007622  001401                            BEQ     TST57           ;;BR IF EQUAL
1524  007624  104011                            ERROR   11              ;ERROR, VC STATUS NOT = 2200
1525                                    ;;*****************************************************************
1526                                    ;*TEST 57        TEST THAT WRITE THRU (BIT 11) CAN BE SET
1527                                    ;;*****************************************************************
1528  007626  000004                    TST57:  SCOPE
1529  007630  012737  004000  001670            MOV     #BIT11,$TMDAT           ;LOAD DISPLAY STATUS
1530
1531                                    ;*          MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1532  007646  012737  004200  001124            MOV     #BIT11!BIT7,$GDDAT      ;LOAD EXPECTED
1533
1534                                    ;*          MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1535  007664  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1536  007672  001401                            BEQ     TST60           ;;BR IF EQUAL
```

```
                                ERROR   11                      ;ERROR, VC STATUS NOT = 4200
                        ;;*******************************************************
                        ;*TEST 60        TEST THAT ERASE (BIT 12) CAN BE SET
                        ;;*******************************************************
007676  000004          TST60:  SCOPE
007700  012737  010000  001670          MOV     #BIT12,$TMDAT           ;SET ERASE BIT
                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
007716  012737  010000  001124          MOV     #BIT12,$GDDAT           ;LOAD EXPECTED
                        ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
007734  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
007742  001401                          BEQ     1$                      ;;BR IF SET
007744  104011                          ERROR   11                      ;ERROR, ERASE BIT FAILED TO SET
007746  005037  001670          1$:     CLR     $TMDAT
                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                        ;;*******************************************************
                        ;*TEST 61        TEST THAT THE X REGISTER CAN BE CLEARED
                        ;;*******************************************************
007762  000004          TST61:  SCOPE
007764  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED
                        ;*      MOV     $GDDAT,@VCXREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
                        ;*      MOV     @VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
010012  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
010020  001401                          BEQ     TST62                   ;;BR IF EQUAL
010022  104012                          ERROR   12                      ;ERROR, VC XREGISTER NOT = 0
                        ;;*******************************************************
                        ;*TEST 62        TEST THAT THE X REGISTER CAN BE LOADED WITH #1777
                        ;;*******************************************************
010024  000004          TST62:  SCOPE
010026  012737  001777  001124          MOV     #1777,$GDDAT            ;LOAD EXPECTED
                        ;*      MOV     $GDDAT,@VCXREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
                        ;*      MOV     @VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
010054  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
010062  001401                          BEQ     TST63                   ;;BR IF EQUAL
010064  104012                          ERROR   12                      ;ERROR, VC X REGISTER NOT = 1777
                        ;;*******************************************************
                        ;*TEST 63        TEST THAT THE X REGISTER CAN BE LOADED WITH #525
                        ;;*******************************************************
010066  000004          TST63:  SCOPE
010070  012737  000525  001124          MOV     #525,$GDDAT             ;LOAD EXPECTED
                        ;*      MOV     $GDDAT,@VCXREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
                        ;*      MOV     @VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
010116  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
010124  001401                          BEQ     TST64                   ;;BR IF EQUAL
```

```
1591  010126  104012                          ERROR   12                      ;ERROR, VC X REGISTER NOT = 525
1592
1593                                  ;;*****************************************************************
1594                                  ;*TEST 64        TEST THAT THE X REGISTER CAN BE LOADED WITH #1252
1595                                  ;;*****************************************************************
1596  010130  000004                  TST64:  SCOPE
1597  010132  012737  001252  001124          MOV     #1252,$GDDAT            ;LOAD EXPECTED
1598
1599                                  ;*      MOV     $GDDAT,@VCXREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
1600
1601                                  ;*      MOV     @VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1602  010160  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1603  010166  001401                          BEQ     TST65           ;;BR IF EQUAL
1604  010170  104012                          ERROR   12                      ;ERROR, VC X REGISTER NOT = 1252
1605
1606                                  ;;*********************************  ******************************
1607                                  ;*TEST 65        TEST THAT THE X REGISTER CAN HOLD A COUNT PATTERN
1608                                  ;;*****************************************************************
1609  010172  000004                  TST65:  SCOPE
1610  010174  012737  010206  001110          MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
1611  010202  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
1612  010206                          1$:
1613
1614                                  ;*      MOV     $GDDAT,@VCXREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCXREG
1615
1616                                  ;*      MOV     @VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1617  010226  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1618  010234  001401                          BEQ     2$              ;;BR IF EQUAL
1619  010236  104012                          ERROR   12                      ;VC X REG FAILED TO HOLD A COUNT PATTERN
1620
1621  010240  005237  001124          2$:     INC     $GDDAT                  ;UPDATE PATTERN
1622  010244  022737  002000  001124          CMP     #2000,$GDDAT            ;FINISHED?
1623  010252  001355                          BNE     1$                      ;BR IF NOT
1624
1625                                  ;;*****************************************************************
1626                                  ;*TEST 66        TEST THAT THE Y REGISTER CAN BE CLEARED
1627                                  ;;*****************************************************************
1628  010254  000004                  TST66:  SCOPE
1629  010256  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED
1630
1631                                  ;*      MOV     $GDDAT,@VCYREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1632
1633                                  ;*      MOV     @VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1634  010304  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1635  010312  001401                          BEQ     TST67           ;;BR IF EQUAL
1636  010314  104013                          ERROR   13                      ;ERROR, VC Y REGISTER NOT = 0
1637
1638                                  ;;*****************************************************************
1639                                  ;*TEST 67        TEST THAT THE Y REGISTER CAN BE LOADED WITH #1777
1640                                  ;;*****************************************************************
1641  010316  000004                  TS  7:  SCOPE
1642  010320  012737  001777  001124          MOV     #1777,$GDDAT            ;LOAD EXPECTED
1643
1644                                  ;*      MOV     $GDDAT,@VCYREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
```

```
1645
1646                                    ;*      MOV     @VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1647  010346  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1648  010354  001401                            BEQ     TST70           ;;BR IF EQUAL
1649  010356  104013                            ERROR   13                      ;ERROR, VC Y REGISTER NOT = 1777
1650
1651                                    ;;******************************************************************
1652                                    ;*TEST 70        TEST THAT THE Y REGISTER CAN BE LOADED WITH #525
1653                                    ;;******************************************************************
1654  010360  000004                    TST70:  SCOPE
1655  010362  012737  000525  001124            MOV     #525,$GDDAT             ;LOAD EXPECTED
1656
1657                                    ;*      MOV     $GDDAT,@VCYREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1658
1659                                    ;*      MOV     @VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1660  010410  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1661  010416  001401                            BEQ     TST71           ;;BR IF EQUAL
1662  010420  104013                            ERROR   13                      ;ERROR, VC Y REGISTER NOT = 525
1663
1664                                    ;;******************************************************************
1665                                    ;*TEST 71        TEST THAT THE Y REGISTER CAN BE LOADED WITH #1252
1666                                    ;;******************************************************************
1667  010422  000004                    TST71:  SCOPE
1668  010424  012737  001252  001124            MOV     #1252,$GDDAT            ;LOAD EXPECTED
1669
1670                                    ;*      MOV     $GDDAT,@VCYREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1671
1672                                    ;*      MOV     @VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1673  010452  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1674  010460  001401                            BEQ     TST72           ;;BR IF EQUAL
1675  010462  104013                            ERROR   13                      ;ERROR, VC Y REGISTER NOT = 1252
1676
1677                                    ;;******************************************************************
1678                                    ;*TEST 72        TEST THAT THE Y REGISTER CAN HOLD A COUNT PATTERN
1679                                    ;;******************************************************************
1680  010464  000004                    TST72:  SCOPE
1681  010466  012737  010500  001110            MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
1682  010474  005037  001124                    CLR     $GDDAT                  ;CLEAR EXPECTED
1683  010500                            1$:
1684
1685                                    ;*      MOV     $GDDAT,@VCYREG  ;/ PUT DATA FROM $GDDAT TO DEVICE REG VCYREG
1686
1687                                    ;*      MOV     @VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1688  010520  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;COMPARE
1689  010526  001401                            BEQ     2$              ;;BR IF EQUAL
1690  010530  104013                            ERROR   13                      ;VC Y REG FAILED TO HOLD A COUNT PATTERN
1691
1692  010532  005237  001124          2$:       INC     $GDDAT                  ;UPDATE PATTERN
1693  010536  022737  002000  001124            CMP     #2000,$GDDAT            ;FINISHED?
1694  010544  001355                            BNE     1$                      ;BR IF NOT
1695
1696                                    ;;******************************************************************
1697                                    ;*TEST 73        TEST THAT THE X-Y REGISTERS CAN HOLD DIFFERENT DATA
1698                                    ;;******************************************************************
```

```
                        000004                  TST73:  SCOPE
010550  012737  001252  001670                  MOV     #1252,$TMDAT            ;LOAD X REGISTER

                                                ;*      MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
010566  012737  000525  001124                  MOV     #525,$GDDAT             ;LOAD EXPECTED

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

                                                ;*      MOV     @VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
010614  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
010622  001401                                  BEQ     1$                      ;;BR IF EQUAL
010624  104013                                  ERROR   13                      ;ERROR, SELECTED Y REGISTER INCORRECTLY

010626  012737  001252  001124          1$:     MOV     #1252,$GDDAT            ;LOAD EXPECTED

                                                ;*      MOV     @VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
010644  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
010652  001401                                  BEQ     TST74                   ;;BR IF EQUAL
010654  104012                                  ERROR   12                      ;ERROR, SELECTED X REGISTER INCORRECTLY

                                                ;;*****************************************************************
                                                ;*TEST 74        TEST THAT WHEN INTENSIFY BIT IS SET THAT THE VC READY BIT RESETS
                                                ;;*****************************************************************
010656  000004                                  TST74:  SCOPE
                                                ; AND THEN SETS AFTER A DELAY
010660  012700  001000                          MOV     #1000,R0
010664  012737  000000  001124                  MOV     #0,$GDDAT               ;LOAD EXPECTED
010672  012737  000001  001670                  MOV     #BIT0,$TMDAT            ;INTENSIFY

                                                ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
010710                                          1$:

                                                ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
010720  105737  001126                          TSTB    $BDDAT
010724  100412                                  BMI     TST75                   ;;NEXT TEST
010726  005300                                  DEC     R0                      ;DELAY
010730  001367                                  BNE     1$                      ;
010732  012737  000200  001124                  MOV     #BIT7,$GDDAT            ;LOAD EXPECTED

                                                ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
010750  104011                                  ERROR   11                      ;READY FAILED TO SET AFTER A DELAY

                                                ;;*****************************************************************
                                                ;*TEST 75        TEST THAT VC MODE 1 (INTENSIFY ON X) RESETS THE READY FLAG
                                                ;;*****************************************************************
010752  000004                                  TST75:  SCOPE
                                                ; AND THEN SETS IT
010754  012700  001000                          MOV     #1000,R0                ;SET UP DELAY
010760  012737  000204  001124                  MOV     #BIT7!BIT2,$GDDAT       ;LOAD EXPECTED
010766  012737  000004  001670                  MOV     #BIT2,$TMDAT            ;LOAD MODE 1

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

011004  012737  000000  001670          2$:     MOV     #0,$TMDAT               ;ADDRESS X AXIS
```

```
1753
1754  011012                                1$:
1755
1756                                        ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1757  011022  105737  001126                      TSTB    $BDDAT
1758  011026  100416                            BMI     TST76           ;;NEXT TEST
1759  011030  005300                            DEC     R0                      ;DELAY
1760  011032  001367                            BNE     1$              ;;TEST READY AGAIN
1761  011034  012737  000204  001124            MOV     #BIT7!BIT2,$GDDAT       ;LOAD EXPECTED
1762
1763                                        ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1764  011052  023737  001124  001126            CMP     $GDDAT,$BDDAT
1765  011060  001401                            BEQ     TST76           ;;BR IF EQUAL
1766  011062  104011                            ERROR   11                      ;ERROR, READY FAILED TO SET
1767                                                                            ; AFTER MODE 1 OPERATION
1768
1769                                        ;;**************************************************************
1770                                        ;*TEST 76        TEST THAT VC MODE 2 (INTENSIFY ON Y) RESETS THE READY FLAG
1771                                        ;;**************************************************************
1772  011064  000004                        TST76:  SCOPE
1773                                        ; AND THEN SETS IT
1774  011066  012700  001000                        MOV     #1000,R0                ;SET UP DELAY
1775  011072  012737  000210  001124                MOV     #BIT7!BIT3,$GDDAT       ;LOAD EXPECTED
1776  011100  012737  000010  001670                MOV     #BIT3,$TMDAT            ;LOAD MODE 2
1777
1778                                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
1779  011116  012737  000000  001670  2$:   MOV     #0,$TMDAT               ;ADDRESS Y AXIS
1780
1781                                        ;*      MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1782  011134                                1$:
1783
1784                                        ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1785  011144  105737  001126                      TSTB    $BDDAT
1786  011150  100416                            BMI     TST77           ;;NEXT TEST
1787  011152  005300                            DEC     R0                      ;DELAY
1788  011154  001367                            BNE     1$              ;;TEST READY AGAIN
1789  011156  012737  000210  001124            MOV     #BIT7!BIT3,$GDDAT       ;LOAD EXPECTED
1790
1791                                        ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
1792  011174  023737  001124  001126            CMP     $GDDAT,$BDDAT
1793  011202  001401                            BEQ     TST77           ;;BR IF EQUAL
1794  011204  104011                            ERROR   11                      ;ERROR, READY FAILED TO sET
1795                                                                            ; AFTER MODE 2 OPERATION
1796
1797                                        ;;**************************************************************
1798                                        ;*TEST 77        TEST WHEN ERASE IS SET, VC READY BIT RESETS
1799                                        ;;**************************************************************
1800  011206  000004                        TST77:  SCOPE
1801  011210  032777  010000  167722                BIT     #BIT12,@SWR             ;TEST BIT 12
1802  011216  001443                            BEQ     TST100          ;;BYPASS IF NO STORAGE SCOPE
1803  011220  012700  000010                        MOV     #10,R0
1804  011224  005037  001660                        CLR     TEMP                    ;CLEAR DELAY
1805  011230  012737  002000  001670                MOV     #BIT10,$TMDAT           ;SET STORE MODE
1806
```

```
                                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
                                        ;*      MOV     @VCSTAT,$TMDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
011256  052737  010000  001670          BIS     #BIT12,$TMDAT
                                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
011274                                  1$:
                                        ;*      MOV     @VCSTAT,$TMDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $TMDAT.
011304  105737  001670                  TSTB    $TMDAT
011310  100406                          BMI     TST100          ;;BR IF SET
011312  005337  001660                  DEC     TEMP                    ;DELAY
011316  001366                          BNE     1$              ;;BR IF NOT READY
011320  005300                          DEC     R0                      ;DECREMENT COUNTER
011322  001364                          BNE     1$              ;;BR IF NOT DONE
011324  104011                          ERROR   11                      ;ERROR, ERASE CLEARED READY AND FAILED
                                                                        ;TO SET READY AFTER A DELAY
                                ;;**************************************************************
                                ;*TEST 100       TEST THAT RESET CLEARS VC MODE BITS
                                ;;**************************************************************
011326  000004                  TST100: SCOPE
011330  012737  000002  001166          MOV     #2,$TIMES       ;;DO 2 ITERATIONS
011336  012737  000014  001670          MOV     #BIT3!BIT2,$TMDAT
                                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
011354  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
011362  004737  023174                  JSR     PC,$RESET
                                        ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
011376  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
011404  001401                          BEQ     TST101          ;;BR IF EQUAL
011406  104011                          ERROR   11                      ;ERROR, RESET FAILED TO CLEAR VC STATUS REG
                                ;;**************************************************************
                                ;*TEST 101       TEST THAT RESET CLEARS INTERRUPT ENABLE, CHANNEL, STORE, WRITE THRU
                                ;;**************************************************************
011410  000004                  TST101: SCOPE
011412  012737  000002  001166          MOV     #2,$TIMES       ;;DO 2 ITERATIONS
011420  012737  007100  001670          MOV     #BIT11!BIT10!BIT9!BIT6,$TMDAT
                                        ;*      MOV     $TMDAT,@VCSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCSTAT
011436  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
011444  004737  023174                  JSR     PC,$RESET
                                        ;*      MOV     @VCSTAT,$BDDAT  ;/READ DEVICE REG VCSTAT,PUT DATA IN $BDDAT.
011460  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
011466  001401                          BEQ     TST102          ;;BR IF EQUAL
011470  104011                          ERROR   11                      ;ERROR, RESET FAILED TO CLEAR VC STATUS
                                ;;**************************************************************
                                ;*TEST 102       TEST THAT RESET CLEARS X REGISTER
                                ;;**************************************************************
011472  000004                  TST102: SCOPE
```

```
1861  011474  012737  000002  001166          MOV     #2,$TIMES       ;;DO 2 ITERATIONS
1862  011502  012737  177777  001670          MOV     #-1,$TMDAT
1863
1864                                  ;*        MOV     $TMDAT,@VCXREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCXREG
1865  011520  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED
1866  011526  004737  023174                  JSR     PC,$RESET
1867
1868                                  ;*        MOV     @VCXREG,$BDDAT  ;/READ DEVICE REG VCXREG,PUT DATA IN $BDDAT.
1869  011542  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1870  011550  001401                          BEQ     TST103          ;;BR IF EQUAL
1871  011552  104011                          ERROR   11                      ;ERROR, RESET FAILED TO CLEAR VC X REGISTER
1872
1873                                  ;;**************************************************************
1874                                  ;*TEST 103       TEST THAT RESET CLEARS Y REGISTER
1875                                  ;;**************************************************************
1876  011554  000004                  TST103: SCOPE
1877  011556  012737  000002  001166          MOV     #2,$TIMES       ;;DO 2 ITERATIONS
1878  011564  012737  177777  001670          MOV     #-1,$TMDAT
1879
1880                                  ;*        MOV     $TMDAT,@VCYREG  ;/ PUT DATA FROM $TMDAT TO DEVICE REG VCYREG
1881  011602  012737  000000  001124          MOV     #0,$GDDAT               ;LOAD EXPECTED
1882  011610  004737  023174                  JSR     PC,$RESET
1883
1884                                  ;*        MOV     @VCYREG,$BDDAT  ;/READ DEVICE REG VCYREG,PUT DATA IN $BDDAT.
1885  011624  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1886  011632  001401                          BEQ     TST104          ;;BR IF EQUAL
1887  011634  104011                          ERROR   11                      ;ERROR, RESET FAILED TO CLEAR VC Y REGISTER
1888
1889                                  ;;**************************************************************
1890                                  ;*TEST 104       DOES EXTERNAL ENABLE (BIT 4) SET
1891                                  ;;**************************************************************
1892  011636  000004                  TST104: SCOPE
1893  011640  012737  000020  001124          MOV     #BIT4,$GDDAT            ;LOAD EXPECTED
1894
1895                                  ;*        MOV     $GDDAT,@ADCS    ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1896
1897                                  ;*        MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1898  011666  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1899  011674  001401                          BEQ     TST105          ;;BR IF SET
1900  011676  104001                          ERROR   1                       ;EXT ENABLE BIT 4 FAILED TO sET
1901
1902                                  ;;**************************************************************
1903                                  ;*TEST 105       DOES CLOCK OVERFLOW ENABLE (BIT 5) SET
1904                                  ;;**************************************************************
1905  011700  000004                  TST105: SCOPE
1906                                                                  ;SET BIT 5
1907  011702  012737  000040  001124          MOV     #BIT5,$GDDAT            ;LOAD EXPECTED
1908
1909                                  ;*        MOV     $GDDAT,@ADCS    ;/ PUT DATA FROM $GDDAT TO DEVICE REG ADCS
1910
1911                                  ;*        MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
1912  011730  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1913  011736  001401                          BEQ     TST106          ;;BR IF SET
1914  011740  104001                          ERROR   1                       ;CLOCK OVERFLOW ENABLE FAILED TO SET
```

```
                                ;;*****************************************************************
                                ;*TEST 106       DOES AD INTERRUPT ENABLE (BIT 6) SET
                                ;;*****************************************************************
011742  000004                  TST106: SCOPE
                                                                        ;SET BIT 6
011744  012737  000100  001124          MOV     #BIT6,$GDDAT          ;LOAD EXPECTED

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

                                ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
011772  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
012000  001401                        BEQ     TST107          ;;BR IF SET
012002  104001                        ERROR   1                       ;AD INTERRUPT ENABLE FAILED TO SET

                                ;;*****************************************************************
                                ;*TEST 107       DOES MUX CHANNEL (BIT 8) SET
                                ;;*****************************************************************
012004  000004                  TST107: SCOPE
                                                                        ;SET BIT 8
012006  012737  000400  001124          MOV     #BIT8,$GDDAT          ;LOAD EXPECTED

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

                                ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
012034  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
012042  001401                        BEQ     TST110          ;;BR IF SET
012044  104001                        ERROR   1                       ;MUX BIT 8 FAILED TO SET

                                ;;*****************************************************************
                                ;*TEST 110       DOES MUX CHANNEL (BIT 9) SET
                                ;;*****************************************************************
012046  000004                  TST110: SCOPE
                                                                        ;SET BIT 9
012050  012737  001000  001124          MOV     #BIT9,$GDDAT          ;LOAD EXPECTED

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

                                ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
012076  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
012104  001401                        BEQ     TST111          ;;BR IF EQUAL
012106  104001                        ERROR   1                       ;MUX BIT 9 FAILED TO SET

                                ;;*****************************************************************
                                ;*TEST 111       DOES MUX CHANNEL (BIT 10) SET
                                ;;*****************************************************************
012110  000004                  TST111: SCOPE
                                                                        ;SET BIT 10
012112  012737  002000  001124          MOV     #BIT10,$GDDAT         ;LOAD EXPECTED

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

                                ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
012140  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
012146  001401                        BEQ     TST112          ;;BR IF EQUAL
```

```
                                        ERROR   1                       ;MUX BIT 10 FAILED TO SET

                        ;;**************************************************************
                        ;*TEST 112       DOES MUX CHANNEL (BIT 11) SET
                        ;;**************************************************************
                        TST112: SCOPE
                                                                        ;SET BIT 11
                                        MOV     #BIT11,$GDDAT           ;LOAD EXPECTED

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

                        ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
                                CMP     $GDDAT,$BDDAT           ;COMPARE
                                BEQ     TST113          ;;BR IF EQUAL
                                ERROR   1                       ;MUX BIT 11 FAILED TO SET

                        ;;**************************************************************
                        ;*TEST 113       DOES UNIPOLAR/BIPOLAR (BIT 13) SET
                        ;;**************************************************************
                        TST113: SCOPE
                                                                        ;SET BIT 13
                                        MOV     #BIT13,$GDDAT           ;LOAD EXPECTED

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

                        ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
                                CMP     $GDDAT,$BDDAT           ;COMPARE
                                BEQ     TST114          ;;BR IF EQUAL
                                ERROR   1                       ;UNIPOLAR /BIPOLAR BIT FAILED TO SET

                        ;;**************************************************************
                        ;*TEST 114       DOES AD DONE (BIT 7) SET AND CLEAR
                        ;;**************************************************************
                        TST114: SCOPE

                        ;*      MOV     @ADDBR,$TMDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
                                MOV     $TMDAT,R0
                                MOV     #0,R0
                                MOV     #BIT0,$TMDAT

                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
                                MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
                        1$:

                        ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
                                TSTB    $TMDAT
                                BMI     2$
                                INC     R0
                                BNE     1$

                        ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
                                ERROR   1                       ;ERROR, A TO D DONE FAILED TO SET
```

```
2023  012360                                  2$:
2024
2025                                  ;*      MOV     @ADDBR,$TMDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2026  012370  013700  001670                  MOV     $TMDAT,R0
2027  012374  012737  000000  001124          MOV     #0,$GDDAT                       ;CLEAR EXPECTED
2028
2029                                  ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2030  012412  105737  001126                  TSTB    $BDDAT
2031  012416  100001                          BPL     3$              ;;BR IF CLEARED
2032  012420  104001                          ERROR   1                       ;ERROR, DON& FAILED TO CLEAR UPON
2033                                                                  ;READING CONVERTED VALUE
2034  012422                                  3$:
2035
2036                                  ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2037  012432  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
2038  012436  032737  000001  001126          BIT     #BIT0,$BDDAT
2039  012444  001401                          BEQ     TST115          ;;BR IF CLEARED
2040  012446  104001                          ERROR   1                       ;ERROR, AD GO FAILED TO CLEAR
2041
2042
2043                                  ;;*****************************************************************
2044                                  ;*TEST 115       TEST THAT THE CONVERTED NUMBER = 1777 (SW BIT 10=1)
2045                                  ;*****************************************************************
2046  012450  000004                  TST115: SCOPE
2047  012452  012737  000000  001670          MOV     #0,$TMDAT               ;INIT
2048
2049                                  ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2050
2051                                  ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2052  012500  005237  001670                  INC     $TMDAT
2053
2054                                  ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2055  012514                                  1$:
2056
2057                                  ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2058  012524  105737  001670                  TSTB    $TMDAT
2059  012530  100371                          BPL     1$
2060  012532  012737  001777  001124          MOV     #1777,$GDDAT            ;LOAD EXPECTED
2061
2062                                  ;*      MOV     @ADDBR,$BDDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
2063  012550  032777  002000  166362          BIT     #BIT10,@SWR             ;TEST BIT 10
2064  012556  001405                          BEQ     TST116          ;;BR IF CLEARED
2065  012560  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
2066  012566  001401                          BEQ     TST116          ;;BR IF EQUAL
2067  012570  104016                          ERROR   16                      ;CONVERTED VALUE NOT = 1777
2068                                                          ;OPERATOR INFORMED THE PROGRAM THAT THE
2069                                                          ; ALL 1'S JUMPER HAD BEEN INSTALLED
2070
2071                                  ;;*****************************************************************
2072                                  ;*TEST 116       TEST THAT NO EXTERNAL CONVERSIONS INPUT
2073                                  ;*****************************************************************
2074  012572  000004                  TST116: SCOPE
2075  012574  012737  000004  001166          MOV     #4,$TIMES               ;;DO 4 ITERATIONS
2076  012602  012737  000000  001670          MOV     #0,$TMDAT
```

```
2077
2078                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2079  012620  012737  000020  001124                MOV     #BIT4,$GDDAT            ;LOAD EXPECTED
2080
2081                                        ;*      MOV     @ADDBR,$TMDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $TMDAT.
2082  012636  013700  001670                        MOV     $TMDAT,R0
2083
2084                                        ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2085  012652  052737  000020  001670                BIS     #BIT4,$TMDAT
2086
2087                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2088  012670  012700  000000                        MOV     #0,R0
2089  012674  005200                        1$:     INC     R0
2090  012676  001376                                BNE     1$
2091
2092                                        ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2093  012710  105737  001126                        TSTB    $BDDAT
2094  012714  100001                                BPL     TST117          ;;BR IF PLUS
2095  012716  104001                                ERROR   1                       ;ERROR EXTERNAL CONVERSION
2096
2097                                        ;;**************************************************************
2098                                        ;*TEST 117      TEST THAT CLOCK CAN START A CONVERSION
2099                                        ;;**************************************************************
2100  012720  000004                        TST117: SCOPE
2101  012722  012737  000004  001166                MOV     #4,$TIMES       ;;DO 4 ITERATIONS
2102  012730  012737  000000  001670                MOV     #0,$TMDAT               ;
2103
2104                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2105
2106                                        ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2107  012756  012737  000240  001124                MOV     #BIT7!BIT5,$GDDAT       ;LOAD EXPECTED DATA
2108  012764  012737  000003  014610                MOV     #3,RATE                 ;LOAD CLOCK RATE
2109
2110                                        ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2111  013002  052737  000040  001670                BIS     #BIT5,$TMDAT
2112
2113                                        ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2114  013020  004737  014002                        JSR     PC,UPCNT                ;ENABLE THE CLOCK
2115  013024  012700  000040                        MOV     #BIT5,R0                ;SETUP DELAY
2116  013030  005300                        1$:     DEC     R0                      ;DELAY
2117  013032  001376                                BNE     1$
2118
2119                                        ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2120  013044  023737  001124  001126                CMP     $GDDAT,$BDDAT           ;COMPARE
2121  013052  001401                                BEQ     TST120          ;;BR IF EQUAL
2122  013054  104001                                ERROR   1                       ;CLOCK OVERFLOW FAILED TO START A CONVERSION
2123
2124                                        ;;**************************************************************
2125                                        ;*TEST 120      TEST THAT RESET CLEARS MUX AND UNIPOLAR BITS
2126                                        ;;**************************************************************
2127  013056  000004                        TST120: SCOPE
2128  013060  012737  000002  001166                MOV     #2,$TIMES       ;;DO 2 ITERATIONS
2129  013066  005037  001124                        CLR     $GDDAT
2130  013072  012737  027400  001670                MOV     #BIT13!BIT11!BIT10!BIT9!BIT8,$TMDAT
```

```
2131
2132                                    ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2133  013110  004737  023174                    JSR     PC,$RESET
2134
2135                                    ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2136  013124  005737  001126                    TST     $BDDAT
2137  013130  001401                            BEQ     TST121          ;;BR IF CLEARED
2138  013132  104000                            ERROR
2139
2140                                    ;;**************************************************************
2141                                    ;*TEST 121       TEST THAT RESET CLEARS EXT AND INTERRUPT ENABLE BITS
2142                                    ;;**************************************************************
2143  013134  000004                    TST121: SCOPE
2144  013136  012737  000002  001166            MOV     #2,$TIMES       ;;DO 2 ITERATIONS
2145  013144  012737  000120  001670            MOV     #BIT6!BIT4,$TMDAT
2146
2147                                    ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2148  013162  005037  001124                    CLR     $GDDAT                  ;CLEAR EXPECTED
2149  013166  004737  023174                    JSR     PC,$RESET
2150
2151                                    ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2152  013202  001401                            BEQ     TST122          ;;BR IF CLEARED
2153  013204  104001                            ERROR   1                       ;ERROR, RESET FAILED TO CLEAR ENABLES
2154
2155                                    ;;**************************************************************
2156                                    ;*TEST 122       TEST THAT RESET CLEARS AD DONE
2157                                    ;;**************************************************************
2158  013206  000004                    TST122: SCOPE
2159  013210  012737  000002  001166            MOV     #2,$TIMES       ;;DO 2 ITERATIONS
2160  013216  012737  000001  001670            MOV     #BIT0,$TMDAT
2161
2162                                    ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2163  013234  005037  001124                    CLR     $GDDAT
2164
2165                                    ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2166  013250  105737  001670                    TSTB    $TMDAT
2167  013254  100375                            BPL     .-4
2168  013256  004737  023174                    JSR     PC,$RESET
2169
2170                                    ;*      MOV     @ADCS,$BDDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $BDDAT.
2171  013272  001401                            BEQ     TST123          ;;BR IF CLEARED
2172  013274  104001                            ERROR   1                       ;ERROR, RESET FAILED TO cLEAR DONE FLAG
2173                                    ;;**************************************************************
2174                                    ;*TEST 123       TEST THAT RESET CLEARS AD BUFFER REG
2175                                    ;;**************************************************************
2176  013276  000004                    TST123: SCOPE
2177  013300  012737  000002  001166            MOV     #2,$TIMES       ;;DO 2 ITERATIONS
2178  013306  012737  000001  001670            MOV     #1,$TMDAT               ;CONVERT
2179
2180                                    ;*      MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2181  013324                            1$:
2182
2183                                    ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2184  013334  105737  001670                    TSTB    $TMDAT                  ;WAIT
```

```
2185  013340  100371                          BPL     1$
2186  013342  004737  023174                  JSR     PC,$RESET
2187
2188                                  ;*      MOV     @ADDBR,$BDDAT   ;/READ DEVICE REG ADDBR,PUT DATA IN $BDDAT.
2189  013356  005037  001124                  CLR     $GDDAT                  ;LOAD EXPECTED
2190  013362  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
2191  013370  001401                          BEQ     TST124          ;;BR IF CLEARED
2192  013372  104016                          ERROR   16                      ;RESET FAILED TO CLEAR AD BUFFER REGISTER
2193                                  ;;*************************************************************
2194                                  ;;*TEST 124      LOAD DIFFERENT NUMBERS INTO DIFFERENT REG.
2195                                  ;;*************************************************************
2196  013374  000004                  TST124: SCOPE
2197  013376  004737  023174                  JSR     PC,$RESET
2198  013402  013700  001626                  MOV     CSC,R0                  ;LOAD STARTING ADDRESS
2199  013406  062700  000002                  ADD     #2,R0                   ;ADJUST ADDRESS
2200  013412  012701  013566                  MOV     #BUFNUM+20,R1           ;LOAD STARTING TABLE ADDRESS
2201  013416  012702  000010                  MOV     #8.,R2                  ;LOAD COUNT
2202  013422  010037  001670          1$:     MOV     R0,$TMDAT
2203  013426  162700  000002                  SUB     #2,R0
2204  013432  162737  000002  001670          SUB     #2,$TMDAT
2205  013440  014137  001124                  MOV     -(R1),$GDDAT
2206
2207                                  ;*      MOV     $GDDAT,@$TMDAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG $TMDAT
2208  013454  005302                          DEC     R2                      ;DONE ALL
2209  013456  001361                          BNE     1$                      ;BR IF NOT
2210
2211  013460  013700  001610                  MOV     ADCS,R0                 ;LOAD STARTING POINTER
2212  013464  012701  013546                  MOV     #BUFNUM,R1              ;LOAD STARTING POINTER <EXPECTED>
2213  013470  012702  000010                  MOV     #8.,R2                  ;LOAD # OF REG
2214
2215  013474  011137  001124          3$:     MOV     (R1),$GDDAT             ;READ REG
2216  013500  010037  001670                  MOV     R0,$TMDAT
2217
2218                                  ;*      MOV     @$TMDAT,$BDDAT  ;/READ DEVICE REG $TMDAT,PUT DATA IN $BDDAT.
2219  013514  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
2220  013522  001403                          BEQ     4$              ;;BR IF EQUAL
2221  013524  010037  013566                  MOV     R0,BUFADR               ;SAVE BUS ADDRESS
2222  013530  104017                          ERROR   17                      ;INCORRECT DATA, REG. WAS CHANGED IN ERROR
2223  013532  005721                  4$:     TST     (R1)+           ;UPDATE POINTERS
2224  013534  062700  000002                  ADD     #2,R0
2225  013540  005302                          DEC     R2                      ;DONE ALL REG
2226  013542  001354                          BNE     3$                      ;BR IF NOT
2227  013544  000411                          BR      TST125          ;;NEXT TEST
2228
2229                                  ;# TO BE LOADED INTO DIFFERENT REG
2230
2231  013546  027560                  BUFNUM: 27560                           ;A TO D STATUS
2232  013550  000000                          0                               ;A TO D BUFFER
2233  013552  140736                          140736                          ;CLOCK STATUS
2234  013554  000377                          377                             ;CLOCK PRESET
2235  013556  007214                          7214                            ;VC STATUS
2236  013560  001252                          1252                            ;VC X POS
2237  013562  000525                          525                             ;VC Y POS
2238  013564  000377                          377                             ;CLOCK COUNTER
```

```
                                ;BUS ADDRESS OF REG IN ERROR

                        ;;*****************************************************************
                        ;*TEST 125      DETERMINE IF MORE AR11'S ARE TO BE TESTED
                        ;;*****************************************************************
                        ;ST125: SCOPE
                        BYPASS: TST     NBEXT                   ;TEST IF ANY
                                BEQ     1$                      ;BR IF NONE
                                ADD     #20,ARBADD              ;UPDATE DEVICE ADDRESS
                                ADD     #20,ARBVCT              ;UPDATE DEVICE VECTOR
                                DEC     NBEXT                   ;ANOTHER ONE ?
                                BR      BYPAS1                  ;BR IF ANOTHER
                        1$:     MOV     $BASE,ARBADD            ;RELOAD ADDRESS
                                MOV     $VECT1,ARBVCT           ;RELOAD VECTOR
                                MOV     NMBEXT,NBEXT            ;RELOAD NUMBER
                                JMP     $EOP                    ;DONE
                        BYPAS1: MOV     #-1,R0
                                JMP     RBEG2                   ;TEST ANOTHER UNIT
                        .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

                        $EOP:
                                SCOPE
                                CLR     $TSTNM          ;;ZERO THE TEST NUMBER
                                CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
                                INC     $PASS           ;;INCREMENT THE PASS NUMBER
                                BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
                                DEC     (PC)+           ;;LOOP?
                        $EOPCT: .WORD   1
                                BGT     $DOAGN          ;;YES
                                MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
                        $ENDCT: .WORD   1
                                $EOPCT
                                TYPE    ,$ENDMG         ;;TYPE "END PASS #"
                                MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
                        $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
                                BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
                                RESET                   ;;CLEAR THE WORLD
                        $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
                                NOP                     ;;SAVE ROOM
                                NOP                     ;;FOR
                                NOP                     ;;ACT11
                        $DOAGN:
                                JMP     @(PC)+          ;;RETURN
                        $RTNAD: .WORD   BYPAS1
```

```
2293  013762  377     377     000     $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
2294  013765  015     042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS */
2295  013772  050040  051501  020123
2296  014000  000043
2297
2298
2299                                    ;SUBROUTINE TO LOAD -2 INTO THE CLOCK PRESET REGISTER
2300                                    ; AND LOAD CLOCK RATE INTO CLOCK STATUS REGISTER
2301                                    ; AND START THE CLOCK. WAIT A PERIOD OF TIME THEN EXIT
2302
2303  014002  012737  000000  001670  UPCNT:  MOV     #0,$TMDAT               ;CLEAR CLOCK STATUS
2304
2305                                    ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2306  014020  012737  177776  001670          MOV     #-2,$TMDAT              ;MOVE -2 INT PRESET
2307
2308                                    ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
2309  014036  013737  014610  001670          MOV     RATE,$TMDAT             ;LOAD RATE AND ENABLE CLOCK
2310
2311                                    ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2312  014054  012737  000000  001656          MOV     #0,DELAY
2313  014062                          UPCNTA:
2314
2315                                    ;*      MOV     @CSR,$TMDAT     ;/READ DEVICE REG CSR,PUT DATA IN $TMDAT.
2316  014072  105737  001670                  TSTB    $TMDAT
2317  014076  100407                          BMI     UPCNTB                  ;
2318  014100  022727  000000  000000          CMP     #0,#0
2319  014106  062737  000001  001656          ADD     #1,DELAY                ;
2320  014114  001362                          BNE     UPCNTA                  ;
2321  014116  000207                  UPCNTB: RTS     7                       ;EXIT
2322
2323                                    .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2324
2325                                    ;;**********************************************************************
2326                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2327                                    ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2328                                    ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2329                                    ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2330                                    ;*REPLACED WITH SPACES.
2331                                    ;*CALL:
2332                                    ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
2333                                    ;*      TYPDS                   ;;GO TO THE ROUTINE
2334
2335  014120                          $TYPDS:
2336  014120  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2337  014122  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2338  014124  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2339  014126  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2340  014130  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2341  014132  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
2342  014136  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
2343  014142  100004                          BPL     1$              ;;BR IF INPUT IS POS.
2344  014144  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
2345  014146  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
2346  014154  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
```

```
2347  014156  012703  014334                    MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
2348  014162  112723  000040                    MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
2349  014166  005002                  2$:       CLR     R2              ;;CLEAR THE BCD NUMBER
2350  014170  016001  014324                    MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
2351  014174  160105                  3$:       SUB     R1,R5           ;;FORM THIS BCD DIGIT
2352  014176  002402                            BLT     4$              ;;BR IF DONE
2353  014200  005202                            INC     R2              ;;INCREASE THE BCD DIGIT BY 1
2354  014202  000774                            BR      3$
2355  014204  060105                  4$:       ADD     R1,R5           ;;ADD BACK THE CONSTANT
2356  014206  005702                            TST     R2              ;;CHECK IF BCD DIGIT=0
2357  014210  001002                            BNE     5$              ;;FALL THROUGH IF 0
2358  014212  105716                            TSTB    (SP)            ;;STILL DOING LEADING 0'S?
2359  014214  100407                            BMI     7$              ;;BR IF YES
2360  014216  106316                  5$:       ASLB    (SP)            ;;MSD?
2361  014220  103003                            BCC     6$              ;;BR IF NO
2362  014222  116663  000001  177777            MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
2363  014230  052702  000060          6$:       BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
2364  014234  052702  000040          7$:       BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2365  014240  110223                            MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2366  014242  005720                            TST     (R0)+           ;;JUST INCREMENTING
2367  014244  020027  000010                    CMP     R0,#10          ;;CHECK THE TABLE INDEX
2368  014250  002746                            BLT     2$              ;;GO DO THE NEXT DIGIT
2369  014252  003002                            BGT     8$              ;;GO TO EXIT
2370  014254  010502                            MOV     R5,R2           ;;GET THE LSD
2371  014256  000764                            BR      6$              ;;GO CHANGE TO ASC
2372  014260  105726                  8$:       TSTB    (SP)+           ;;WAS THE LSD THE   RST NON-ZERO?
2373  014262  100003                            BPL     9$              ;;BR IF NO
2374  014264  116663  177777  177776            MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
2375  014272  105013                  9$:       CLRB    (R3)            ;;SET THE TERMINATOR
2376  014274  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
2377  014276  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
2378  014300  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
2379  014302  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
2380  014304  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
2381  014306  104401  014334                    TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
2382  014312  016666  000002  000004            MOV     2(SP),4(SP)     ;;ADJUST THE STACK
2383  014320  012616                            MOV     (SP)+,(SP)
2384  014322  000002                            RTI                     ;;RETURN TO USER
2385  014324  023420                  $DTBL:    10000.
2386  014326  001750                            1000.
2387  014330  000144                            100.
2388  014332  000012                            10.
2389  014334  000004                  $DBLK:    .BLKW   4
2390                                  ;SUBROUTINE TO TEST THE CLOCK REPEATIBILITY
2391                                  ;FIRST CLEAR CLOCK STATUS AND PRESET BUFFER
2392                                  ; THEN ENABLE THE CLOCK TO COUNT AT A RATE.
2393                                  ; DECrEMENT R0 FOR SOME PERIOD OF TIME, WHEN R0 = 0
2394                                  ; SAVE THE COUNTER VALUE AND REPEAT THIS OPERATION AGAIN
2395                                  ;THEN COMPARE THE FIRST TIMED VALUE TO THE SECOND TIMED VALUE
2396                                  ; <MACHINE AND MEMORY TIMING NOT IMPORTANT>
2397                                  ; TO BE WITHIN THE VALUE SPECIFIED BY LOCATION CNTDEV
2398                                  ; IF GREATER THAN EXPECTED IT IS AN ERROR.
2399                                  ;ALSO TEST THAT THE COUNTER HAS REACHED A MIN. COUNT
2400
```

```
2401  014344  012737  000000  001670  REPEAT: MOV     #0,$TMDAT               ;STOP THE CLOCK
2402
2403                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2404
2405                                  ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
2406  014372  012537  014610                  MOV     (P5)+,RATE              ;SET UP RATE
2407  014376  012537  014612                  MOV     (R5)+,CNTDEV            ;SET UP CNT. DEV
2408  014402  012537  014614                  MOV     (R5)+,MINCNT            ;SET UP MIN COUNT
2409  014406  012537  014616                  MOV     (5)+,CKDLY              ;SAVE DELAY
2410  014412  004737  014502                  JSR     PC,10$                  ;DUMMY TO CHARGE THE "CACHE"
2411  014416  004737  014502                  JSR     PC,10$                  ;ENABLE THE CLOCK
2412  014422  010037  001124                  MOV     R0,$GDDAT               ;SAVE FIRST TIME
2413  014426  004737  014502                  JSR     PC,10$                  ;ENABLE THE CLOCK
2414  014432  010037  001126                  MOV     R0,$BDDAT               ;SAVE SECOND TIME
2415  014436  013700  001124                  MOV     $GDDAT,R0               ;GET FIRST TIME AGAIN
2416  014442  163700  001126                  SUB     $BDDAT,R0               ;SUBTRACT SECOND TIME
2417  014446  100001                          BPL     3$
2418  014450  005400                          NEG     R0                      ;MAGNITUDE OF DIFFERENCE IN R0
2419  014452  023737  001126  014614  3$:     CMP     $BDDAT,MINCNT           ;COMPARE TO MIN. COUNT
2420  014460  002004                          BGE     4$                      ;BRANCH IF GREATER
2421  014462  013737  014614  001124          MOV     MINCNT,$GDDAT           ;LOAD $GDDAT FOR TYPE-OUT
2422  014470  000205                          RTS     R5
2423  014472  005725                  4$:     TST     (R5)+                   ;UPDATE THE STACK
2424  014474  020037  014612                  CMP     R0,CNTDEV               ;COMPARE TO DEVIATION
2425  014500  000205                          RTS     R5
2426
2427  014502  013700  014616          10$:    MOV     CKDLY,R0                ;GET DELAY
2428  014506  013737  014610  001670          MOV     RATE,$TMDAT             ;LOAD RATE
2429
2430                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2431  014524  005237  001670                  INC     $TMDAT                  ;ENABLE CLOCK
2432
2433                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2434  014540  005300                  1$:     DEC     R0                      ;DELAY
2435  014542  001376                          BNE     1$
2436
2437                                  ;*      MOV     @CSC,$TMDAT     ;/READ DEVICE REG CSC,PUT DATA IN $TMDAT.
2438  014554  013700  001670                  MOV     $TMDAT,R0
2439  014560  012737  000000  001670          MOV     #0,$TMDAT               ;STOP CLOCK
2440
2441                                  ;*      MOV     $TMDAT,@CSR     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSR
2442
2443                                  ;*      MOV     $TMDAT,@CSB     ;/ PUT DATA FROM $TMDAT TO DEVICE REG CSB
2444  014606  000207                          RTS     PC                      ;EXIT
2445
2446  014610  000000          RATE:   0                                       ;CLOCK RATE
2447  014612  000000          CNTDEV: 0                                       ;CLOCK DEV.
2448  014614  000000          MINCNT: 0                                       ;MIN. COUNT
2449  014616  000000          CKDLY:  0
2450  014620  020101  047524  042040  EM1:    .ASCIZ  /A TO D STATUS REGISTER IN ERROR/
2451  014626  051440  040524  052524
2452  014634  020123  042522  044507
2453  014642  052123  051105  044440
2454  014650  020116  051105  047522
```

```
2455  014656  000122
2456  014660  020101  047524  042040  EM2:    .ASCIZ  /A TO D INTERRUPT ERROR/
2457  014666  044440  052116  051105
2458  014674  052522  052120  042440
2459  014702  051122  051117     000
2460  014707     103  047514  045503  EM3:    .ASCIZ  /CLOCK STATUS REGISTER IN ERROR/
2461  014714  051440  040524  052524
2462  014722  020123  042522  044507
2463  014730  052123  051105  044440
2464  014736  020116  051105  047522
2465  014744  000122
2466  014746  046103  041517  020113  EM4:    .ASCIZ  /CLOCK PRESET REGISTER IN ERROR/
2467  014754  051120  051505  052105
2468  014762  051040  043505  051511
2469  014770  042524  020122  047111
2470  014776  042440  051122  051117
2471  015004     000
2472  015005     103  047514  045503  EM5:    .ASCIZ  /CLOCK INTERRUPT ERROR/
2473  015012  044440  052116  051105
2474  015020  052522  052120  042440
2475  015026  051122  051117     000
2476  015033     103  047514  045503  EM6:    .ASCIZ  /CLOCK COUNTER REGISTER IN ERROR/
2477  015040  041440  052517  052116
2478  015046  051105  051040  043505
2479  015054  051511  042524  020122
2480  015062  047111  042440  051122
2481  015070  051117     000
2482  015073     103  047514  045503  EM7:    .ASCIZ  /CLOCK COUNTED IN ERROR/
2483  015100  041440  052517  052116
2484  015106  042105  044440  020116
2485  015114  051105  047522  000122
2486  015122  046103  041517  020113  EM10:   .ASCIZ  /CLOCK REPEATABILITY FAILED/
2487  015130  042522  042520  052101
2488  015136  041101  046111  052111
2489  015144  020131  040506  046111
2490  015152  042105     000
2491
2492  015155     126  020103  052123  EM11:   .ASCIZ  /VC STATUS REGISTER IN ERROR/
2493  015162  052101  051525  051040
2494  015170  043505  051511  042524
2495  015176  020122  047111  042440
2496  015204  051122  051117     000
2497  015211     130  051040  043505  EM12:   .ASCIZ  /X REGISTER IN ERROR/
2498  015216  051511  042524  020122
2499  015224  047111  042440  051122
2500  015232  051117     000
2501  015235     131  051040  043505  EM13:   .ASCIZ  /Y REGISTER IN ERROR/
2502  015242  051511  042524  020122
2503  015250  047111  042440  051122
2504  015256  051117     000
2505  015261     123  047503  042520  EM14:   .ASCIZ  /SCOPE INTERRUPT ERROR/
2506  015266  044440  052116  051105
2507  015274  052522  052120  042440
2508  015302  051122  051117     000
```

```
2509  015307     104  053105  041511  E"15:   .ASCIZ  /DEVICE BUS ERROR/
2510  015314  020105  052502  020123
2511  015322  051105  047522  000122
2512
2513  015330  047111  047503  051122  EM16:   .ASCIZ  "INCORRECT A/D BUFFER DATA"
2514  015336  041505  020124  027501
2515  015344  020104  052502  043106
2516  015352  051105  042040  052101
2517  015360  000101
2518  015362  044103  047101  042507  EM17:   .ASCIZ  /CHANGED REGISTER IN ERROR/
2519  015370  020104  042522  044507
2520  015376  052123  051105  044440
2521  015404  020116  051105  047522
2522  015412  000122
2523  015414  051105  050122  020103  DH1:    .ASCIZ  /ERRPC    ADCS    ADSTAT EXPECTED/
2524  015422  020040  040440  041504
2525  015430  020123  020040  042101
2526  015436  052123  052101  042440
2527  015444  050130  041505  042524
2528  015452  000104
2529  015454  051105  050122  020103  DH2:    .ASCIZ  /ERRPC    ADCS/
2530  015462  020040  040440  041504
2531  015470  000123
2532  015472  051105  050122  020103  DH3:    .ASCIZ  /ERRPC    CSR     CKSTAT EXPECTED/
2533  015500  020040  041440  051123
2534  015506  020040  020040  045503
2535  015514  052123  052101  042440
2536  015522  050130  041505  042524
2537  015530  000104
2538  015532  051105  050122  020103  DH4:    .ASCIZ  /ERRPC    CSB     CKBUFF EXPECTED/
2539  015540  020040  041440  041123
2540  015546  020040  020040  045503
2541  015554  052502  043106  042440
2542  015562  050130  041505  042524
2543  015570  000104
2544  015572  051105  050122  020103  DH5:    .ASCIZ  /ERRPC    CSR/
2545  015600  020040  041440  051123
2546  015606     000
2547  015607     105  051122  041520  DH6:    .ASCIZ  /ERRPC    CSC     CKCNTR EXPECTED/
2548  015614  020040  020040  051503
2549  015622  020103  020040  041440
2550  015630  041513  052116  020122
2551  015636  054105  042520  052103
2552  015644  042105     000
2553  015647     105  051122  041520  DH10:   .ASCIZ  /ERRPC    CSC     TIME1   TIME2/
2554  015654  020040  020040  051503
2555  015662  020103  020040  052040
2556  015670  046511  030505  020040
2557  015676  052040  046511  031105
2558  015704     000
2559  015705     105  051122  041520  DH11:   .ASCIZ  /ERRPC    VCADR   VCSTAT EXPECTED/
2560  015712  020040  053040  040503
2561  015720  051104  020040  053040
2562  015726  051503  040524  020124
```

```
2563  015734  054105  042520  052103
2564  015742  042105  000
2565  015745  105  051122  041520  DH12:   .ASCIZ  /ERRPC   VCXREG  X POS.  EXPECTED/
2566  015752  020040  053040  054103
2567  015760  042522  020107  054040
2568  015766  050040  051517  020056
2569  015774  042440  050130  041505
2570  016002  042524  000104
2571  016006  051105  050122  020103  DH13:   .ASCIZ  /ERRPC   VCYREG  Y POS.  EXPECTED/
2572  016014  020040  041526  051131
2573  016022  043505  020040  020131
2574  016030  047520  027123  020040
2575  016036  054105  042520  052103
2576  016044  042105  000
2577  016047  105  051122  041520  DH14:   .ASCIZ  /ERRPC   VCSTAT/
2578  016054  020040  053040  051503
2579  016062  040524  000124
2580  016066  051105  050122  020103  DH15:   .ASCIZ  /ERRPC    BASE    ACTUAL/
2581  016074  020040  041040  051501
2582  016102  020105  020040  040440
2583  016110  052103  040525  000114
2584  016116  051105  050122  020103  DH16:   .ASCIZ  /ERRPC   ADDBR    READ   EXPECTED/
2585  016124  020040  042101  041104
2586  016132  020122  020040  051040
2587  016140  040505  020104  020040
2588  016146  054105  042520  052103
2589  016154  042105  000
2590  016157  105  051122  041520  DH17:   .ASCIZ  /ERRPC   BUFADR   READ   EXPECTED/
2591  016164  020040  041040  043125
2592  016172  042101  020122  020040
2593  016200  042522  042101  020040
2594  016206  042440  050130  041505
2595  016214  042524  000104
2596                                        .EVEN
2597  016220  001116  001610  001126  DT1:    $ERRPC,ADCS,$BDDAT,$GDDAT,0
2598  016226  001124  000000
2599  016232  001116  001610  000000  DT2:    $ERRPC,ADCS,0
2600  016240  001116  001614  001126  DT3:    $ERRPC,CSR,$BDDAT,$GDDAT,0
2601  016246  001124  000000
2602  016252  001116  001616  001126  DT4:    $ERrPC,CSB,$BDDAT,$GDDAT,0
2603  016260  001124  000000
2604  016264  001116  001614  000000  DT5:    $ERRPC,CSR,0
2605  016272  001116  001626  001126  DT6:    $ERRPC,CSC,$BDDAT,$GDDAT,0
2606  016300  001124  000000
2607  016304  001116  001620  001126  DT11:   $ERRPC,VCSTAT,$BDDAT,$GDDAT,0
2608  016312  001124  000000
2609  016316  001116  001622  001126  DT12:   $ERRPC,VCXREG,$BDDAT,$GDDAT,0
2610  016324  001124  000000
2611  016330  001116  001624  001126  DT13:   $ERRPC,VCYREG,$BDDAT,$GDDAT,0
2612  016336  001124  000000
2613  016342  001116  001620  000000  DT14:   $ERRPC,VCSTAT,0
2614  016350  001116  001644  001126  DT15:   $ERRPC,ARBADD,$BDDAT,0
2615  016356  000000
2616  016360  001116  001612  001126  DT16:   $ERRPC,ADDBR,$BDDAT,$GDDAT,0
```

```
 2617  016366  001124  000000
 2618  016372  001116  013566  001126  DT17:   $ERRPC,BUFADR,$BDDAT,$GDDAT,0
 2619  016400  001124  000000
 2620
 2621                                  .SBTTL  SCOPE HANDLER ROUTINE
 2622
 2623                                  ;;**********************************************************************
 2624                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 2625                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 2626                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
 2627                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 2628                                  ;*SW14=1         LOOP ON TEST
 2629                                  ;*SW11=1         INHIBIT ITERATIONS
 2630                                  ;*SW09=1         LOOP ON ERROR
 2631                                  ;*SW08=1         LOOP ON TEST IN SWR<7:0>
 2632                                  ;*CALL
 2633                                  ;*      SCOPE           ;;SCOPE=IOT
 2634
 2635  016404                          $SCOPE:
 2636  016404  104407                         CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
 2637  016406  104407                         CKSWR
 2638  016410  032777  040000  162522  1$:    BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 2639  016416  001114                         BNE     $OVER           ;;YES IF SW14=1
 2640                                  ;*****START OF CODE FOR THE XOR TESTER*****
 2641  016420  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
 2642                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 2643  016422  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 2644  016426  012737  016446  000004         MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 2645  016434  005737  177060                 TST     @#177060        ;;TIME OUT ON XOR?
 2646  016440  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 2647  016444  000463                         BR      $SVLAD          ;;GO TO THE NEXT TEST
 2648  016446  022626                  5$:    CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 2649  016450  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 2650  016454  000423                         BR      7$              ;;LOOP ON THE PRESENT TEST
 2651  016456                          6$:;*****END OF CODE FOR THE XOR TESTER*****
 2652  016456  032777  000400  162454         BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 2653  016464  001404                         BEQ     2$              ;;BR IF NO
 2654  016466  127737  162446  001102         CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
 2655  016474  001465                         BEQ     $OVER           ;;BR IF YES
 2656  016476  105737  001103          2$:    TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 2657  016502  001421                         BEQ     3$              ;;BR IF NO
 2658  016504  123737  001115  001103         CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 2659  016512  101015                         BHI     3$              ;;BR IF NO
 2660  016514  032777  001000  162416         BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 2661  016522  001404                         BEQ     4$              ;;BR IF NO
 2662  016524  013737  001110  001106  7$:    MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 2663  016532  000446                         BR      $OVER
 2664  016534  105037  001103          4$:    CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 2665  016540  005037  001166                 CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 2666  016544  000415                         BR      1$              ;;ESCAPE TO THE NEXT TEST
 2667  016546  032777  004000  162364  3$:    BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
 2668  016554  001011                         BNE     1$              ;;BR IF YES
 2669  016556  005737  001204                 TST     $PASS           ;;IF FIRST PASS OF PROGRAM
 2670  016562  001406                         BEQ     1$              ;;      INHIBIT ITERATIONS
```

```
2671  016564  005237  001104                  INC     $ICNT                   ;;INCREMENT ITERATION COUNT
2672  016570  023737  001166  001104          CMP     $TIMES,$ICNT            ;;CHECK THE NUMBER OF ITERATIONS MADE
2673  016576  002024                          BGE     $OVER                   ;;BR IF MORE ITERATION REQUIRED
2674  016600  012737  000001  001104  1$:     MOV     #1,$ICNT                ;;REINITIALIZE THE ITERATION COUNTER
2675  016606  013737  016664  001166          MOV     $MXCNT,$TIMES           ;;SET NUMBER OF ITERATIONS TO DO
2676  016614  105237  001102          $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
2677  016620  113737  001102  001202          MOVB    $TSTNM,$TESTN           ;;SET TEST NUMBER IN APT MAILBOX
2678  016626  011637  001106                  MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
2679  016632  011637  001110                  MOV     (SP),$LPERR             ;;SAVE ERROR LOOP ADDRESS
2680  016636  005037  001170                  CLR     $ESCAPE                 ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
2681  016642  112737  000001  001115          MOVB    #1,$ERMAX               ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2682  016650  013777  001102  162264  $OVER:  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER
2683  016656  013716  001106                  MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
2684  016662  000002                          RTI                             ;;FIXES PS
2685  016664  000006          $MXCNT: 6.                                      ;;MAX. NUMBER OF ITERATIONS
2686                           .SBTTL  ERROR HANDLER ROUTINE
2687
2688                           ;;**********************************************************************
2689                           ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2690                           ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2691                           ;*AND GO TO $ERRTYP ON ERROR
2692                           ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2693                           ;*SW15=1        HALT ON ERROR
2694                           ;*SW13=1        INHIBIT ERROR TYPEOUTS
2695                           ;*SW09=1        LOOP ON ERROR
2696                           ;*CALL
2697                           ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2698
2699  016666                  $ERROR:
2700  016666  104407                          CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
2701  016670  105237  001103          7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
2702  016674  001775                          BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
2703  016676  013777  001102  162236          MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
2704  016704  005237  001112                  INC     $ERTTL                  ;;INC THE ERROR COUNT
2705  016710  011637  001116                  MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
2706  016714  162737  000002  001116          SUB     #2,$ERRPC
2707  016722  117737  162170  001114          MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
2708  016730  032777  020000  162202          BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
2709  016736  001004                          BNE     20$                     ;;SKIP TYPEOUTS
2710  016740  004737  017730                  JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
2711  016744  104401  001173                  TYPE    ,$CRLF
2712  016750                          20$:
2713  016750  122737  000001  001216          CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
2714  016756  001007                          BNE     2$                      ;;NO,SKIP APT ERROR REPORT
2715  016760  113737  001114  016772          MOVB    $ITEMB,21$              ;;SET ITEM NUMBER AS ERROR NUMBER
2716  016766  004737  021032                  JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
2717  016772     000          21$:    .BYTE   0
2718  016773     000                  .BYTE   0
2719  016774  000777          22$:    BR      22$                             ;;APT ERROR LOOP
2720  016776  005777  162136  2$:     TST     @SWR                            ;;HALT ON ERROR
2721  017002  100002                  BPL     3$                              ;;SKIP IF CONTINUE
2722  017004  000000                  HALT                                    ;;HALT ON ERROR!
2723  017006  104407                  CKSWR                                   ;;TEST FOR CHANGE IN SOFT-SWR
2724  017010  032777  001000  162122  3$:     BIT     #BIT09,@SWR             ;;LOOP ON ERROR SWITCH SET?
```

```
2725  017016  001402                        BEQ     4$                      ;;BR IF NO
2726  017020  013716  001110                MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
2727  017024  005737  001170        4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
2728  017030  001402                        BEQ     5$                      ;;BR IF NONE
2729  017032  013716  001170                MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
2730  017036                          5$:
2731  017036  022737  013746  000042        CMP     #$ENDAD,@#42            ;;ACT-11 AUTO-ACCEPT?
2732  017044  001001                        BNE     6$                      ;;BRANCH IF NO
2733  017046  000000                        HALT                            ;;YES
2734  017050                          6$:
2735  017050  000002                        RTI                             ;;RETURN
2736                                  .SBTTL  TTY INPUT ROUTINE
2737
2738                                  ;;**************************************************************
2739                                  .ENABL  LSB
2740
2741                                  ;;**************************************************************
2742                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2743                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2744                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2745                                  ;*WHEN OPERATING IN TTY FLAG MODE.
2746  017052  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR              ;;IS THE SOFT-SWR SELECTED?
2747  017060  001074                        BNE     15$                     ;;BRANCH IF NO
2748  017062  105777  162056                TSTB    @$TKS                   ;;CHAR THERE?
2749  017066  100071                        BPL     15$                     ;;IF NO, DON'T WAIT AROUND
2750  017070  117746  162052                MOVB    @$TKB,-(SP)             ;;SAVE THE CHAR
2751  017074  042716  177600                BIC     #^C177,(SP)             ;;STRIP-OFF THE ASCII
2752  017100  022726  000007                CMP     #7,(SP)+                ;;IS IT A CONTROL G?
2753  017104  001062                        BNE     15$                     ;;NO, RETURN TO USER
2754  017106  123727  001134  000001        CMPB    $AUTOB,#1               ;;ARE WE RUNNING IN AUTO-MODE?
2755  017114  001456                        BEQ     15$                     ;;BRANCH IF YES
2756
2757  017116  104401  017577                TYPE    ,$CNTLG                 ;;ECHO THE CONTROL-G (^G)
2758  017122  104401  017604        $GTSWR: TYPE    ,$MSWR                  ;;TYPE CURRENT CONTENTS
2759  017126  013746  000176                MOV     $SWREG,-(SP)            ;;SAVE SWREG FOR TYPEOUT
2760  017132  104402                        TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2761  017134  104401  017615                TYPE    ,$MNEW                  ;;PROMPT FOR NEW SWR
2762  017140  005046                19$:    CLR     -(SP)                   ;;CLEAR COUNTER
2763  017142  005046                        CLR     -(SP)                   ;;THE NEW SWR
2764  017144  105777  161774        7$:     TSTB    @$TKS                   ;;CHAR THERE?
2765  017150  100375                        BPL     7$                      ;;IF NOT TRY AGAIN
2766
2767  017152  117746  161770                MOVB    @$TKB,-(SP)             ;;PICK UP CHAR
2768  017156  042716  177600                BIC     #^C177,(SP)             ;;MAKE IT 7-BIT ASCII
2769
2770
2771
2772  017162  021627  000025        9$:     CMP     (SP),#25                ;;IS IT A CONTROL-U?
2773  017166  001005                        BNE     10$                     ;;BRANCH IF NOT
2774  017170  104401  017572                TYPE    ,$CNTLU                 ;;YES, ECHO CONTROL-U (^U)
2775  017174  062706  000006        20$:    ADD     #6,SP                   ;;IGNORE PREVIOUS INPUT
2776  017200  000757                        BR      19$                     ;;LET'S TRY IT AGAIN
2777
2778
```

```
2779  017202  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
2780  017206  001022                          BNE     16$             ;;BRANCH IF NO
2781  017210  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
2782  017214  001403                          BEQ     11$             ;;BRANCH IF YES
2783  017216  016677  000002  161714          MOV     2(SP),@SWR      ;;SAVE NEW SWR
2784  017224  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
2785  017230  104401  001173          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
2786  017234  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
2787  017242  001003                          BNE     15$             ;;BRANCH IF NOT
2788  017244  012777  000100  161672          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
2789  017252  000002          15$:    RTI                             ;;RETURN
2790  017254  004737  020744          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
2791  017260  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
2792  017264  002420                          BLT     18$             ;;BRANCH IF YES
2793  017266  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
2794  017272  003015                          BGT     18$             ;;BRANCH IF YES
2795  017274  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2796  017300  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
2797  017304  001403                          BEQ     17$             ;;BRANCH IF YES
2798  017306  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
2799  017310  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
2800  017312  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
2801  017314  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
2802  017320  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2803  017324  000707                          BR      7$              ;;GET THE NEXT ONE
2804  017326  104401  001172          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
2805  017332  000720                          BR      20$             ;;SIMULATE CONTROL-U
2806                                  .DSABL  LSB
2807
2808
2809                                  ;;*******************************************************************

2810                                  ;*CALL:ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY-
2811                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
2812                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
2813                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
2814                                  ;
2815
2816
2817  017334  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
2818  017336  016666  000004  000002  MOV     4(SP),2(SP)     ;;SAVE THE PS
2819  017344  105777  161574          1$:     TSTB    @$TKS           ;;WAIT FOR
2820  017350  100375                          BPL     1$              ;;A CHARACTER
2821  017352  117766  161570  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
2822  017360  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
2823  017366  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
2824  017374  001013                          BNE     3$              ;;BRANCH IF NO
2825  017376  105777  161542          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
2826  017402  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
2827  017404  117746  161536                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
2828  017410  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2829  017414  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
2830  017420  001366                          BNE     2$              ;;IF NOT DISCARD IT
2831  017422  000750                          BR      1$              ;;YES, RESUME
2832  017424  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
```

```
2833  017432  002407                        BLT     4$                      ;;BRANCH IF YES
2834  017434  026627  000004  000175        CMP     4(SP),#175              ;;IS IT A SPECIAL CHAR?
2835  017442  003003                        BGT     4$                      ;;BRANCH IF YES
2836  017444  042766  000040  000004        BIC     #40,4(SP)               ;;MAKE IT UPPER CASE
2837  017452  000002              4$:       RTI                             ;;GO BACK TO USER
2838                                ;;**********************************************************************
2839                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2840                                ;*CALL:
2841                                ;*      RDLIN                           ;;INPUT A STRING FROM THE TTY
2842                                ;*      RETURN HERE                     ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2843                                ;*                                      ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2844
2845  017454  010346              $RDLIN: MOV     R3,-(SP)                ;;SAVE R3
2846  017456  012703  017562      1$:     MOV     #$TTYIN,R3              ;;GET ADDRESS
2847  017462  022703  017572      2$:     CMP     #$TTYIN+8.,R3           ;;BUFFER FULL?
2848  017466  101405                      BLOS    4$                      ;;BR IF YES
2849  017470  104410                      RDCHR                           ;;GO READ ONE CHARACTER FROM THE TTY
2850  017472  112613                      MOVB    (SP)+,(R3)              ;;GET CHARACTER
2851  017474  122713  000177      10$:    CMPB    #177,(R3)               ;;IS IT A RUBOUT
2852  017500  001003                      BNE     3$                      ;;SKIP IF NOT
2853  017502  104401  001172      4$:     TYPE    ,$QUES                  ;;TYPE A '?'
2854  017506  000763                      BR      1$                      ;;CLEAR THE BUFFER AND LOOP
2855  017510  111337  017560      3$:     MOVB    (R3),9$                 ;;ECHO THE CHARACTER
2856  017514  104401  017560              TYPE    ,9$
2857  017520  122723  000015              CMPB    #15,(R3)+               ;;CHECK FOR RETURN
2858  017524  001356                      BNE     2$                      ;;LOOP IF NOT RETURN
2859  017526  105063  177777              CLRB    -1(R3)                  ;;CLEAR RETURN (THE 15)
2860  017532  104401  001174              TYPE    ,$LF                    ;;TYPE A LINE FEED
2861  017536  012603                      MOV     (SP)+,R3                ;;RESTORE R3
2862  017540  011646                      MOV     (SP),-(SP)              ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2863  017542  016666  000004  000002      MOV     4(SP),2(SP)             ;;      FIRST ASCII CHARACTER ON IT
2864  017550  012766  017562  000004      MOV     #$TTYIN,4(SP)
2865  017556  000002                      RTI                             ;;RETURN
2866  017560     000              9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
2867  017561     000                      .BYTE   0                       ;;TERMINATOR
2868  017562  000010              $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
2869  017572  052536  005015  000  $CNTLU: .ASCIZ  /↑U/<15><12>            ;;CONTROL "U"
2870  017577     136  006507  000012 $CNTLG: .ASCIZ  /↑G/<15><12>          ;;CONTROL "G"
2871  017604  005015  053523  020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
2872  017612  020075     000
2873  017615     040  047040  053505 $MNEW:  .ASCIZ  /  NEW = /
2874  017622  036440  000040
2875                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2876
2877                                ;;**********************************************************************
2878                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2879                                ;*CHANGE IT TO BINARY.
2880                                ;*CALL:
2881                                ;*      RDOCT                           ;;READ AN OCTAL NUMBER
2882                                ;*      RETURN HERE                     ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2883                                ;*                                      ;;HIGH ORDER BITS ARE IN $HIOCT
2884
2885  017626  011646              $RDOCT: MOV     (SP),-(SP)              ;;PROVIDE SPACE FOR THE
2886  017630  016666  000004  000002      MOV     4(SP),2(SP)             ;;INPUT NUMBER
```

```
2887  017636  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2888  017640  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2889  017642  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2890  017644  104411          1$:     RDLIN                   ;;READ AN ASCIZ LINE
2891  017646  012600                  MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2892  017650  005001                  CLR     R1              ;;CLEAR DATA WORD
2893  017652  005002                  CLR     R2
2894  017654  112046          2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
2895  017656  001412                  BEQ     3$              ;;IF ZERO GET OUT
2896  017660  006301                  ASL     R1              ;;*2
2897  017662  006102                  ROL     R2
2898  017664  006301                  ASL     R1              ;;*4
2899  017666  006102                  ROL     R2
2900  017670  006301                  ASL     R1              ;;*8
2901  017672  006102                  ROL     R2
2902  017674  042716  177770          BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
2903  017700  062601                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
2904  017702  000764                  BR      2$              ;;LOOP
2905  017704  005726          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
2906  017706  010166  000012          MOV     R1,12(SP)       ;;SAVE THE RESULT
2907  017712  010237  017726          MOV     R2,$HIOCT
2908  017716  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
2909  017720  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
2910  017722  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
2911  017724  000002                  RTI                     ;;RETURN
2912  017726  000000          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
2913
2914                          .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
2915
2916                          ;;**************************************************************
2917                          ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
2918                          ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
2919                          ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
2920
2921  017730          $ERRTYP:
2922  017730  104401  001173          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
2923  017734  010046                  MOV     R0,-(SP)        ;;SAVE R0
2924  017736  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
2925  017740  153700  001114          BISB    @#$ITEMB,R0
2926  017744  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
2927                                                          ;;TYPE THE PC OF THE ERROR
2928  017746  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
2929                                                          ;;ERROR ADDRESS
2930  017752  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2931  017754  000426                  BR      6$              ;;GET OUT
2932  017756  005300          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
2933  017760  006300                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
2934  017762  006300                  ASL     R0
2935  017764  006300                  ASL     R0
2936  017766  062700  001322          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
2937  017772  012037  020002          MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
2938  017776  001404                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
2939  020000  104401                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
2940  020002  000000          2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
```

```
2941  020004  104401  001173            TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
2942  020010  012037  020020    3$:     MOV     @(R0)+,4$       ;;PICKUP "DATA HEADER" POINTER
2943  020014  001404                    BEQ     5$              ;;SKIP TYPEOUT IF 0
2944  020016  104401                    TYPE                    ;;TYPE THE "DATA HEADER"
2945  020020  000000    4$:             .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
2946  020022  104401  001173            TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
2947  020026  011000    5$:             MOV     @R0,R0          ;;PICKUP "DATA TABLE" POINTER
2948  020030  001004                    BNE     7$              ;;GO TYPE THE DATA
2949  020032  012600    6$:             MOV     (SP)+,R0        ;;RESTORE R0
2950  020034  104401  001173            TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
2951  020040  000207                    RTS     PC              ;;RETURN
2952  020042            7$:
2953  020042  013046                    MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
2954  020044  104402                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2955  020046  005710                    TST     (R0)            ;;IS THERE ANOTHER NUMBER?
2956  020050  001770                    BEQ     6$              ;;BR IF NO
2957  020052  104401  020060            TYPE    ,8$             ;;TYPE TWO(2) SPACES
2958  020056  000771                    BR      7$              ;;LOOP
2959  020060  020040      000   8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
2960            020064                  .EVEN
2961
2962                                    .SBTTL  POWER DOWN AND UP ROUTINES
2963
2964                                    ;;**************************************************************
2965                                    ;POWER DOWn ROUTINE
2966  020064  012737  020230  000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
2967  020072  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
2968  020100  010046                       MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2969  020102  010146                       MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2970  020104  010246                       MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2971  020106  010346                       MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2972  020110  010446                       MOV     R4,-(SP)        ;;PUSH R4 ON STACK
2973  020112  010546                       MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2974  020114  017746  161020               MOV     @$WR,-(SP)      ;;PUSH @$WR ON STACK
2975  020120  010637  020234               MOV     SP,$SAVR6       ;;SAVE SP
2976  020124  012737  020136  000024       MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
2977  020132  000000                       HALT
2978  020134  000776                       BR      .-2             ;;HANG UP
2979
2980                                    ;;**************************************************************
2981                                    ;POWER UP ROUTINE
2982  020136  012737  020230  000024 $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
2983  020144  013706  020234               MOV     $SAVR6,SP       ;;GET SP
2984  020150  005037  020234               CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
2985  020154  005237  020234       1$:     INC     $SAVR6          ;;WAIT FOR THE INC
2986  020160  001375                       BNE     1$              ;;OF  WORD
2987  020162  012677  160752               MOV     (SP)+,@$WR      ;;POP STACK INTO @$WR
2988  020166  012605                       MOV     (SP)+,R5        ;;POP STACK INTO R5
2989  020170  012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
2990  020172  012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
2991  020174  012602                       MOV     (SP)+,R2        ;;POP STACK INTO R2
2992  020176  012601                       MOV     (SP)+,R1        ;;POP STACK INTO R1
2993  020200  012600                       MOV     (SP)+,R0        ;;POP STACK INTO R0
2994  020202  012737  020064  000024       MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
```

```
2995   020210  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
2996   020216  104401                          TYPE                    ;;REPORT THE POWER FAILURE
2997   020220  020236                  $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
2998   020222  012716                          MOV     (PC)+,(SP)      ;;RESTART AT BEGIN1
2999   020224  001676                  $PWRAD: .WORD   BEGIN1           ;;RESTART ADDRESS
3000   020226  000002                          RTI
3001   020230  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
3002   020232  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
3003   020234  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
3004   020236  005015  042522  052123  PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12><12>
3005   020244  051101  044524  043516
3006   020252  040440  052106  051105
3007   020260  040440  050040  053517
3008   020266  051105  043040  044501
3009   020274  052514  042522  005015
3010   020302  000012
3011                                   .EVEN
3012
3013                                   .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3014
3015                                   ;;**************************************************************
3016                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3017                                   ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3018                                   ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3019                                   ;*CALL:
3020                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3021                                   ;*      TYPOS                   ;;CALL FOR TYPEOUT
3022                                   ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3023                                   ;*      .BYTE   M               ;;M=1 OR 0
3024                                   ;*                                      ;;1=TYPE LEADING ZEROS
3025                                   ;*                                      ;;0=SUPPRESS LEADING ZEROS
3026                                   ;*
3027                                   ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3028                                   ;*$TYPOS OR $TYPOC
3029                                   ;*CALL:
3030                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3031                                   ;*      TYPON                   ;;CALL FOR TYPEOUT
3032                                   ;*
3033                                   ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3034                                   ;*CALL:
3035                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3036                                   ;*      TYPOC                   ;;CALL FOR TYPEOUT
3037
3038   020304  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
3039   020310  116637  000001  020527          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
3040   020316  112637  020531                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
3041   020322  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
3042   020326  000406                          BR      $TYPON
3043   020330  112737  000001  020527  $TYpOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
3044   020336  112737  000006  020531          MOVB    #6,$OMoDE+1     ;;SET FOR SIX(6) DIGITS
3045   020344  112737  000005  020526  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
3046   020352  010346                          MOV     R3,-(SP)        ;;SAVE R3
3047   020354  010446                          MOV     R4,-(SP)        ;;SAVE R4
3048   020356  010546                          MOV     R5,-(SP)        ;;SAVE R5
```

```
3049  020360  113704  020531              MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
3050  020364  005404                      NEG     R4
3051  020366  062704  000006              ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
3052  020372  110437  020530              MOVB    R4,$OMODE       ;;SAVE IT FOR USE
3053  020376  113704  020527              MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
3054  020402  016605  000012              MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
3055  020406  005003                      CLR     R3              ;;CLEAR THE OUTPUT WORD
3056  020410  006105              1$:     ROL     R5              ;;ROTATE MSB INTO "C"
3057  020412  000404                      BR      3$              ;;GO DO MSB
3058  020414  006105              2$:     ROL     R5              ;;FORM THIS DIGIT
3059  020416  006105                      ROL     R5
3060  020420  006105                      ROL     R5
3061  020422  010503                      MOV     R5,R3
3062  020424  006103              3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
3063  020426  105337  020530              DECB    $OMODE          ;;TYPE THIS DIGIT?
3064  020432  100016                      BPL     7$              ;;BR IF NO
3065  020434  042703  177770              BIC     #177770,R3      ;;GET RID OF JUNK
3066  020440  001002                      BNE     4$              ;;TEST FOR 0
3067  020442  005704                      TST     R4              ;;SUPPRESS THIS 0?
3068  020444  001403                      BEQ     5$              ;;BR IF YES
3069  020446  005204              4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
3070  020450  052703  000060              BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
3071  020454  052703  000040      5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3072  020460  110337  020524              MOVB    R3,8$           ;;SAVE FOR TYPING
3073  020464  104401  020524              TYPE    ,8$             ;;GO TYPE THIS DIGIT
3074  020470  105337  020526      7$:     DECB    $OCNT           ;;COUNT BY 1
3075  020474  003347                      BGT     2$              ;;BR IF MORE TO DO
3076  020476  002402                      BLT     6$              ;;BR IF DONE
3077  020500  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3078  020502  000744                      BR      2$              ;;GO DO THE LAST DIGIT
3079  020504  012605              6$:     MOV     (SP)+,R5        ;;RESTORE R5
3080  020506  012604                      MOV     (SP)+,R4        ;;RESTORE R4
3081  020510  012603                      MOV     (SP)+,R3        ;;RESTORE R3
3082  020512  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3083  020520  012616                      MOV     (SP)+,(SP)
3084  020522  000002                      RTI                     ;;RETURN
3085  020524     000              8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3086  020525     000                      .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3087  020526     000              $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3088  020527     000              $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
3089  020530  000000              $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3090                               .SBTTL  TYPE ROUTINE
3091
3092                               ;;******************************************************************
3093                               ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3094                               ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3095                               ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3096                               ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3097                               ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3098                               ;*
3099                               ;*CALL:
3100                               ;*1) USING A TRAP INSTRUCTION
3101                               ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3102                               ;*OR
```

```
                                        ;*      TYPE
                                        ;*      MESADR
                                        ;*
                                        
020532  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
020536  100002                          BPL     1$              ;;BR IF YES
020540  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
020542  000430                          BR      3$              ;;LEAVE
020544  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
020546  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
020552  122737  000001  001216          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
020560  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
020562  132737  000100  001217          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
020570  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
020572  010037  020602                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
020576  004737  021022                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
020602  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
020604  132737  000040  001217  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
020612  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
020614  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
020616  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
020620  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
020622  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
020624  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
020630  000002                          RTI                     ;;RETURN
020632  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
020636  001430                          BEQ     8$
020640  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
020644  001006                          BNE     5$
020646  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
020650  104401                          TYPE                    ;;TYPE A CR AND LF
020652  001173                          $CRLF
020654  105037  021010                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
020660  000755                          BR      2$              ;;GET NEXT CHARACTER
020662  004737  020744          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
020666  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
020672  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
020674  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                        ;;AND THE NULL CHAR.
020700  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
020704  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
020706  004737  020744                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
020712  105337  021010                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
020716  000770                          BR      7$              ;;LOOP

                        ;HORIZONTAL TAB PROCESSOR

020720  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
020724  004737  020744          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
020730  132737  000007  021010          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
020736  001372                          BNE     9$              ;;TAB STOP
020740  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
020742  000724                          BR      2$              ;;GET NEXT CHARACTER
020744  105777  160200          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
```

```
3157  020750  100375                        BPL     $TYPEC
3158  020752  116677  000002  160172        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3159  020760  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
3160  020766  001003                        BNE     1$              ;;BRANCH IF NO
3161  020770  105037  021010                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
3162  020774  000406                        BR      $TYPEX          ;;EXIT
3163  020776  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
3164  021004  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
3165  021006  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
3166  021010  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
3167  021012  000207          $TYPEX: RTS     PC
3168
3169                          .SBTTL  APT COMMUNICATIONS ROUTINE
3170
3171                          ;;**************************************************************
3172  021014  112737  000001  021260  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
3173  021022  112737  000001  021256  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
3174  021030  000403                        BR      $ATYC
3175  021032  112737  000001  021260  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
3176  021040                          $ATYC:
3177  021040  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3178  021042  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3179  021044  105737  021256                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
3180  021050  001450                        BEQ     5$              ;;IF NOT:  BR
3181  021052  122737  000001  001216        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
3182  021060  001031                        BNE     3$              ;;IF NOT:  BR
3183  021062  132737  000100  001217        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
3184  021070  001425                        BEQ     3$              ;;IF NOT:  BR
3185  021072  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
3186  021076  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
3187  021104  005737  001176          1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
3188  021110  001375                        BNE     1$              ;;IF NOT:  WAIT
3189  021112  010037  001212                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
3190  021116  105720                  2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
3191  021120  001376                        BNE     2$
3192  021122  163700  001212                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
3193  021126  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
3194  021130  010037  001214                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
3195  021134  012737  000004  001176        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
3196  021142  000413                        BR      5$
3197  021144  017637  000004  021170  3$:   MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
3198  021152  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
3199  021160  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
3200  021164  004737  020532                JSR     PC,$TYPE        ;;CALL TYPE MACRO
3201  021170  000000                  4$:   .WORD   0
3202  021172                          5$:
3203  021172  105737  021260          10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
3204  021176  001416                        BEQ     12$             ;;IF NOT:  BR
3205  021200  005737  001216                TST     $ENV            ;;RUNNING UNDER APT?
3206  021204  001413                        BEQ     12$             ;;IF NOT:  BR
3207  021206  005737  001176          11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
3208  021212  001375                        BNE     11$             ;;IF NOT:  WAIT
3209  021214  017637  000004  001200        MOV     @4(SP),$FATAL   ;;GET ERROR #
3210  021222  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
```

```
3211  021230  005237  001176          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
3212  021234  105037  021260  12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
3213  021240  105037  021257          CLRB    $LFLG           ;;CLEAR LOG FLAG
3214  021244  105037  021256          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
3215  021250  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
3216  021252  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
3217  021254  000207                  RTS     PC              ;;RETURN
3218  021256     000  $MFLG:  .BYTE   0               ;;MESSG. FLAG
3219  021257     000  $LFLG:  .BYTE   0               ;;LOG FLAG
3220  021260     000  $FFLG:  .BYTE   0               ;;FATAL FLAG
3221          021262          .EVEN
3222          000200  APTSIZE=200
3223          000001  APTENV=001
3224          000100  APTSPOOL=100
3225          000040  APTCSUP=040
3226                  .SBTTL  TRAP DECODER
3227
3228                  ;;*****************************************************************
3229                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3230                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3231                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3232                  ;*GO TO THAT ROUTINE.
3233
3234  021262  010046  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
3235  021264  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
3236  021270  005740                  TST     -(R0)           ;;BACKUP BY 2
3237  021272  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
3238  021274  006300                  ASL     R0              ;;POSITION FOR INDEXING
3239  021276  016000  021316          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
3240  021302  000200                  RTS     R0              ;;GO TO ROUTINE
3241
3242
3243                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3244
3245  021304  011646  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
3246  021306  016666  000004  000002  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
3247  021314  000002                  RTI                     ;;RESTORE THE PSW
3248
3249                  .SBTTL  TRAP TABLE
3250
3251                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3252                  ;*BY THE "TRAP" INSTRUCTION.
3253
3254                  ;       ROUTINE
3255                  ;       -------
3256  021316  021304  $TRPAD: .WORD   $TRAP2
3257  021320  020532          $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
3258  021322  020330          $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3259  021324  020304          $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3260  021326  020344          $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3261  021330  014120          $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
3262
3263  021332  017122          $GTSWR  ;;CALL=GTSWR     TRAP+6(104406)  GET SOFT-SWR SETTING
3264
```

```
                                        $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
                                        $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
                                        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
                                        $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY

                                        ;*
                                        ;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
                                        ;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
                                        ;*NEXT WE WILL INIT BOTH UPROCESSORS
                                        ;*THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
                                        ;*THE ORDER OF LOAD IS DETERMINED BY THE USER.
                                        ;*
                                        ;*      CALL=   JSR     R5,$LPAI
                                        ;*              .WORD   0               ;ADDR. OF DEVICE ADDRESS.
                                        ;* ROUTINES REQUIRED:    .LOADLP
                                        ;* PROGRAMS REQUIRED:    DRLPX2
                                        ;*

                                        ;*              ;RETURNS WITH $AERR=1 IF SLAVE
                                        ;*              ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
                                        ;*
021344                          $LPAI:
021344  013746  000004                  MOV     4,-(SP)

021350  000413                          BR      31$             ;FIELD DOES NOT HAVE A BUS SWITCH TO
                                                                ;WORRY ABOUT,SO WE WILL UNCONDITIONALLY
                                                                ;BRANCH ARROUD THE NEXT CODE THAT
                                                                ;WORKS BASED ON A BUS SWITCH.
                                                                ;CODE LEFT IN HERE FOR IN HOUSE
                                                                ;PERSONAL WHO MAY PATCH THIS BRANCH
                                                                ;INSTRUCTION TO A <NOP> OCTAL <240>
                                                                ;IN ORDER TO rUN PROGRAM WITH A SWITCH.

                                                                ;NOTE THIS "SWITCH" IS A PIECE OF INHOUSE
                                                                ;TEST EQUIPMENT ONLY IT CONNECTS
                                                                ;THE UNIBUS TO THE I/O BUS FOR
                                                                ;CERTAIN TESTING.
021352  012737  021376  000004          MOV     #30$,4
021360  005237  170000                  INC     170000
021364  104401  021372                  TYPE    ,65$            ;;TYPE ASCIZ STRING
021370  000401                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <7>##
021374                          64$:
021374  000401                          BR      31$
021376  022626                  30$:    CMP     (SP)+,(SP)+
021400  012637  000004          31$:    MOV     (SP)+,4         ;ALL THIS JUNK MUST BE REMOVED!!
021404  005037  022222                  CLR     $AERR
021410  004537  022224                  JSR     R5,$LOAD        ;LOAD MICRO-CODE.
021414  000000G                         .WORD   DRLPX2          ;FILE "DRLPX2.OBJ"

021416  052777  040000  160066          BIS     #BIT14,@KMADO   ;ISSUE KMC+DMC INIT.

021424                          1$:
                                                                ;"HANGS" HERE THEN KMC-11 ERROR.
```

```
3319  021424  010146                          MOV     R1,-(SP)
3320  021426  005001                          CLR     R1
3321  021430  005201                  2$:     INC     R1              ;STALL FOR DMC-UP
3322  021432  001376                          BNE     2$
3323  021434  012777  104000  160050          MOV     #BIT15!BIT11,@KMADO     ;SET RUN, AND ENABLE ARBITRATION.
3324  021442  105201                  25$:    INCB    R1
3325  021444  001376                          BNE     25$
3326
3327  021446  032777  000040  160036          BIT     #BIT5,@KMADO    ;SLAVE READY? (READING IPBM SR)
3328  021454  001401                          BEQ     3$
3329                                                                  ;FATAL LPA-11 ERROR SLAVE NOT READY.
3330  021456  104000                          ERROR
3331
3332  021460  012777  000004  160030  3$:     MOV     #4,@KMAD2       ;READ FAST PATH
3333  021466                          4$:
3334  021466  004537  023134                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
3335
3336  021472  104000                          ERROR                   ;/TIME-OUT ERROR
3337                                                                  ;/WE FAILED TO COMPLETE
3338                                                                  ;/CURRENT OPERATION.
3339                                                                  ;/CONTINUES IN THIS LOOP
3340                                                                  ;/WOULD MAKE US "HANG" HERE
3341
3342  021474  000774                          BR              4$
3343
3344                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3345  021476  122777  000377  160012          CMPB    #377,@KMAD2     ;WAIT TILL KMC DONE COMMAND.
3346  021504  001370                          BNE     4$
3347  021506  122777  000377  160006          CMPB    #377,@KMAD4     ;IF FAST PATH=377 THEN ERROR.
3348  021514  001001                          BNE     35$
3349  021516  104000                          ERROR                   ;IPBM ERROR (SLAVE SIDE)
3350                                                                  ;YOU MUST RUN IPBM DIAGNOSTIC.
3351
3352  021520  122777  000004  157774  35$:     CMPB    #4,@KMAD4       ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
3353  021526  001543                          BEQ     5$              ;YES-CONTINUE.
3354  021530  005227  177777                  INC     #-1
3355  021534  001140                          BNE     5$
3356  021536  005227  177777                  INC     #-1
3357  021542  001135                          BNE     5$
3358  021544  104401  021552                  TYPE    ,67$            ;;TYPE ASCIZ STRING
3359  021550  000440                          BR      66$             ;;GET OVER THE ASCIZ
3360                                  ;;67$:   .ASCIZ  <200>"W A R N I N G  THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
3361  021652                          66$:
3362  021652  104401  021660                  TYPE    ,69$            ;;TYPE ASCIZ STRING
3363  021656  000430                          BR      68$             ;;GET OVER THE ASCIZ
3364                                  ;;69$:   .ASCIZ  <200>"MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED."
3365  021740                          68$:
3366  021740  104401  021746                  TYPE    ,71$            ;;TYPE ASCIZ STRING
3367  021744  000434                          BR      70$             ;;GET OVER THE ASCIZ
3368                                  ;;71$:   .ASCIZ  <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
3369  022036                          70$:
3370
3371  022036  112737  177777  022170  5$:      MOVB    #0-1,11$        ;DAC CODE FOR SLAVE.
3372  022044  012501                          MOV     (5)+,R1         ;GET NEXT DEVICE ADDR.
```

```
3373  022046  021127  000000          6$:     CMP     (R1),#0         ;TERM REACHED?
3374  022052  001444                          BEQ     10$
3375  022054  105237  022170                  INCB    11$
3376  022060  113777  022170  157434          MOVB    11$,@KMAD4      ;FIFO DATA
3377  022066  004737  022172                  JSR     PC,20$          ;ISSUE SEND
3378  022072  112177  157424                  MOVB    (R1)+,@KMAD4    ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
3379  022076  004737  022172                  JSR     PC,20$          ;ISSUE SEND
3380  022102  112177  157414                  MOVB    (R1)+,@KMAD4    ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
3381  022106  004737  022172                  JSR     PC,20$
3382
3383  022112  032777  000002  157372  7$:     BIT     #BIT1,@KMAD0    ;WAIT FOR FIFO DATA
3384  022120  001374                          BNE     7$              ;=1 NO DATA. =0 DATA.
3385  022122  112777  000002  157366          MOVB    #2,@KMAD2       ;READ FIFO.
3386
3387  022130                          8$:
3388  022130  004537  023134                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
3389
3390  022134  104000                          ERROR                   ;/TIME-OUT ERROR
3391                                                                  ;/WE FAILED TO COMPLETE
3392                                                                  ;/CURRENT OPERATION.
3393                                                                  ;/CONTINUES IN THIS LOOP
3394                                                                  ;/WOULD MAKE US "HANG" HERE
3395
3396  022136  000774                          BR              8$
3397
3398                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3399  022140  122777  000377  157350          CMPB    #377,@KMAD2     ;WAIT FOR READ.
3400  022146  001370                          BNE     8$
3401  022150  105777  157346                  TSTB    @KMAD4          ;WAS A ZERO RETURNED?
3402  022154  001734                          BEQ     6$              ;YES GET NEXT ADDR.
3403                                                                  ;SLAVE WILL RETURN CODE 0 IF
3404  022156  005237  022222                  INC     $AERR           ;DEV PRESENT.   ELSE
3405                                                                  ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
3406  022162  005041                          CLR     -(1)            ;GET RID OF REFERENCE TO BAD ADDR.
3407  022164  012601                  10$:    MOV     (SP)+,R1
3408  022166  000205                          RTS     R5              ;RETURN ALL ADDR. CHECKED.
3409
3410  022170  000000                  11$:    .WORD   0               ;HOLDS DAC CODE PLUS OFFSET
3411                                                                  ;TO SLAVES ADDR. TABLE.
3412
3413  022172  112777  000003  157316  20$:    MOVB    #3,@KMAD2       ;ISSUE FIFO WRITE
3414  022200                          21$:
3415  022200  004537  023134                  JSR     R5, $TOuT       ;-TOUT-CHECK FOR TIMEOUT
3416
3417  022204  104000                          ERROR                   ;/TIME-OUT ERROR
3418                                                                  ;/WE FAILED TO COMPLETE
3419                                                                  ;/CURRENT OPERATION.
3420                                                                  ;/CONTINUES IN THIS LOOP
3421                                                                  ;/WOULD MAKE US "HANG" HERE
3422
3423  022206  000774                          BR              21$
3424
3425                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3426  022210  122777  000377  157300          CMPB    #377,@KMAD2     ;KMC CODE WILL RETURN A "377"
```

```
3427  022216  001370                        BNE     21$             ;WHEN DONE COMMAND.
3428  022220  000207                        RTS     PC
3429
3430  022222  000000              $AERR:    .WORD   0               ;=0 IF ADDR. LIST OK,=1 IF BAD.
3431
3432                                        ;*
3433                                        ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
3434                                        ;*      CALL =  JSR     R5,$LOAD
3435                                        ;*              .WORD   XX              ;ADDR. OF MICRO CODE.
3436                                        ;*              ;RETURNS HERE
3437                                        ;*      NOTE:   MICRO CODE FILE MUST END IN -1 DATA.
3438                                        ;*
3439
3440  022224  010446              $LOAD:    MOV     R4,-(SP)        ;SAVE R4.
3441  022226  010046                        MOV     R0,-(SP)        ;SAVE R0.
3442  022230  012500              1$:       MOV     (5)+,R0         ;GET PROG. ADDR.
3443  022232  005077  157254               CLR     @KMAD0          ;CLEAR CSR
3444  022236  005077  157260               CLR     @KMAD4          ;CLEAR CRAM ADDR.
3445  022242  052777  002000  157242  2$:  BIS     #2000,@KMAD0    ;SELECT CRAM.
3446  022250  012077  157252               MOV     (0)+,@KMAD6     ;WRITE DATA.
3447  022254  052777  020000  157230       BIS     #20000,@KMAD0   ;SET CRAM WRITE
3448  022262  005077  157224               CLR     @KMAD0          ;DISABLE CRAM.
3449  022266  005277  157230               INC     @KMAD4          ;UPDATE CRAM ADDR.
3450  022272  021027  177777               CMP     (0),#-1         ;ALL DONE?
3451  022276  001361                        BNE     2$              ;NO LOOP.
3452  022300  005077  157216               CLR     @KMAD4          ;CLEAR CRAM ADDR.
3453  022304  016500  177776               MOV     -2(5),R0        ;GET MICRO CODE ADDR.
3454
3455  022310  052777  002000  157174  3$:  BIS     #2000,@KMAD0    ;SELECT CRAM
3456  022316  022077  157204               CMP     (R0)+,@KMAD6    ;DATA OK?
3457  022322  001013                        BNE     5$              ;NO - REPORT AN ERROR.
3458  022324  021027  177777               CMP     (0),#-1         ;ALL DONE?
3459  022330  001405                        BEQ     4$              ;YES - EXIT
3460  022332  005077  157154               CLR     @KMAD0          ;NO - DESELECT CRAM.
3461  022336  005277  157160               INC     @KMAD4          ;UPDATE CRAM ADDR.
3462  022342  000762                        BR      3$
3463
3464  022344  012600              4$:       MOV     (SP)+,R0        ;RESTORE R0
3465  022346  012604                        MOV     (SP)+,R4        ;RESTORE R4
3466  022350  000205                        RTS     R5              ;EXIT
3467
3468  022352                      5$:                               ;COME HERE ON LOAD ERROR
3469  022352  005745                        TST     -(5)
3470  022354  105204                        INCB    R4              ;UPDATE ERROR COUNTER.
3471  022356  100324                        BPL     1$              ;IF NOT TOO MANY, TRY AGAIN.
3472  022360  000000                        HALT                    ;MICRO CODE LOAD ERROR.
3473                                                                ;KMC-11 FAULT. YOU COULD TRY
3474  022362  000722                        BR      1$              ;TO PRESS CONTINUE TO GIVE IT
3475                                                                ;ANOTHER CHANCE, BUT I DOUBT
3476                                                                ;THAT THAT WOULD WORK. SINCE I'VE
3477                                                                ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
3478                                                                ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
3479
3480
```

```
3481                                                    ;
3482                                                    ;*THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
3483                                                    ;*
3484                                                    ;*      CALL =  JSR     R5,$TLKW
3485                                                    ;*              .WORD   0               ;OFFSET OF DEVICE ADDR.
3486                                                    ;*              .WORD   0               ;DATA TO BE WRITTEN
3487                                                    ;*
3488  022364  010046                          $TLKW:   MOV     R0,-(SP)       ;SAVE R0
3489  022366  012500                                   MOV     (5)+,R0        ;GET DEVICE OFFSET
3490  022370  052700  000340                          BIS     #340,R0        ;ADD WRITE CODE.
3491  022374  004737  022646                          JSR     PC,$LPW        ;WAIT FOR FAST PATH READY
3492  022400  010037  022472                          MOV     R0,W1
3493  022404  010077  157112                          MOV     R0,@KMAD4
3494  022410  112777  000005  157100                  MOVB    #5,@KMAD2      ;ISSUE FAST PATH WRITE
3495  022416  004737  022646                          JSR     PC,$LPW        ;WAIT FOR RDY
3496  022422  011537  022474                          MOV     (5),W2
3497  022426  112577  157070                          MOVB    (5)+,@KMAD4    ;WRITE LOW BYTE DATA.
3498
3499  022432  112777  000005  157056                  MOVB    #5,@KMAD2      ;FP WRITE
3500  022440  004737  022646                          JSR     PC,$LPW
3501  022444  111537  022476                          MOVB    (5),W3
3502  022450  112577  157046                          MOVB    (5)+,@KMAD4    ;WRITE HIGH BYTE
3503  022454  112777  000005  157034                  MOVB    #5,@KMAD2
3504  022462  004737  022646                          JSR     PC,$LPW
3505  022466  012600                                   MOV     (SP)+,R0
3506  022470  000205                                   RTS     R5             ;EXIT DONE.
3507  022472  000000                          W1:      0
3508  022474  000000                          W2:      0
3509  022476  000000                          W3:      0
3510
3511                                                    ;*
3512                                                    ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
3513                                                    ;*
3514                                                    ;*      CALL =  JSR     R5,$TLKR
3515                                                    ;*              .WORD   0               ;OFFSET OF DEVICE
3516                                                    ;*              ;RETURNS HERE
3517                                                    ;*DATA IN WORD $DATR
3518                                                    ;*
3519
3520  022500  010046                          $TLKR:   MOV     R0,-(SP)       ;SAVE R0
3521  022502  012500                                   MOV     (5)+,R0        ;GET OFFSET
3522  022504  052700  000300                          BIS     #300,R0        ;ADD READ CODE
3523  022510  004737  022646                          JSR     PC,$LPW        ;WAIT TILL READY
3524  022514  110077  157002                          MOVB    R0,@KMAD4
3525  022520  112777  000005  156770                  MOVB    #5,@KMAD2      ;ISSUE WRITE FP
3526  022526  004737  022646                          JSR     PC,$LPW
3527  022532  010037  022642                          MOV     R0,RD1
3528  022536                                  1$:
3529  022536  004537  023134                          JSR     R5, $TOUt      ;-TOUT-CHECK FOR TIMEOUT
3530
3531  022542  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  022544  000774                    BR          1$
3538
3539                                                    ;/RETURNS HERE-FROM-TIMED OUT.
3540  022546  032777  000040  156736    BIT     #BIT5,@KMADO    ;FAST PATH GOT DATA?
3541  022554  001370                    BNE     1$
3542  022556  112777  000004  156732    MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
3543  022564  004737  022646            JSR     PC,$LPW
3544  022570  117737  156726  022644    MOVB    @KMAD4,$DATR    ;GET LOW BYTE
3545  022576                    2$:
3546  022576  004537  023134            JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
3547
3548  022602  104000                    ERROR                   ;/TIME-OUT ERROR
3549                                                    ;/WE FAILED TO COMPLETE
3550                                                    ;/CURRENT OPERATION.
3551                                                    ;/CONTINUES IN THIS LOOP
3552                                                    ;/WOULD MAKE US "HANG" HERE
3553
3554  022604  000774                    BR          2$
3555
3556                                                    ;/RETURNS HERE-FROM-TIMED OUT.
3557  022606  032777  000040  156676    BIT     #BIT5,@KMADO    ;FAST PATH READY?
3558  022614  001370                    BNE     2$
3559  022616  112777  000004  156672    MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
3560  022624  004737  022646            JSR     PC,$LPW
3561  022630  117737  156666  022645    MOVB    @KMAD4,$DATR+1  ;SAVE HIGH BYTE
3562  022636  012600                    MOV     (SP)+,R0
3563  022640  000205                    RTS     R5
3564  022642  000000            RD1:    0
3565  022644  000000            $DATR:  .WORD   0
3566                                    ;
3567                                    ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
3568                                    ;AS FAST PATH TO BE READ.
3569                                    ;
3570                                    ;       CALL =  JSR     PC,$LPW
3571                                    ;
3572                                    ;IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
3573                                    ;THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
3574                                    ;
3575
3576  022646  010146            $LPW:   MOV     R1,-(SP)        ;SAVE R1
3577  022650  005001                    CLR     R1
3578  022652  122777  000377  156636    CMPB    #377,@KMAD2     ;FINISHED INSTRUCTION?
3579  022660  001403                    BEQ     2$
3580  022662  005201                    INC     R1              ;TIME OUT?
3581  022664  001372                    BNE     1$
3582  022666  000411                    BR      10$
3583
3584  022670  032777  000020  156614    BIT     #BIT4,@KMADO    ;FAST PATH READ?
3585  022676  001403                    BEQ     3$
3586  022700  005201                    INC     R1              ;NO - TIME OUT?
3587  022702  001372                    BNE     2$
3588  022704  000402                    BR      10$             ;YES - REPORT AN ERROR
```

```
                                ;RESTORE R1
022706  012601          3$:     MOV     (SP)+,R1
022710  000207                  RTS     PC              ;EXIT
                        10$:
022712
022712  104401  022720          TYPE    ,65$            ;;TYPE ASCIZ STRING
022716  000407                  BR      64$             ;;GET OVER THE ASCIZ
                        ;;65$:  .ASCIZ  <200>#LPA-11 FAULT#
022736                  64$:
                                ;LPA-11 FAULT RUN LPA-11
022736  000000          11$:    HALT
022740  000776                  BR      11$             ;DIAGNOSTICS.
                                ;*
                                ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
                                ;*A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.
                                ;*
                                ;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED
                                ;* BEFORE. IF NOT WE HAVE TO INITIALIZE THE LPA WITH
                                ;* THAT ADDRESS.
                                ;* WHEN THE ADDR. IS KNOWN BY THE LPA, DO THE OUTPUT BY
                                ;* $TLKW
                                ;*
                                ;SAVE R0
022742  010046          $OUTLP: MOV     R0,-(SP)
022744  010146                  MOV     R1,-(SP)        ;SAVE R1
                                ;PROGRAM DEFINED LIST.
022746  012700  001540          MOV     #.DVLS,R0
022752  005001                  CLR     R1
                                ;TERMINATOR REACHED?
022754  005710          1$:     TST     (0)
022756  001421                  BEQ     10$             ;YES NEXT STEP.
                                ;MATCH WITH ADDR IN LIST?
022760  027520  000000          CMP     @(5),(0)+
022764  001402                  BEQ     2$
022766  005201                  INC     R1
022770  000771                  BR      1$
                                ;SAVE OFFSET, DEVICE KNOWN.
022772  010137  023010  2$:     MOV     R1,3$
022776  005725                  TST     (5)+
                                ;GET DATA TO BE WRITTEN
023000  013537  023012          MOV     @(5)+,4$
                                ;DO WRITE
023004  004537  022364          JSR     R5,$TLKW
                                ;DEVICE OFFSET
023010  000000          3$:     .WORD   0
                                ;DATA TO BE WRITTEN.
023012  000000          4$:     .WORD   0
023014  012601                  MOV     (SP)+,R1
023016  012600                  MOV     (SP)+,R0
023020  000205                  RTS     R5
                                ;SAVE ADDR.
023022  017520  000000  10$:    MOV     @(5),(0)+
023026  005010                  CLR     (0)
023030  004537  021344          JSR     R5,$LPAI
023034  001540                  .WORD   .DVLS
023036  000755                  BR      2$

                                ;*
```

```
                                        ;*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 $TORE READ DATA.

023040  010046          $INLP:  MOV     R0,-(SP)        ;SAVE R0
023042  010146                  MOV     R1,-(SP)        ;SAVE R1

023044  012700  001540          MOV     #.DVLS,R0       ;PROG DEFINED ADDR. LIST.
023050  005001                  CLR     R1
023052  005710          1$:     TST     (0)             ;EOL REACHED?
023054  001420                  BEQ     10$             ;YES - DEFINE NEW ADDR.

023056  027520  000000          CMP     @(5),(0)+       ;ADDR. MATCH?
023062  001402                  BEQ     2$
023064  005201                  INC     R1
023066  000771                  BR      1$

023070  010137  023102  2$:     MOV     R1,3$           ;SAVE LIST OFFSET
023074  005725                  TST     (5)+
023076  004537  022500          JSR     R5,$TLKR        ;GO READ DEVICE
        023102          $OFS=.
023102  000000          3$:     .WORD   0               ;OFFSET OF DEVICE

023104  013735  022644          MOV     $DATR,@(5)+     ;STORE DATA.
023110  012601                  MOV     (SP)+,R1        ;RESTORE R1
023112  012600                  MOV     (SP)+,R0        ;RESTORE R2
023114  000205                  RTS     R5              ;EXIT

023116  017520  000000  10$:     MOV     @(5),(0)+
023122  005010                  CLR     (0)
023124  004537  021344          JSR     R5,$LPAI
023130  001540                  .WORD   .DVLS
023132  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
                                ;*

023134  020537  023170  $TOUT:  CMP     R5,$SAD         ;SAME ADDR?
023140  001405                  BEQ     1$
```

```
3697  023142  010537  023170            MOV     R5,$SAD         ;NO-SAVE THIS ADDR.
3698  023146  005037  023172            CLR     $CNT            ;CLR CNT AT ADDR.
3699  023152  000403                    BR      2$
3700  023154  005237  023172    1$:     INC     $CNT            ;OVERFLOW?
3701  023160  100402                    BMI     3$              ;YES-ERROR RETURN
3702  023162  062705  000004    2$:     ADD     #4,R5           ;NO-NON ERROR RETURN
3703  023166  000205            3$:     RTS     R5              ;RETURN.
3704
3705  023170  000000            $SAD:   .WORD   0               ;CONTAINS LOOP ADDR.
3706  023172  000000            $CNT:   .WORD   0               ;# OF TIMES AT ADDR.
3707
3708                                    ;*
3709                                    ;* THIS ROUTINE REPLACES WHAT THE USER WOULD ORDINARILY
3710                                    ;*USE FOR A RESET.  FIRST,WE DO A RESET INSTRUCTION.
3711                                    ;*THEN WE CLR ".DVLST" WHICH FORCES US TO RESET BOTH THE
3712                                    ;*KMC AND DMC AS SOON AS A DEVICE IS REFERENCED.
3713                                    ;*
3714                                    ;*      CALL=JSR        PC,$RESET       ;REPLACES "RESET INSTRUCTION
3715                                    ;*              ;RETURNS HERE.
3716                                    ;*
3717  023174  000005            $RESET: RESET                   ;RESET THE WORLD.
3718
3719                            ;*      MOV     @2$,1$  ;/READ DEVICE REG 2$,PUT DATA IN 1$.
3720  023206  005737  022222            TST     $AERR           ;IF NO ERROR,LOOP
3721  023212  001004                    BNE     10$             ;THERE WAS AN ERROR.
3722  023214  062737  000002  023230    ADD     #2,2$           ;UPDATE DEVICE ADDR.
3723                                                            ;YOU SEE ,WE HAVE TO PROTECT OUR SELF!
3724                                                            ;IF 2$ CONTAINED A VALID ADDR,WE
3725                                                            ;MUST KEEP TRYING UNTIL WE GENERATE
3726                                                            ;AN INVALID ADDR.
3727  023222  000764                    BR      $RESET
3728  023224                    10$:
3729  023224  000207                    RTS     PC
3730  023226  000000            1$:     .WORD   0               ;JUNK LOC.
3731  023230  160000            2$:     .WORD   160000          ;DUMB ADDR. FORCES INIT OF DMC/KMC.
3732
3733
3734                                    ;
3735                                    ;SDELAY- ROUTINE TO GIVE A MINOR DELAY.
3736                                    ;        IS NOT TIME DEPENDENT CODE SENCE
3737                                    ;        NOT USED TO GET SPECIFIC TIME BUT
3738                                    ;        JUST A LITTLE DELAY.
3739                                    ;
3740                                    ;       THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
3741                                    ;       THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
3742                                    ;
3743                                    ;
3744                                    ;       CALL=   JSR PC,  SDELAY
3745                                    ;
3746  023232                    SDELAY:
3747  023232  005737  023314            TST     RTCCSR          ;CLOCK PRESENT?
3748  023236  100016                    BPL     10$
3749  023240  012737  000002  023304    MOV     #2,TIME
3750  023246  052777  000115  000040    BIS     #115,@RTCCSR    ;START CLOCK
```

```
3751  023254  005037  177776          CLR     PS
3752  023260  005737  023304  1$:     TST     TIME
3753  023264  001375                  BNE     1$
3754  023266  005077  000022          CLR     @RTCCSR         ;STOP CLOCK
3755
3756  023272  000207                  RTS PC
3757  023274  105237  023304  10$:    INCB    TIME
3758  023300  001375                  BNE     10$
3759  023302  000207                  RTS     PC
3760
3761  023304  000000          TIME:   .WORD   0
3762
3763  023306  005337  023304  CLKINT: DEC     TIME
3764  023312  000002                  RTI
3765  023314  000000          RTCCSR: .WORD   0               ;CLOCK CSR IF USED.
3766                                  ;*
3767                                  ;*THIS MACRO ALLOWS THE OPERATOR TO TALK TO
3768                                  ;*ANY DEVICE ON THE I/O BUS
3769                                  ;*USER MUST START AT THIS ADDR.
3770                                  ;*HE MUST SAY EITHER "E" FOR EXAMINE, OR "D" FOR DEPOSIT.
3771                                  ;*"E" IS DEFAULT.
3772                                  ;*NEXT, HE MUST SUPPLY AN ADDR.
3773                                  ;*NOTE IF ADDR. IS NOT FOUND ON I/O BUS, A HALT
3774                                  ;*WILL OCCUR.
3775
3776
3777  023316                  $UTK:
3778  023316  005037  001540          CLR     .DVLS
3779  023322                  21$:
3780  023322  104401  023330          TYPE    ,65$            ;;TYPE ASCIZ STRING
3781  023326  000405                  BR      64$             ;;GET OVER THE ASCIZ
3782                          ;;65$:  .ASCIZ  <200>*E OR D?*
3783  023342                  64$:
3784  023342  105777  155576  1$:     TSTB    @$TKS
3785  023346  100375                  BPL     1$
3786  023350  117737  155572  023472  MOVB    @$TKB,20$       ;GET INPUT
3787  023356  104401  023472          TYPE,   20$             ;ECHO, NEXT MESSAGE.
3788  023362  142737  000240  023472  BICB    #240,20$        ;STRIP PARITY, LC
3789  023370  104412                  RDOCT                   ;GET ADDR.
3790  023372  012637  023470          MOV     (SP)+,14$
3791  023376  123727  023472  000104  CMPB    20$,#'D         ;DEPOSIT?
3792  023404  001411                  BEQ     10$
3793
3794  023406  004537  023040          JSR     R5,$INLP        ;GET DATA
3795  023412  023470          2$:     .WORD   14$
3796  023414  023426                  .WORD   5$
3797
3798  023416  013746  023426          MOV     5$,-(SP)        ;;SAVE 5$ FOR TYPEOUT
3799  023422  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3800  023424  000736                  BR      21$             ;LOOP.
3801  023426  000000          5$:     .WORD   0
3802
3803  023430                  10$:
3804  023430  104401  023436          TYPE    .67$            ;;TYPE ASCIZ STRING
```

```
3805  023434  000404                          BR      66$             ;;GET OVER THE ASCIZ
3806                                  ;;67$:  .ASCIZ  <200>*DATA= *
3807  023446                          66$:
3808  023446  104412                          RDOCT
3809  023450  012637  023466                  MOV     (SP)+,13$
3810
3811  023454  004537  022742          11$:    JSR     R5,$OUTLP       ;OUTPUT ROUTINE.
3812  023460  023470                  12$:    .WORD   14$             ;DEVICE ADDR.
3813  023462  023466                          .WORD   13$             ;DATA
3814  023464  000716                          BR      21$
3815
3816  023466  000000                  13$:    .WORD   0
3817  023470  000000                  14$:    .WORD   0
3818  023472  100001  042504  044526  20$:    .ASCIZ  <1><200>*DEVICE ADDR= *
3819  023500  042503  040440  042104
3820  023506  036522  000040
3821                                          .EVEN
3822
3823
3824
3825                                  ;
3826                                  ;THIS ROUTINE LOOKS THROUGH CURENT .DVLS FOR A/D ADDR.
3827                                  ;IF UNFOUND,GENERATES IT. THIS ROUTINE'S WHOLE PURPOSE IS
3828                                  ;TO SET UP THE USER PROGRAM TO LINK TO FILE "DRLPX2" FOR
3829                                  ;SAMPLE TAKEING PURPOSES.
3830                                  ;       TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
3831                                  ;       A/D CSR IN BSEL 4,AND 5.
3832                                  ;       (2) HE MUST CALL THIS ROUTINE:
3833                                  ;               JSR     R5,$PUTS        ;CALL SET UP ROUTINE.
3834                                  ;               .WORD   ADCSR           ;ADDR. OF A/D CSR.
3835                                  ;               ;RETURNS HERE ;KMC BSEL 3,6,7 PERMINENTLY SET UP
3836                                  ;                               ;(UNTILL ONE DOES A RESET)
3837                                  ;
3838                                  ;       (3)THE USER MUST PUT CODE 006 INTO KMC REG 2 TO
3839                                  ;          START CONVERSION  CAUTION*DO WITH MOVB INSTR.!
3840                                  ;       (4)MONITOR KMC REG 2 FOR CODE 377 (DRLPX2 IS DONE)
3841                                  ;       (5)READ KMC REG 4,5 FOR A/D RESULT.
3842                                  ;       (6)  TO TAKE MORE SAMPLES,SIMPLY PUT A/D CSR INTO
3843                                  ;          BSEL 4,5 AND CODE 6 INTO BSEL 2.
3844                                  ;
3845  023512  012537  023522  $PUTS:  MOV     (5)+,1$                 ;GET ADDR OF ADDR. OF A/D
3846  023516  004537  023040          JSR     R5,$INLP
3847  023522  000000          1$:     .WORD   0
3848  023524  023620                  .WORD   10$
3849  023526  113777  023102  155772  MOVB    $OFS,@KMAD6
3850  023534  113777  023102  155766  MOVB    $OFS,@KMAD7
3851  023542  013737  023522  023562  MOV     1$,2$
3852  023550  062737  000002  023562  ADD     #2,2$
3853  023556  004537  023040          JSR     R5,$INLP
3854  023562  000000          2$:     .WORD   0
3855  023564  023620                  .WORD   10$
3856  023566  113777  023102  155724  MOVB    $OFS,@KMAD3
3857  023574  152777  000340  155724  BISB    #340,@KMAD6
3858  023602  152777  000300  155720  BISB    #300,@KMAD7
```

```
 3859  023610  152777  000300  155702          BISB    #300,@KMAD3
 3860  023616  000205                          RTS     R5
 3861  023620  000000                  10$:    .WORD   0
 3862
 3863          000001                          .END
```

```
ABASE = 170400            173#    284     325
ACDW1 = 000000            284     327
ACDW2 = 000000            284     328
ACPUOP= 000000            284     299
ADBRL   001602            489#
ADCS    001610            495#    667     692    1896   1898   1910   1912   1924   1926   1938   1940   1952   1954
                         1965    1967    1979   1981   1993   1995   2012   2016   2022   2030   2037   2050   2052
                         2055    2058    2079   2085   2088   2093   2105   2111   2114   2120   2133   2136   2148
                         2152    2163    2166   2171   2181   2184   2211   2597   2599
ADDBR   001612            496#    674     2007   2026   2063   2082   2189   2616
ADDW0 = 000000            284     329
ADDW1 = 000000            284     330
ADDW10= 000000            284     339
ADDW11= 000000            284     340
ADDW12= 000000            284     341
ADDW13= 000000            284     342
ADDW14= 000000            284     343
ADDW15= 000000            284     344
ADDW2 = 000000            284     331
ADDW3 = 000000            284     332
ADDW4 = 000000            284     333
ADDW5 = 000000            284     334
ADDW6 = 000000            284     335
ADDW7 = 000000            284     336
ADDW8 = 000000            284     337
ADDW9 = 000000            284     338
ADEVCT= 000000            284     290
ADEVM = 000000            284     326
ADINT   001630            508#    669
ADINT1  001632            509#
AENV  = 000000            284     295
AENVM = 000000            284     296
AFATAL= 000000            284     287
AMADR1= 000000            284     312
AMADR2= 000000            284     316
AMADR3= 000000            284     319
AMADR4= 000000            284     322
AMAMS1= 000000            284     306
AMAMS2= 000000            284     314
AMAMS3= 000000            284     317
AMAMS4= 000000            284     320
AMSGAD= 000000            284     292
AMSGLG= 000000            284     293
AMSGTY= 000000            284     286
AMTYP1= 000000            284     307
AMTYP2= 000000            284     315
AMTYP3= 000000            284     318
AMTYP4= 000000            284     321
APASS = 000000            284     289
APRIOR= 000200            175#    284
APTCSU= 000040           3120    3225#
APTENV= 000001           2713    3113    3181    3223#
APTSIZ= 000200            589    3222#
APTSPO= 000100           3115    3183    3224#
```

```
ARBADD  001644  517*  613*  668  672  2248*  2252*  2614
ARBVCT  001646  518*  614*  671  2249*  2253*
ASWREG= 000000  284  297
ATESTN= 000000  284  288
AUNIT = 000000  284  291
AUSWR = 000000  284  298
AVECT1= 000340  174*  284  323  481  462
AVECT2= 000000  284  324
BEGIN   001672  198  530*
BEGIN1  001676  199  532*  2999
BIT0  = 000001  158*  887  1293  1726  2009  2038  2160
BIT00 = 000001  148*  158
BIT01 = 000002  147*  157
BIT02 = 000004  146*  156
BIT03 = 000010  145*  155
BIT04 = 000020  144*  154
BIT05 = 000040  143*  153
BIT06 = 000100  142*  152
BIT07 = 000200  141*  151
BIT08 = 000400  140*  150  2652
BIT09 = 001000  139*  149  2660  2724
BIT1  = 000002  157*  900  1068  1071  1242  1311  1353  1382  3383
BIT10 = 002000  138*  1516  1519  1805  1846  1962  2063  2130
BIT11 = 004000  137*  1047  1052  1074  1079  1114  1119  1153  1158  1183  1188  1219  1224
                1529  1532  1846  1976  2130  2667  3323
BIT12 = 010000  136*  1542  1545  1801  1810
BIT13 = 020000  135*  1990  2130  2708
BIT14 = 040000  134*  965  1258  2638  3315
BIT15 = 100000  133*  990  1274  3323
BIT2  = 000004  156*  913  1242  1332  1353  1451  1454  1747  1748  1761  1830
BIT3  = 000010  155*  926  1242  1368  1382  1474  1477  1775  1776  1789  1830
BIT4  = 000020  154*  887  900  913  926  939  952  965  978  990  1005  1071  1311
                1332  1353  1368  1382  1893  2079  2085  2145  3584
BIT5  = 000040  153*  1907  2107  2111  2115  3327  3540  3557
BIT6  = 000100  152*  939  1258  1488  1491  1846  1921  2145
BIT7  = 000200  151*  978  1071  1274  1311  1332  1353  1368  1382  1439  1454  1463  1477
                1491  1505  1519  1532  1736  1747  1761  1775  1789  1833  1849  2012  2107
BIT8  = 000400  150*  952  1242  1935  2130
BIT9  = 001000  149*  1502  1505  1846  1949  2130
BPTVEC= 000014  165*
BRLEV1  001664  525*
BRLEV2  001666  526*
BUFADR  013566  2221*  2240*  2618
BUFNUM  013546  2200  2212  2231*
BYPASS  013572  707  2246*
BYPAS1  013650  2251  2256*  2292
CKBRL   001604  490*
CKDLY   014616  2409*  2427  2449*
CKSWR = 104407  2636  2637  2700  2723  3265*
CLKINT  023306  3763*
CNTDEV  014612  2407*  2424  2447*
COUNT   001654  521*  1110*  1122*  1149*  1161*  1179*  1191*  1215*  1227*
CR    = 000015  73*  3159  3169
CRLF  = 000200  74*  657  3130  3169
```

```
CSB     001616          499*    717     719     730     732     743     745     756     758     771     773     789     792
                        803     816     830     843     858     875    1025    1068    1097    1137    1175    1211    1293
                       2309    2406    2444    2602
CSC     001626          506*    699     805     818     832     845     860     878    1032    1105    1144    1195    1231
                       1321    1342    1407    1430    2198    2438    2605
CSR     001614          498*    890     892     903     905     916     918     929     931     942     944     955     957
                        968     970     981     983     993     995    1005    1010    1023    1028    1039    1047    1050
                       1052    1055    1065    1071    1074    1077    1079    1082    1084    1095    1101    1114    1117
                       1119    1122    1135    1140    1153    1156    1158    1161    1173    1179    1183    1186    1188
                       1191    1209    1215    1219    1222    1224    1227    1245    1249    1261    1265    1277    1281
                       1296    1300    1315    1336    1357    1372    1386    1400    1423    2107    2306    2312    2316
                       2404    2431    2434    2442    2600    2604
DDI$P = 177570          80*     262     577
DELAY   001656          522*   2312*   2319*
DH1     015414          367    2523*
DH10    015647          409    2553*
DH11    015705          415    2559*
DH12    015745          421    2565*
DH13    016006          427    2571*
DH14    016047          433    2577*
DH15    016066          439    2580*
DH16    016116          445    2584*
DH17    016157          451    2590*
DH2     015454          373    2529*
DH3     015472          379    2532*
DH4     015532          385    2538*
DH5     015572          391    2544*
DH6     015607          397     403    2547*
DISPLA  001142          262*    577*    585*   2682*   2703*
DISPRE  000174          195*    585
DRLPX2= ******  G        14*   3313
DSWR  = 177570           79*     261     576
DT1     016220          368    2597*
DT11    016304          416    2607*
DT12    016316          422    2609*
DT13    016330          428    2611*
DT14    016342          434    2613*
DT15    016350          440    2614*
DT16    016360          446    2616*
DT17    016372          452    2618*
DT2     016232          374    2599*
DT3     016240          380    2600*
DT4     016252          386    2602*
DT5     016264          392    2604*
DT6     016272          398     404     410    2605*
EMTVEC= 000030          168*    560*    561*
EM1     014620          366    2450*
EM10    015122          408    2486*
EM11    015155          414    2492*
EM12    015211          420    2497*
EM13    015235          426    2501*
EM14    015261          432    2505*
EM15    015307          438    2509*
EM16    015330          444    2513*
```

```
EM17    005362            450    2518#
EM2     014660            372    2456#
EM3     014707            378    2460#
EM4     014746            384    2466#
EM5     015005            390    2472#
EM6     015033            396    2476#
EM7     015073            402    2482#
ERRVEC= 000004            161#    574     575*    586*   2643    2644*   2646*   2649*
GNS   = ****** U          194     635     656     663    3257    3258    3259    3260    3261    3263    3265    3266    3267
                         3268    3306    3360    3364    3368    3596    3782    3806
GTSWR = 104406            651    3263#
HT    = 000011             71#   3128    3169
IOTVEC= 000020            166#    558*    559*
KMAD0   001512            464#    540    3315*   3323*   3327    3383    3443*   3445*   3447*   3448*   3455*   3460*   3540
                         3557    3584
KMAD1   001514            467#    541
KMAD2   001516            469#   3332*   3345    3385*   3399    3413*   3426    3494*   3499*   3503*   3525*   3542*   3559*
                         3578
KMAD3   001520            471#   3856*   3859*
KMAD4   001522            473#   3347    3352    3376*   3378*   3380*   3401    3444*   3449*   3452*   3461*   3493*   3497*
                         3502*   3524*   3544    3561
KMAD5   001524            475#
KMAD6   001526            477#   3446*   3456    3849*   3857*
KMAD7   001530            479#    545    3850*   3858*
KWIV    001634            511#
KWIVS   001636            512#
LF    = 000012             72#   3163    3169
LPADH   001524            474#
LPADL   001522            472#
LPCI    001512            463#
LPCO    001516            468#
LPMR    001514            466#
LPMS1   001526            476#
LPMS2   001530            478#
LPSO    001520            470#
MINCNT  014614           2408#   2419    2421    2448#
NBEXT   001652            520#    615*   2246    2250*   2254*
NMBEXT  001650            519#    610*    615     658    2254
PC    =%000007             92#    789*    875*   1246*   1262*   1278*   1297*   1312*   1333*   1354*   1369*   1383*   1397*
                         1420*   1440*   1834*   1850*   1866*   1882*   2114*   2133*   2149*   2168*   2186*   2197*   2273*
                         2276*   2286*   2291    2410*   2411*   2413*   2444*   2710*   2716*   2790*   2951*   2998    3118*
                         3137*   3144*   3151*   3165*   3167*   3200*   3217*   3377*   3379*   3381*   3428*   3491*   3495*
                         3500*   3504*   3523*   3526*   3543*   3560*   3591*   3729*   3756*   3759*
PIRQ  = 177772             78#
PIRQVE= 000240            172#
PR0   = 000000             95#
PR1   = 000040             96#
PR2   = 000100             97#
PR3   = 000140             98#
PR4   = 000200             99#
PR5   = 000240            100#
PR6   = 000300            101#
PR7   = 000340            102#
PS    = 177776             75#     76    3751*
```

```
                    2152*  2163*  2166*  2171*  2181*  2184*  2189*  2208*  2219*  2306*  2309*  2312*  2316*
                    2340   2342*  2344*  2351*  2355*  2370   2376*  2404*  2406*  2407   2408   2422*  2423
                    2425*  2431*  2434*  2438*  2442*  2444*  2973   2988*  3048   3054*  3056*  3058*  3059*
                    3060*  3061   3079*  3312*  3334*  3388*  3408*  3415*  3466*  3506*  3529*  3546*  3563*
                    3630*  3635*  3638*  3672*  3679*  3683*  3695   3697   3702*  3703*  3720*  3794*  3811*
                    3846*  3853*  3860*
R6      =%000006      89*    552*   553*   554
R7      =%000007      90*
SDELAY  023232      3746*
SP      =%000006      91*    538*   539*   548    549    556*   574*   582*   586    593*   658*   2280*  2336*
                    2337*  2338*  2339*  2340*  2341*  2342   2345*  2358   2360*  2362   2372   2374   2376
                    2377   2378   2379   2380   2382*  2383*  2643*  2646   2648   2649   2678   2679   2683*
                    2705   2726*  2729*  2750*  2751*  2752   2759*  2762*  2763*  2767*  2768*  2772   2775*
                    2779   2781   2783   2784*  2791   2793   2795*  2796   2798*  2799*  2800*  2801*  2802*
                    2817*  2818*  2821*  2822*  2823   2827*  2828*  2829   2832   2834   2836*  2845*  2850
                    2861   2862*  2863*  2864*  2885*  2886*  2887*  2888*  2889*  2891   2894*  2902*  2903
                    2905   2906*  2908   2909   2910   2923*  2928*  2949   2953*  2968*  2969*  2970*  2971*
                    2972*  2973*  2974*  2975   2983*  2987   2988   2989   2990   2991   2992   2993   2998*
                    3038*  3039   3040   3041*  3046*  3047*  3048*  3054   3079   3080   3081   3082*  3083*
                    3111*  3112   3122*  3124   3125   3126*  3128   3130   3132   3138   3140*  3142*  3150*
                    3154   3158   3159   3163   3177*  3178*  3185   3186*  3197   3198*  3199*  3209   3210*
                    3215   3216   3234*  3235   3245*  3246*  3287*  3309   3310   3319*  3407   3440*  3441*
                    3464   3465   3488*  3505   3520*  3562   3576*  3590   3615*  3616*  3633   3634   3657*
                    3658*  3677   3678   3790   3798*  3809
STACK = 001100      66*    556    593
STKLMT= 177774      77*
SWITCH  001662     524*
SWR     001140     261*    554    576*   578    584*   591*   649    1801   2063   2638   2652   2654   2660
                   2667   2708   2720   2724   2746   2783*  2974   2987*
SWREG   000176     196*    584    649    2746   2759
SW0   = 000001     130*
SW00  = 000001     120*   130
SW01  = 000002     119*   129
SW02  = 000004     118*   128
SW03  = 000010     117*   127
SW04  = 000020     116*   126
SW05  = 000040     115*   125
SW06  = 000100     114*   124
SW07  = 000200     113*   123
SW08  = 000400     112*   122
SW09  = 001000     111*   121
SW1   = 000002     129*
SW10  = 002000     110*
SW11  = 004000     109*
SW12  = 010000     108*
SW13  = 020000     107*
SW14  = 040000     106*
SW15  = 100000     105*
SW2   = 000004     128*
SW3   = 000010     127*
SW4   = 000020     126*
SW5   = 000040     125*
SW6   = 000100     124*
SW7   = 000200     123*
```

```
SW8    = 000400            122#
SW9    = 001000            121#
TBITVE= 000014             163#
TEMP    001660             523#   1006*   1013*   1028*   1029    1055*   1804*   1819*
TIME    023304            3749*   3752    3757*   3761#   3763*
TKVEC  = 000060             170#
TPVEC  = 000064             171#
TRAPVE= 000034              169#    562*    563*
TRTVEC= 000014              164#
TST1    002736              689#
TST10   003364              793     799#
TST100  011326             1802    1818    1828#
TST101  011410             1838    1844#
TST102  011472             1854    1860#
TST103  011554             1870    1876#
TST104  011636             1886    1892#
TST105  011700             1899    1905#
TST106  011742             1913    1919#
TST107  012004             1927    1933#
TST11   003426              806     812#
TST110  012046             1941    1947#
TST111  012110             1955    1960#
TST112  012152             1968    1974#
TST113  012214             1982    1988#
TST114  012256             1996    2004#
TST115  012450             2039    2046#
TST116  012572             2064    2066    2074#
TST117  012720             2094    2100#
TST12   003470              819     826#
TST120  013056             2121    2127#
TST121  013134             2137    2143#
TST122  013206             2152    2158#
TST123  013276             2171    2176#
TST124  013374             2191    2196#
TST125  013570             2227    2245#
TST13   003532              833     839#
TST14   003574              846     852#
TST15   003650              870#
TST16   003722              879     886#
TST17   003764              893     899#
TST2    003020              704     713#
TST20   004026              906     912#
TST21   004070              919     925#
TST22   004132              932     938#
TST23   004174              945     951#
TST24   004236              958     964#
TST25   004300              971     977#
TST26   004342              984     989#
TST27   004404              996    1001#
TST3    003062              720     726#
TST30   004464             1018#
TST31   004704             1043    1061#
TST32   005062             1085    1090#
TST33   005266             1130#
```

```
TST34   005466          1168#
TST35   005662          1196    1204#
TST36   006056          1232    1240#
TST37   006140          1250    1256#
TST4    003124           733     739#
TST40   006222          1266    1272#
TST41   006304          1282    1288#
TST42   006404          1301    1308#
TST43   006506          1322    1329#
TST44   006610          1343    1350#
TST45   006662          1358    1365#
TST46   006734          1373    1379#
TST47   007006          1387    1393#
TST5    003166           746     752#
TST50   007110          1408    1416#
TST51   007212          1431    1438#
TST52   007250          1444    1450#
TST53   007366          1467    1473#
TST54   007436          1481    1487#
TST55   007506          1495    1501#
TST56   007556          1509    1515#
TST57   007626          1523    1528#
TST6    003230           759     765#
TST60   007676          1536    1541#
TST61   007762          1557#
TST62   010024          1564    1570#
TST63   010066          1577    1583#
TST64   010130          1590    1596#
TST65   010172          1603    1609#
TST66   010254          1628#
TST67   010316          1635    1641#
TST7    003304           783#
TST70   010360          1648    1654#
TST71   010422          1661    1667#
TST72   010464          1674    1680#
TST73   010546          1699#
TST74   010656          1716    1722#
TST75   010752          1733    1744#
TST76   011064          1758    1765    1772#
TST77   011206          1786    1793    1800#
TYPDS = 104405          2281    3261#
TYPE  = 104401           633     643     661    2279    2282    2381    2711    2757    2758    2761    2774    2785    2804
                        2853    2856    2860    2922    2939    2941    2944    2946    2950    2957    2996    3073    3133
                        3257#   3304    3358    3362    3366    3594    3780    3787    3804
TYPOC = 104402          2760    2930    2954    3258#   3799
TYPON = 104404          3260#
TYPOS = 104403           659    3259#
UPCNT   014002          1312    1333    1354    1369    1383    1397    1420    2114    2303#
UPCNTA  014062          2313#   2320
UPCNTB  014116          2317    2321#
VCBRL   001606           491#
VCIV    001640           514#
VCIVS   001642           515#
VCSTAT  001620           501#   1443    1454    1457    1463    1466    1477    1480    1491    1494    1505    1508    1519
```

```
                    1522    1532    1535    1545    1548    1554    1729    1732    1739    1751    1757    1764    1779
                    1785    1792    1808    1810    1813    1817    1833    1837    1849    1853    2607    2613
VCXREG  001622       502#    1561    1563    1574    1576    1587    1589    1600    1602    1615    1617    1703    1715
                    1865    1869    2609
VCYREG  001624       503#    1632    1634    1645    1647    1658    1660    1671    1673    1686    1688    1706    1708
                    1782    1881    1885    2611
VECTOR  001532       481#
VECTPS  001534       482#
VERSN   001536       484#
W1      022472      3492*   3507#
W2      022474      3496*   3508#
W3      022476      3501*   3509#
$AERR   022222        600     697    3311*   3404*   3430#   3720
$APTHD  001000        221     227#
$ASTAT= ****** U     3203    3218
$ATYC   021040       3174    3176#
$ATY1   021014       3172#
$ATY3   021022       3118    3173#
$ATY4   021032       2716    3175#
$AUTOB  001134        258#    653*    2754    2875
$BASE   001252        325#    595      613    2252
$BDADR  001122        253#
$BDDAT  001126        255#    692*    696     699     701*    719     732     745     758     773     792     805     818
                      832     845     860     878     892     905     918     931     944     957     970     983     995
                     1010    1032    1039    1084    1105    1144    1195    1231    1249    1265    1281    1296*   1300
                     1315    1321    1336    1342    1357    1372    1386    1400    1407    1423    1430    1443    1457
                     1466    1480    1494    1508    1522    1535    1548    1563    1576    1589    1602    1617    1634
                     1647    1660    1673    1688    1708    1715    1732    1739    1757    1764    1785    1792    1837
                     1853    1869    1885    1898    1912    1926    1940    1954    1967    1981    1995    2022    2030
                     2037    2038    2063    2065    2093    2120    2136    2152    2171    2189    2190    2219    2414*
                     2416    2419    2597    2600    2602    2605    2607    2609    2611    2614    2616    2618
$CDW1   001256        327#
$CDW2   001260        328#
$CHARC  021010       3135*   3145*   3152    3161*   3166#
$CKSWR  017052       2746#   3265
$CMTAG  001100        241#    551     552     560     566     567     568
$CM1    = 000002      273#    274#    275#
$CM2    = 000004      273#    274#    275#
$CM3    = 000002      271#    273
$CNT    023172       3698*   3700*   3706#
$CNTLG  017577       2757    2870#
$CNTLU  017572       2774    2869#
$CPUOP  001224        299#
$CRLF   001173        278#    2711    2736    2785    2869    2922    2941    2946    2950    3134    3169
$DAIR   022644       3544#   3561*   3565#   3676
$DBLK   014334       2347    2381    2389#
$DDW0   001262        329#
$DDW1   001264        330#
$DDW10  001306        339#
$DDW11  001310        340#
$DDW12  001312        341#
$DDW13  001314        342#
$DDW14  001316        343#
$DDW15  001320        344#
```

```
$DDW2   001266  331#
$DDW3   001270  332#
$DDW4   001272  333#
$DDW5   001274  334#
$DDW6   001276  335#
$DDW7   001300  336#
$DDW8   001302  337#
$DDW9   001304  338#
$DEVCT  001206  290#
$DEVM   001254  326#
$DOAGN  013756  2275  2284  2290#
$DTBL   014324  2350  2385#
$ENDAD  013746  207  641  2286#  2731
$ENDCT  013714  566  2277#
$ENDMG  013765  2279  2294#
$ENULL  013762  2282  2293#
$ENV    001216  295#  647  2713  3113  3181  3205
$ENVM   001217  296#  589  3115  3120  3183
$EOP    013660  2255  2267#
$EOPCT  013706  566*  2274#  2278
$ERFLG  001103  244#  2626  2656  2658  2664*  2686  2701*  2736
$ERMAX  001115  250#  569*  2658  2681*  2686
$ERROR  016666  560  2699#
$ERRPC  001116  251#  2597  2599  2600  2602  2604  2605  2607  2609  2611  2613  2614  2616
                2618  2705*  2706*  2707  2736  2928
$ERRTB  001322  363#  2936
$ERRTY  017730  2710  2921#
$ERTTL  001112  248#  2704*  2736
$ESCAP  001170  276#  568*  2680*  2727  2729  2736
$ETABL  001216  294#
$ETEND  001322  233  347#
$FATAL  001200  287#  3209*
$FFLG   021260  3172*  3175*  3203  3212*  3220#
$FILLC  001156  269#  3138  3169
$FILLS  001155  268#  3169
$GDADR  001120  252#
$GDDAT  001124  254#  600  696  714*  717  719  727*  730  732  740*  743  745  753*
                756  758  767*  771  773  777*  785*  800*  803  805  813*  816  818
                827*  830  832  840*  843  845  854*  858  860  864*  871*  887*  890
                892  900*  903  905  913*  916  918  926*  929  931  939*  942  944
                952*  955  957  965*  968  970  978*  981  983  990*  993  995  1005*
                1029*  1032  1041*  1071*  1097*  1101*  1105  1124*  1140*  1144  1163*  1175*  1195
                1211*  1231  1245*  1249  1261*  1265  1277*  1281  1300  1311*  1315  1318*  1321
                1332*  1336  1339*  1342  1353*  1357  1368*  1372  1382*  1386  1396*  1400  1404*
                1407  1419*  1423  1427*  1430  1439*  1443  1454*  1457  1463*  1466  1477*  1480
                1491*  1494  1505*  1508  1519*  1522  1532*  1535  1545*  1548  1558*  1561  1563
                1571*  1574  1576  1584*  1587  1589  1597*  1600  1602  1611*  1615  1617  1621*
                1622  1629*  1632  1634  1642*  1645  1647  1655*  1658  1660  1668*  1671  1673
                1682*  1686  1688  1692*  1693  1703*  1706  1708  1712*  1715  1725*  1736*  1747*
                1761*  1764  1775*  1789*  1792  1833*  1837  1849*  1853  1865*  1869  1881*  1885
                1893*  1896  1898  1907*  1910  1912  1921*  1924  1926  1935*  1938  1940  1949*
                1952  1954  1962*  1965  1967  1976*  1979  1981  1990*  1993  1995  2012*  2027*
                2037*  2060*  2065  2079*  2107*  2120  2129*  2148*  2163*  2189*  2190  2205*  2208
                2215*  2219  2412*  2415  2421*  2597  2600  2602  2605  2607  2609  2611  2616
```

```
                        2618
$GET42  013736          2283#
$GTSWR  017122          2758#   3263
$HD   = 000000            63
$HIBTS  001000           228#
$HIOCT  017726          2907*   2912#
$ICNT   001104           245*   2671*   2672    2674*   2665
$ILLUP  020230          2966    2982    3001#
$INLP   023040           600     719     732     745     758     773     792     805     818     832     845     860     878
                         892     905     918     931     944     957     970     983     995    1010    1032    1039    1047
                        1052    1074    1079    1084    1105    1114    1119    1144    1153    1158    1183    1188    1195
                        1219    1224    1231    1249    1265    1281    1300    1315    1321    1336    1342    1357    1372
                        1386    1400    1407    1423    1430    1443    1457    1466    1480    1494    1508    1522    1535
                        1548    1563    1576    1589    1602    1617    1634    1647    1660    1673    1688    1708    1715
                        1732    1739    1757    1764    1785    1792    1810    1817    1837    1853    1869    1885    1898
                        1912    1926    1940    1954    1967    1981    1995    2007    2016    2022    2026    2030    2037
                        2052    2058    2063    2082    2085    2093    2111    2120    2136    2152    2166    2171    2184
                        2189    2219    2316    2438    3657*   3720    3794    3846    3853
$INTAG  001135           259*   2786    2875
$ITEMB  001114           249*   2707*   2715    2736    2925
$LF     001174           279*   2736    2860    2869    3169
$LFLG   021257          3213*   3219#
$LOAD   022224          3312    3440#
$LPADR  001106           246*    570*   2662*   2678*   2683    2685
$LPAI   021344          3286*   3638    3683
$LPERR  001110           247*    571*    766*    853*   1610*   1681*   2662    2679*   2685    2726
$LPW    022646          3491    3495    3500    3504    3523    3526    3543    3560    3576#
$MADR1  001230           312#
$MADR2  001234           316#
$MADR3  001240           319#
$MADR4  001244           322#
$MAIL   001176           229     233     285#    588     647    2677    2713    3113
$MAMS1  001226           306#
$MAMS2  001232           314#
$MAMS3  001236           317#
$MAMS4  001242           320#
$MBADR  001002           229#
$MFLG   021256          3173*   3179    3214*   3218#
$MNEW   017615          2761    2873#
$MSGAD  001212           292#   3189*   3192
$MSGLG  001214           293#   3194*
$MSGTY  001176           286#   3187    3195*   3207    3211*
$MSWR   017604          2758    287'#
$MTYP1  001227           307#
$MTYP2  001233           315#
$MTYP3  001237           318#
$MTYP4  001243           321#
$MXCNT  016664          2675    2685#
$NULL   001154           267#   3140    3169
$NWTST= 000001           686#    710#    723#    736#    749#    762#    780#    796#    809#    823#    836#    849#    867#
                         883#    896#    909#    922#    935#    948#    961#    974#    986#    998#   1015#   1058#   1087#
                        1127#   1165#   1201#   1237#   1253#   1269#   1285#   1305#   1326#   1347#   1362#   1376#   1390#
                        1413#   1435#   1447#   1470#   1484#   1498#   1512#   1525#   1538#   1554#   1567#   1580#   1593#
                        1606#   1625#   1638#   1651#   1664#   1677#   1696#   1719#   1741#   1769#   1797#   1825#   1841#
```

```
                    1857#  1873#  1889#  1902#  1916#  1930#  1944#  1957#  1971#  1985#  2001#  2043#  2071#
                    2097#  2124#  2140#  2155#  2173#  2193#  2242#
$OCNT   020526      3045*  3074*  3087#
$OFs  = 023102      3673#  3849   3850   3856
$OMoDE  020530      3040*  3044*  3049   3052*  3063*  3089#
$OUTLP  022742       696    717    730    743    756    771    789    803    816    830    843    858    875
                     890    903    916    929    942    955    968    981    993   1005   1023   1025   1028
                    1050   1055   1065   1068   1071   1077   1082   1095   1097   1101   1117   1122   1135
                    1137   1140   1156   1161   1173   1175   1179   1186   1191   1209   1211   1215   1222
                    1227   1245   1261   1277   1293   1296   1454   1463   1477   1491   1505   1519   1532
                    1545   1554   1561   1574   1587   1600   1615   1632   1645   1658   1671   1686   1703
                    1706   1729   1751   1779   1782   1808   1813   1833   1849   1865   1881   1896   1910
                    1924   1938   1952   1965   1979   1993   2012   2050   2055   2079   2088   2105   2107
                    2114   2133   2148   2163   2181   2208   2306   2309   2312   2404   2406   2431   2434
                    2442   2444   3615#  3811
$OVER   016650      2639   2655   2663   2673   2682#
$PASS   001204       289#   588*  2271*  2272*  2280   2293   2669   2686
$PASTM  001006       231#
$PUTS   023512      3845#
$PWRAD  020224      2999#
$PWRDN  020064       564   2966#  2994
$PWRMG  020220      2997#
$PWRUP  020136      2976   2982#
$QUES   001172       277#  2736   2804   2853   2869   3169
$RDCHR  017334      2817#  3266
$RDDEC= ****** U    3269
$RDLIN  017454      2845#  3267
$RDOCT  017626      2885#  3268
$RDSZ = 000010      2838#
$REGAD  001160       271#
$REGO   001162       273#
$REG1   001164       274#
$RESET  023174       789    875   1246   1262   1278   1297   1440   1834   1850   1866   1882   2133   2149
                    2168   2186   2197   3717#  3727
$RTNAD  013760      2292#
$R2A  = ****** U    3269
$SAD    023170      3695   3697*  3705#
$SAVRE= ****** U    3269
$SAVR6  020234      2975*  2983   2984*  2985*  3003#
$SCOPE  016404       558   2635#
$SETUP= 000137       442#   557    558    560    562    564    566    567    568    570    641    644   2269
                    2636   2700   2723   2731   2741   2875
$STUP = 177777       442#
$SVLAD  016614      2647   2676#
$SVPC = 000210       205#   210
$SWR  = 165400        30#    63    180    181    182    193    184    185    186    187    275    276    277
                     567    568    570    571    690    714    727    740    753    766    784    800    813
                     827    840    853    871    887    900    913    926    939    952    965    978    990
                    1002   1019   1062   1091   1131   1169   1205   1241   1257   1273   1289   1309   1330
                    1351   1366   1380   1394   1417   1439   1451   1474   1488   1502   1516   1529   1542
                    1558   1571   1584   1597   1610   1629   1642   1655   1668   1681   1700   1723   1745
                    1773   1801   1829   1845   1861   1877   1893   1906   1920   1934   1948   1961   1975
                    1989   2005   2047   2075   2101   2128   2144   2159   2177   2197   2246   2264   2270
                    2285   2291   2293   2627   2628   2629   2630   2631   2638   2650   2652   2653   2656
```

```
                    2657    2658    2665    2666    2667    2679    2682    2685    2692    2693    2694    2695    2704
                    2708    2720    2724    2736    3000
$SWREG  001220       297*     591
$SWRMK= 000000       187      188    2631    2632    2654
$TESTN  001202       288*    2677*
$TIMES  001166       275*     567*    784*   1019*   1091*   1131*   1169*   1205*   1241*   1257*   1273*   1289*   1309*
                    1330*   1351*   1366*   1380*   1394*   1417*   1829*   1845*   1861*   1877*   2075*   2101*   2128*
                    2144*   2159*   2177*   2270*   2665*   2672    2675*   2685
$TKB    001146       264*    2739    2750    2767    2821    2827    3786
$TKS    001144       263*    2739    2748    2764    2788*   2819    2825    3784
$TLKR   022500      3520*    3672
$TLKW   022364      3488*    3630
$TMDAT  001670       527*     597*    600     786*    789     872*    875    1002*   1005    1020*   1023    1025*   1028
                    1047*   1050    1052*   1055    1062*   1065*   1068*   1071    1074*   1077    1079*   1082    1092*
                    1095    1097    1098*   1101    1114*   1117    1119*   1122    1132*   1135    1137*   1140    1153*
                    1156    1158*   1161    1170*   1173    1175    1176*   1179    1183*   1186    1188*   1191    1206*
                    1209    1211    1212*   1215    1219*   1222    1224*   1227    1242*   1245    1258*   1261    1274*
                    1277    1290*   1293*   1296    1451*   1454    1460*   1463    1474*   1477    1488*   1491    1502*
                    1505    1516*   1519    1529*   1532    1542*   1545    1551*   1554    1700*   1703    1726*   1729
                    1748*   1751    1752*   1776*   1779*   1782    1805*   1808    1810*   1813    1817    1830*   1833
                    1846*   1849    1862*   1865    1878*   1881    2007    2009*   2012    2016    2026    2047*   2050
                    2052*   2055    2058    2076*   2079    2082    2085*   2088    2102*   2105    2107    2111*   2114
                    2130*   2133    2145*   2148    2160*   2163    2166    2178*   2181    2184    2202*   2204*   2208
                    2216*   2219    2303*   2306*   2309*   2312    2316    2401*   2404    2406    2428*   2431*   2434
                    2438    2439*   2442    2444
$TN     000126       30*      63      686     690*    704     710     714*    720     723     727*    733     736     740*
                     746     749     753*    759     762     766*    780     784*    793     796     800*    806     809
                     813*    819     823     827*    833     836     840*    846     849     853*    867     871*    879
                     883     887*    893     896     900*    906     909     913*    919     922     926*    932     935
                     939*    945     948     952*    958     961     965*    971     974     978*    984     986     990*
                     996     998    1002*   1015    1019*   1043    1058    1062*   1085    1087    1091*   1127    1131*
                    1165    1169*   1196    1201    1205*   1232    1237    1241*   1250    1253    1257*   1266    1269
                    1273*   1282    1285    1289*   1301    1305    1309*   1322    1326    1330*   1343    1347    1351*
                    1358    1362    1366*   1373    1376    1380*   1387    1390    1394*   1408    1413    1417*   1431
                    1435    1439*   1444    1447    1451*   1467    1470    1474*   1481    1484    1488*   1495    1498
                    1502*   1509    1512    1516*   1523    1525    1529*   1536    1538    1542*   1554    1558*   1564
                    1567    1571*   1577    1580    1584*   1590    1593    1597*   1603    1606    1610*   1625    1629*
                    1635    1638    1642*   1648    1651    1655*   1661    1664    1668*   1674    1677    1681*   1696
                    1700*   1716    1719    1723*   1733    1741    1745*   1758    1765    1769    1773*   1786    1793
                    1797    1801*   1802    1818    1825    1829*   1838    1841    1845*   1854    1857    1861*   1870
                    1873    1877*   1886    1889    1893*   1899    1902    1906*   1913    1916    1920*   1927    1930
                    1934*   1941    1944    1948*   1955    1957    1961*   1968    1971    1975*   1982    1985    1989*
                    1996    2001    2005*   2039    2043    2047*   2064    2066    2071    2075*   2094    2097    2101*
                    2121    2124    2128*   2137    2140    2144*   2152    2155    2159*   2171    2173    2177*   2191
                    2193    2197*   2227    2242    2246*
$TOUT   023134      3334    3388    3415    3529    3546    3695*
$TPB    001152       266*   3158*   3169
$TPFLG  001157       270*   3107    3169
$TPS    001150       265*   3156    3169
$TRAP   021262       562    3234*
$TRAP2  021304      3245*   3256
$TRP    000013      3249*   3258*   3259*   3260*   3261*   3262*   3263    3264*   3265    3266*   3267*   3268*   3269*
$TRPAD  021316      3239    3256*
$TSTM   001004       230*
```

```
$TSTNM  001102            243#  2269#  2626  2654  2676#  2677  2682  2686  2703  2736
$TTYIN  017562           2846   2847   2864  2868#
$TYPBN= ****** U         3262
$TYPDS  014120           2335#  3261
$TYPE   020532           3107#  3200   3249  3257
$TYPEC  020744           2790   3137   3144  3151  3156#  3157
$TYPEX  021012           3162   3164   3167#
$TYPOC  020330           3043#  3258
$TYPON  020344           3042   3045#  3260
$TYPOS  020304           3038#  3259
$UNIT   001210            291#
$UNITM  001010            232#
$USWR   001222            298#
$UTK    023316           3777#
$VECT1  001246            323#   614   2253
$VECT2  001250            324#
$XTSTR  016420           2641#
$$GET4= 000000           2285#
$OFILL  020527           3039#  3043#  3053  3088#
$40CAT= ****** U         2638   2710
.       = 023622          190#   194#   205   206#  208#  210#  211#  217   218#  220#  222#  240#  280
                          487#   555    570   571   657#  2167  2293  2297  2389#  2685  2686  2736  2739
                         2868#  2869   2875  2960#  2978  3002  3169  3221#  3673  3783#
.DVLS   001540            486#   547#   691#  3618  3639  3660  3684  3778#
.$ASTA= ****** U         3173   3176
.$X     001000            217#   222
```

```
            1996    2031    2039    2064    2066    2094    2121    2137    2152    2171    2191    2220    2227
SLASH        173#
SPACE        173#
STARS        173#     203     214     216     223     236     280     283     686     688     710     712     723     725     736
             738      749     751     762     764     780     782     796     798     809     811     823     825     836     838
             849      851     867     869     883     885     896     898     909     911     922     924     935     937     948
             950      961     963     974     976     986     968     998    1000    1015    1017    1058    1060    1087    1089
            1127     1129    1165    1167    1201    1203    1237    1239    1253    1255    1269    1271    1285    1287    1305
            1307     1326    1328    1347    1349    1362    1364    1376    1378    1390    1392    1413    1415    1435    1437
            1447     1449    1470    1472    1484    1486    1498    1500    1512    1514    1525    1527    1538    1540    1554
            1556     1567    1569    1580    1582    1593    1595    1606    1608    1625    1627    1638    1640    1651    1653
            1664     1666    1677    1679    1696    1698    1719    1721    1741    1743    1769    1771    1797    1799    1825
            1827     1841    1843    1857    1859    1873    1875    1889    1891    1902    1904    1916    1918    1930    1932
            1944     1946    1957    1959    1971    1973    1985    1987    2001    2003    2043    2045    2071    2073    2097
            2099     2124    2126    2140    2142    2155    2157    2173    2175    2193    2195    2242    2244    2260    2325
            2623     2688    2738    2741    2809    2838    2877    2916    2964    2980    3015    3092    3171    3228
SWRSU        173#     572#
TOUT         489#    3333    3387    3414    3528    3545
TRMTRP      3249#
TYPBIN       173#
TYPDEC       173#    2280
TYPNAM       173#     637
TYPNUM       173#
TYPOCS       173#
TYPOCT       173#    2759    2928    2952    3798
TYPTXT       173#     632     661    3304    3358    3362    3366    3593    3779    3804
$CAL.         30#    3269
$DMAST        30#
$DMDT         30#
$MMAST        30#
$$CMRE       234#     273     274
$$CMTM       234#
$$ESCA       173#
$$NEWT       173#     686     710     723     736     749     762     780     796     809     823     836     849     867     883
             896      909     922     935     948     961     974     986     998    1015    1058    1087    1127    1165    1201
            1237     1253    1269    1285    1305    1326    1347    1362    1376    1390    1413    1435    1447    1470    1484
            1498     1512    1525    1538    1554    1567    1580    1593    1606    1625    1638    1651    1664    1677    1696
            1719     1741    1769    1797    1825    1841    1857    1873    1889    1902    1916    1930    1944    1957    1971
            1985     2001    2043    2071    2097    2124    2140    2155    2173    2193    2242
$$SET       3249#    3258    3259    3260    3261    3263    3265    3266    3267    3268
$$SETM       588#
$$SKIP       173#     704     720     733     746     759     793     806     819     833     846     879     893     906     919
             932      945     958     971     984     996    1043    1085    1196    1232    1250    1266    1282    1301    1322
            1343     1358    1373    1387    1408    1431    1444    1467    1481    1495    1509    1523    1536    1564    1577
            1590     1603    1635    1648    1661    1674    1716    1733    1758    1765    1786    1793    1802    1818    1838
            1854     1870    1886    1899    1913    1927    1941    1955    1968    1982    1996    2039    2064    2066    2094
            2121     2137    2152    2171    2191    2227
.EQUAT        30#      63
.HEADE        30#      53
.KMADR        17#     455
.KSIS         22#     533
.LOADL        25#    3432
.LPAIN        22#    3270
.PUTCS        24#    3825
```

```
.RESET      22#   3707
.SETUP      30#    442
.SWRHI      30#    176
.SWRLO     188#
.UTK        29#   3767
.$ACTI      30#    201
.$APTB      30#    281#
.$APTH      30#    212
.$APTY      30#   3169
.$CATC      30#    188
.$CMTA      30#    234
.$EOP       30#   2258
.$ERRC      30#   2686
.$ERRT      30#   2914
.$INLP      28#   3642
.$MMAC      18#
.$OUTL      27#   3603
.$PARM      30#
.$POWE      30#   2962
.$RDOC      30#   2875
.$READ      30#   2736
.$SAVE      30#
.$SCOP      30#   2621
.$SPAC      30#
.$SWDO      30#
.$TLKW      26#   3481
.$TOUT     489#   3686
.$TRAP      30#   3226
.$TYPD      30#   2323
.$TYpE      30#   3090
.$TYPO      30#   3013
```

```
                1527  1529  1536  1538  1540  1542  1549  1554  1556  1558  1564  1567  1569  1571  1577
                1580  1582  1584  1590  1593  1595  1597  1603  1606  1608  1610  1618  1625  1627  1629
                1635  1638  1640  1642  1648  1651  1653  1655  1661  1664  1666  1668  1674  1677  1679
                1681  1689  1696  1698  1700  1709  1716  1719  1721  1723  1733  1741  1743  1745  1758
                1760  1765  1769  1771  1773  1786  1788  1793  1797  1799  1801  1802  1818  1820  1822
                1825  1827  1829  1830  1838  1841  1843  1845  1846  1854  1857  1859  1861  1862  1870
                1873  1875  1877  1878  1886  1889  1851  1893  1899  1902  1904  1906  1913  1916  1918
                1920  1927  1930  1932  1934  1941  1944  1946  1948  1955  1957  1959  1961  1968  1971
                1973  1975  1982  1985  1987  1989  1996  2001  2003  2005  2031  2039  2043  2045  2047
                2064  2066  2071  2073  2075  2076  2094  2097  2099  2101  2102  2121  2124  2126  2128
                2129  2137  2140  2142  2144  2145  2152  2155  2157  2159  2160  2171  2173  2175  2177
                2178  2191  2193  2195  2197  2220  2227  2242  2244  2246  2260  2261  2262  2263  2264
                2265  2266  2268  2274  2277  2279  2283  2285  2291  2293  2294  2325  2623  2626  2631
                2637  2638  2650  2652  2653  2654  2656  2657  2658  2667  2669  2677  2679  2684  2685
                2686  2688  2691  2701  2704  2708  2710  2711  2713  2720  2724  2731  2735  2736  2738
                2740  2741  2742  2770  2809  2810  2838  2846  2847  2851  2852  2868  2869  2875  2877
                2880  2892  2916  2931  2947  2964  2974  2975  2980  2987  2988  2996  2998  3000  3004
                3015  3092  3113  3171  3173  3176  3203  3218  3228  3234  3238  3249  3258  3259  3260
                3261  3262  3263  3265  3266  3267  3268  3269  3306  3336  3342  3360  3364  3368  3390
                3396  3417  3423  3531  3537  3548  3554  3596  3767  3782  3806
.IFF               65   183   185   187   188   204   208   210   215   217   224   237   240   243   271
                 281   284   556   640   642   687   688   689   690   705   711   712   713   714   721
                 724   725   726   727   734   737   738   739   740   747   750   751   752   753   760
                 763   764   765   766   774   781   782   783   784   794   797   798   799   800   807
                 810   811   812   813   820   824   825   826   827   834   837   838   839   840   847
                 850   851   852   853   861   868   869   870   871   880   884   885   886   887   894
                 897   898   899   900   907   910   911   912   913   920   923   924   925   926   933
                 936   937   938   939   946   949   950   951   952   959   962   963   964   965   972
                 975   976   977   978   985   987   988   989   990   997   999  1000  1001  1002  1011
                1016  1017  1018  1019  1033  1040  1044  1059  1060  1061  1062  1086  1088  1089  1090
                1091  1106  1128  1129  1130  1131  1145  1166  1167  1168  1169  1197  1202  1203  1204
                1205  1233  1238  1239  1240  1241  1251  1254  1255  1256  1257  1267  1270  1271  1272
                1273  1283  1286  1287  1288  1289  1302  1306  1307  1308  1309  1316  1323  1327  1328
                1329  1330  1337  1344  1348  1349  1350  1351  1359  1363  1364  1365  1366  1374  1377
                1378  1379  1380  1388  1391  1392  1393  1394  1401  1409  1414  1415  1416  1417  1424
                1432  1436  1437  1438  1439  1445  1448  1449  1450  1451  1458  1468  1471  1472  1473
                1474  1482  1485  1486  1487  1488  1496  1499  1500  1501  1502  1510  1513  1514  1515
                1516  1524  1526  1527  1528  1529  1537  1539  1540  1541  1542  1549  1555  1556  1557
                1558  1565  1568  1569  1570  1571  1578  1581  1582  1583  1584  1591  1594  1595  1596
                1597  1604  1607  1608  1609  1610  1618  1626  1627  1628  1629  1636  1639  1640  1641
                1642  1649  1652  1653  1654  1655  1662  1665  1666  1667  1668  1675  1678  1679  1680
                1681  1689  1697  1698  1699  1700  1709  1717  1720  1721  1722  1723  1734  1742  1743
                1744  1745  1759  1760  1766  1770  1771  1772  1773  1787  1788  1794  1798  1799  1800
                1801  1803  1819  1820  1822  1826  1827  1828  1829  1839  1842  1843  1844  1845  1855
                1858  1859  1860  1861  1871  1874  1875  1876  1877  1887  1890  1891  1892  1893  1900
                1903  1904  1905  1906  1914  1917  1918  1919  1920  1928  1931  1932  1933  1934  1942
                1945  1946  1947  1948  1956  1958  1959  1960  1961  1969  1972  1973  1974  1975  1983
                1986  1987  1988  1989  1997  2002  2003  2004  2005  2031  2040  2044  2045  2046  2047
                2065  2067  2072  2073  2074  2075  2095  2098  2099  2100  2101  2122  2125  2126  2127
                2128  2138  2141  2142  2143  2144  2153  2156  2157  2158  2159  2172  2174  2175  2176
                2177  2192  2194  2195  2196  2197  2220  2228  2243  2244  2245  2246  2261  2265  2269
                2274  2277  2293  2326  2624  2651  2654  2655  2658  2685  2689  2691  2705  2731  2736
                2739  2742  2810  2812  2817  2838  2839  2848  2852  2869  2878  2917  2932  2961  2965
                2981  2996  3016  3093  3172  3229  3235  3336  3342  3390  3396  3417  3423  3531  3537
```

```
                792   796   800   803   805   809   813   816   818   823   827   830   832   836   840
                843   845   849   853   858   860   867   871   875   878   883   887   890   892   896
                900   903   905   909   913   916   918   922   926   929   931   935   939   942   944
                948   952   955   957   961   965   968   970   974   978   981   983   986   990   993
                995   998  1002  1005  1010  1015  1019  1023  1025  1028  1032  1039  1047  1050  1052
               1055  1058  1062  1065  1068  1071  1074  1077  1079  1082  1084  1087  1091  1095  1097
               1101  1105  1114  1117  1119  1122  1127  1131  1135  1137  1140  1144  1153  1156  1158
               1161  1165  1169  1173  1175  1179  1183  1186  1188  1191  1195  1201  1205  1209  1211
               1215  1219  1222  1224  1227  1231  1237  1241  1245  1249  1253  1257  1261  1265  1269
               1273  1277  1281  1285  1289  1293  1296  1300  1305  1309  1315  1321  1326  1330  1336
               1342  1347  1351  1357  1362  1366  1372  1376  1380  1386  1390  1394  1400  1407  1413
               1417  1423  1430  1435  1439  1443  1447  1451  1454  1457  1463  1466  1470  1474  1477
               1480  1484  1488  1491  1494  1498  1502  1505  1508  1512  1516  1519  1522  1525  1529
               1532  1535  1538  1542  1545  1548  1554  1558  1561  1563  1567  1571  1574  1576  1580
               1584  1587  1589  1593  1597  1600  1602  1606  1610  1615  1617  1625  1629  1632  1634
               1638  1642  1645  1647  1651  1655  1658  1660  1664  1668  1671  1673  1677  1681  1686
               1688  1696  1700  1703  1706  1708  1715  1719  1723  1729  1732  1739  1741  1745  1751
               1757  1764  1769  1773  1779  1782  1785  1792  1797  1801  1808  1810  1813  1817  1825
               1829  1833  1837  1841  1845  1849  1853  1857  1861  1865  1869  1873  1877  1881  1885
               1889  1893  1896  1898  1902  1906  1910  1912  1916  1920  1924  1926  1930  1934  1938
               1940  1944  1948  1952  1954  1957  1961  1965  1967  1971  1975  1979  1981  1985  1989
               1993  1995  2001  2005  2007  2012  2016  2022  2026  2030  2037  2043  2047  2050  2052
               2055  2058  2063  2071  2075  2079  2082  2085  2088  2093  2097  2101  2105  2107  2111
               2114  2120  2124  2128  2133  2136  2140  2144  2148  2152  2155  2159  2163  2166  2171
               2173  2177  2181  2184  2189  2193  2197  2208  2219  2242  2246  2269  2285  2306  2309
               2312  2316  2404  2406  2431  2434  2438  2442  2444  2631  2731  2838  3249  3257  3258
               3259  3260  3261  3262  3263  3264  3265  3266  3267  3268  3269  3307  3361  3365  3369
               3597  3720  3783  3807
.PAGE           234   349
.PSECT           14
.REM              1    14    30
.REPT           194   273
.SBTTL           63   176   188   197   201   212   234   281   349   550   637   644   686   710   723
                736   749   762   780   796   809   823   836   849   867   883   896   909   922   935
                948   961   974   986   998  1015  1058  1087  1127  1165  1201  1237  1253  1269  1285
               1305  1326  1347  1362  1376  1390  1413  1435  1447  1470  1484  1498  1512  1525  1538
               1554  1567  1580  1593  1606  1625  1638  1651  1664  1677  1696  1719  1741  1769  1797
               1825  1841  1857  1873  1889  1902  1916  1930  1944  1957  1971  1985  2001  2043  2071
               2097  2124  2140  2155  2173  2193  2242  2258  2323  2621  2686  2736  2875  2914  2962
               3013  3090  3169  3226  3249
.TITLE           53
.WORD           194   195   196   209   228   229   230   231   232   233   242   245   246   247   248
                251   252   253   254   255   256   257   260   261   262   271   273   274   286   287
                288   289   290   291   292   293   297   298   299   312   316   319   322   323   324
                325   326   327   328   329   330   331   332   333   334   335   336   337   338   339
                340   341   342   343   344   464   467   469   471   473   475   477   479   481   482
                484   486   600   660   696   717   719   730   732   743   745   756   758   771   773
                789   792   803   805   816   818   830   832   843   845   858   860   875   878   890
                892   903   905   916   918   929   931   942   944   955   957   968   970   981   983
                993   995  1005  1010  1023  1025  1028  1032  1039  1047  1050  1052  1055  1065  1068
               1071  1074  1077  1079  1082  1084  1095  1097  1101  1105  1114  1117  1119  1122  1135
               1137  1140  1144  1153  1156  1158  1161  1173  1175  1179  1183  1186  1188  1191  1195
               1209  1211  1215  1219  1222  1224  1227  1231  1245  1249  1261  1265  1277  1281  1293
               1296  1300  1315  1321  1336  1342  1357  1372  1386  1400  1407  1423  1430  1443  1454
```

1457	1463	1466	1477	1480	1491	1494	1505	1508	1519	1522	1532	1535	1545	1548
1554	1561	1563	1574	1576	1587	1589	1600	1602	1615	1617	1632	1634	1645	1647
1658	1660	1671	1673	1686	1688	1703	1706	1708	1715	1729	1732	1739	1751	1757
1764	1779	1782	1785	1792	1808	1810	1813	1817	1833	1837	1849	1853	1865	1869
1881	1885	1896	1898	1910	1912	1924	1926	1938	1940	1952	1954	1965	1967	1979
1981	1993	1995	2007	2012	2016	2022	2026	2030	2037	2050	2052	2055	2058	2063
2079	2082	2085	2088	2093	2105	2107	2111	2114	2120	2133	2136	2148	2152	2163
2166	2171	2181	2184	2189	2208	2219	2274	2277	2292	2306	2309	2312	2316	2404
2406	2431	2434	2438	2442	2444	2912	2940	2945	2997	2999	3089	3119	3166	3201
3256	3313	3410	3430	3565	3631	3632	3639	3674	3684	3705	3706	3720	3730	3731
3761	3765	3795	3796	3801	3812	3813	3816	3817	3847	3848	3854	3855	3861	

000000

ERRORS DETECTED: 0

```
*DRLPC,DRLPC/SOL/CRF=DRLPA.MAC,DRLPC
RUN-TIME:  35 23 3 SECONDS
CORE USED:  40K
```