LPA/DR11-K

DIAGNOSTIC TEST

MD-11-DRLPF-A

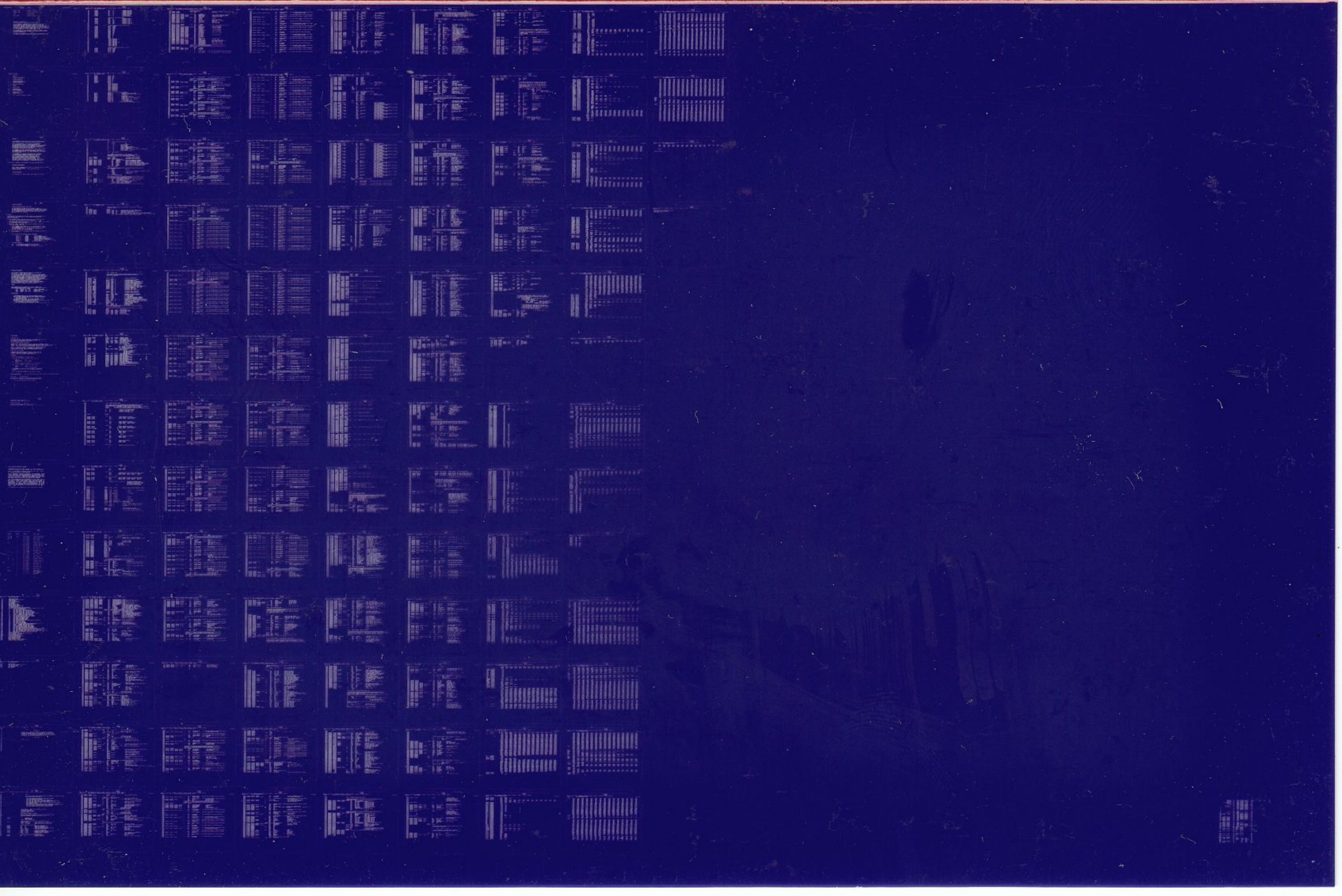
EP-DRLPF-A-DL

FICHE 1 OF 1

MAR 1978

digital

MADE IN USA



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

PRODUCT NAME: LPA/DR11-K DIAGNOSTIC TEST

DATE: JANUARY 1978

MAINTAINER: DIAGNOSTIC GROUP

TABLE OF CONTENTS

1. ABSTRACT

THIS PROGRAM IS A LOGIC TEST OF THE DR11-K DIGITAL INPUT OUTPUT CONTROL OPTION. MOST OPTION FUNCTIONS CAN BE TESTED.

DUE TO THE FLEXIBILITY OF THE OPTION, THE OPERATOR WILL BE REQUIRED TO SUPPLY OPTION CHARACTERSTICS. THE PROGRAM WILL HANDLE ALL CONFIGURATIONS OF INPUT INTERRUPT SWITCHES AND INPUT DATA LATCHING JUMPERS. FOR SYSTEMS WITH CONSECUTIVE MULTIPLE DR11-K'S, THESE CONGIGURATIONS MUST BE THE SAME. THE FOLLOWING JUMPERS MUST BE INSERTED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A.

THIS PROGRAM WILL TEST SEQUENTIAL DR11-K'S STARTING AT THE BUS ADDRESS AND VECTOR IN LOCATIONS "$BASE" AND "$VECTI". FOR NORMAL FACTORY CONFIG., ALL QUESTIONS SHOULD BE ANSWERED WITH A VALUE OF 0.

THIS PROGRAM IS A MODIFIED VERSION OF "MD-11-DZDRG-E" IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE DR11-K OPTION WHEN IT IS ON THE LPA11-KX I/O BUS. NO RECABLING IS NEEDED. SOME TEST DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION TEST, WERE DELETED AS THEY COULD NOT BE CHECKED IF THIS DIAGNOSTIC DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO RUN "MD-DZDRG-E" YOU SHOULD RUN "MD-11-DRLPA" BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 10.

AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC TEST MUST BE RUN.

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERmINAL AND 16K OF MEMORY
DR11-K OPTION INSTALLED
BC08R-1 ONE FOOT OUTPUT TO INPJT WRAPAROUND CABLE
LPA11-KX OPTION

3. LOADING PROCEDURE

THE PROCEDURE FOR LOADING BINARY TAPES SHOULD BE FOLLOWED.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS (LOGIC TEST)

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

1) TYPE CONTROL G <↑G>; THIS WILL ALLOW THE TTY TO eNTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.

2) THE MACHINE WILL THEN TYPE: SWR=XXXXXX NEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)

3) AFTER THE ''NEW='' HAS BEEN TYPED THEN THE OPERATOR CAN DO oNE OF THE FOLLOWING AT THE TTY:

   A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)
   IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.

   B) IF A CONTROL U <↑U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BA K TO STEP 2.

WILL BE USED AS THE SOFTWARE DISPLAY REGISTER.

SW 15 = 1	100000	HALT ON ERROR
SW 14 = 1	040000	LOOP ON CURRENT SUB-TEST
SW 13 = 1	020000	INHIBIT ERROR TYPOUTS
SW 12 = 1	010000	LOOP ON CURRENTLY SELECTED DR11-K
SW 11 = 1	004000	INHIBIT SUB-TEST INTERATIONS
SW 10 = 1	002000	NO OUTPUT TO INPUT WRAPAROUND CABLE
SW 09 = 1	001000	LOOP ON ERROR
SW 08 = 1	000400	LOOP ON TEST IN SWR <7:0>

4.2 STARTING ADDRESSES

200	IS THE STARTING ADDRESS OF THE LOGIC TEST.
204	IS THE RESTART ADDRESS OF THE LOGIC TEST.
210	IS THE STARTING ADDRESS OF THE CONTROL LINE LOOP.

5. OPERATING PROCEDURE

```
THE FOLLOWING JUMPERS MUST BE INSTALLED TO EXECUTE THE
LOGIC TEST:     W21A, W22A AND W23A
IF THE CUSTOMER HAS SELECTED THE "B" SECTION OF THESE JUMPERS
IT MUST BE RETURNED TO THE "FACTORY" POSITION BEFORE RUNNING
THE LOGIC TEST. ** WORST CASE WILL ONLY BE CHANGING THREE JUMPERS. **
THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH
REGISTER WHEN REQUESTED BY THE PROGRAM OR AN ERROR WILL OCCUR.
WITH THE INPUT INTERRUPT SWITCHES AND DATA LATCHING JUMPERS
IN THE FACTORY POSITION, ALL SWITCH REGISTER BITS SHOULD BE RESET.
ONCE STARTED THE TEST WILL RUN IN IT'S NORMAL MANNER
WITHOUT OPERATOR INTERVENTION OR SWITCH SELECTION.
*** NOTE: OPERATOR MUST INSERT INFORMATION WHEN REQUESTED BY
PROGRAM. THE MACHINE WILL TYPE:  NEW =  AFTER NEW =
THE INFORMATION IS INSERTED. REFER TO SECTION 4.1  2) ***
```

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.
        BYTE    $TSTNM: (LOC. 1102)     CURRENT TEST NUMBER
        BYTE    $ITEMB: (LOC. 1114)     ITEM #N ERROR TABLE INDEX
        WORD    $ERRPC: (LOC. 1116)     ERRORING P.C.
        WORD    $PASS:  (LOC. 1176)     CURRENT PASS COUNT
```

7. RESTRICTIONS

1. IF SEQUENTIAL DR11-K'S, ALL DR11-K'S MUST BE IN THE SAME INTERRUPT SWITCHES AND DATA PATH JUMPER CONFIGERATION.
2. THE FOLLOWING JUMPERS MUST BE IN THE "FACTORY" POSITION:
   W21A, W22A AND W23A
3. THE OPERATOR MUST SUPPLY THE CORRECT INTERRUPT SWITCHES AND DATA PATH JUMPER AND SWITCH CONFIGURATION INFORMATION TO THE INITALIZATION QUESTIONS OR AN ERROR WILL OCCUR.
4. FOR MULTIPLE GROUPS OF CONSECUTIVE DR11-K'S:
   THIS DIAGNOSTIC MUST BE RUN FOR EACH GROUP.
5. AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC MUST BE RUN FIRST

8. MISCELANEOUS

8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 90 SECONDS FOR COMPLETION
ON A PDP11/05 TYPE AND WILL TYPE 'END PASS NNNN.'.
THE CONTROL LINE LOOP WILL NEVER EXIT.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS

"$BASE" (LOC. 1244) CONTAINS THE DR11-K BASE DEVICE ADDRESS (767770)
"$VECT1" (LOC. 1240) THE LOW 9 BITS CONTAIN THE DR11-K BASE INTERRUPT VECTOR (300)
"$VECT1" (LOC. 1240) THE HIGH 3 BITS CONTAIN THE DR11-K BR LEVEL #4 (200)

*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 USER LINK TO I/O DEVICE

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

PROCEDURE:

1) START THE PROCESSOR AT LOCATION $UTK:

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

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

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

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

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

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

NOTE: THE OPERATORS RESPONSE IS ENCLOSED IN QUOTES.

9. PROGRAM DEScRIPTION THE LOGIC TEST MUST BE RUN FIRST AFTER INITIAL PROGRAM LOADING

9.1 LOGIC TEST (SA 200 AND 204)

THE LOGIC TEST IS A TEST OF THE CONTROL AND INPUT/OUTPUT
REGISTERS. ALL JUMPERS AND SWITCHES COMBINATIONS EXCEPT:

W21A, W22A AND W23A CAN BE DIAGNOSED.

THE PROGRAM CHECKS THAT THE DR11-K
"RESET" WILL WORK CORRECTLY.

SEQ 0007

9.2 CONTROL LINE LOOP <SA 210>

THIS TEST LOOP PROVIDES THE OPERATOR WITH A
SCOPE LOOP FOR CHECKING W21, W22 AND W23 IN THE "B" POSITION.

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 LPA11-KX 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 TESTS DELETED FROM THIS DIAGNOSTIC. THESE TESt COULD
                                NOT BE DONE THROUGH THE LPA-11

                                  TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN
                                  TEST FOR UNEXPECTED INTERRUPT
                                  TEST THAT THE INPUT CAN INTR. USING MAINT. BIT
                                  TESt THAT THE INPUT INTR. CLEAR INT. ENABLE VIA MAINT. BIT
                                  TEST THAT THE OUTPUT CAN INTR. USING MAINT. BIT
                                  TEST FOR INTR. FROM DRA INPUT  TEST FOR INTR. FROM DRA OUTPUT
                                  PRE INTERRUPT SETUP
                                  TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT
                                  TEST FOR NO INTR. FROM DRA INPUT LEVEL INDICATED VIA MAINT. INT.
                                  TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN
                                  TEST THAT RESET CLEARS DIGITAL STATUS REGISTER


                                !
                        .TITLE  DR11-K  LOGIC TEST  MAINDEC-11-DRLPF-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.
                        ;*
167770                          ABASE=167770
100300                          AVECT1=100300
                        .SBTTL  BASIC DEFINITIONS

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

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

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
```

```
                        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
                        PR0=    0               ;;PRIORITY LEVEL 0
                        PR1=    40              ;;PRIORITY LEVEL 1
                        PR2=    100             ;;PRIORITY LEVEL 2
                        PR3=    140             ;;PRIORITY LEVEL 3
                        PR4=    200             ;;PRIORITY LEVEL 4
                        PR5=    240             ;;PRIORITY LEVEL 5
                        PR6=    300             ;;PRIORITY LEVEL 6
                        PR7=    340             ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
                        SW15=   100000
                        SW14=   40000
                        SW13=   20000
                        SW12=   10000
                        SW11=   4000
                        SW10=   2000
                        SW09=   1000
                        SW08=   400
                        SW07=   200
                        SW06=   100
                        SW05=   40
                        SW04=   20
                        SW03=   10
                        SW02=   4
                        SW01=   2
                        SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                        BIT15=  100000
                        BIT14=  40000
                        BIT13=  20000
                        BIT12=  10000
                        BIT11=  4000
                        BIT10=  2000
```

```
                        BIT09=  1000
                        BIT08=  400
                        BIT07=  200
                        BIT06=  100
                        BIT05=  40
                        BIT04=  20
                        BIT03=  10
                        BIT02=  4
                        BIT01=  2
                        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
```

```
                                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                                        ;*
                                        ;*      SWITCH                  USE
                                        ;*      ------          --------------------
                                        ;*        15            HALT ON ERROR
                                        ;*        14            LOOP ON TEST
                                        ;*        13            INHIBIT ERROR TYPEOUTS
                                        ;*        12            LOOP ON CURRENTLY SELECTED DR11-K
                                        ;*        11            INHIBIT ITERATIONS
                                        ;*        10            OUTPUT TO INPUT WRAPAROUND CABLE NOT CONNECTED
                                        ;*         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  001546                  JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001602                  JMP     @#BEGIN1        ;JUMP TO THE RESTART ADDRESS
000210  000137  013600                  JMP     @#EXTTST        ;JUMP TO THE CONTROL LINES LOOP
000214  000137  014036                  JMP     @#COULTR        ;JUMP TO THE COULTER FUNCTION LOOP
000220  000137  001556                  JMP     COULTJ          ;JUMP TO SA WITH THE COULTER JUMPER CONFIG.
000224  000137  001570                  JMP     H322J           ;JUMP TO SA WITH THE LOC-BOX JUMPER CONFIG.
000230  000137  014312                  JMP     LOCBOX          ;JUMP TO SA OF LOC-BOX FUNCTION LOOP
000234  000137  014624                  JMP     XL01AD          ;JUMP TO SA OF XL01 ADJUSTMENT LOOP
                                        .SBTTL  ACT11 HOOKS

                                        ;;*****************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                000240                          $SVPC=.                 ;SAVE PC
                000046                          .=46
000046  013320                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                          .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
                000240                          .=$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  001170                 $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000030                 $TSTM:  .WORD   30       ;;RUN TIM OF LONGEST TEST
001006  000010                 $PASTM: .WORD   10       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000030                 $UNITM: .WORD   30       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000031                         .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS

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

                        001100                  .=1100
001100                  $CMTAG:                                 ;;START OF COMMON TAGS
001100  000000                  .WORD   0
001102  000                     $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000                     $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                  $LFADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                  $LSERR: .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                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001162  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001164  077                     $QUES:  .ASCII  /?/             ;;QUESTION MARK
001165  015                     $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001166  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;**********************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;**********************************************************************
                                .EVEN
001170                  $MAIL:                                  ;;APT MAILBOX
001170  000000                 $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001172  000000                 $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001174  000000                 $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001176  000000                 $PASS:  .WORD   APASS           ;;PASS COUNT
001200  000000                 $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001202  000000                 $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
```

```
  290  001204  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
  291  001206  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
  292  001210                        $ETABLE:                ;;APT ENVIRONMENT TABLE
  293  001210      000               $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
  294  001211      000               $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
  295  001212  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
  296  001214  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
  297  001216  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
  298                                ;*                      BITS 15-11=CPU TYPE
  299                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
  300                                ;*                              11/70=06,PDQ=07,Q=10
  301                                ;*                      BIT 10=REAL TIME CLOCK
  302                                ;*                      BIT  9=FLOATING POINT PROCESSOR
  303                                ;*                      BIT  8=MEMORY MANAGEMENT
  304  001220      000               $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
  305  001221      000               $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
  306                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
  307                                ;*                              900 NSEC CORE=001
  308                                ;*                              300 NSEC BIPOLAR=002
  309                                ;*                              500 NSEC MOS=003
  310  001222  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
  311                                ;*                       MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
  312  001224      000               $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
  313  001225      000               $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
  314  001226  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
  315  001230      000               $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
  316  001231      000               $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
  317  001232  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
  318  001234      000               $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
  319  001235      000               $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
  320  001236  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
  321  001240  100300                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
  322  001242  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
  323  001244  167770                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
  324  001246  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
  325  001250  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
  326  001252                        $ETEND:
  327                                .MEXIT
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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


001252                          $ERRTB:

                                ;ITEM   1
001252  016435                          EM1             ;STATUS REGISTER IN ERROR
001254  017055                          DH1             ;ERRPC  DRADD   STATUS EXPECTED
001256  017346                          DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
001260  017410                          DF0

                                ;ITEM   2
001262  016466                          EM2             ;INPUT REGISTER IN ERROR
001264  017125                          DH2             ;ERRPC  DRADD   INPUT EXPECTED
001266  017346                          DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
001270  017410                          DF0

                                ;ITEM 3
001272  016516                          EM3             ;OUTPUT REGISTER IN ERROR
001274  017173                          DH3             ;ERRPC  DRADD   OUTPUT EXPECTED
001276  017346                          DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
001300  017410                          DF0

                                ;ITEM   4
001302  016547                          EM4             ;INPUT FAILED TO INTERRUPT
001304  017241                          DH4             ;ERRPC  DRADD
001306  017362                          DT4             ;$ERRPC DRADD
001310  017410                          DF0

                                ;ITEM   5
001312  016601                          EM5             ;OUTPUT FAILED TO INTERRUPT
001314  017241                          DH4             ;ERRPC  DRADD
001316  017362                          DT4             ;$ERRPC DRADD
001320  017410                          DF0

                                ;ITEM   6
001322  016634                          EM6             ;UNEXPECTED INTERRUPT
001324  017241                          DH4             ;ERRPC  DRADD
001326  017362                          DT4             ;$ERRPC DRADD
001330  017410                          DF0

                                ;ITEM   7
001332  016661                          EM7             ;OPERATOR INTERVENTION ERROR
```

```
                                        DH4             ;ERRPC  DRADD
                                        DT4             ;$ERRPC DRADD
                                        DF0

                                ;ITEM   10
                                        EM10            ;INTERRUPT INPUT BIT FAILED TO SET INPUT READY
                                        DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
                                        DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
                                        DF0

                                ;ITEM   11
                                        EM11            ;NON-INTERRUPTING INPUT BIT SET INPUT READY
                                        DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
                                        DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
                                        DF0
                                        ;
                                        ;ADDRESS OF KMC-11 OF LPA-11    THE ADDR FOR KMAD0 MAY BE
                                        ;                               CHANGED BY THE USER TO REFLECT
                                        ;                               A DIFFERENT KMC-11 ADDR. THE
                                        ;                               REST OF THE ADDRESSES WILL
                                        ;                               BE CHANGED BY THE PROGRAM.
                                        ;

001362                          LPCI:
001362  170460                 KMAD0:  .WORD   170460          ;BASE KMC ADDR. MAY BE PATCHED BY USER.

001364                          LPMR:
001364  170461                 KMAD1:  .WORD   170460+1                ;>DO NOT        <;KMC-CSR ADDR
001366                          LPC0:
001366  170462                 KMAD2:  .WORD   170460+2                ;>PATCH         <;
001370                          LPS0:
001370  170463                 KMAD3:  .WORD   170460+3                ;>THIS AREA     <
001372                          LPADL:
001372  170464                 KMAD4:  .WORD   170460+4                ;
001374                          LPADH:
001374  170465                 KMAD5:  .WORD   170460+5                ;>DO NOT        <
001376                          LPMS1:
001376  170466                 KMAD6:  .WORD   170460+6                ;>PATCH         <
001400                          LPMS2:
001400  170467                 KMAD7:  .WORD   170460+7                ;>THIS AREA     <

001402  000300                 VECTOR: .WORD   AVECT1&777      ;BASE VECTOR OF KMC
001404  000304                 VECTPS: .WORD   4+AVECT1&777    ;VECOTR ADDR.+2

001406  000004                 VERSN:  .WORD   4               ;CURRENT VERSION NUMBER OF MICROCODE.

001410  000000                 .DVLS:  .WORD   0               ;/DEVICE LIST OF I/O ADDR. DEFINED
001412  000020                         .BLKW   16.             ;/BY INIT.


001452  167770                 BASEBA: 167770                  ;STARTING BASE BUS ADDRESS
001454  000300                 BASEIV: 300                     ;STARTING BASE INTERRUPT ADDRESS
001456  000200                 BASEBR: 200
001460  000240                 CPU:    240                     ;1 MS. CPU DELAY FACTOR 240 FOR 11/05
```

```
                                                                                    620 FOR 11/40
001462  000000                  NMBEXT: 0                       ;ADDITIONAL DR-11-K
001464  000000                  NBEXT:  0

001466  167770                  DRADD:  167770                  ;CURRENT DR11-K STARTING ADDRESS
001470  000300                  DRIV:   300                     ;CURRENT DR11-K STARTING INTERRUPT VECTOR


001472  167770                  LOC1:   167770                  ;LOC BOX #1 ADDR
001474  167760                  LOC2:   167760                  ;LOC BOX #2 ADDR
001476  167750                  LOC3:   167750                  ;LOC BOX #3 ADDR
001500  167770                  GRSTAT: 167770          ;DR STATUS
001502  167772                  GRDAI:  167772          ;INPUT REG.
001504  167774                  GRDIO:  167774          ;OUTPUT REG.
001506  167775                  GRBHIO: 167775          ;OUTPUT REG HIGH BYTE
001510  000000                  NOTLCH: 0
001512  000000                  INTBIT: 0
001514  000300                  GRIVA:  300
001516  000302                  GRIVSA: 302
001520  000304                  GRIVB:  304
001522  000306                  GRIVSB: 306
001524  000200                  DIOBRL: 200
001526  000000                  BRLEV1: 0
001530  000000                  BRLEV2: 0
001532  000000                  BRLEV3: 0
001534  000000                  ODDJMP: 0
001536  000000                  MINSIN: 0
001540  000000                  TRANST: 0
001542  000000                  JUMPER: 0       ;1 IF RUNNING H322 JUMPER CONFIG ;2 IF COULTER JUMPER CONFIG
001544  000000                  $TMDAT: 0

                                ;;*******************************************************
                                ;       DIGITAL I-O LOGIC TEST
                                ;*******************************************************
001546  005000          BEGIN:  CLR     R0                      ;CLEAR R0
001550  005037  001542          CLR     JUMPER                  ;INDICATE NORMAL FACTORY BUILD
001554  000414                  BR      RBEG
001556  012737  000002  001542  COULTJ: MOV     #2,JUMPER               ;INDICATE COULTER JUMPER CONFIG
001564  005000                  CLR     R0
001566  000407                  BR      RBEG
001570  012737  000001  001542  H322J:  MOV     #1,JUMPER               ;INDICATE LOC BOX JUMPER CONFIG
001576  005000                  CLR     R0
001600  000402                  BR      RBEG
001602  012700  177777          BEGIN1: MOV     #-1,R0                  ;LOAD R0
001606  000240                  RBEG:   NOP
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001610  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001614  005026                  CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001616  022706  001140                  CMP     #SWR,R6 ;;DONE?
001622  001374                  BNE     .-6             ;;LOOP BACK IF NO
001624  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
001630  012737  017430  000020          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
                                                MOV     #340,@#PWRVEC+2 ;;LEVEL 7
                                                MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
                                                CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
                                                CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
                                                MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
                                                MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
                                                MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
                                                MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
                                                MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
                                                MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
                                                MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
                                                CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
                                                BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
                                                BR      65$             ;;BRANCH IF NO TIMEOUT
                                        64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
                                                RTI
                                        65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
                                                MOV     #DISPREG,DISPLAY
                                        66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                                                CLR     $PASS           ;;CLEAR PASS COUNT
                                                BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
                                                BEQ     67$             ;;YES,USE NON-APT SWITCH
                                                MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
                                        67$:
                                                ;
                                                ;THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
                                                ;

                                                MOV     R0,-(SP)
                                                MOV     R1,-(SP)
                                                MOV     KMADO,R0        ;GET KMC-11 ADDRESS.
                                                MOV     #KMAD1,R1       ;GET ADDR. OF ADDR. LIST.

                                        68$:    INC     R0              ;UPDATE ADDR.
                                                MOV     R0,(1)+         ;WRITE ADDR.
                                                CMP     R1,#KMAD7+2     ;DONE ALL ADDRESSES?
                                                BNE     68$             ;NO - DO NEXT ADDR.
                                                CLR     .DVLS           ;CLR ADDR. LIST.
                                                MOV     (SP)+,R1
                                                MOV     (SP)+,R0
                                                MOV     $BASE,BASEBA            ;GET APT DEFINED ADDRESS
                                                MOV     $VECT1,BASEIV           ;GET VECTOR
                                                BIC     #160000,BASEIV
                                                MOVB    $VECT1+1,BASEBR         ;GET PRIORITY
```

```
                                        BIC     #177437,BASEBR
                                        MOV     #1$,@#ERRVEC            ;LOAD BUS ERROR
                                        MOV     BASEBA,R2               ;LOAD STARTING ADDRESS
                                        CLR     R3                      ;CLEAR COUNT
                                2$:     MOV     R2,$TMDAT               ;TEST IF EXISTENT

                                ;*      MOV     @$TMDAT,$GDDAT  ;/READ DEVICE REG $TMDAT,PUT DATA IN $GDDAT.
                                        TST     $AERR
                                        BNE     1$
                                        SUB     #10,R2                  ;EXIST, UPDATE TEST ADDRESS
                                        INC     R3                      ;UPDATE # OF DR11'S
                                        BR      2$
                                1$:     TST     R3                      ;TEST IF FIRST DOES EXIST
                                        BNE     3$                      ;BR
                                        BIT     #SW13,@SWR              ;TEST IF INHIBIT TYPEOUT
                                        BNE     4$                      ;BR IF YES
                                        TYPE,   BUSTRP                  ;TELL OPERATOR BASE DR11K DOESN'T EXIST
                                4$:     BIT     #SW15,@SWR              ;TEST HALT ON ERROR
                                        BEQ     5$                      ;BR IF NO HALT
                                        HALT                            ;FIRST DR11-K DOES NOT EXIST
                                                                        ;CHECK THE PROGRAM DEVICE ADDRESS
                                5$:     BR      2$                      ;TRY AGAIN

                                3$:     DEC     R3                      ;ADJUST R3
                                        MOV     R3,NMBEXT               ;SAVE THE NUMBER OF ADDITIONAL DR11-K
                                        MOV     #VECTRP,@#ERRVEC        ;RESET BUS ERROR
                                        MOV     #340,@#ERRVEC+2
                                RBEG1:  JSR     PC,$RESET
                                        TST     @#42                    ;TEST IF UNDER A MONITOR
                                        BNE     4$                      ;BR IF
                                        TST     R0                      ;TEST R0
                                        BEQ     2$                      ;BR IF CLEARED
                                4$:     JMP     IOTEST                  ;JUMP IF SET
                                2$:
                                        TYPE    ,65$                    ;;TYPE ASCIZ STRING
                                        BR      64$                     ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12><15><12>/DR11-K DIGITAL INPUT OUTPUT LOGIC TEST/
                                64$:
                                .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$
```

```
                                        ;;SET AUTO-MODE INDICATOR
                                        ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  <CRLF><15><12>/MAINDEC-11-DRLPF-A/<15><12><CRLF>
                                        ;;66$:
                                        ;;TYPE ASCIZ STRING
                                        ;;GET OVER THE ASCIZ
                                        ;;71$:  .ASCIZ  /(8) ADDITIONAL DR11-K'S CONNECTED/<15><12>
                                        ;;70$:
                                        ;TEST IF LOC-BOX CONFIG.
                                        ;BR IF NOT
                                        ;LOAD LOC-BOX JUMPER CONFIG
                                        ;
                                        ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
                                        ;TEST IF COULTER CONFIG.
                                        ;BR IF NOT
                                        ;LOAD COULTER JUMPER CONFIG.
                                        ;RUN THE LOGIC TEST WITH COULTER JUMPERS
                                        ;TALK TO THE ANIMALS
                                        ;ABOUT NON LATCH INPUT BITS
                                        ;TALK TO THE ANIMAL AGAIN
                                        ;ABOUT THE INTERRUPTING BITS
                                        ;ISN'T THIS BOARING
                                        ;AGAIN-ABOUT THE MINUS INPUT BITS
                                        ;HO-HUM
                                        ;AGAIN-ABOUT THE TRANSITION BITS

                                        ;DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
                                        ;LOAD TEST BIT
                                        ;TEST IF NEG INPUT BIT
                                        ;BR IF NOT
                                        ;TEST IF TRANS. INPUT
                                        ;BR IF NOT
                                        ;LOGICAL OPERATOR ERROR
                                        ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
                                        ;
                                        ;MOVE LEFT
                                        ;BR UNTIL DONE
                                        ;ASK THE OPERATOR
```

```
                        BRLEV3                          ;ABOUT PROGRAM OPTIONS
003016  000137  003332  JMP     IOTEST

                        ;INTERRUPT TO UNEXPECTED VECTOR
003022  021627  001000  VECTRP: CMP     (SP),#1000      ;TEST IF IN PROGRAM CODE
003026  103402          BLO     1$                      ;BR IF NOT
003030  000000  70$:    HALT                            ;FATAL BUS TRAP IN PROGRAM AREA
003032  000776          BR      70$
003034  021627  000200  1$:     CMP     (SP),#200       ;TEST IF IN SACRED VECTOR AREA
003040  101002          BHI     2$                      ;BR IF NOT
003042  000000  71$:    HALT                            ;FATAL VECTOR TRAP
003044  000776          BR      71$
003046  012637  003244  2$:     MOV     (SP)+,72$       ;GET ADDR THE TRAP OCCURRED TO
003052  162737  000004  003244  SUB     #4,72$          ;MAKE REAL ADDRESS
003060  032777  020000  176052  BIT     #SW13,@SWR      ;TEST IF TYPE ERROR INHIBIT
003066  001050          BNE     3$                      ;BR IF INHIBIT
003070  104401  003076          TYPE    ,65$            ;;TYPE ASCIZ STRING
003074  000417          BR      64$                     ;;GET OVER THE ASCIZ
                        ;;65$:  .ASCIZ  <15><12>/DR11K INTERRUPTED TO LOC. /
003134                  64$:
003134  013746  003244          MOV     72$,-(SP)       ;LOAD VALUE TO BE TYPED
003140  104402          TYPOC
003142  104401  003150          TYPE    ,67$            ;;TYPE ASCIZ STRING
003146  000420          BR      66$                     ;;GET OVER THE ASCIZ
                        ;;67$:  .ASCIZ  / AND WILL NOW USE THAT VECTOR/<15><12>
003210                  66$:
003210  042737  000007  003244  3$:     BIC     #7,72$          ;MASK OFF LOWER 3 BITS
003216  032777  100000  175714          BIT     #SW15,@SWR      ;TEST IF HALT ON ERROR
003224  001401          BEQ     4$                      ;BR IF NOT
003226  000000          HALT            ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
003230  022626  4$:     CMP     (SP)+,(SP)+             ;CLEAN THE STACK
003232  013737  003244  001470          MOV     72$,@RIV        ;LOAD NEW VECTOR ADDRESS
003240  000137  003360          JMP     RBEG2           ;TEST THE DR11K AT THE NEW VECTOR
003244  000000  72$:     0

                        ;OPERATOR QUESTIONS AND ANSWER ROUTINE

003246  012537  003260  TALK:   MOV     (R5)+,10$       ;GET ASCII POINTER
003252  012537  003324          MOV     (R5)+,11$       ;GET POINTER TO DATA FROM OPERATOR
003256  104401          TYPE
003260  015740  10$:    SWNLB                           ;TELL OPERATOR TO LOAD SWITCHES
003262  023727  001140  000176  CMP     SWR,#SWREG      ;TEST IF SWITCH REG. EXISTS
003270  001006          BNE     1$                      ;BR IF NO
003272  104401  022147          TYPE    ,$MSWR          ;TYPE "SWR ="
003276  104412          RDOCT                           ;READ OCTAL
003300  012677  175634          MOV     (SP)+,@SWR      ;SAVE THE SWITCHES
003304  000403          BR      2$
003306  104401  1$:     TYPE
003310  016413          DEPCNT                          ;TELL OPERATOR TO DEPRESS CONT
003312  000000          HALT                            ;WAIT FOR OPERATOR
003314  017777  175620  000002  2$:     MOV     @SWR,@11$       ;LOAD SWITCH VALUE
003322  000205          RTS     R5                      ;EXIT
003324  001510  11$:    NOTLCH                          ;POINTER TO DATA TO BE LOADED
```

```
                                        ;UNEXPECTED INTERRUPT
003326  104006                  UNEXPT: ERROR   6                       ;UNEXPECTED INTERRUPT DUR;NG A SUB-TEST
003330  000002                          RTI                             ;EXIT

003332  004737  024354          IOTEST: JSR     PC,$RESET
003336  013737  001452  001466          MOV     BASEBA,DRADD            ;LOAD INITIAL STARTING ADDRESS
003344  013737  001454  001470          MOV     BASEIV,DRIV             ;LOAD INITIAL STARTING VECTOR
003352  013737  001462  001464          MOV     NMBEXT,NBEXT            ;RELOAD # AVAILABLE
003360  004737  024354          RBEG2:  JSR     PC,$RESET
003364  012701  000240                  MOV     #240,R1                 ;LOAD R1
003370  012702  000242                  MOV     #242,R2
003374  010221                  5$:     MOV     R2,(R1)+                ;SAVE THE ADDRESS
003376  012721  004700                  MOV     #4700,(R1)+             ;LOAD "JSR      PC,R0"
003402  022222                          CMP     (R2)+,(R2)+             ;BUMP R2
003404  020227  001076                  CMP     R2,#$CMTAG-2            ;TEST FOR LAST
003410  001371                          BNE     5$                      ;BR UNTIL DONE

003412  004737  024354          IOTST1: JSR     PC,$RESET
003416  004737  003424                  JSR     PC,SETADD               ;SET UP BUS ADDRESS AND VECTOR
003422  000453                          BR      IOTTS1
003424  013737  001466  001500  SETADD: MOV     DRADD,GRSTAT            ;LOAD 1ST ADDRESS
003432  013737  001466  001502          MOV     DRADD,GRDAI             ;LOAD 2ND ADDRESS
003440  062737  000002  001502          ADD     #2,GRDAI
003446  013737  001466  001504          MOV     DRADD,GRDIO             ;LOAD 3RD ADDRESS
003454  062737  000004  001504          ADD     #4,GRDIO
003462  013737  001466  001506          MOV     DRADD,GR8HIO            ;LOAD 4TH ADDRESS
003470  062737  000005  001506          ADD     #5,GR8HIO
003476  013737  001470  001514          MOV     DRIV,GRIVA              ;LOAD FIRST VECTOR
003504  013737  001470  001516          MOV     DRIV,GRIVSA
003512  062737  000002  001516          ADD     #2,GRIVSA
003520  013737  001470  001520          MOV     DRIV,GRIVB              ;LOAD 2ND VECTOR
003526  062737  000004  001520          ADD     #4,GRIVB
003534  013737  001470  001522          MOV     DRIV,GRIVSB
003542  062737  000006  001522          ADD     #6,GRIVSB
003550  000207                          RTS     PC
003552  013737  001456  001524  IOTTS1: MOV     BASEBR,DIOBRL           ;LOAD BR LEVEL
003560  005037  001534                  CLR     ODDJMP
003564  053737  001536  001534          BIS     MINSIN,ODDJMP           ;SET MINUS INPUT BITS
003572  053737  001540  001534          BIS     TRANST,ODDJMP           ;SET TRANSITION INPUT BITS
003600  012777  003326  175706          MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
003606  012777  000340  175702          MOV     #340,@GRIVSA
003614  012777  003326  175676          MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
003622  012777  000340  175672          MOV     #340,@GRIVSB
                                ;;**************************************************************
                                ;*TEST 1         TEST FOR NO BUS ERRORS
                                ;;**************************************************************
003630  000004                  TST1:   SCOPE
003632  012737  000001  001174          MOV     #1,$TESTN
003640  012737  000001  001102          MOV     #1,$TSTNM
003646  012737  000000  001544          MOV     #0,$TMDAT

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

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

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

                                        ;;*****************************************************************
                                        ;*TEST 2         TEST THAT OUTPUT REG. CAN HOLD #-1
                                        ;;*****************************************************************
003704  000004                        TST2:   SCOPE
003706  012737  177777  001124                MOV     #-1,$GDDAT              ;LOAD EXPECTED
003714  012737  177777  001544                MOV     #-1,$TMDAT              ;ALL ONES TO rEGISTER

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

                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
003742  023737  001124  001126                CMP     $GDDAT,$BDDAT           ;COMPARE
003750  001401                               BEQ     TST3                    ;;BR IF EQUAL
003752  104003                               ERROR   3                       ;REG WILL NOT HOLD ONES
                                        ;;*****************************************************************
                                        ;*TEST 3         TEST THAT RESET CLEARS OUTPUT REG.
                                        ;;*****************************************************************
003754  000004                        TST3:   SCOPE
003756  012737  000002  001160                MOV     #2,$TIMES               ;;DO 2 ITERATIONS
003764  005037  001124                        CLR     $GDDAT
003770  012737  177777  001544                MOV     #-1,$TMDAT

                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
004006  004737  024354                        JSR     PC,$RESET                       ;SET DATA TO ALL ONES

                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
004022  005737  001126                        TST     $BDDAT
004026  001401                               BEQ     TST4                    ;;BR IF EQUAL
004030  104003                               ERROR   3                       ;REG FAILED TO CLEAR

                                        ;;*****************************************************************
                                        ;*TEST 4         TEST THAT OUTPUT REG. CAN HOLD #52525
                                        ;;*****************************************************************
004032  000004                        TST4:   SCOPE
004034  012737  052525  001124                MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
004042  012737  052525  001544                MOV     #52525,$TMDAT

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

                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
004070  023737  001124  001126                CMP     $GDDAT,$BDDAT           ;COMPARE
004076  001401                               BEQ     TST5                    ;;BR IF EQUAL
004100  104003                               ERROR   3                       ;DATA NOT=52525

                                        ;;*****************************************************************
                                        ;*TEST 5         TEST THAT OUTPUT REG. CAN HOLD #125252
                                        ;;*****************************************************************
004102  000004                        TST5:   SCOPE
004104  012737  125252  001124                MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
004112  012737  125252  001544                MOV     #125252,$TMDAT

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

```
                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
004140  023737  001124  001126                 CMP     $GDDAT,$BDDAT           ;COMPARE
004146  001401                                 BEQ     TST6            ;;BR IF EQUAL
004150  104003                                 ERROR   3                       ;DATA NOT=125252

                                        ;;*****************************************************************
                                        ;*TEST 6        TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
                                        ;;*****************************************************************
004152  000004                          TST6:   SCOPE
004154  012737  000001  001160                 MOV     #1,$TIMES       ;;DO 1 ITERATION
004162  012737  004174  001110                 MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
004170  005037  001124                         CLR     $GDDAT                  ;CLEAR PATTERN
004174                                  2$:

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

                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
004214  023737  001124  001126                 CMP     $GDDAT,$BDDAT           ;COMPARE
004222  001401                                 BEQ     1$              ;;BR IF EQUAL
004224  104003                                 ERROR   3                       ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
004226  005237  001124                  1$:    INC     $GDDAT                  ;UPDATE THE PATTERN
004232  001360                                 BNE     2$                      ;TRY AGAIN
                                        ;;*****************************************************************
                                        ;*TEST 7        FLOAT A 1 ACROSS THE OUTPUT REGISTER
                                        ;;*****************************************************************
004234  000004                          TST7:   SCOPE
004236  012737  004252  001110                 MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
004244  012737  000001  001124                 MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE

004252  012737  000000  001544          1$:    MOV     #0,$TMDAT               ;CLEAR OUTPUT

                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
004270  053737  001124  001544                 BIS     $GDDAT,$TMDAT           ;SET THAT BIT

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

                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
004316  023737  001124  001126                 CMP     $GDDAT,$BDDAT           ;TEST RESULTS
004324  001401                                 BEQ     2$                      ;BR IF EQUAL ?
004326  104003                                 ERROR   3

004330  006337  001124                  2$:    ASL     $GDDAT                  ;SHIFT EXPECTED DATA
004334  001346                                 BNE     1$                      ;BR UNTIL DONE
                                        ;;*****************************************************************
                                        ;*TEST 10       FLOAT A 0 ACROSS THE OUTPUT REGISTER
                                        ;;*****************************************************************
004336  000004                          TST10:  SCOPE
004340  012737  004354  001110                 MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
004346  012737  000001  001124                 MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE

004354  012737  177777  001544          1$:    MOV     #-1,$TMDAT              ;LOAD OUTPUT TO A ONE

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

```
004372  043737  001124  001544          BIC     $GDDAT,$TMDAT           ;CLEAR A BIT
                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
004420  005137  001126                  COM     $BDDAT                  ;COMPLEMENT IT
004424  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;EQUAL ?
004432  001401                          BEQ     2$                      ;BR IF EQUAL
004434  104003                          ERROR   3
004436  006337  001124          2$:     ASL     $GDDAT                  ;SHIFT LEFT
004442  001344                          BNE     1$                      ;BRANCH UNTIL DONE
                                ;;*************************************************************
                                ;*TEST 11        TEST FOR SLOW OUTPUT GATES WITH #125252
                                ;;*************************************************************
004444  000004                  TST11:  SCOPE
004446  012737  125252  001124          MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
                                ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004474  005137  001544                  COM     $TMDAT
                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004520  005137  001544                  COM     $TMDAT
                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004544  005137  001544                  COM     $TMDAT
                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004570  005137  001544                  COM     $TMDAT
                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004614  005137  001544                  COM     $TMDAT
                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004640  005137  001544                  COM     $TMDAT
                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004664  005137  001544                  COM     $TMDAT
```

```
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004710  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
004734  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;TEST REGISTER
004742  001401                                  BEQ     TST12           ;;BR IF CORRECT
004744  104003                                  ERROR   3
                                        ;;*****************************************************************
                                        ;*TEST 12       TEST FOR SLOW OUTPUT GATES WITH #52525
                                        ;;*****************************************************************
004746  000004                          TST12:  SCOPE
004750  012737  052525  001124                  MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
                                        ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
004776  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
005022  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
005046  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
005072  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
005116  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ,/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
005142  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
005166  005137  001544                          COM     $TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
```

```
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
005212  005137  001544                          COM     $TMDAT

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

                                        ;*      MOV     @GRDIO,$BDDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
005236  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;TEST PATTERN
005244  001401                                  BEQ     TST13           ;;BR IF EQUAL
005246  104003                                  ERROR   3                       ;OUTPUT REGISTER IN ERROR

                                        ;;*************************************************************
                                        ;*TEST 13       TEST OUTPUT DATA ACCEPT FLAG
                                        ;;*************************************************************
005250  000004                          TST13:  SCOPE
005252  012737  000000  001544                  MOV     #0,$TMDAT

                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
005270  012737  177777  001544                  MOV     #-1,$TMDAT

                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
005306  012737  100000  001124                  MOV     #BIT15,$GDDAT           ;LOAD EXPECTED

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

                                        ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
005334  033737  001124  001126                  BIT     $GDDAT,$BDDAT
005342  001001                                  BNE     TST14           ;;BR IF SET
005344  104001                                  ERROR   1                       ;ERROR, BIT 15 FAILED TO SET

                                        ;;*************************************************************
                                        ;*TEST 14       TEST OUTPUT INTERRUPT ENABLE
                                        ;;*************************************************************
005346  000004                          TST14:  SCOPE
005350  012737  000000  001544                  MOV     #0,$TMDAT               ;CLEAR STATUS

                                        ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
005366  012737  040000  001124                  MOV     #BIT14,$GDDAT           ;LOAD EXPECTED

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

                                        ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
005414  033737  001124  001126                  BIT     $GDDAT,$BDDAT
005422  001001                                  BNE     TST15           ;;BR IF SET
005424  104001                                  ERROR   1                       ;ERROR BIT 14 FAILED TO SET

                                        ;;*************************************************************
                                        ;*TEST 15       TEST INPUT DATA READY FLAG
                                        ;;*************************************************************
005426  000004                          TST15:  SCOPE
005430  012737  000200  001124                  MOV     #BIT7,$GDDAT            ;LOAD EXPECTED

                                        ;*      MOV     $GDDAT,@GRSTAT  ;/ PJT DATA FROM $GDDAT TO DEVICE REG GRSTAT

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

```
1030  005456  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1031  005464  001401                          BEQ     TST16           ;;BR IF EQUAL
1032  005466  104001                          ERROR   1               ;ERROR, BIT 7 FAILED TO SET
1033
1034                                  ;;*****************************************************************
1035                                  ;*TEST 16        TEST INPUT INTERRUPT ENABLE
1036                                  ;;*****************************************************************
1037  005470  000004                  TST16:  SCOPE
1038  005472  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR STATUS
1039
1040                                  ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1041  005510  012737  000100  001124          MOV     #BIT6,$GDDAT            ;LOAD EXPECTED
1042
1043                                  ;*      MOV     $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
1044
1045                                  ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
1046  005536  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1047  005544  001401                          BEQ     TST17           ;;BR IF EQUAL
1048  005546  104001                          ERROR   1               ;ERROR, BIT 6 FAILED TO SET
1049
1050
1051
1052                                  ;;*****************************************************************
1053                                  ;*TEST 17        TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
1054                                  ;;*****************************************************************
1055  005550  000004                  TST17:  SCOPE
1056  005552  032777  002000  173360          BIT     #BIT10,@SWR             ;TEST SWITCH BIT
1057  005560  001405                          BEQ     1$              ;BRANCH IF DOWN
1058  005562  112737  000036  001102          MOVB    #36,$TSTNM              ;LOAD NEW TEST NUMBER
1059  005570  000137  012250                  JMP     DRT21           ;BYPASS SOME TEST USING THE EXTERNAL CABLE
1060  005574                          1$:
1061  005574  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTFUT REGISTER
1062
1063                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1064  005612  012737  177777  001544          MOV     #-1,$TMDAT
1065
1066                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1067  005630  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
1068  005634  012737  000000  001544          MOV     #0,$TMDAT               ;LOAD THE OUTPUT
1069
1070                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1071
1072                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1073  005662  043737  001534  001126          BIC     ODDJMP,$BDDAT           ;MASK
1074  005670  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1075  005676  001401                          BEQ     TST20           ;;BR IF EQUAL
1076  005700  104002                          ERROR   2               ;ERROR, INPUT DID NOT EQUAL THE OUTPUT REG.
1077
1078                                  ;;*****************************************************************
1079                                  ;*TEST 20        TEST INPUT WITH #-1
1080                                  ;;*****************************************************************
1081  005702  000004                  TST20:  SCOPE
1082  005704  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1083
```

```
1084                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1085  005722  012737  177777  001544              MOV     #-1,$TMDAT
1086
1087                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1088  005740  012737  177777  001124              MOV     #-1,$GDDAT              ;LOAD EXPECTED
1089  005746  043737  001534  001124              BIC     ODDJMP,$GDDAT           ;MASK
1090
1091                                    ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1092
1093                                    ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1094  005774  043737  001534  001126              BIC     ODDJMP,$BDDAT           ;MASK
1095  006002  023737  001124  001126              CMP     $GDDAT,$BDDAT           ;COMPARE
1096  006010  001401                              BEQ     TST21           ;;BR IF EQUAL
1097  006012  104002                              ERROR   2                       ;ERROR, INPUT DID NOT EQUAL THE OUTPUT
1098
1099                                    ;;*****************************************************************
1100                                    ;*TEST 21        TEST INPUT WITH #52525
1101                                    ;;*****************************************************************
1102  006014  000004                    TST21:  SCOPE
1103  006016  012737  000000  001544              MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1104
1105                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1106  006034  012737  177777  001544              MOV     #-1,$TMDAT
1107
1108                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1109  006052  012737  052525  001124              MOV     #52525,$GDDAT           ;LOAD EXPECTED
1110  006060  043737  001534  001124              BIC     ODDJMP,$GDDAT           ;MASK
1111
1112                                    ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1113
1114                                    ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1115  006106  043737  001534  001126              BIC     ODDJMP,$BDDAT           ;MASK
1116  006114  023737  001124  001126              CMP     $GDDAT,$BDDAT           ;COMPARE
1117  006122  001401                              BEQ     TST22           ;;BR IF EQUAL
1118  006124  104002                              ERROR   2                       ;ERROR, INPUT DID NOT EQUAL OUTPUT
1119
1120                                    ;;*****************************************************************
1121                                    ;*TEST 22        TEST INPUT WITH #125252
1122                                    ;;*****************************************************************
1123  006126  000004                    TST22:  SCOPE
1124  006130  012737  000000  001544              MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1125
1126                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1127  006146  012737  177777  001544              MOV     #-1,$TMDAT
1128
1129                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1130  006164  012737  125252  001124              MOV     #125252,$GDDAT          ;LOAD EXPECTED
1131  006172  043737  001534  001124              BIC     ODDJMP,$GDDAT           ;MASK
1132
1133                                    ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1134
1135                                    ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1136  006220  043737  001534  001126              BIC     ODDJMP,$BDDAT           ;MASK
1137  006226  023737  001124  001126              CMP     $GDDAT,$BDDAT           ;COMPARE
```

```
1138  006234  001401                          BEQ     TST23           ;;BR IF EQUAL
1139  006236  104002                          ERROR   2                       ;ERROR, INPUT DID NOT EQUAL OUTPUT
1140                                  ;;**************************************************************
1141                                  ;*TEST 23       TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
1142                                  ;;**************************************************************
1143  006240  000004                  TST23:  SCOPE
1144  006242  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
1145  006250  005737  001534                  TST     ODDJMP                  ;TEST IF ANY ODD JUMPERS
1146  006254  001514                          BEQ     TST24           ;;BR IF NONE
1147  006256  012737  010000  001124          MOV     #BIT12,$GDDAT           ;LOAD TEST BIT
1148  006264  033737  001534  001124  1$:     BIT     ODDJMP,$GDDAT           ;TEST IF ODD JUMPER BIT
1149  006272  001502                          BEQ     2$                      ;BR IF NOT
1150  006274  033737  001510  001124          BIT     NOTLCH,$GDDAT           ;TEST IF LATCHING INPUT BIT
1151  006302  001076                          BNE     2$                      ;BR IF NOT
1152
1153                                  ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1154  006314  012737  177777  001544          MOV     #-1,$TMDAT              ;CLEAR INPUT
1155
1156                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1157
1158                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1159  006342  043737  001124  001544          BIC     $GDDAT,$TMDAT
1160
1161                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1162
1163                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1164  006370  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1165  006376  001401                          BEQ     3$                      ;BR IF EQUAL
1166  006400  104002                          ERROR   2                       ;ERROR, NEG. INPUT OR NEG. TRANSITION
1167                                                                          ; INPUT BIT FAILED TO sET INPUT REGISTER
1168
1169  006402  033737  001124  001536  3$:     BIT     $GDDAT,MINSIN           ;TEST IF NEG. INPUT BIT
1170  006410  001033                          BNE     2$                      ;BR IF
1171  006412  012737  177777  001544          MOV     #-1,$TMDAT              ;CLEAR INPUT
1172
1173                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1174
1175                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1176  006440  053737  001124  001544          BIS     $GDDAT,$TMDAT
1177
1178                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1179
1180                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1181  006466  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1182  006474  001401                          BEQ     2$                      ;BR IF EQUAL
1183  006476  104002                          ERROR   2                       ;ERROR, POSITIVE INPUT TRANSITION
1184                                                                          ;LOGIC FAILED TO SET INPUT REGISTER BIT
1185
1186  006500  006137  001124          2$:     ROL     $GDDAT                  ;CHANGE DATA PATTERN
1187  006504  103267                          BCC     1$                      ;BR IF MORE DATA
1188                                  ;;**************************************************************
1189                                  ;*TEST 24       FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
1190                                  ;;**************************************************************
1191  006506  000004                  TST24:  SCOPE
```

```
1192  006510  012737  006536  001110          MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
1193
1194  006516  012737  000001  001530          MOV     #BIT0,BRLEV2            ;LOAD EXPECTED
1195  006524  013737  001510  001532          MOV     NOTLCH,BRLEV3           ;GET NON-LATCH
1196  006532  005137  001532                  COM     BRLEV3                  ;COMPLEMENT
1197  006536  013737  001530  001124  2$:     MOV     BRLEV2,$GDDAT           ;LOAD GOOD
1198  006544  033737  001124  001534          BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
1199  006552  001055                          BNE     1$                      ;BYPASS IF ODD JUMPER
1200  006554  033737  001124  001510          BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCH
1201  006562  001451                          BEQ     1$                      ;BR IF LATCHING
1202
1203
1204                                  ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
1205
1206                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1207  006604  043737  001532  001126          BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
1208  006612  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1209  006620  001401                          BEQ     3$              ;;BR IF EQUAL
1210  006622  104002                          ERROR   2                       ;INPUT REGISTER IN ERROR
1211                                                                  ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
1212
1213                                  ;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
1214
1215  006624                          3$:
1216
1217                                  ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1218  006634  043737  001124  001544          BIC     $GDDAT,$TMDAT
1219
1220                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1221
1222                                  ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1223  006662  043737  001532  001126          BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
1224  006670  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
1225  006674  033737  001530  001126          BIT     BRLEV2,$BDDAT           ;TEST FOR BIT
1226  006702  001401                          BEQ     1$              ;;BR IF CLEARED
1227  006704  104002                          ERROR   2                       ;INPUT BIT LATCHED IN ERROR
1228                                                                  ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
1229
1230  006706  006337  001530          1$:     ASL     BRLEV2                  ;CHANGE PATTERN
1231  006712  001311                          BNE     2$                      ;BR UNTIL DONE
1232
1233                                  ;;*****************************************************************
1234                                  ;*TEST 25       FLOAT A 1 ACROSS LATCHING INPUT BITS
1235                                  ;;*****************************************************************
1236  006714  000004                  TST25:  SCOPE
1237  006716  012737  006766  001110          MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
1238  006724  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE
1239  006732  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REG
1240
1241                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1242  006750  012737  177777  001544          MOV     #-1,$TMDAT              ;CLEAR INPUT
1243
1244                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1245
```

```
1246  006766  033737  001124  001510  2$:     BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCHING
1247  006774  001030                          BNE     1$                      ;BR IF NON-LATCH
1248  006776  033737  001124  001534          BIT     $GDDAT,CDDJMP           ;TEST IF ODD JUMPER
1249  007004  001024                          BNE     1$                      ;BYPASS IF ODD JUMPER
1250
```

```
                                        ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
007016  012737  000000  001544                  MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER

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

                                        ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
007044  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
007052  001401                                  BEQ     1$              ;;BR IF EQUAL
007054  104002                                  ERROR   2                       ;INPUT REGISTER FAILED TO LATCH DATA

007056                                  1$:

                                        ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
007066  053737  001124  001544                  BIS     $GDDAT,$TMDAT

                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
007104  006337  001124                  2$:     ASL     $GDDAT                  ;CHANGE PATTERN
007110  001326                                  BNE     2$                      ;BR UNTIL DONE

                                        ;;*******************************************************************
                                        ;*TEST 26       FLOAT A 0 ACROSS LATCHING INPUT BITS
                                        ;*******************************************************************
007112  000004                          TST26:  SCOPE
007114  012737  007164  001110                  MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
007122  012737  000001  001532                  MOV     #BIT0,BRLEV3            ;LOAD EXPECTED
007130  012737  000000  001544                  MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER

                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
007146  012737  177777  001544                  MOV     #-1,$TMDAT

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

007164  033737  001532  001510           2$:    BIT     BRLEV3,NOTLCH           ;TEST FOR LATCHING
007172  001041                                  BNE     1$                      ;BR IF NOT
007174  033737  001124  001534                  BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
007202  001035                                  BNE     1$                      ;BYPASS IF ODD JUMPER BIT

007204  012737  177777  001124                  MOV     #-1,$GDDAT              ;LOAD
007212  043737  001510  001124                  BIC     NOTLCH,$GDDAT
007220  043737  001532  001124                  BIC     BRLEV3,$GDDAT           ;MAKE BRLEV3

                                        ;*      MOV     $GDDAT,@GRDIO   ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
007236  012737  000000  001544                  MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER

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


                                        ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
007264  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
007272  001401                                  BEQ     1$              ;;BR IF EQUAL
007274  104002                                  ERROR   2                       ;INPUT REGISTER FAILED TO LATCH DATA

007276                                  1$:
```

```
1305
1306                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1307  007306  053737  001532  001544            BIS     BRLEV3,$TMDAT
1308
1309                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1310  007324  006337  001532                    ASL     BRLEV3                  ;CHANGE PATTERN
1311  007330  001315                            BNE     2$                      ;BR UNTIL DONE
1312                                    ;;*****************************************************************
1313                                    ;*TEST 27        TEST FOR SLOW INPUT GATES WITH #125252
1314                                    ;;*****************************************************************
1315  007332  000004                    TST27:  SCOPE
1316
1317  007334  012737  125252  001124            MOV     #125252,$GDDAT          ;LOAD EXPECTED
1318  007342  043737  001510  001124            BIC     NOTLCH,$GDDAT           ;CONVERT
1319  007350  043737  001534  001124            BIC     ODDJMP,$GDDAT           ;MASK
1320  007356  013700  001124                    MOV     $GDDAT,R0               ;LOAD PATTERN
1321  007362  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1322
1323                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1324  007400  012737  177777  001544            MOV     #-1,$TMDAT
1325
1326                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1327
1328
1329                                    ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1330  007426  010037  001544                    MOV     R0,$TMDAT
1331
1332                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1333  007442  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1334
1335                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1336
1337                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1338  007470  013701  001544                    MOV     $TMDAT,R1
1339
1340                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1341  007504  005100                            COM     R0
1342
1343                                    ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1344  007516  010037  001544                    MOV     R0,$TMDAT
1345
1346                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1347  007532  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1348
1349                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1350
1351                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMdAT.
1352  007560  013701  001544                    MOV     $TMDAT,R1
1353
1354                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1355  007574  005100                            COM     R0
1356
1357                                    ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1358  007606  010037  001544                    MOV     R0,$TMDAT
```

```
1359
1360                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1361  007622  012737  000000  001544                MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1362
1363                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1364
1365                                        ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1366  007650  013701  001544                        MOV     $TMDAT,R1
1367
1368                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1369  007664  005100                                COM     R0
1370
1371                                        ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1372  007676  010037  001544                        MOV     R0,$TMDAT
1373
1374                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1375  007712  012737  000000  001544                MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1376
1377                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1378
1379                                        ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1380  007740  013701  001544                        MOV     $TMDAT,R1
1381
1382                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1383  007754  005100                                COM     R0
1384
1385                                        ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1386  007766  010037  001544                        MOV     R0,$TMDAT
1387
1388                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1389  010002  012737  000000  001544                MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1390
1391                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1392
1393                                        ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1394  010030  013701  001544                        MOV     $TMDAT,R1
1395
1396                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1397  010044  005100                                COM     R0
1398
1399                                        ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1400  010056  010037  001544                        MOV     R0,$TMDAT
1401
1402                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1403  010072  012737  000000  001544                MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1404
1405                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1406
1407                                        ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1408  010120  013701  001544                        MOV     $TMDAT,R1
1409
1410                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1411  010134  005100                                COM     R0
1412
```

```
1413                                    ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1414  010146  010037  001544                    MOV     R0,$TMDAT
1415
1416                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1417  010162  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1418
1419                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1420
1421                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1422  010210  013701  001544                    MOV     $TMDAT,R1
1423
1424                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1425  010224  005100                            COM     R0
1426
1427                                    ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1428  010236  010037  001544                    MOV     R0,$TMDAT
1429
1430                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1431  010252  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1432
1433                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1434
1435                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMdAT.
1436  010300  013701  001544                    MOV     $TMDAT,R1
1437
1438                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1439  010314  005100                            COM     R0
1440
1441                                    ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1442  010326  010037  001544                    MOV     R0,$TMDAT
1443
1444                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1445  010342  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1446
1447                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1448
1449                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1450  010370  013701  001544                    MOV     $TMDAT,R1
1451
1452                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMdAT TO DEVICE REG GRDAI
1453  010404  005100                            COM     R0
1454
1455                                    ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1456  010416  010037  001544                    MOV     R0,$TMDAT
1457
1458                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1459  010432  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1460
1461                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1462
1463                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
14E   010460  013701  001544                    MOV     $TMDAT,R1
14E
146E                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
```

```
1467  010474  005100                          COM     R0
1468
1469                                  ;*      MOV     R0,@GRDIO       ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
1470  010506  010037  001544                  MOV     R0,$TMDAT
1471
1472                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1473  010522  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1474
1475                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1476
1477                                  ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1478  010550  013701  001544                  MOV     $TMDAT,R1
1479
1480                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1481  010564  005100                          COM     R0
1482
1483  010566  010137  001126                  MOV     R1,$BDDAT               ;LOAD READ
1484  010572  000240                          NOP
1485  010574  000240                          NOP
1486  010576  000240                          NOP
1487  010600  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
1488  010606  001401                          BEQ     TST30           ;;BR IF EQUAL
1489  010610  104002                          ERROR   2                       ;INPUT GATE  LOW
1490
1491                                  ;;*****************************************************************
1492                                  ;*TEST 30       TEST FOR SLOW INPUT GATES WITH #52525
1493                                  ;;*****************************************************************
1494  010612  000004                  TST30:  SCOPE
1495
1496  010614  012737  052525  001124          MOV     #52525,$GDDAT           ;SETUP EXPECTED
1497  010622  043737  001534  001124          BIC     ODDJMP,$GDDAT           ;MASK ODD JUMPER BITS
1498  010630  043737  001510  001124          BIC     NOTLCH,$GDDAT           ;CONVERT
1499  010636  013700  001124                  MOV     $GDDAT,R0
1500  010642  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1501
1502                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1503  010660  012737  177777  001544          MOV     #-1,$TMDAT
1504
1505                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1506
1507  010676  010037  001544                  MOV     R0,$TMDAT
1508
1509                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1510  010712  012737  000000  001544          MO'     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1511
1512                                  ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1513
1514                                  ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMdAT.
1515  010740  013701  001544                  MOV     $TMDAT,R1
1516
1517                                  ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1518  010754  005100                          COM     R0
1519  010756  010037  001544                  MOV     R0,$TMDAT
1520
```

```
1521                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1522    010772  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1523
1524                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1525
1526                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1527    011020  013701  001544                  MOV     $TMDAT,R1
1528
1529                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1530    011034  005100                          COM     R0
1531    011036  010037  001544                  MOV     R0,$TMDAT
1532
1533                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1534    011052  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1535
1536                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1537
1538                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1539    011100  013701  001544                  MOV     $TMDAT,R1
1540
1541                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1542    011114  005100                          COM     R0
1543    011116  010037  001544                  MOV     R0,$TMDAT
1544
1545                                    ;*      MOV     $TMdAT,@GRDIO   ;/ PUT DATA FROM $TMdAT TO DEVICE REG GRDIO
1546    011132  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1547
1548                                    ;*      MOV     $TMdAT,@GRDIO   ;/ PUT DATA FROM $TMdAT TO DEViCE REG GRDIO
1549
1550                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEViCE REG GRDAI,PUT DATA IN $TMDAT.
1551    011160  013701  001544                  MOV     $TMDAT,R1
1552
1553                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1554    011174  005100                          COM     R0
1555    011176  010037  001544                  MOV     R0,$TMDAT
1556
1557                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1558    011212  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1559
1560                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1561
1562                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1563    011240  013701  001544                  MOV     $TMDAT,R1
1564
1565                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1566    011254  005100                          COM     R0
1567    011256  010037  001544                  MOV     R0,$TMDAT
1568
1569                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1570    011272  012737  000000  001544          MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1571
1572                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1573
1574                                    ;*      MOV     @GRDAI,$TMdAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
```

```
1575  011320  013701  001544                    MOV     $TMDAT,R1
1576
1577                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1578  011334  005100                            COM     R0
1579  011336  010037  001544                    MOV     R0,$TMDAT
1580
1581                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1582  011352  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1583
1584                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1585
1586                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1587  011400  013701  001544                    MOV     $TMDAT,R1
1588
1589                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1590  011414  005100                            COM     R0
1591  011416  010037  001544                    MOV     R0,$TMDAT
1592
1593                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1594  011432  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1595
1596                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1597
1598                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMdAT.
1599  011460  013701  001544                    MOV     $TMDAT,R1
1600
1601                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1602  011474  005100                            COM     R0
1603  011476  010037  001544                    MOV     R0,$TMDAT
1604
1605                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1606  011512  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1607
1608                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1609
1610                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1611  011540  013701  001544                    MOV     $TMDAT,R1
1612
1613                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1614  011554  005100                            COM     R0
1615  011556  010037  001544                    MOV     R0,$TMDAT
1616
1617                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEViCE REG GRDIO
1618  011572  012737  000000  001544            MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1619
1620                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEViCE REG GRDIO
1621
1622                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1623  011620  013701  001544                    MOV     $TMDAT,R1
1624
1625                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1626  011634  005100                            COM     R0
1627  011636  010037  001544                    MOV     R0,$TMDAT
1628
```

```
1629                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1630  011652  012737  000000  001544              MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1631
1632                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1633
1634                                    ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
1635  011700  013701  001544              MOV     $TMDAT,R1
1636
1637                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1638  011714  005100                      COM     R0
1639
1640  011716  010137  001126              MOV     R1,$BDDAT               ;LOAD VALUE READ
1641  011722  000240                      NOP
1642  011724  000240                      NOP
1643  011726  000240                      NOP
1644  011730  023737  001124  001126      CMP     $GDDAT,$BDDAT           ;COMPARE
1645  011736  001401                      BEQ     TST31           ;;BR IF EQUAL
1646  011740  104002                      ERROR   2
1647                                    ;;*****************************************************************
1648                                    ;*TEST 31        TEST THAT RESET CLEARS INPUT REGISTER BITS
1649                                    ;;*****************************************************************
1650  011742  000004              TST31:  SCOPE
1651  011744  012737  000002  001160      MOV     #2,$TIMES       ;;DO 2 ITERATIONS
1652  011752  012737  000000  001544      MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1653
1654                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1655  011770  012737  177777  001544      MOV     #-1,$TMDAT
1656
1657                                    ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1658  012006  012737  177777  001544      MOV     #-1,$TMDAT              ;LOAD OUTPUT
1659
1660                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1661  012024  005037  001124              CLR     $GDDAT                  ;LOAD EXPECTED
1662  012030  004737  024354              JSR     PC,$RESET
1663
1664                                    ;*      MOV     @GRDAI,$BDDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
1665  012044  043737  001534  001126      BIC     ODDJMP,$BDDAT           ;MASK ODD JUMPERS
1666  012052  005737  001126              TST     $BDDAT
1667  012056  001401                      BEQ     TST32           ;;BR IF ALL BITS CLEARED
1668  012060  104002                      ERROR   2                       ;INPUT REG. FAILED TO CLEAR UPON RESET INST.
1669
1670                                    ;;*****************************************************************
1671                                    ;*TEST 32        TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
1672                                    ;;*****************************************************************
1673  012062  000004              TST32:  SCOPE
1674  012064  012737  000200  001124      MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1675  012072  012737  000000  001544      MOV     #0,$TMDAT
1676
1677                                    ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1678
1679                                    ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1680  012120  022727  000000  000000      CMP     #0,#0                   ;DELAY
1681
1682                                    ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
```

```
                                        CMP     $GDDAT,$BDDAT           ;COMPARE
                                        BEQ     TST33           ;;BR IF EQUAL
                                        ERROR   1                       ;INPUT DATA READY FLAG FAILED TO SET

                                ;TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET

                                ;;*****************************************************************
                                ;*TEST 33       TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
                                ;;*****************************************************************
                                TST33:  SCOPE
                                        MOV     #BIT15,$GDDAT           ;LOAD EXPECTED
                                        MOV     #0,$TMDAT

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

                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        CMP     #0,#0

                                ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
                                        MOV     $TMDAT,R0

                                ;*      MOV     @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
                                        TST     $BDDAT
                                        BMI     TST34           ;;BR IF SET
                                        ERROR   1                       ;INPUT DATA READY FLAG FAILED TO SET


                                DRT21:
                                ;;*****************************************************************
                                ;*TEST 34       TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
                                ;;*****************************************************************
                                TST34:  SCOPE
                                        BIT     #BIT10,@SWR             ;TEST CABLE SWITCH
                                        BNE     TST35           ;;BYPASS IF NO I/O CABLE

                                        MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
                                        MOV     #BIT0,BRLEV3            ;LOAD INTERRUPT BIT
                                1$:     CLR     $BDDAT                  ;CLEAR BAD DATA
                                        MOV     #BIT7,$GDDAT            ;LOAD GOOD DATA

                                        BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
                                        BEQ     3$              ;;NO TRY NEXT BIT
                                        MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER

                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        MOV     #-1,$TMDAT

                                ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
                                        MOV     #0,$TMDAT               ;CLEAR STATUS

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

                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
                                        BIS     BRLEV3,$TMDAT
```

```
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
012430  043737  001532  001544          BIC     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
012456  053737  001532  001544          BIS     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
012474  005077  167000                  CLR     @GRSTAT                 ;CLEAR FLAG FROM DATA READY
                                        ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
                                                                        ;IF INTERRUPT INPUT SWITCH IS ON
                                        ;*      MOV     @GRSTAT,$TMDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
012520  105737  001544                  TSTB    $TMDAT
012524  100401                          BMI     2$                      ;;BR IF SET
012526  104010                          ERROR   10                      ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
                                                                        ;?? DID OPERATOR GIVE CORRECT
                                                                        ;INPUT INTERRUPT BITS ??

012530                                  2$:
                                        ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
012540  043737  001532  001544          BIC     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                        ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
012566  053737  001532  001544          BIS     BRLEV3,$TMDAT
                                        ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
012604  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
012610  005077  166664                  CLR     @GRSTAT                 ;CLEAR STATUS
012614  117737  166660  001126          MOVB    @GRSTAT,$BDDAT          ;READ STATUS
012622  100001                          BPL     3$                      ;;BR IF CLEARED
012624  104001                          ERROR   1                       ;INPUT READY FAILED TO CLEAR

012626  006337  001532          3$:     ASL     BRLEV3                  ;CHANGE BIT
012632  001221                          BNE     1$                      ;BR IF NOT DONe
                                        ;;*****************************************************************
                                        ;*TEST 35        TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
                                        ;;*****************************************************************
012634  000004                  TST35:  SCOPE
012636  032777  002000  166274          BIT     #BIT10,@SWR             ;TEST CABLE SWITCH
012644  001127                          BNE     TST36           ;;BYPASS IF NO I/O CABLE

012646  012737  012662  001110          MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
012654  012737  000001  001532          MOV     #BIT0,BRLEV3            ;LOAD NON-INTERRUPT BIT
012662  012737  000200  001126  1$:     MOV     #200,$BDDAT             ;LOAD BAD DATA
```

```
1791  012670  005037  001124                CLR     $GDDAT                  ;CLEAR GOOD DATA
1792
1793  012674  033737  001532  001512        BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
1794  012702  001105                        BNE     3$              ;;YES SKIP AND TRY NEXT BIT
1795  012704  012737  000000  001544        MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER
1796
1797                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1798  012722  012737  177777  001544        MOV     #-1,$TMDAT
1799
1800                                ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
1801  012740  012737  000000  001544        MOV     #0,$TMDAT               ;CLEAR STATUS
1802
1803                                ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
1804
1805                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1306  012766  053737  001532  001544        BIS     BRLEV3,$TMDAT
1807
1808                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1809  013004  043737  001532  001544        BIC     BRLEV3,$TMDAT           ;CLEAR OUTPUT
1810
1811                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1812
1813                                ;*      MOV     @GRDIO,$TMDAT   ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
1814  013032  053737  001532  001544        BIS     BRLEV3,$TMDAT
1815
1816                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1817
1818                                ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
1819  013060  012737  000000  001544        MOV     #0,$TMDAT               ;CLEAR FLAG FROM DATA READY
1820
1821                                ;*      MOV     $TMDAT,@        ;/ PUT DATA FROM $TMDAT TO DEVICE REG
1822                                                                        ;SHOULD REMAIN SET VIA DIRECT SET SIDE
1823
1824                                ;*      MOV     @GRSTAT,$TMDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
1825  013106  105737  001544                TSTB    $TMDAT
1826  013112  100001                        BPL     3$              ;;BR IF CLEAR
1827  013114  104011                        ERROR   11                      ;INPUT NON-INTERRUPT BIT SET INPUT READY
1828                                                                        ;?? DID OPERATOR GIVE CORRECT
1829                                                                        ;INPUT INTERRUPT BITS ??
1830
1831
1832  013116  006337  001532        3$:     ASL     BRLEV3                  ;CHANGE BIT
1833  013122  001257                        BNE     1$                      ;BR IF NOT DONE
1834                                ;;**************************************************************
1835                                ;*TEST 36        DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
1836                                ;;**************************************************************
1837  013124  000004                TST36:  SCOPE
1838  013126  012737  000001  001160        MOV     #1,$TIMES       ;;DO 1 ITERATION
1839  013134  005737  001464        BYPASS: TST     NBEXT                   ;TEST IF ANY
1840  013140  001415                        BEQ     1$                      ;BR IF NONE
1841  013142  032777  010000  165770        BIT     #SW12,@SWR              ;TEST BIT12 OF SWR
1842  013150  001024                        BNE     BYPAS1                  ;INHIBIT TESTING NEXT DR11-K
1843  013152  162737  000010  001466        SUB     #10,DRADD               ;UPDATE DEVICE ADDRESS
1844  013160  062737  000010  001470        ADD     #10,DRIV                ;UPDATE DEVICE VECTOR
```

```
1845  013166  005337  001464                    DEC     NBEXT                   ;ANOTHER ONE ?
1846  013172  000413                            BR      BYPAS1                  ;BR IF ANOTHER
1847  013174  013737  001452  001466  1$:       MOV     BASEBA,DRADD            ;RELOAD ADDRESS
1848  013202  013737  001454  001470            MOV     BASEIV,DRIV             ;RELOAD VECTOR
1849  013210  013737  001462  001464            MOV     NMBEXT,NBEXT            ;RELOAD NUMBER
1850  013216  000137  013232                    JMP     $EOP                    ;DONE
1851  013222  012700  177777          BYPAS1:   MOV     #-1,R0
1852  013226  000137  003360                    JMP     RBEG2                   ;TEST ANOTHER UNIT
1853
1854                                            .SBTTL  END OF PASS ROUTINE
1855
1856                                            ;;**************************************************************
1857                                            ;*INCREMENT THE PASS NUMBER ($PASS)
1858                                            ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
1859                                            ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
1860                                            ;*IF THERES A MONITOR GO TO IT
1861                                            ;*IF THERE ISN'T JUMP TO BYPAS1
1862
1863  013232                            $EOP:
1864  013232  000004                            SCOPE
1865  013234  005037  001102                    CLR     $TSTNM                  ;;ZERO THE TEST NUMBER
1866  013240  005037  001160                    CLR     $TIMES                  ;;ZERO THE NUMBER OF ITERATIONS
1867  013244  005237  001176                    INC     $PASS                   ;;INCREMENT THE PASS NUMBER
1868  013250  042737  100000  001176            BIC     #100000,$PASS           ;;DON'T ALLOW A NEG. NUMBER
1869  013256  005327                            DEC     (PC)+                   ;;LOOP?
1870  013260  000001                  $EOPCT:   .WORD   1
1871  013262  003022                            BGT     $DOAGN                  ;;YES
1872  013264  012737                            MOV     (PC)+,@(PC)+            ;;RESTORE COUNTER
1873  013266  000001                  $ENDCT:   .WORD   1
1874  013270  013260                            $EOPCT
1875  013272  104401  013337                    TYPE    ,$ENDMG                 ;;TYPE "END PASS #"
1876  013276  013746  001176                    MOV     $PASS,-(SP)             ;;SAVE $PASS FOR TYPEOUT
1877  013302  104405                            TYPDS                           ;;GO TYPE--DECIMAL ASCII WITH SIGN
1878  013304  104401  013334                    TYPE    ,$ENULL                 ;;TYPE A NULL CHARACTER
1879  013310  013700  000042          $GET42:   MOV     @#42,R0                 ;;GET MONITOR ADDRESS
1880  013314  001405                            BEQ     $DOAgN                  ;;BRANCH IF NO MONITOR
1881  013316  000005                            RESET                           ;;CLEAR THE WORLD
1882  013320  004710                  $ENDAD:   JSR     PC,(R0)                 ;;GO TO MONITOR
1883  013322  000240                            NOP                             ;;SAVE ROOM
1884  013324  000240                            NOP                             ;;FOR
1885  013326  000240                            NOP                             ;;ACT11
1886  013330                            $DOAgN:
1887  013330  000137                            JMP     @(PC)+                  ;;RETURN
1888  013332  013222                  $RTNAD:   .WORD   BYPAS1
1889  013334     377     377     000  $ENULL:   .BYTE   -1,-1,0                 ;;NULL CHARACTER STRING
1890  013337     015  042412  042116  $ENDMG:   .ASCIZ  <15><12>/END PASS #/
1891  013344  050040  051501  020123
1892  013352  000043
1893                                            .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1894
1895                                            ;;**************************************************************
1896                                            ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1897                                            ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1898                                            ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
```

```
                                        ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                        ;*REPLACED WITH SPACES.
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                        ;*      TYPDS                   ;;GO TO THE ROUTINE
                                        $TYPDS:
013354  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013356  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013360  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
013362  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
013364  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
013366  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
013372  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
013376  100004                          BPL     1$              ;;BR IF INPUT IS POS.
013400  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
013402  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
013410  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
013412  012703  013570                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
013416  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
013422  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
013424  016001  013560                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
013430  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
013432  002402                          BLT     4$              ;;BR IF DONE
013434  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
013436  000774                          BR      3$
013440  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
013442  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
013444  001002                          BNE     5$              ;;FALL THROUGH IF 0
013446  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
013450  100407                          BMI     7$              ;;BR IF YES
013452  106316                  5$:     ASLB    (SP)            ;;MSD?
013454  103003                          BCC     6$              ;;BR IF NO
013456  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
013464  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
013470  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
013474  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
013476  005720                          TST     (R0)+           ;;JUST INCREMENTING
013500  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
013504  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
013506  003002                          BGT     8$              ;;GO TO EXIT
013510  010502                          MOV     R5,R2           ;;GET THE LSD
013512  000764                          BR      6$              ;;GO CHANGE TO ASCII
013514  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
013516  100003                          BPL     9$              ;;BR IF NO
013520  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
013526  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
013530  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
013532  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
013534  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
013536  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
013540  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
013542  104401  013570                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
013546  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
```

```
                                                MOV     (SP)+,(SP)
                                                RTI                             ;;RETURN TO USER
                                        $DTBL:  10000.
                                                1000.
                                                100.
                                                10.
                                        $DBLK:  .BLKW   4

                                                .SBTTL  MISC. EXTERNAL LOGIC TEST
                                        EXTTST: MOV     #STACK,SP
                                                JSR     PC,SETADD               ;SET UP ADDRESS AND VECTOR
                                        2$:     MOV     #BIT14,EXTCNT           ;LOAD COUNT
                                        1$:
                                                MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER

                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                                MOV     #125252,$TMDAT          ;LOAD OUTPUT

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

                                        ;*      MOV     @GRDAI,EXTTMP   ;/READ DEVICE REG GRDAI,PUT DATA IN EXTTMP.

                                        ;*      MOV     EXTTMP,@GRDAI   ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
                                                MOV     #0,$TMDAT               ;CLEAR OUTPUT REGISTER

                                        ;*      MOV     $TMDAT,@GRDIO   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
                                                MOV     #52525,$TMDAT           ;LOAD OUTPUT

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

                                        ;*      MOV     @GRDAI,EXTTMP   ;/READ DEVICE REG GRDAI,PUT DATA IN EXTTMP.

                                        ;*      MOV     EXTTMP,@GRDAI   ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
                                                DEC     EXTCNT                  ;FINISHED COUNT
                                                BNE     1$
                                                BR      2$                      ;LOOP

                                        EXTTMP: 0
                                        EXTCNT: 0

                                        ;SUBROUTINE TO LOAD TRAP AND POWER FAIL VECTORS
                                        SETTRP: MOV     #$TRAP,@#TRAPVEC        ;LOAD TRAP VECTOR
                                                MOV     #340,@#TRAPVEC+2
                                                MOV     #$PWRDN,@#PWRVEC        ;LOAD POWER FAIL VECTOR
                                                MOV     #340,@#PWRVEC+2
                                                MOV     #1$,@#4                 ;SET UP FOR UNEXPECTED TRAP
                                                RTS     PC                      ;EXIT
                                        1$:     CMP     (SP)+,(SP)+
                                                TYPE    ,BUSTRP
                                                HALT
                                                JMP     BEGIN
                                                .SBTTL  COULTER INTERFACE TEST
                                        COULTR: MOV     #STACK,SP
                                                JSR     PC,SETTRP               ;LOAD TRAP AND POWER FAIL VECTORS
```

```
2007  014046  004737  003424                      JSR     PC,SETADD               ;SETUP BUS ADDRESS AND VECTOR
2008  014052  012737  000000  001544              MOV     #0,$TMDAT               ;CLEAR INPUT STATUR REG
2009
2010                                      ;*      MOV     $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
2011                                              ;LOAD INPUT VECTOR
2012  014070  104401  015710              1$:     TYPE,RUNMSG                     ;TELL OPERATOR TO "RUN SAMPLE"
2013  014074  012701  022522                      MOV     #BUFFER,R1              ;LOAD POINTER FOR RESULTS
2014                                              ;WAIT FOR OPERATOR
2015  014100                              4$:
2016
2017                                      ;*      MOV     @GRSTAT,$TMDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
2018  014110  105737  001544                      TSTB    $TMDAT
2019  014114  100371                              BPL     4$
2020  014116  012737  000106  013756              MOV     #70.,EXTTMP             ;LOAD # OF 1 MSEC DELAYS
2021  014124  013700  001460              6$:     MOV     CPU,R0  ;LOAD 1 MSEC. DELAY WEIGHT ;240 FOR 11/05  ;620 FOR 11/40
2022  014130  005300                      3$:     DEC     R0                      ;DELAY
2023  014132  001376                              BNE     3$
2024  014134  005337  013756                      DEC     EXTTMP                  ;FINISHED ALL MSEC. DELAY ?
2025  014140  001371                              BNE     6$                      ;BR IF NOT
2026
2027                                      ;*      MOV     @GRDAI,$TMDAT   ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
2028  014152  013700  001544                      MOV     $TMDAT,R0
2029  014156  010021                              MOV     R0,(R1)+                ;SAVE IN BUFFER
2030  014160  012737  177777  001544              MOV     #-1,$TMDAT              ;CLEAR INPUT
2031
2032                                      ;*      MOV     $TMDAT,@GRDAI   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
2033  014176  042700  007777                      BIC     #7777,R0                ;MASK
2034  014202  001336                              BNE     4$                      ;BR IF NOT LAST SAMPLE
2035  014204  010103                              MOV     R1,R3                   ;COPY R1
2036  014206  014300                      5$:     MOV     -(R3),R0                ;GET RESULTS
2037  014210  004737  014224                      JSR     PC,40$                  ;CONVERT AND TYPE
2038  014214  022703  022522                      CMP     #BUFFER,R3              ;FINISHED ?
2039  014220  001723                              BEQ     1$                      ;BR IF YES
2040  014222  000771                              BR      5$                      ;CONT. TYPING UNTIL DONE
2041                                      ;SUBROUTINE FOR THE COULTER TEST
2042  014224  012737  000055  015732      40$:    MOV     #55,MSGRUS+2            ;FIX ASCII MESSAGE
2043  014232  012702  015737                      MOV     #MSPNT1,R2              ;LOAD DEST. POINTER
2044  014236  012737  000004  013756              MOV     #4,EXTTMP               ;LOAD COUNT
2045  014244  000404                              BR      12$
2046  014246  006000                      10$:    ROR     R0
2047  014250  006000                              ROR     R0
2048  014252  006000                              ROR     R0
2049  014254  006000                              ROR     R0
2050  014256  010001                      12$:    MOV     R0,R1                   ;LOAD R1
2051  014260  042701  177760                      BIC     #177760,R1              ;MASK
2052  014264  052701  000060                      BIS     #60,R1                  ;MAKE DIGIT
2053  014270  110142                              MOVB    R1,-(R2)                ;SAVE DIGIT
2054  014272  005337  013756                      DEC     EXTTMP                  ;FINISHED ?
2055  014276  001363                              BNE     10$                     ;BR IF NOT
2056  014300  000337  015732                      SWAB    MSGRUS+2                ;ADJUST MESSAGE
2057  014304  104401  015730                      TYPE,   MSGRUS
2058  014310  000207                              RTS     PC                      ;EXIT
2059                                              .SBTTL  LOC-BOX LAMP AND SWITCH LOOP
2060  014312  012706  001100              LOCBOX: MOV     #STACK,SP
```

```
2061  014316  004737  013762                  JSR     PC,SETTRP
2062  014322  104401  015470                  TYPE    ,LOCHDR         ;TELL OPERATOR WHAT THIS IS
2063  014326  004537  014432                  JSR     R5,INADP        ;GET THE ADDRESSES
2064  014332  001472  014616                  LOC1,LOC1Y
2065  014336  000412                          BR      LOCA            ;BR IF DONE
2066  014340  004537  014432                  JSR     R5,INADR        ;GET NEXT ADDRESS
2067  014344  001474  014620                  LOC2,LOC2Y
2068  014350  000405                          BR      LOCA            ;BR IF DONE
2069  014352  004537  014432                  JSR     R5,INADR        ;GET NEXT ADDRESS
2070  014356  001476  014622                  LOC3,LOC3Y
2071  014362  000400                          BR      LOCA
2072
2073  014364  005737  014616          LOCA:   TST     LOC1Y           ;TEST IF SELECTED
2074  014370  001403                          BEQ     1$              ;BR IF NOT
2075  014372  004537  014544                  JSR     R5,LOUP1        ;TEST FOR SWITCHES AND LOAD LAMPS
2076  014376  001472                          LOC1
2077  014400  005737  014620          1$:     TST     LOC2Y           ;TEST IF SELECTED
2078  014404  001403                          BEQ     2$              ;BR IF NOT
2079  014406  004537  014544                  JSR     R5,LOUP1        ;LOAD NEXT SET
2080  014412  001474                          LOC2
2081  014414  005737  014622          2$:     TST     LOC3Y           ;TEST IF SELECTED
2082  014420  001403                          BEQ     3$
2083  014422  004537  014544                  JSR     R5,LOUP1        ;LOAD NEXT SET
2084  014426  001476                          LOC3
2085  014430  000755                  3$:     BR      LOCA            ;LOOP BACK
2086
2087  014432  013537  014536          INADR:  MOV     @(R5)+,10$      ;GET BUS ADDRESS
2088  014436  012537  014542                  MOV     (R5)+,12$       ;GET INDICATOR
2089  014442  005077  000074                  CLR     @12$            ;CLEAR FLAG
2090  014446  104401  015531          1$:     TYPE,   INADRH          ;ASK FOR INPUT
2091  014452  013746  014536                  MOV     10$,-(SP)       ;TYPE CURRENT ADDRESS
2092  014456  104402                          TYPOC
2093  014460  104401  015574                  TYPE,   INDADR          ;ADD YES NO
2094  014464  104411                          RDLIN                   ;READ CHAR. AND ECHO
2095  014466  000240                          NOP
2096  014470  000240                          NOP
2097  014472  013637  014540                  MOV     @(SP)+,11$      ;GET THE ANSWER
2098  014476  042737  177640  014540          BIC     #177640,11$     ;MASK OFF BITS
2099  014504  001413                          BEQ     3$              ;BR IF CR
2100  014506  022737  000116  014540          CMP     #'N,11$         ;TEST IF NO
2101  014514  001406                          BEQ     2$              ;BR IF NO
2102  014516  022737  000131  014540          CMP     #'Y,11$         ;TEST IF YES
2103  014524  001350                          BNE     1$              ;BR IF OTHER
2104  014526  005277  000010                  INC     @12$            ;SET YES FLAG
2105  014532  005725                  2$:     TST     (R5)+
2106  014534  000205                  3$:     RTS     R5              ;EXIT
2107  014536  000000                  10$:    0
2108  014540  000000                  11$:    0
2109  014542  000000                  12$:    0
2110
2111                                  ;SUBROUTINE TO LOAD THE LAMPS FROM THE LOC BOX SWITCHES
2112  014544  013537  014612          LOUP1:  MOV     @(R5)+,40$      ;GET ARG.
2113  014550  001001                          BNE     1$              ;BR IF YES
2114  014552  000205                          RTS     R5              ;EXIT
```

```
2115  014554  012702  014612          1$:     MOV     40$,R2
2116  014560  062702  000002                  ADD     #2,R2
2117  014564  010203                          MOV     R2,R3
2118  014566  062703  000002                  ADD     #2,R3
2119  014572  011200                          MOV     (R2),R0
2120  014574  011301                          MOV     (R3),R1
2121  014576  040100                          BIC     R1,R0                   ;MASK OFF BITS
2122  014600  041213                          BIC     (R2),(R3)               ;LOAD OUTPUT BITS
2123  014602  060013                          ADD     R0,(R3)         ;LOAD
2124  014604  012712  177777                  MOV     #-1,(R2)                ;CLEAR INPUT
2125  014610  000205                          RTS     R5                      ;EXIT
2126
2127  014612  000000          40$:    0
2128  014614  000000          41$:    0
2129  014616  000000          LOC1Y:  0               ;YES/NO ANSWER TO IOC BOX #1
2130  014620  000000          LOC2Y:  0
2131  014622  000000          LOC3Y:  0
2132
2133                                  .SBTTL  XLO1 ADJUSTMENT ROUTINE
2134  014624  012706  001100  XLO1AD: MOV     #STACK,SP
2135  014630  004737  013762          JSR     PC,SETTRP
2136  014634  104401  015612          TYPE    ,XLOHDR         ;TELL OPERATOR
2137  014640  004537  014432          JSR     R5,INADR                ;GET BUS ADDRESS
2138  014644  001472  014616          LOC1,LOC1Y
2139  014650  000412                  BR      XLO10
2140  014652  004537  014432          JSR     R5,INADR                ;
2141  014656  001474  014620          LOC2,LOC2Y
2142  014662  000405                  BR      XLO10
2143  014664  004537  014432          JSR     R5,INADR
2144  014670  001476  014622          LOC3,LOC3Y
2145  014674  000400                  BR      XLO10
2146
2147  014676  005737  014616  XLO10:  TST     LOC1Y                   ;TEST IF SELECTED
2148  014702  001431                  BEQ     1$                      ;BR IF NOT SELECTED
2149  014704  004537  015152          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2150  014710     000     000          .BYTE   0,0                     ;LOAD CH# AND SHIFT COUNT
2151  014712  000007                  7                               ;LOAD DATA MASK
2152  014714  001472                  LOC1                            ;DR11K BUS ADDRESS
2153  014716  004537  015152          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2154  014722     001     003          .BYTE   1,3                     ;LOAD CH# AND SHIFT COUNT
2155  014724  000070                  70                              ;LOAD DATA MASK
2156  014726  001472                  LOC1                            ;DR11K BUS ADDRESS
2157  014730  004537  015152          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2158  014734     002     006          .BYTE   2,6                     ;LOAD CH# AND SHIFT COUNT
2159  014736  000700                  700                             ;LOAD DATA MASK
2160  014740  001472                  LOC1                            ;DR11K BUS ADDRESS
2161  014742  004537  015152          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2162  014746     003     011          .BYTE   3,11                    ;LOAD CH# AND SHIFT COUNT
2163  014750  007000                  7000                            ;LOAD DATA MASK
2164  014752  001472                  LOC1                            ;DR11K BUS ADDRESS
2165  014754  004537  015152          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
2166  014760     004     014          .BYTE   4,14                    ;LOAD CH# AND SHIFT COUNT
2167  014762  070000                  70000                           ;LOAD DATA MASK
2168  014764  001472                  LOC1                            ;DR11K BUS ADDRESS
```

```
 2169  014766  005737  014620          1$:     TST     LOC2Y           ;TEST IF SELECTED
 2170  014772  001431                          BEQ     2$              ;BR IF NOT
 2171  014774  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2172  015000     005     000                  .BYTE   5,0             ;LOAD CH# AND SHIFT COUNT
 2173  015002  000007                          7                       ;LOAD DATA MASK
 2174  015004  001474                          LOC2                    ;DR11K BUS ADDRESS
 2175  015006  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2176  015012     006     003                  .BYTE   6,3             ;LOAD CH# AND SHIFT COUNT
 2177  015014  000070                          70                      ;LOAD DATA MASK
 2178  015016  001474                          LOC2                    ;DR11K BUS ADDRESS
 2179  015020  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2180  015024     007     006                  .BYTE   7,6             ;LOAD CH# AND SHIFT COUNT
 2181  015026  000700                          700                     ;LOAD DATA MASK
 2182  015030  001474                          LOC2                    ;DR11K BUS ADDRESS
 2183  015032  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2184  015036     010     011                  .BYTE   10,11           ;LOAD CH# AND SHIFT COUNT
 2185  015040  007000                          7000                    ;LOAD DATA MASK
 2186  015042  001474                          LOC2                    ;DR11K BUS ADDRESS
 2187  015044  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2188  015050     011     014                  .BYTE   11,14           ;LOAD CH# AND SHIFT COUNT
 2189  015052  070000                          70000                   ;LOAD DATA MASK
 2190  015054  001474                          LOC2                    ;DR11K BUS ADDRESS
 2191  015056  005737  014622          2$:     TST     LOC3Y           ;TEST IF SELECTED
 2192  015062  001431                          BEQ     3$              ;BR IF NOT
 2193  015064  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2194  015070     012     000                  .BYTE   12,0            ;LOAD CH# AND SHIFT COUNT
 2195  015072  000007                          7                       ;LOAD DATA MASK
 2196  015074  001476                          LOC3                    ;DR11K BUS ADDRESS
 2197  015076  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2198  015102     013     003                  .BYTE   13,3            ;LOAD CH# AND SHIFT COUNT
 2199  015104  000070                          70                      ;LOAD DATA MASK
 2200  015106  001476                          LOC3                    ;DR11K BUS ADDRESS
 2201  015110  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2202  015114     014     006                  .BYTE   14,6            ;L AD CH# AND SHIFT COUNT
 2203  015116  000700                          700                     ;LOAD DATA MASK
 2204  015120  001476                          LOC3                    ;DR11K BUS ADDRESS
 2205  015122  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2206  015126     015     011                  .BYTE   15,11           ;LOAD CH# AND SHIFT COUNT
 2207  015130  007000                          7000                    ;LOAD DATA MASK
 2208  015132  001476                          LOC3                    ;DR11K BUS ADDRESS
 2209  015134  004537  015152                  JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 2210  015140     016     014                  .BYTE   16,14           ;LOAD CH# AND SHIFT COUNT
 2211  015142  070000                          70000                   ;LOAD DATA MASK
 2212  015144  001476                          LOC3                    ;DR11K BUS ADDRESS
 2213  015146  000137  014676          3$:     JMP     XLO10
 2214                                  ;ANALOG CONVERSION AND LOC BOX DRIVER
 2215  015152                          XLOADJ:
 2216
 2217                                  ;*      MOV     @ADCS1,$TMDAT   ;/READ DEVICE REG ADCS1,PUT DATA IN $TMDAT.
 2218  015162  112537  001544                  MOVB    (R5)+,$TMDAT
 2219
 2220                                  ;*      MOV     $TMDAT,@ADCS1   ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS1
 2221
 2222                                  ;*      MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
```

```
2223  015206  052737  020000  001544          BIS     #BIT13,$TMDAT
2224
2225                                  ;*        MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2226  015224  112537  015454                  MOVB    (R5)+,40$               ;GET SHIFT COUNT
2227  015230  012537  015460                  MOV     (R5)+,42$               ;GET DATA MASK
2228  015234  013537  015456                  MOV     @(R5)+,41$              ;GET DR11K BUS ADDRESS
2229  015240  001504                          BEQ     70$             ;BR IF NONE
2230
2231                                  ;*        MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2232  015252  105237  001544                  INCB    $TMDAT
2233
2234                                  ;*        MOV     $TMDAT,@ADCS    ;/ PUT DATA FROM $TMDAT TO DEVICE REG ADCS
2235  015266  005001                          CLR     R1
2236  015270  062737  000004  015456          ADD     #4,41$          ;GET LAMP ADDRESS
2237  015276                          1$:
2238
2239                                  ;*        MOV     @ADCS,$TMDAT    ;/READ DEVICE REG ADCS,PUT DATA IN $TMDAT.
2240  015306  105737  001544                  TSTB    $TMDAT
2241  015312  100371                          BPL     1$
2242
2243                                  ;*        MOV     @ADBR,$TMDAT    ;/READ DEVICE REG ADBR,PUT DATA IN $TMDAT.
2244  015324  013700  001544                  MOV     $TMDAT,R0
2245  015330  162700  000004                  SUB     #4,R0                   ;TEST IF AT UPPER END
2246  015334  100432                          BMI     7$                      ;BR IF YES
2247  015336  162700  000014                  SUB     #14,R0                  ;TEST IF NEAR UPPER END
2248  015342  100425                          BMI     6$                      ;BR IF YES
2249  015344  162700  000060                  SUB     #60,R0                  ;TEST IF GETTING NEAR UPPER END
2250  015350  100420                          BMI     5$                      ;BR IF YES
2251  015352  162700  001530                  SUB     #1530,R0                ;TEST IF GETTING NEAR LOWER END
2252  015356  100407                          BMI     2$                      ;BR IF YES
2253  015360  162700  000060                  SUB     #60,R0                  ;TEST IF NEAR LOWER END
2254  015364  100406                          BMI     3$                      ;BR IF YES
2255  015366  162700  000014                  SUB     #14,R0                  ;TEST IF AT LOWER END
2256  015372  100405                          BMI     4$                      ;BR IF YES
2257  015374  000412                          BR      7$                      ;BR IF AT STOP
2258  015376  012701  000004          2$:     MOV     #4,R1                   ;LOAD #
2259  015402  062701  000002          3$:     ADD     #2,R1
2260  015406  005201                  4$:     INC     R1
2261  015410  000404                          BR      7$                      ;BR AND LOAD LAMP'S
2262  015412  012701  000002          5$:     MOV     #2,R1                   ;LOAD #
2263  015416  062701  000004          6$:     ADD     #4,R1
2264  015422  013700  015454          7$:     MOV     40$,R0                  ;LOAD SHIFT COUNT
2265  015426  006301                  10$:    ASL     R1                      ;SHUFFEL THE DATA
2266  015430  005300                          DEC     R0
2267  015432  100375                          BPL     10$                     ;BR IF NOT DONE
2268  015434  006201                          ASR     R1                      ;RE ADJUST
2269  015436  043777  015460  000012          BIC     42$,@41$                ;CLEAR OLD DATA IN LAMPS
2270  015444  050177  000006                  BIS     R1,@41$                 ;LOAD THE LAMPS
2271  015450  005001                          CLR     R1
2272  015452  000205                  70$:    RTS     R5                      ;EXIT
2273
2274  015454  000000                  40$:    0
2275  015456  000000                  41$:    0
2276  015460  000000                  42$:    0
```

```
2277  015462  170400                          ADCS:   170400
2278  015464  170401                          ADCS1:  170401
2279  015466  170402                          ADBR:   170402
2280  015470  005015  047514  026503  LOCHDR: .ASCIZ  <15><12>/LOC-BOX SWITCH AND LAMP LOOP/<15><12>
2281  015476  047502  020130  053523
2282  015504  052111  044103  040440
2283  015512  042116  046040  046501
2284  015520  020120  047514  050117
2285  015526  005015    000
2286  015531     015  052412  042523  INADRH: .ASCIZ  <15><12>/USE LOC-BOX <DR11K AT BUS ADDR. /
2287  015536  046040  041517  041055
2288  015544  054117  036040  051104
2289  015552  030461  020113  052101
2290  015560  041040  051525  040440
2291  015566  042104  027122  000040
2292  015574  037040  020040  020131  INDADR: .ASCIZ  / >  Y OR N ? /
2293  015602  051117  047040  037440
2294  015610  000040
2295  015612  005015  046130  030460  XLOHDR: .ASCIZ  <15><12>/XLO1 POT ADJUSTMENT LOOP/<15><12>
2296  015620  050040  052117  040440
2297  015626  045104  051525  046524
2298  015634  047105  020124  047514
2299  015642  050117  005015    000
2300  015647     015  041012  051525  BUSTRP: .ASCIZ  <15><12>/BUS TIME-OUT ON SELECTED DR11K/
2301  015654  052040  046511  026505
2302  015662  052517  020124  047117
2303  015670  051440  046105  041505
2304  015676  042524  020104  051104
2305  015704  030461  000113
2306  015710  005015  051012  047125  RUNMSG: .ASCIZ  <15><12><12>/RUN SAMPLE/<15><12>
2307  015716  051440  046501  046120
2308  015724  006505  000012
2309  015730  005015  047055  047116  MSGRUS: .ASCII  <15><12>/-NNNN/
2310  015736     116
2311  015737     000                  MSPNT1: .BYTE   0
2312                                          .EVEN
2313
2314
2315  015740  042523  020124  053523  SWNLB:  .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING INPUT BITS/
2316  015746  052111  044103  051040
2317  015754  043505  051511  042524
2318  015762  020122  044502  051524
2319  015770  042440  052521  046101
2320  015776  052040  020117  044124
2321  016004  020105  047516  026516
2322  016012  040514  041524  044510
2323  016020  043516  044440  050116
2324  016026  052125  041040  052111
2325  016034  000123
2326  016036  042523  020124  053523  SWINTB: .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING INPUT BITS/
2327  016044  052111  044103  051040
2328  016052  043505  051511  042524
2329  016060  020122  044502  051524
2330  016066  042440  052521  046101
```

```
2331  016074  052040  020117  044124
2332  016102  020105  047111  042524
2333  016110  051122  050125  044524
2334  016116  043516  044440  050116
2335  016124  052125  041040  052111
2336  016132  000123
2337  016134  042523  020124  053523  SWPOSB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO POSITIVE INPUT BITS/
2338  016142  052111  044103  051040
2339  016150  043505  051511  042524
2340  016156  020122  044502  051524
2341  016164  030440  026465  031061
2342  016172  042440  052521  046101
2343  016200  052040  020117  047520
2344  016206  044523  044524  042526
2345  016214  044440  050116  052125
2346  016222  041040  052111  000123
2347  016230  042523  020124  053523  SWTRAB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO TRANSITION INPUT BITS/
2348  016236  052111  044103  051040
2349  016244  043505  051511  042524
2350  016252  020122  044502  051524
2351  016260  030440  026465  031061
2352  016266  042440  052521  046101
2353  016274  052040  020117  051124
2354  016302  047101  044523  044524
2355  016310  047117  044440  050116
2356  016316  052125  041040  052111
2357  016324  000123
2358  016326  042523  020124  053523  SWDPOB: .ASCIZ  /SET SWITCH REGISTER WITH THE DESIRED PROGRAM OPTIONS/
2359  016334  052111  044103  051040
2360  016342  043505  051511  042524
2361  016350  020122  044527  044124
2362  016356  052040  042510  042040
2363  016364  051505  051111  042105
2364  016372  050040  047522  051107
2365  016400  046501  047440  052120
2366  016406  047511  051516     000
2367  016413     015  042012  050105  DEPCNT: .ASCIZ  <15><12>/DEPRESS CONT./<15><12>
2368  016420  042522  051523  041440
2369  016426  047117  027124  005015
2370  016434     000
2371  016435     123  040524  052524  EM1:    .ASCIZ  /STATUS REGISTER IN ERROR/
2372  016442  020123  042522  044507
2373  016450  052123  051105  044440
2374  016456  020116  051105  047522
2375  016464  000122
2376  016466  047111  052520  020124  EM2:    .ASCIZ  /INPUT REGISTER IN ERROR/
2377  016474  042522  044507  052123
2378  016502  051105  044440  020116
2379  016510  051105  047522  000122
2380  016516  052517  050124  052125  EM3:    .ASCIZ  /OUTPUT REGISTER IN ERROR/
2381  016524  051040  043505  051511
2382  016532  042524  020122  047111
2383  016540  042440  051122  051117
2384  016546     000
```

```
2385  016547     111  050116  052125  EM4:    .ASCIZ  /INPUT FAILED TO INTERRUPT/
2386  016554  043040  044501  042514
2387  016562  020104  047524  044440
2388  016570  052116  051105  052522
2389  016576  052120     000
2390  016601     117  052125  052520  EM5:    .ASCIZ  /OUTPUT FAILED TO INTERRUPT/
2391  016606  020124  040506  046111
2392  016614  042105  052040  020117
2393  016622  047111  042524  051122
2394  016630  050125  000124
2395  016634  047125  054105  042520  EM6:    .ASCIZ  /UNEXPECTED INTERRUPT/
2396  016642  052103  042105  044440
2397  016650  052116  051105  052522
2398  016656  052120     000
2399  016661     117  042520  040522  EM7:    .ASCIZ  /OPERATOR INTERVENTION ERROR/
2400  016666  047524  020122  047111
2401  016674  042524  053122  047105
2402  016702  044524  047117  042440
2403  016710  051122  051117     000
2404  016715     111  052116  051105  EM10:   .ASCIZ  /INTERRUPT INPUT BIT FAILED TO sET INPUT READY FLAG/
2405  016722  052522  052120  044440
2406  016730  050116  052125  041040
2407  016736  052111  043040  044501
2408  016744  042514  020104  047524
2409  016752  051440  052105  04[illegible]
2410  016760  050116  052125  05[illegible]
2411  016766  040505  054504  043040
2412  016774  040514  000107
2413  017000  047516  026516  047111  EM11:   .ASCIZ  /NON-INTERRUPT INPUT BIT SET INPUT READY FLAG/
2414  017006  042524  051122  050125
2415  017014  020124  047111  052520
2416  017022  020124  044502  020124
2417  017030  042523  020124  047111
2418  017036  052520  020124  042522
2419  017044  042101  020131  046106
2420  017052  043501     000
2421
2422  017055     105  051122  041520  DH1:    .ASCIZ  /ERRPC   DRADD  TSTNUM   STATUS  EXPECTED/
2423  017062  020040  042040  040522
2424  017070  042104  052011  052123
2425  017076  052516  020115  020040
2426  017104  052123  052101  051525
2427  017112  020040  054105  042520
2428  017120  052103  042105     000
2429  017125     105  051122  041520  DH2:    .ASCIZ  /ERRPC   DRADD  TSTNUM   INPUT   EXPECTED/
2430  017132  020040  042040  040522
2431  017140  042104  052011  052123
2432  017146  052516  004515  047111
2433  017154  052520  020124  020040
2434  017162  054105  042520  052103
2435  017170  042105     000
2436  017173     105  051122  041520  DH3:    .ASCIZ  /ERRPC   DRADD  TSTNUM   OUTPUT  EXPECTED/
2437  017200  020040  042040  040522
2438  017206  042104  052011  052123
```

```
2439  017214  052516  004515  052517
2440  017222  050124  052125  020040
2441  017230  054105  042520  052103
2442  017236  042105     000
2443  017241     105  051122  041520  DH4:    .ASCIZ  /ERRPC   DRADD  TSTNUM/
2444  017246  020040  042040  040522
2445  017254  042104  052011  052123
2446  017262  052516  000115
2447  017266  051105  050122  020103  DH10:   .ASCIZ  /ERRPC   DRADD  TSTNUM  STATUS  EXPECT  INPUT BIT/
2448  017274  020040  051104  042101
2449  017302  004504  051524  047124
2450  017310  046525  051411  040524
2451  017316  052524  020123  042440
2452  017324  050130  041505  020124
2453  017332  044440  050116  052125
2454  017340  041040  052111     000
2455          017346                  .EVEN
2456  017346  001116  001466  017426  DT1:    $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,0
2457  017354  001126  001124  000000
2458  017362  001116  001466  017426  DT4:    $ERRPC,DRADD,TSTNUM,0
2459  017370  000000
2460  017372  001116  001466  017426  DT10:   $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,ORLEV3,0
2461  017400  001126  001124  001532
2462  017406  000000
2463  017410  000000  000000  000000  DF0:    0,0,0,0,0,0,0
2464  017416  000000  000000  000000
2465  017424  000000
2466  017426  000000                  TSTNUM: 0
2467
2468                                  .SBTTL  SCOPE HANDLER ROUTINE
2469
2470                                  ;;**********;**********
2471                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2472                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2473                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2474                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2475                                  ;*SW14=1         LOOP ON TEST
2476                                  ;*SW11=1         INHIBIT ITERATIONS
2477                                  ;*SW09=1         LOOP ON ERROR
2478                                  ;*SW08=1         LOOP ON TEST IN SWR<7:0>
2479                                  ;*CALL
2480                                  ;*      SCOPE           ;;SCOPE=IOT
2481
2482  017430                          $SCOPE:
2483  017430  104407                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
2484  017432  032777  040000  161500  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
2485  017440  001114                          BNE     $OVER           ;;YES IF SW14=1
2486                                  ;#####START OF CODE FOR THE XOR TESTER#####
2487  017442  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
2488                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
2489  017444  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2490  017450  012737  017470  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
2491  017456  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
2492  017462  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
```

```
2493  017466  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
2494  017470  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
2495  017472  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
2496  017476  000423                          BR      7$              ;;LOOP ON THE PRESENT TEST
2497  017500                          6$:;*****END OF CODE FOR THE XOR TESTER*****
2498  017500  032777  000400  161432          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
2499  017506  001404                          BEQ     2$              ;;BR IF NO
2500  017510  127737  161424  001102          CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
2501  017516  001465                          BEQ     $OVER           ;;BR IF YES
2502  017520  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
2503  017524  001421                          BEQ     3$              ;;BR IF NO
2504  017526  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2505  017534  101015                          BHI     3$              ;;BR IF NO
2506  017536  032777  001000  161374          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
2507  017544  001404                          BEQ     4$              ;;BR IF NO
2508  017546  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
2509  017554  000446                          BR      $OVER
2510  017556  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
2511  017562  005037  001160                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2512  017566  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
2513  017570  032777  004000  161342  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
2514  017576  001011                          BNE     1$              ;;BR IF YES
2515  017600  005737  001176                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
2516  017604  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
2517  017606  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
2518  017612  023737  001160  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
2519  017620  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
2520  017622  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
2521  017630  013737  017706  001160          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
2522  017636  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
2523  017642  113737  001102  001174          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
2524  017650  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
2525  017654  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
2526  017660  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FOR ERROR ADDRESS
2527  017664  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2528  017672  013777  001102  161242  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
2529  017700  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
2530  017704  000002                          RTI                     ;;FIXES PS
2531  017706  000010                  $MXCNT: 8.                      ;;MAX. NUMBER OF ITERATIONS
2532                                  .SBTTL  ERROR HANDLER ROUTINE
2533
2534                                  ;;**************************************************************
2535                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2536                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2537                                  ;*AND GO TO $ERrTYP ON ERROR
2538                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2539                                  ;*SW15=1         HALT ON ERROR
2540                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
2541                                  ;*SW09=1         LOOP ON ERROR
2542                                  ;*CALL
2543                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2544
2545  017710                          $ERROR:
2546  017710  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
```

```
2547  017712  113737  001102  017426          MOVB    $TSTNM,TSTNUM
2548  017720  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
2549  017724  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
2550  017726  013777  001102  161206          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
2551  017734  005237  001112                  INC     $ERTTL          ;;INC THE ERROR COUNT
2552  017740  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
2553  017744  162737  000002  001116          SUB     #2,$ERRPC
2554  017752  117737  161140  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
2555  017760  032777  020000  161152          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
2556  017766  001004                          BNE     20$             ;;SKIP TYPEOUTS
2557  017770  004737  020102                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
2558  017774  104401  001165                  TYPE    ,$CRLF
2559  020000                          20$:
2560  020000  122737  000001  001210          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
2561  020006  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
2562  020010  113737  001114  020022          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
2563  020016  004737  022210                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
2564  020022     000          21$:    .BYTE   0
2565  020023     000                  .BYTE   0
2566  020024  000777          22$:    BR      22$             ;;APT ERROR LOOP
2567  020026  005777  161106          2$:     TST     @SWR            ;;HALT ON ERROR
2568  020032  100002                  BPL     3$              ;;SKIP IF CONTINUE
2569  020034  000000                  HALT                    ;;HALT ON ERROR!
2570  020036  104407                  CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
2571  020040  032777  001000  161072  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
2572  020046  001402                  BEQ     4$              ;;BR IF NO
2573  020050  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
2574  020054  005737  001162          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
2575  020060  001402                  BEQ     5$              ;;BR IF NONE
2576  020062  013716  001162                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
2577  020066                          5$:
2578  020066  022737  013320  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
2579  020074  001001                          BNE     6$              ;;BRANCH IF NO
2580  020076  000000                          HALT                    ;;YES
2581  020100                          6$:
2582  020100  000002                          RTI                     ;;RETURN
2583
2584                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
2585
2586                  ;;**********************************************************************
2587                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
2588                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
2589                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
2590
2591  020102          $ERRTYP:
2592  020102  104401  001165          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
2593  020106  010046                  MOV     R0,-(SP)        ;;SAVE R0
2594  020110  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
2595  020112  153700  001114          BISB    @#$ITEMB,R0
2596  020116  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
2597                                                          ;;TYPE THE PC OF THE ERROR
2598  020120  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
2599                                                          ;;ERROR ADDRESS
2600  020124  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
2601  020126  000426                  BR      6$              ;;GET OUT
2602  020130  005300          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
2603  020132  006300                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
2604  020134  006300                  ASL     R0
2605  020136  006300                  ASL     R0
2606  020140  062700  001252          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
2607  020144  012037  020154          MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
2608  020150  001404                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
2609  020152  104401                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
2610  020154  000000          2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
2611  020156  104401  001165          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
2612  020162  012037  020172  3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
2613  020166  001404                  BEQ     5$              ;;SKIP TYPEOUT IF 0
2614  020170  104401                  TYPE                    ;;TYPE THE "DATA HEADER"
2615  020172  000000          4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
2616  020174  104401  001165          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
2617  020200  011000          5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
2618  020202  001004                  BNE     7$              ;;GO TYPE THE DATA
2619  020204  012600          6$:     MOV     (SP)+,R0        ;;RESTORE R0
2620  020206  104401  001165          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
2621  020212  000207                  RTS     PC              ;;RETURN
2622  020214                  7$:
2623  020214  013046                  MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
2624  020216  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2625  020220  005710                  TST     (R0)            ;;IS THERE ANOTHER NUMBER?
2626  020222  001770                  BEQ     6$              ;;BR IF NO
2627  020224  104401  020232          TYPE    ,8$             ;;TYPE TWO(2) SPACES
2628  020230  000771                  BR      7$              ;;LOOP
2629  020232  020040      000 8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
2630          020236                  .EVEN
2631
2632                                  .SBTTL  BINARY TO oCTAL (ASCII) AND TYPE
2633
2634                                  ;;**************************************************************
2635                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2636                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2637                                  ;*$TYpOS---ENTER HERE TO sETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO tYPE
2638                                  ;*CALL:
2639                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2640                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
2641                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2642                                  ;*      .BYTE   M               ;;M=1 OR 0
2643                                  ;*                                      ;;1=TYPE LEADING ZEROS
2644                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
2645                                  ;*
2646                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2647                                  ;*$TYpOS OR $TYpOC
2648                                  ;*CALL:
2649                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2650                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
2651                                  ;*
2652                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2653                                  ;*CALL:
2654                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
```

```
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
                                        ;
                                        ;
020236  017646  000000                  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
020242  116637  000001  020461                  MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
020250  112637  020463                          MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
020254  062716  000002                          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
020260  000406                                  BR      $TYPON
020262  112737  000001  020461          $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
020270  112737  000006  020463                  MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
020276  112737  000005  020460          $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
020304  010346                                  MOV     R3,-(SP)        ;;SAVE R3
020306  010446                                  MOV     R4,-(SP)        ;;SAVE R4
020310  010546                                  MOV     R5,-(SP)        ;;SAVE R5
020312  113704  020463                          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
020316  005404                                  NEG     R4
020320  062704  000006                          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
020324  110437  020462                          MOVB    R4,$OMODE       ;;SAVE IT FOR USE
020330  113704  020461                          MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
020334  016605  000012                          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
020340  005003                                  CLR     R3              ;;CLEAR THE OUTPUT WORD
020342  006105                          1$:     ROL     R5              ;;ROTATE MSB INTO "C"
020344  000404                                  BR      3$              ;;GO DO MSB
020346  006105                          2$:     ROL     R5              ;;FORM THIS DIGIT
020350  006105                                  ROL     R5
020352  006105                                  ROL     R5
020354  010503                                  MOV     R5,R3
020356  006103                          3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
020360  105337  020462                          DECB    $OMODE          ;;TYPE THIS DIGIT?
020364  100016                                  BPL     7$              ;;BR IF NO
020366  042703  177770                          BIC     #177770,R3      ;;GET RID OF JUNK
020372  001002                                  BNE     4$              ;;TEST FOR 0
020374  005704                                  TST     R4              ;;SUPPRESS THIS 0?
020376  001403                                  BEQ     5$              ;;BR IF YES
020400  005204                          4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
020402  052703  000060                          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
020406  052703  000040                  5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
020412  110337  020456                          MOVB    R3,8$           ;;SAVE FOR TYPING
020416  104401  020456                          TYPE    ,8$             ;;GO TYPE THIS DIGIT
020422  105337  020460                  7$:     DECB    $OCNT           ;;COUNT BY 1
020426  003347                                  BGT     2$              ;;BR IF MORE TO DO
020430  002402                                  BLT     6$              ;;BR IF DONE
020432  005204                                  INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
020434  000744                                  BR      2$              ;;GO DO THE LAST DIGIT
020436  012605                          6$:     MOV     (SP)+,R5        ;;RESTORE R5
020440  012604                                  MOV     (SP)+,R4        ;;RESTORE R4
020442  012603                                  MOV     (SP)+,R3        ;;RESTORE R3
020444  016666  000002  000004                  MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
020452  012616                                  MOV     (SP)+,(SP)
020454  000002                                  RTI                     ;;RETURN
020456      000                 8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
020457      000                         .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
020460      000                 $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
020461      000                 $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
020462  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
2709
2710                                    .SBTTL  POWER DOWN AND UP ROUTINES
2711
2712                                    ;;**********************************************************************
2713                                    ;POWER DOWN ROUTINE
2714  020464  012737  020630  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
2715  020472  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
2716  020500  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2717  020502  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2718  020504  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2719  020506  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2720  020510  010446                         MOV     R4,-(SP)        ;;PUSH R4 ON STACK
2721  020512  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2722  020514  017746  160420                 MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
2723  020520  010637  020634                 MOV     SP,$SAVR6       ;;SAVE SP
2724  020524  012737  020536  000024         MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
2725  020532  000000                         HALT
2726  020534  000776                         BR      .-2             ;;HANG UP
2727
2728                                    ;;**********************************************************************
2729                                    ;POWER UP ROUTINE
2730  020536  012737  020630  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWn
2731  020544  013706  020634                 MOV     $SAVR6,SP       ;;GET SP
2732  020550  005037  020634                 CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
2733  020554  005237  020634          1$:    INC     $SAVR6          ;;WAIT FOR THE INC
2734  020560  001375                         BNE     1$              ;;OF  WORD
2735  020562  012677  160352                 MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
2736  020566  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
2737  020570  012604                         MOV     (SP)+,R4        ;;POP STACK INTO R4
2738  020572  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
2739  020574  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
2740  020576  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
2741  020600  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
2742  020602  012737  020464  000024         MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
2743  020610  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
2744  020616  104401                         TYPE                    ;;REPORT THE POWER FAILURE
2745  020620  020636                 $PWRMG: .WORD   PWRMSG           ;;POWER FAIL MESSAGE POINTER
2746  020622  012716                         MOV     (PC)+,(SP)      ;;RESTART AT IOTEST
2747  020624  003332                 $PWRAD: .WORD   IOTEST           ;;RESTART ADDRESS
2748  020626  000002                         RTI
2749  020630  000000                 $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
2750  020632  000776                         BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
2751  020634  000000                 $SAVR6: 0                       ;;PUT THE SP HERE
2752  020636  005015  042522  052123 PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
2753  020644  051101  044524  043516
2754  020652  040440  052106  051105
2755  020660  040440  050040  053517
2756  020666  051105  043040  044501
2757  020674  052514  042522  005015
2758  020702      000
2759          020704                    .EVEN
2760
2761                                    .SBTTL  TYPE ROUTINE
2762
```

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

020704  105737  001157  $TYPE:  TSTB    $TPFLG           ;;IS THERE A TERmINAL?
020710  100002                  BPL     1$               ;;BR IF YES
020712  000000                  HALT                     ;;HALT HERE IF NO TERmINAL
020714  000430                  BR      3$               ;;LEAVE
020716  010046          1$:     MOV     R0,-(SP)         ;;SAVE R0
020720  017600  000002          MOV     @2(SP),R0        ;;GET ADDRESS OF ASCIZ STRING
020724  122737  000001  001210  CMPB    #APTENV,$ENV     ;;RUNNING IN APT MODE
020732  001011                  BNE     62$              ;;NO,GO CHECK FOR APT CONSOLE
020734  132737  000100  001211  BITB    #APTSPOOL,$ENVM  ;;SPOOL MESSAGE TO APT
020742  001405                  BEQ     62$              ;;NO,GO CHECK FOR CONSOLE
020744  010037  020754          MOV     R0,61$           ;;SETUP MESSAGE ADDRESS FOR APT
020750  004737  022200          JSR     PC,$ATY3         ;;SPOOL MESSAGE TO APT
020754  000000          61$:    .WORD   0                ;;MESSAGE ADDRESS
020756  132737  000040  001211  62$:    BITB    #APTCSUP,$ENvM   ;;APT CONSOLE SUPPRESSED
020764  001003                  BNE     60$              ;;YES,SKIP TYPE OUT
020766  112046          2$:     MOVB    (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
020770  001005                  BNE     4$               ;;BR IF IT ISN'T THE TERmINATOR
020772  005726                  TST     (SP)+            ;;IF TERMINATOR POP IT OFF THE STACK
020774  012600          60$:    MOV     (SP)+,R0         ;;RESTORE R0
020776  062716  000002          ADD     #2,(SP)          ;;ADJUST RETURN PC
021002  000002                  RTI                      ;;RETURN
021004  122716  000011          4$:     CMPB    #HT,(SP)         ;;BRANCH IF <HT>
021010  001430                  BEQ     8$
021012  122716  000200          CMPB    #CRLF,(SP)       ;;BRANCH IF NOT <CRLF>
021016  001006                  BNE     5$
021020  005726                  TST     (SP)+            ;;POP  <CR><LF> EQUIV
021022  104401                  TYPE                     ;;TYPE A CR AND LF
021024  001165                  $CRLF
021026  105037  021162          CLRB    $CHARCNT         ;;CLEAR CHARACTER COUNT
021032  000755                  BR      2$               ;;GET NEXT CHARACTER
021034  004737  021116  5$:     JSR     PC,$TYPEC        ;;GO TYPE THIS CHARACTER
021040  123726  001156  6$:     CMPB    $FILLC,(SP)+     ;;IS IT TIME FOR FILLER CHARS.?
021044  001350                  BNE     2$               ;;IF NO GO GET NEXT CHAR.
021046  013746  001154          MOV     $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
                                                         ;;AND THE NULL CHAR.
021052  105366  000001  7$:     DECB    1(SP)            ;;DOES A NULL NEED TO BE TYPED?
021056  002770                  BLT     6$               ;;BR IF NO--GO POP THE NULL OFF OF STACK
021060  004737  021116          JSR     PC,$TYPEC        ;;GO TYPE A NULL
021064  105337  021162          DECB    $CHARCNT         ;;DO NOT COUNT AS A COUNT
```

```
2817  021070  000770                          BR      7$              ;;LOOP
2818
2819                                  ;HORIZONTAL TAB PROCESSOR
2820
2821  021072  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
2822  021076  004737  021116          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
2823  021102  132737  000007  021162          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
2824  021110  001372                          BNE     9$              ;;TAB STOP
2825  021112  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
2826  021114  000724                          BR      2$              ;;GET NEXT CHARACTER
2827  021116  105777  160026          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
2828  021122  100375                          BPL     $TYPEC
2829  021124  116677  000002  160020        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2830  021132  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
2831  021140  001003                          BNE     1$              ;;BRANCH IF NO
2832  021142  105037  021162                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
2833  021146  000406                          BR      $TYPEX          ;;EXIT
2834  021150  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
2835  021156  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
2836  021160  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
2837  021162  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
2838  021164  000207          $TYPEX: RTS     PC
2839
2840                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2841
2842                                  ;;**************************************************************
2843                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2844                                  ;*CHANGE IT TO BINARY.
2845                                  ;*CALL:
2846                                  ;*      RDOCT                   ;;READ AN OCTAL NUMBER
2847                                  ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2848                                  ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
2849
2850  021166  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2851  021170  016666  000004  000002        MOV     4(SP),2(SP)     ;;INPUT NUMBER
2852  021176  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2853  021200  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2854  021202  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2855  021204  104411                  1$:   RDLIN                   ;;READ AN ASCIZ LINE
2856  021206  012600                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2857  021210  005001                        CLR     R1              ;;CLEAR DATA WORD
2858  021212  005002                        CLR     R2
2859  021214  112046                  2$:   MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
2860  021216  001412                        BEQ     3$              ;;IF ZERO GET OUT
2861  021220  006301                        ASL     R1              ;;*2
2862  021222  006102                        ROL     R2
2863  021224  006301                        ASL     R1              ;;*4
2864  021226  006102                        ROL     R2
2865  021230  006301                        ASL     R1              ;;*8
2866  021232  006102                        ROL     R2
2867  021234  042716  177770                BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
2868  021240  062601                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
2869  021242  000764                        BR      2$              ;;LOOP
2870  021244  005726                  3$:   TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
```

```
2871  021246  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
2872  021252  010237  021266                  MOV     R2,$HIOCT
2873  021256  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
2874  021260  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2875  021262  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2876  021264  000002                          RTI                     ;;RETURN
2877  021266  000000              $HIOCT:     .WORD   0               ;;HIGH ORDER BITS GO HERE
2878                                .SBTTL  TTY INPUT ROUTINE
2879
2880                                ;;**********************************************************************
2881                                .ENABL  LSB
2882
2883                                ;;**********************************************************************
2884                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2885                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2886                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2887                                ;*WHEN OPERATING IN TTY FLAG MODE.
2888  021270  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
2889  021276  001074                          BNE     1$              ;;BRANCH IF NO
2890  021300  105777  157640                  TSTB    @$TKS           ;;CHAR THERE?
2891  021304  100071                          BPL     1$              ;;IF NO, DON'T WAIT AROUND
2892  021306  117746  157634                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
2893  021312  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
2894  021316  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
2895  021322  001062                          BNE     1$              ;;NO, RETURN TO USER
2896  021324  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
2897  021332  001456                          BEQ     1$              ;;BRANCH IF YES
2898
2899  021334  104401  022142                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
2900  021340  104401  022147          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
2901  021344  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
2902  021350  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2903  021352  104401  022160                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
2904  021356  005046              19$:        CLR     -(SP)           ;;CLEAR COUNTER
2905  021360  005046                          CLR     -(SP)           ;;THE NEW SWR
2906  021362  105777  157556          7$:     TSTB    @$TKS           ;;CHAR THERE?
2907  021366  100375                          BPL     7$              ;;IF NOT TRY AGAIN
2908
2909  021370  117746  157552                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
2910  021374  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2911
2912
2913
2914  021400  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
2915  021404  001005                          BNE     10$             ;;BRANCH IF NOT
2916  021406  104401  022135                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
2917  021412  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
2918  021416  000757                          BR      19$             ;;LET'S TRY IT AGAIN
2919
2920
2921  021420  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
2922  021424  001022                          BNE     16$             ;;BRANCH IF NO
2923  021426  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
2924  021432  001403                          BEQ     11$             ;;BRANCH IF YES
```

```
2925  021434  016677  000002  157476          MOV     2(SP),@SWR      ;;SAVE NEW SWR
2926  021442  062706  000006                  ADD     #6,SP           ;;CLEAR UP STACK
2927  021446  104401  001165          11$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
2928  021452  123727  001135  000001  14$:    CMPB    @$INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
2929  021460  001003                          BNE     15$             ;;BRANCH IF NOT
2930  021462  012777  000100  157454          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
2931  021470  000002                  15$:    RTI                     ;;RETURN
2932  021472  004737  021116          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
2933  021476  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
2934  021502  002420                          BLT     18$             ;;BRANCH IF YES
2935  021504  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
2936  021510  003015                          BGT     18$             ;;BRANCH IF YES
2937  021512  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2938  021516  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
2939  021522  001403                          BEQ     17$             ;;BRANCH IF YES
2940  021524  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
2941  021526  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
2942  021530  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
2943  021532  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
2944  021536  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2945  021542  000707                          BR      7$              ;;GET THE NEXT ONE
2946  021544  104401  001164          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
2947  021550  000720                          BR      20$             ;;SIMULATE CONTROL-U
2948                                  .DSABL  LSB
2949
2950
2951                                  ;;*****************************************************************
2952                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2953                                  ;*CALL:
2954                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
2955                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
2956                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
2957                                  ;
2958
2959  021552  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
2960  021554  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
2961  021562  105777  157356          1$:     TSTB    @$TKS           ;;WAIT FOR
2962  021566  100375                          BPL     1$              ;;A CHARACTER
2963  021570  117766  157352  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
2964  021576  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
2965  021604  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
2966  021612  001013                          BNE     3$              ;;BRANCH IF NO
2967  021614  105777  157324          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
2968  021620  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
2969  021622  117746  157320                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
2970  021626  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2971  021632  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
2972  021636  001366                          BNE     2$              ;;IF NOT DISCARD IT
2973  021640  000750                          BR      1$              ;;YES, RESUME
2974  021642  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
2975  021650  002407                          BLT     4$              ;;BRANCH IF YES
2976  021652  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
2977  021660  003003                          BGT     4$              ;;BRANCH IF YES
2978  021662  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
```

```
                                        4$:     RTI                             ;;GO BACK TO USER
                                        ;;**************************************************************
                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                        ;*CALL:
                                        ;*      RDLIN                           ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE                     ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                                      ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
                                        $RDLIN: MOV     R3,-(SP)                ;;SAVE R3
                                                CLR     -(SP)                   ;;CLEAR THE RUBOUT KEY
                                        1$:     MOV     #$TTYIN,R3              ;;GET ADDRESS
                                        2$:     CMP     #$TTYIN+7,R3            ;;BUFFER FULL?
                                                BLOS    4$                      ;;BR IF YES
                                                RDCHR                           ;;GO READ ONE CHARACTER FROM THE TTY
                                                MOVB    (SP)+,(R3)              ;;GET CHARACTER
                                        10$:    CMPB    #177,(R3)               ;;IS IT A RUBOUT
                                                BNE     5$                      ;;BR IF NO
                                                TST     (SP)                    ;;IS THIS THE FIRST RUBOUT?
                                                BNE     6$                      ;;BR IF NO
                                                MOVB    #'\,9$                  ;;TYPE A BACK SLASH
                                                TYPE    ,9$
                                                MOV     #-1,(SP)                ;;SET THE RUBOUT KEY
                                        6$:     DEC     R3                      ;;BACKUP BY ONE
                                                CMP     R3,#$TTYIN              ;;STACK EMPTY?
                                                BLO     4$                      ;;BR IF YES
                                                MOVB    (R3),9$                 ;;SETUP TO TYPEOUT THE DELETED CHAR.
                                                TYPE    ,9$                     ;;GO TYPE
                                                BR      2$                      ;;GO READ ANOTHER CHAR.
                                        5$:     TST     (SP)                    ;;RUBOUT KEY SET?
                                                BEQ     7$                      ;;BR IF NO
                                                MOVB    #'\,9$                  ;;TYPE A BACK SLASH
                                                TYPE    ,9$
                                                CLR     (SP)                    ;;CLEAR THE RUBOUT KEY
                                        7$:     CMPB    #25,(R3)                ;;IS CHARACTER A CTRL U?
                                                BNE     8$                      ;;BR IF NO
                                                TYPE    ,$CNTLU                 ;;TYPE A CONTROL "U"
                                                BR      1$                      ;;GO START OVER
                                        8$:     CMPB    #22,(R3)                ;;IS CHARACTER A "↑R"?
                                                BNE     3$                      ;;BRANCH IF NO
                                                CLRB    (R3)                    ;;CLEAR THE CHARACTER
                                                TYPE    ,$CRLF                  ;;TYPE A "CR" & "LF"
                                                TYPE    ,$TTYIN                 ;;TYPE THE INPUT STRING
                                                BR      2$                      ;;GO PICKUP ANOTHER CHACTER
                                        4$:     TYPE    ,$QUES                  ;;TYPE A '?'
                                                BR      1$                      ;;CLEAR THE BUFFER AND LOOP
                                        3$:     MOVB    (R3),9$                 ;;ECHO THE CHARACTER
                                                TYPE    ,9$
                                                CMPB    #15,(R3)+               ;;CHECK FOR RETURN
                                                BNE     2$                      ;;LOOP IF NOT RETURN
                                                CLRB    -1(R3)                  ;;CLEAR RETURN (THE 15)
                                                TYPE    ,$LF                    ;;TYPE A LINE FEED
                                                TST     (SP)+                   ;;CLEAN RUBOUT KEY FROM THE STACK
                                                MOV     (SP)+,R3                ;;RESTORE R3
                                                MOV     (SP),-(SP)              ;;ADJUST THE STACK AND PUT ADDRESS OF THE
```

```
3033  022106  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
3034  022114  012766  022126  000004          MOV     #$TTYIN,4(SP)
3035  022122  000002                          RTI                     ;;RETURN
3036  022124     000  9$:     .BYTE   0       ;;STORAGE FOR ASCII CHAR. TO TYPE
3037  022125     000          .BYTE   0       ;;TERMINATOR
3038  022126  000007  $TTYIN: .BLKB   7       ;;RESERVE 7 BYTES FOR TTY INPUT
3039  022135     136  006525  000012 $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
3040  022142  043536  005015     000 $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
3041  022147     015  051412  051127 $MSWR:  .ASCIZ  <15><12>/SWR = /
3042  022154  036440  000040
3043  022160  020040  042516  020127 $MNEW:  .ASCIZ  /  NEW = /
3044  022166  020075     000
3045  022172                          .EVEN
3046
3047
3048                                    .SBTTL  APT COMMUNICATIONS ROUTINE
3049
3050                                    ;;*****************************************************************
3051  022172  112737  000001  022436 $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
3052  022200  112737  000001  022434 $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
3053  022206  000403                  BR      $ATYC
3054  022210  112737  000001  022436 $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
3055  022216                          $ATYC:
3056  022216  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3057  022220  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3058  022222  105737  022434          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
3059  022226  001450                  BEQ     5$              ;;IF NOT:  BR
3060  022230  122737  000001  001210 CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
3061  022236  001031                  BNE     3$              ;;IF NOT:  BR
3062  022240  132737  000100  001211 BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
3063  022246  001425                  BEQ     3$              ;;IF NOT:  BR
3064  022250  017600  000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
3065  022254  062766  000002  000004 ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
3066  022262  005737  001170  1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
3067  022266  001375                  BNE     1$              ;;IF NOT:  WAIT
3068  022270  010037  001204          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
3069  022274  105720          2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
3070  022276  001376                  BNE     2$
3071  022300  163700  001204          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
3072  022304  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
3073  022306  010037  001206          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
3074  022312  012737  000004  001170 MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
3075  022320  000413                  BR      5$
3076  022322  017637  000004  022346 3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LI  GE
3077  022330  062766  000002  000004 ADD     #2,4(SP)                ;;BUMP RETURN ADD ESS
3078  022336  013746  177776          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
3079  022342  004737  020704          JSR     PC,$TYPE        ;;CALL TYPE MACRO
3080  022346  000000          4$:     .WORD   0
3081  022350                          5$:
3082  022350  105737  022436 10$:     TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
3083  022354  001416                  BEQ     12$             ;;IF NOT:  BR
3084  022356  005737  001210          TST     $ENV            ;;RUNNING UNDER APT?
3085  022362  001413                  BEQ     12$             ;;IF NOT:  BR
3086  022364  005737  001170 11$:     TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
```

```
3087  022370  001375                          BNE     11$                     ;;IF NOT:  WAIT
3088  022372  017637  000004  001172          MOV     @4(SP),$FATAL           ;;GET ERROR #
3089  022400  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
3090  022406  005237  001170                  INC     $MSGTYPE                ;;TELL APT TO TAKE ERROR
3091  022412  105037  022436          12$:    CLRB    $FFLG                   ;;CLEAR FATAL FLAG
3092  022416  105037  022435                  CLRB    $LFLG                   ;;CLEAR LOG FLAG
3093  022422  105037  022434                  CLRB    $MFLG                   ;;CLEAR MESSAGE FLAG
3094  022426  012601                          MOV     (SP)+,R1                ;;POP STACK INTO R1
3095  022430  012600                          MOV     (SP)+,R0                ;;POP STACK INTO R0
3096  022432  000207                          RTS     PC                      ;;RETURN
3097  022434     000          $MFLG:  .BYTE   0                               ;;MESSG. FLAG
3098  022435     000          $LFLG:  .BYTE   0                               ;;LOG FLAG
3099  022436     000          $FFLG:  .BYTE   0                               ;;FATAL FLAG
3100          022440                  .EVEN
3101          000200          APTSIZE=200
3102          000001          APTENV=001
3103          000100          APTSPOOL=100
3104          000040          APTCSUP=040
3105                          .SBTTL  TRAP DECODER
3106
3107                          ;;**************************************************************
3108                          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3109                          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3110                          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3111                          ;*GO TO THAT ROUTINE.
3112
3113  022440  010046          $TRAP:  MOV     R0,-(SP)                ;;SAVE R0
3114  022442  016600  000002          MOV     2(SP),R0                ;;GET TRAP ADDRESS
3115  022446  005740                  TST     -(R0)                   ;;BACKUP BY 2
3116  022450  111000                  MOVB    (R0),R0                 ;;GET RIGHT BYTE OF TRAP
3117  022452  006300                  ASL     R0                      ;;POSITION FOR INDEXING
3118  022454  016000  022474          MOV     $TRPAD(R0),R0           ;;INDEX TO TABLE
3119  022460  000200                  RTS     R0                      ;;GO TO ROUTINE
3120
3121
3122                          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3123
3124  022462  011646          $TRAP2: MOV     (SP),-(SP)              ;;MOVE THE PC DOWN
3125  022464  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
3126  022472  000002                  RTI                             ;;RESTORE THE PSW
3127
3128                          .SBTTL  TRAP TABLE
3129
3130                          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3131                          ;*BY THE "TRAP" INSTRUCTION.
3132
3133                          ;       ROUTINE
3134                          ;       -------
3135  022474  022462          $TRPAD: .WORD   $TRAP2
3136  022476  020704                  $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
3137  022500  020262                  $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3138  022502  020236                  $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3139  022504  020276                  $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3140  022506  013354                  $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
```

```
3141
3142  022510  021340                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
3143
3144  022512  021270                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
3145  022514  021552                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
3146  022516  021672                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
3147  022520  021166                          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3148
3149  022522  000000              BUFFER: 0                           ;10 WORD BUFFER FOR THE COULTER INTERFACE TEST
3150
3151                                        ;*
3152                                        ;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
3153                                        ;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
3154                                        ;*NEXT WE WILL INIT BOTH UPROCESSORS
3155                                        ;*THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
3156                                        ;*THE ORDER OF LOAD IS DETERMINED BY THE USER.
3157                                        ;*
3158                                        ;*      CALL=   JSR     R5,$LPAI
3159                                        ;*              .WORD   0               ;ADDR. OF DEVICE ADDRESS.
3160                                        ;* ROUTINES REQUIRED:   .LOADLP
3161                                        ;* PROGRAMS REQUIRED:   DRLPX2
3162                                        ;*
3163
3164                                        ;*              ;RETURNS WITH $AERR=1 IF SLAVE
3165                                        ;*              ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
3166                                        ;*
3167  022524                        $LPAI:
3168  022524  013746  000004                MOV     4,-(SP)
3169
3170  022530  000413                        BR      31$             ;FIELD DOES NOT HAVE A BUS SWITCH TO
3171                                                                ;WORRY ABOUT,SO WE WILL UNCONDITIONALLY
3172                                                                ;BRANCH ARROUD THE NEXT CODE THAT
3173                                                                ;WORKS BASED ON A BUS SWITCH.
3174                                                                ;CODE LEFT IN HERE FOR IN HOUSE
3175                                                                ;PERSONAL WHO MAY PATCH THIS BRANCH
3176                                                                ;INSTRUCTION TO A <NOP> OCTAL <240>
3177                                                                ;IN ORDER TO RUN PROGRAM WITH A SWITCH.
3178
3179                                                                ;NOTE THIS "SWITCH" IS A PIECE OF INHOUSE
3180                                                                ;TEST EQUIPMENT ONLY IT CONNECTS
3181                                                                ;THE UNIBUS TO THE I/O BUS FOR
3182                                                                ;CERTAIN TESTING.
3183  022532  012737  022556  000004                MOV     #30$,4
3184  022540  005237  170000                       INC     170000
3185  022544  104401  022552                       TYPE    ,65$            ;;TYPE ASCIZ STRING
3186  022550  000401                                BR      64$             ;;GET OVER THE ASCIZ
3187                                        ;;65$:  .ASCIZ  <7>##
3188  022554                                64$:
3189  022554  000401                                BR      31$
3190  022556  022626                        30$:    CMP     (SP)+,(SP)+
3191  022560  012637  000004                31$:    MOV     (SP)+,4         ;ALL THIS JUNK MUST BE REMOVED!!
3192  022564  005037  023402                        CLR     $AERR
3193  022570  004537  023404                        JSR     R5,$LOAD        ;LOAD MICRO-CODE.
3194  022574  000000G                               .WORD   DRLPX2          ;FILE "DRLPX2.OBJ"
```

```
3195
3196  022576  052777  040000  156556          BIS     #BIT14,@KMADO   ;ISSUE KMC+DMC INIT.
3197
3198  022604                          1$:
3199                                                                  ;"HANGS" HERE THEN KMC-11 ERROR.
3200  022604  010146                          MOV     R1,-(SP)
3201  022606  005001                          CLR     R1
3202  022610  005201                  2$:     INC     R1              ;STALL FOR DMC-UP
3203  022612  001376                          BNE     2$
3204  022614  012777  104000  156540          MOV     #BIT15!BIT11,@KMADO     ;SET RUN, AND ENABLE ARBITRATION.
3205  022622  105201                  25$:    INCB    R1
3206  022624  001376                          BNE     25$
3207
3208  022626  032777  000040  156526          BIT     #BIT5,@KMADO    ;SLAVE READY? (READING IPBM SR)
3209  022634  001401                          BEQ     3$
3210                                                                  ;FATAL LPA-11 ERROR SLAVE NOT READY.
3211  022636  104000                          ERROR
3212
3213  022640  012777  000004  156520  3$:     MOV     #4,@KMAD2       ;READ FAST PATH
3214  022646                          4$:
3215  022646  004537  024314                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
3216
3217  022652  104000                          ERROR                   ;/TIME-OUT ERROR
3218                                                                  ;/WE FAILED TO COMPLETE
3219                                                                  ;/CURRENT OPERATION.
3220                                                                  ;/CONTINUES IN THIS LOOP
3221                                                                  ;/WOULD MAKE US "HANG" HERE
3222
3223  022654  000774                          BR              4$
3224
3225                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3226  022656  122777  000377  156502          CMPB    #377,@KMAD2     ;WAIT TILL KMC DONE COMMAND.
3227  022664  001370                          BNE     4$
3228  022666  122777  000377  156476          CMPB    #377,@KMAD4     ;IF FAST PATH=377 THEN ERROR.
3229  022674  001001                          BNE     35$
3230  022676  104000                          ERROR                   ;IPBM ERROR (SLAVE SIDE)
3231                                                                  ;YOU MUST RUN IPBM DIAGNOSTIC.
3232
3233  022700  122777  000004  156464  35$:     CMPB    #4,@KMAD4       ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
3234  022706  001543                          BEQ     5$              ;YES-CONTINUE.
3235  022710  005227  177777                  INC     #-1
3236  022714  001140                          BNE     5$
3237  022716  005227  177777                  INC     #-1
3238  022722  001135                          BNE     5$
3239  022724  104401  022732                  TYPE    ,67$            ;;TYPE ASCIZ STRING
3240  022730  000440                          BR      66$             ;;GET OVER THE ASCIZ
3241                                  ;;67$:   .ASCIZ  <200>"W A R N I N G  THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4"
3242  023032                          66$:
3243  023032  104401  023040                  TYPE    ,69$            ;;TYPE ASCIZ STRING
3244  023036  000430                          BR      68$             ;;GET OVER THE ASCIZ
3245                                  ;;69$:   .ASCIZ  <200>"MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED."
3246  023120                          68$:
3247  023120  104401  023126                  TYPE    ,71$            ;;TYPE ASCIZ STRING
3248  023124  000434                          BR      70$             ;;GET OVER THE ASCIZ
```

```
3249                                          ;;71$:  .ASCIZ  <200>"THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED."<200><200>
3250  023216                                  70$:
3251
3252  023216  112737  177777  023350  5$:     MOVB    #0-1,11$        ;DAC CODE FOR SLAVE.
3253  023224  012501                          MOV     (5)+,R1         ;GET NEXT DEVICE ADDR.
3254  023226  021127  000000          6$:     CMP     (R1),#0         ;TERM REACHED?
3255  023232  001444                          BEQ     10$
3256  023234  105237  023350                  INCB    11$
3257  023240  113777  023350  156124          MOVB    11$,@KMAD4      ;FIFO DATA
3258  023246  004737  023352                  JSR     PC,20$          ;ISSUE SEND
3259  023252  112177  156114                  MOVB    (R1)+,@KMAD4    ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
3260  023256  004737  023352                  JSR     PC,20$          ;ISSUE SEND
3261  023262  112177  156104                  MOVB    (R1)+,@KMAD4    ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
3262  023266  004737  023352                  JSR     PC,20$
3263
3264  023272  032777  000002  156062  7$:     BIT     #BIT1,@KMAD0    ;WAIT FOR FIFO DATA
3265  023300  001374                          BNE     7$              ;=1 NO DATA. =0 DATA.
3266  023302  112777  000002  156056          MOVB    #2,@KMAD2       ;READ FIFO.
3267
3268  023310                                  8$:
3269  023310  004537  024314                  JSR     R5, $TOUt       ;-TOUT-CHECK FOR TIMEOUT
3270
3271  023314  104000                          ERROR                   ;/TIME-OUT ERROR
3272                                                                  ;/WE FAILED TO COMPLETE
3273                                                                  ;/CURRENT OPERATION.
3274                                                                  ;/CONTINUES IN THIS LOOP
3275                                                                  ;/WOULD MAKE US "HANG" HERE
3276
3277  023316  000774                          BR              8$
3278
3279                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3280  023320  122777  000377  156040          CMPB    #377,@KMAD2     ;WAIT FOR READ.
3281  023326  001370                          BNE     8$
3282  023330  105777  156036                  TSTB    @KMAD4          ;WAS A ZERO RETURNED?
3283  023334  001734                          BEQ     6$              ;YES GET NEXT ADDR.
3284                                                                  ;SLAVE WILL RETURN CODE 0 IF
3285  023336  005237  023402                  INC     $AERR           ;DEV PRESENT.   ELSE
3286                                                                  ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
3287  023342  005041                          CLR     -(1)            ;GET RID OF REFERENCE TO BAD ADDR.
3288  023344  012601                  10$:    MOV     (SP)+,R1
3289  023346  000205                          RTS     R5              ;RETURN ALL ADDR. CHECKED.
3290
3291  023350  000000                  11$:    .WORD   0               ;HOLDS DAC CODE PLUS OFFSET
3292                                                                  ;TO SLAVES ADDR. TABLE.
3293
3294  023352  112777  000003  156006  20$:    MOVB    #3,@KMAD2       ;ISSUE FIFO WRITE
3295  023360                          21$:
3296  023360  004537  024314                  JSR     R5, $TOUT       ;-TOUT-CHECK FOR TIMEOUT
3297
3298  023364  104000                          ERROR                   ;/TIME-OUT ERROR
3299                                                                  ;/WE FAILED TO COMPLETE
3300                                                                  ;/CURRENT OPERATION.
3301                                                                  ;/CONTINUES IN THIS LOOP
3302                                                                  ;/WOULD MAKE US "HANG" HERE
```

```
3303
3304  023366  000774                          BR          21$
3305
3306                                                      ;/RETURNS HERE-FROM-TIMED OUT.
3307  023370  122777  000377  155770          CMPB    #377,@KMAD2     ;KMC CODE WILL RETURN A "377"
3308  023376  001370                          BNE     21$             ;WHEN DONE COMMAND.
3309  023400  000207                          RTS     PC
3310
3311  023402  000000                  $AERR:  .WORD   0               ;=0 IF ADDR. LIST OK,=1 IF BAD.
3312
3313                                          ;*
3314                                          ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
3315                                          ;*      CALL =  JSR     R5,$LOAD
3316                                          ;*              .WORD   XX              ;ADDR. OF MICRO CODE.
3317                                          ;*              ;RETURNS HERE
3318                                          ;*      NOTE:   MICRO CODE FILE MUST END IN -1 DATA.
3319                                          ;*
3320
3321  023404  010446                  $LOAD:  MOV     R4,-(SP)        ;SAVE R4.
3322  023406  010046                          MOV     R0,-(SP)        ;SAVE R0.
3323  023410  012500                  1$:     MOV     (5)+,R0         ;GET PROG. ADDR.
3324  023412  005077  155744                  CLR     @KMAD0          ;CLEAR CSR
3325  023416  005077  155750                  CLR     @KMAD4          ;CLEAR CRAM ADDR.
3326  023422  052777  002000  155732  2$:     BIS     #2000,@KMAD0    ;SELECT CRAM.
3327  023430  012077  155742                  MOV     (0)+,@KMAD6     ;WRITE DATA.
3328  023434  052777  020000  155720          BIS     #20000,@KMAD0   ;SET CRAM WRITE
3329  023442  005077  155714                  CLR     @KMAD0          ;DISABLE CRAM.
3330  023446  005277  155720                  INC     @KMAD4          ;UPDATE CRAM ADDR.
3331  023452  021027  177777                  CMP     (0),#-1         ;ALL DONE?
3332  023456  001361                          BNE     2$              ;NO LOOP.
3333  023460  005077  155706                  CLR     @KMAD4          ;CLEAR CRAM ADDR.
3334  023464  016500  177776                  MOV     -2(5),R0        ;GET MICRO CODE ADDR.
3335
3336  023470  052777  002000  155664  3$:     BIS     #2000,@KMAD0    ;SELECT CRAM
3337  023476  022077  155674                  CMP     (R0)+,@KMAD6    ;DATA OK?
3338  023502  001013                          BNE     5$              ;NO - REPORT AN ERROR.
3339  023504  021027  177777                  CMP     (0),#-1         ;ALL DONE?
3340  023510  001405                          BEQ     4$              ;YES - EXIT
3341  023512  005077  155644                  CLR     @KMAD0          ;NO - DESELECT CRAM.
3342  023516  005277  155650                  INC     @KMAD4          ;UPDATE CRAM ADDR.
3343  023522  000762                          BR      3$
3344
3345  023524  012600                  4$:     MOV     (SP)+,R0        ;RESTORE R0
3346  023526  012604                          MOV     (SP)+,R4        ;RESTORE R4
3347  023530  000205                          RTS     R5              ;EXIT
3348
3349  023532                          5$:                             ;COME HERE ON LOAD ERROR
3350  023532  005745                          TST     -(5)
3351  023534  105204                          INCB    R4              ;UPDATE ERROR COUNTER.
3352  023536  100324                          BPL     1$              ;IF NOT TOO MANY, TRY AGAIN.
3353  023540  000000                          HALT                    ;MICRO CODE LOAD ERROR.
3354                                                                  ;KMC-11 FAULT. YOU COULD TRY
3355  023542  000722                          BR      1$              ;TO PRESS CONTINUE TO GIVE IT
3356                                                                  ;ANOTHER CHANCE, BUT I DOUBT
```

```
                                                        ;THAT THAT WOULD WORK. SINCE I'VE
                                                        ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
                                                        ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
                                                ;
                                                ;*THIS ROUTINE ISSUES A WRITE COMMAND TO tHE LPA-11
                                                ;*
                                                ;*      CALL =  JSR     R5,$TLKW
                                                ;*              .WORD   0               ;OFFSET OF DEVICE ADDR.
                                                ;*              .WORD   0               ;DATA TO BE WRITTEN
                                                ;*
023544  010046                  $TLKW:  MOV     R0,-(SP)        ;SAVE R0
023546  012500                          MOV     (5)+,R0         ;GET DEVICE OFFSET
023550  052700  000340                  BIS     #340,R0         ;ADD WRITE CODE.
023554  004737  024026                  JSR     PC,$LPW         ;WAIT FOR FAST PATH READY
023560  010037  023652                  MOV     R0,W1
023564  010077  155602                  MOV     R0,@KMAD4
023570  112777  000005  155570          MOVB    #5,@KMAD2       ;ISSUE FAST PATH WRITE
023576  004737  024026                  JSR     PC,$LPW         ;WAIT FOR RDY
023602  011537  023654                  MOV     (5),W2
023606  112577  155560                  MOVB    (5)+,@KMAD4     ;WRITE LOW BYTE DATA.

023612  112777  000005  155546          MOVB    #5,@KMAD2       ;FP WRITE
023620  004737  024026                  JSR     PC,$LPW
023624  111537  023656                  MOVB    (5),W3
023630  112577  155536                  MOVB    (5)+,@KMAD4     ;WRITE HIGH BYTE
023634  112777  000005  155524          MOVB    #5,@KMAD2
023642  004737  024026                  JSR     PC,$LPW
023646  012600                          MOV     (SP)+,R0
023650  000205                          RTS     R5              ;EXIT DONE.
023652  000000                  W1:     0
023654  000000                  W2:     0
023656  000000                  W3:     0

                                                ;*
                                                ;*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
                                                ;*
                                                ;*      CALL =  JSR     R5,$TLKR
                                                ;*              .WORD   0               ;OFFSET OF DEVICE
                                                ;*              ;RETURNS HERE
                                                ;*DATA IN WORD $DATR
                                                ;*
023660  010046                  $TLKR:  MOV     R0,-(SP)        ;SAVE R0
023662  012500                          MOV     (5)+,R0         ;GET OFFSET
023664  052700  000300                  BIS     #300,R0         ;ADD READ CODE
023670  004737  024026                  JSR     PC,$LPW         ;WAIT TILL READY
023674  110077  155472                  MOVB    R0,@KMAD4
023700  112777  000005  155460          MOVB    #5,@KMAD2       ;ISSUE WRITE FP
023706  004737  024026                  JSR     PC,$LPW
023712  010037  024022                  MOV     R0,RD1
023716                          1$:
023716  004537  024314                  JSR     R5, $TOuT       ;-TOUT-CHECK FOR TIMEOUT
```

```
3411
3412  023722  104000                          ERROR                   ;/TIME-OUT ERROR
3413                                                                  ;/WE FAILED TO COMPLETE
3414                                                                  ;/CURRENT OPERATION.
3415                                                                  ;/CONTINUES IN THIS LOOP
3416                                                                  ;/WOULD MAKE US "HANG" HERE
3417
3418  023724  000774                          BR              1$
3419
3420                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3421  023726  032777  000040  155426          BIT     #BIT5,@KMAD0    ;FAST PATH GOT DATA?
3422  023734  001370                          BNE     1$
3423  023736  112777  000004  155422          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
3424  023744  004737  024026                  JSR     PC,$LPW
3425  023750  117737  155416  024024          MOVB    @KMAD4,$DATR    ;GET LOW BYTE
3426  023756                          2$:
3427  023756  004537  024314                  JSR     R5, $TOUT       ;-TOUt-CHECK FOR TIMEOUT
3428
3429  023762  104000                          ERROR                   ;/TIME-OUT ERROR
3430                                                                  ;/WE FAILED TO COMPLETE
3431                                                                  ;/CURRENT OPERATION.
3432                                                                  ;/CONTINUES IN THIS LOOP
3433                                                                  ;/WOULD MAKE US "HANG" HERE
3434
3435  023764  000774                          BR              2$
3436
3437                                                                  ;/RETURNS HERE-FROM-TIMED OUT.
3438  023766  032777  000040  155366          BIT     #BIT5,@KMAD0    ;FAST PATH READY?
3439  023774  001370                          BNE     2$
3440  023776  112777  000004  155362          MOVB    #4,@KMAD2       ;ISSUE FAST PATH READ
3441  024004  004737  024026                  JSR     PC,$LPW
3442  024010  117737  155356  024025          MOVB    @KMAD4,$DATR+1  ;SAVE HIGH BYTE
3443  024016  012600                          MOV     (SP)+,R0
3444  024020  000205                          RTS     R5
3445  024022  000000                  RD1:    0
3446  024024  000000                  $DATR:  .WORD   0
3447                                          ;
3448                                          ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
3449                                          ;AS FAST PATH TO BE READ.
3450                                          ;
3451                                          ;       CALL =  JSR     PC,$LPw
3452                                          ;
3453                                          ;IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
3454                                          ;THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
3455                                          ;
3456
3457  024026  010146                  $LPW:   MOV     R1,-(SP)        ;SAVE R1
3458  024030  005001                          CLR     R1
3459  024032  122777  000377  155326  1$:     CMPB    #377,@KMAD2     ;FINISHED INSTRUCTION?
3460  024040  001403                          BEQ     2$
3461  024042  005201                          INC     R1              ;TIME OUT?
3462  024044  001372                          BNE     1$
3463  024046  000411                          BR      10$
3464
```

```
3465  024050  032777  000020  155304  2$:     BIT     #BIT4,@KMADO    ;FAST PATH READ?
3466  024056  001403                          BEQ     3$
3467  024060  005201                          INC     R1              ;NO - TIME OUT?
3468  024062  001372                          BNE     2$
3469  024064  000402                          BR      10$             ;YES - REPORT AN ERROR
3470
3471  024066  012601                  3$:     MOV     (SP)+,R1        ;RESTORE R1
3472  024070  000207                          RTS     PC              ;EXIT
3473
3474  024072                          10$:
3475  024072  104401  024100                  TYPE    ,65$            ;;TYPE ASCIZ STRING
3476  024076  000407                          BR      64$             ;;GET OVER THE ASCIZ
3477                                  ;;65$:  .ASCIZ  <200>#LPA-11 FAULT#
3478  024116                          64$:
3479
3480  024116  000000                  11$:    HALT                    ;LPA-11 FAULT RUN LPA-11
3481  024120  000776                          BR      11$             ;DIAGNOSTICS.
3482
3483
3484
3485                                          ;*
3486                                          ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
3487                                          ;*A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.
3488                                          ;*
3489                                          ;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED
3490                                          ;* BEFORE. IF NOT WE HAVE TO INITIALIZE THE LPA WITH
3491                                          ;* THAT ADDRESS.
3492                                          ;* WHEN THE ADDR. IS KNOWN BY THE LPA, DO THE OUTPUT BY
3493                                          ;* $TLKW
3494                                          ;*
3495
3496  024122  010046                  $OUTLP: MOV     R0,-(SP)        ;SAVE R0
3497  024124  010146                          MOV     R1,-(SP)        ;SAVE R1
3498
3499  024126  012700  001410                  MOV     #.DVLS,R0       ;PROGRAM DEFINED LIST.
3500  024132  005001                          CLR     R1
3501  024134  005710                  1$:     TST     (0)             ;TERMINATOR REACHED?
3502  024136  001421                          BEQ     10$             ;YES NEXT STEP.
3503  024140  027520  000000                  CMP     @(5),(0)+       ;MATCH WITH ADDR IN LIST?
3504  024144  001402                          BEQ     2$
3505  024146  005201                          INC     R1
3506  024150  000771                          BR      1$
3507
3508  024152  010137  024170          2$:     MOV     R1,3$           ;SAVE OFFSET, DEVICE KNOWN.
3509  024156  005725                          TST     (5)+
3510  024160  013537  024172                  MOV     @(5)+,4$        ;GET DATA TO BE WRITTEN
3511  024164  004537  023544                  JSR     R5,$TLKW        ;DO WRITE
3512  024170  000000                  3$:     .WORD   0               ;DEVICE OFFSET
3513  024172  000000                  4$:     .WORD   0               ;DATA TO BE WRITTEN.
3514  024174  012601                          MOV     (SP)+,R1
3515  024176  012600                          MOV     (SP)+,R0
3516  024200  000205                          RTS     R5
3517  024202  017520  000000          10$:    MOV     @(5),(0)+       ;SAVE ADDR.
3518  024206  005010                          CLR     (0)
```

```
3519  024210  004537  022524                  JSR     R5,$LPAI
3520  024214  001410                          .WORD   .DVLS
3521  024216  000755                          BR      2$
3522
3523                                          ;*
3524                                          ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE
3525                                          ;*TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
3526                                          ;*
3527                                          ;*FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN
3528                                          ;*USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
3529                                          ;*WITH THE NEW ADDR.
3530                                          ;*WHEN THE ADDR IS KNOWN WE CAN DO oUTPUT THROUGH
3531                                          ;*$TLkR
3532                                          ;*      CALL THROUGH    MOVEI   DATA,ADDR.
3533                                          ;*              WHICH EQUALS:
3534                                          ;*                      JSR     R5,$INLP
3535                                          ;*                      .WORD   XX      ADDR OF DEVICE
3536                                          ;*                      .WORD   YY      ADDR TO STORE READ DATA.
3537
3538  024220  010046          $INLP:  MOV     R0,-(SP)        ;SAVE R0
3539  024222  010146                  MOV     R1,-(SP)        ;SAVE R1
3540
3541  024224  012700  001410          MOV     #.DVLS,R0       ;PROG DEFINED ADDR. LIST.
3542  024230  005001                  CLR     R1
3543  024232  005710          1$:     TST     (0)             ;EOL REACHED?
3544  024234  001420                  BEQ     10$             ;YES - DEFINE NEW ADDR.
3545
3546  024236  027520  000000          CMP     @(5),(0)+       ;ADDR. MATCH?
3547  024242  001402                  BEQ     2$
3548  024244  005201                  INC     R1
3549  024246  000771                  BR      1$
3550
3551  024250  010137  024262  2$:     MOV     R1,3$           ;SAVE LIST OFFSET
3552  024254  005725                  TST     (5)+
3553  024256  004537  023660          JSR     R5,$TLKr        ;GO READ DEVICE
3554          024262          $OFS=.
3555  024262  000000          3$:     .WORD   0               ;OFFSET OF DEVICE
3556
3557  024264  013735  024024          MOV     $DATR,@(5)+     ;STORE DATA.
3558  024270  012601                  MOV     (SP)+,R1        ;RESTORE R1
3559  024272  012600                  MOV     (SP)+,R0        ;RESTORE R2
3560  024274  000205                  RTS     R5              ;EXIT
3561
3562  024276  017520  000000  10$:     MOV     @(5),(0)+
3563  024302  005010                  CLR     (0)
3564  024304  004537  022524          JSR     R5,$LPAI
3565  024310  001410                  .WORD   .DVLS
3566  024312  000756                  BR      2$
3567                                          ;*
3568                                          ;*$TOUT ROUTINE USED TO WATCH IF
3569                                          ;*      WE'RE IN A LOOP TOO-LONG
3570                                          ;*      CALL=   JSR R5, $TOUT
3571                                          ;*              ERROR X     ;RETURNS HERE ON TIMEOUT
3572                                          ;*              BR
```

```
3573                                                    ;*              ;RETURNS HERE NO ERROR
3574                                                    ;*
3575
3576  024314  020537  024350          $TOUT:  CMP     R5,$SAD         ;SAME ADDR?
3577  024320  001405                  BEQ     1$
3578  024322  010537  024350          MOV     R5,$SAD         ;NO-SAVE THIS ADDR.
3579  024326  005037  024352          CLR     $CNT            ;CLR CNT AT ADDR.
3580  024332  000403                  BR      2$
3581  024334  005237  024352          1$:     INC     $CNT            ;OVERFLOW?
3582  024340  100402                  BMI     3$              ;YES-ERROR RETURN
3583  024342  062705  000004          2$:     ADD     #4,R5           ;NO-NON ERROR RETURN
3584  024346  000205                  3$:     RTS     R5              ;RETURN.
3585
3586  024350  000000          $SAD:   .WORD   0               ;CONTAINS LOOP ADDR.
3587  024352  000000          $CNT:   .WORD   0               ;# OF TIMES AT ADDR.
3588
3589                                          ;*
3590                                          ;* THIS ROUTINE REPLACES WHAT THE USER WOULD ORDINARILY
3591                                          ;*USE FOR A RESET.  FIRST,WE DO A RESET INSTRUCTION.
3592                                          ;*THEN WE CLR ".DVLST" WHICH FORCES US TO rESET BOTH THE
3593                                          ;*KMC AND DMC AS SOON AS A DEVICE IS REFERENCED.
3594                                          ;*
3595                                          ;*      CALL=JSR        PC,$RESET       ;REPLACES "RESET INSTRUCTION
3596                                          ;*              ;RETURNS HERE.
3597                                          ;*
3598  024354  000005          $RESET: RESET                   ;RESET THE WORLD.
3599
3600                                  ;*      MOV     #2$,1$  ;/READ DEVICE REG 2$,PUT DATA IN 1$.
3601  024366  005737  023402                  TST     $AERR           ;IF NO ERROR,LOOP
3602  024372  001004                  BNE     10$             ;THERE WAS AN ERROR.
3603  024374  062737  000002  024410  ADD     #2,2$           ;UPDATE DEVICE ADDR.
3604                                                          ;YOU SEE, WE HAVE TO PROTECT OUR SELF!
3605                                                          ;IF 2$ CONTAINED A VALID ADDR,WE
3606                                                          ;MUST KEEP TRYING UNTIL WE GENERATE
3607                                                          ;AN INVALID ADDR.
3608  024402  000764                  BR      $RESET
3609  024404                  10$:
3610  024404  000207                  RTS     PC
3611  024406  000000          1$:     .WORD   0               ;JUNK LOC.
3612  024410  160000          2$:     .WORD   160000          ;DUMB ADDR. FORCES INIT OF DMC/KMC.
3613
3614
3615                                  ;
3616                                  ;SDELAY- ROUTINE TO GIVE A MINOR DELAY.
3617                                  ;        IS NOT TIME DEPENDENT CODE SENCE
3618                                  ;        NOT USED TO GET SPECIFIC TIME BUT
3619                                  ;        JUST A LITTLE DELAY.
3620
3621                                  ;       THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
3622                                  ;       THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
3623                                  ;
3624                                  ;
3625                                  ;       CALL=   JSR PC,  SDELAY
3626                                  ;
```

```
3627  024412                                SDELAY:
3628  024412  005737  024474                        TST     RTCCSR          ;CLOCK PRESENT?
3629  024416  100016                        BPL     10$
3630  024420  012737  000002  024464        MOV     #2,TIME
3631  024426  052777  000115  000040        BIS     #115,@RTCCSR    ;START CLOCK
3632  024434  005037  177776                CLR     PS
3633  024440  005737  024464        1$:     TST     TIME
3634  024444  001375                        BNE     1$
3635  024446  005077  000022                CLR     @RTCCSR         ;STOP CLOCK
3636
3637  024452  000207                        RTS PC
3638  024454  105237  024464        10$:    INCB    TIME
3639  024460  001375                        BNE     10$
3640  024462  000207                        RTS     PC
3641
3642  024464  000000                TIME:   .WORD   0
3643
3644  024466  005337  024464        CLKINT: DEC     TIME
3645  024472  000002                        RTI
3646  024474  000000                RTCCSR: .WORD   0               ;CLOCK CSR IF USED.
3647                                        ;*
3648                                        ;*THIS MACRO ALLOWS THE OPERATOR TO TALK TO
3649                                        ;*ANY DEVICE ON THE I/O BUS
3650                                        ;*USER MUST START AT THIS ADDR.
3651                                        ;*HE MUST SAY EITHER "E" FOR EXAMINE, OR "D" FOR DEPOSIT.
3652                                        ;*"E" IS DEFAULT.
3653                                        ;*NEXT, HE MUST SUPPLY AN ADDR.
3654                                        ;*NOTE IF ADDR. IS NOT FOUND ON I/O BUS, A !!ALT
3655                                        ;*WILL OCCUR.
3656
3657
3658  024476                                $UTK:
3659  024476  005037  001410                        CLR     .DVLS
3660  024502                                21$:
3661  024502  104401  024510                        TYPE    ,65$            ;;TYPE ASCIZ STRING
3662  024506  000405                        BR      64$             ;;GET OVER THE ASCIZ
3663                                ;;65$:  .ASCIZ  <200>#E OR D?#
3664  024522                                64$:
3665  024522  105777  154416        1$:     TSTB    @$TKS
3666  024526  100375                        BPL     1$
3667  024530  117737  154412  024652        MOVB    @$TKB,20$       ;GET INPUT
3668  024536  104401  024652                TYPE,   20$             ;ECHO, NEXT MESSAGE.
3669  024542  142737  000240  024652        BICB    #240,20$        ;STRIP PARITY, LC
3670  024550  104412                        RDOCT                   ;GET ADDR.
3671  024552  012637  024650                MOV     (SP)+,14$
3672  024556  123727  024652  000104        CMPB    20$,#'D         ;DEPOSIT?
3673  024564  001411                        BEQ     10$
3674
3675  024566  004537  024220                JSR     R5,$INLP        ;GET DATA
3676  024572  024650                2$:     .WORD   14$
3677  024574  024606                        .WORD   5$
3678
3679  024576  013746  024606                MOV     5$,-(SP)        ;;SAVE 5$ FOR TYPEOUT
3680  024602  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
3681  024604  000736                          BR      21$             ;LOOP.
3682  024606  000000                  5$:     .WORD   0
3683
3684  024610                          10$:
3685  024610  104401  024616                  TYPE    ,67$            ;;TYPE ASCIZ STRING
3686  024614  000404                          BR      66$             ;;GET OVER THE ASCIZ
3687                                  ;;67$:  .ASCIZ  <200>#DATA= #
3688  024626                          66$:
3689  024626  104412                          RDOCT
3690  024630  012637  024646                  MOV     (SP)+,13$
3691
3692  024634  004537  024122          11$:    JSR     R5,$OUTLP       ;OUTPUT ROUTINE.
3693  024640  024650                  12$:    .WORD   14$             ;DEVICE ADDR.
3694  024642  024646                          .WORD   13$             ;DATA
3695  024644  000716                          BR      21$
3696
3697  024646  000000                  13$:    .WORD   0
3698  024650  000000                  14$:    .WORD   0
3699  024652  100001  042504  044526  20$:    .ASCIZ  <1><200>#DEVICE ADDR= #
3700  024660  042503  040440  042104
3701  024666  036522  000040
3702                                          .EVEN
3703
3704
3705
3706                                  ;
3707                                  ;THIS ROUTINE LOOKS THROUGH CURENT .DVLS FOR A/D ADDR.
3708                                  ;IF UNFOUND,GENERATES IT. THIS ROUTINE'S WHOLE PURPOSE IS
3709                                  ;TO SET UP THE USER PROGRAM TO LINK TO FILE "DRLPX2" FOR
3710                                  ;SAMPLE TAKEING PURPOSES.
3711                                  ;       TO TAKE SAMPLES, THE USER PROGRAM MUST SET UP
3712                                  ;       A/D CSR IN BSEL 4 AND 5.
3713                                  ;       (2) HE MUST CALL THIS ROUTINE:
3714                                  ;               JSR     R5,$PUTS        ;CALL SET UP ROUTINE.
3715                                  ;               .WORD   ADCSR           ;ADDR. OF A/D CSR.
3716                                  ;               ;RETURNS HERE ;KMC BSEL 3,6,7 PERMINENTLY SET UP
3717                                  ;                               ;(UNTILL ONE DOES A RESET)
3718                                  ;
3719                                  ;       (3)THE USER MUST PUT CODE 006 INTO KMC REG 2 TO
3720                                  ;          START CONVERSION  CAUTION*DO WITH MOVB INSTR.!
3721                                  ;       (4)MONITOR KMC REG 2 FOR CODE 377 (DRLPX2 IS DONE)
3722                                  ;       (5)READ KMC REG 4,5 FOR A/D RESULT.
3723                                  ;       (6) TO TAKE MORE SAMPLES,SIMPLY PUT A/D CSR INTO
3724                                  ;          BSEL 4,5 AND CODE 6 INTO BSEL 2.
3725
3726  024672  012537  024702          $PUTS:  MOV     (5)+,1$                 ;GET ADDR OF ADDR. OF A/D
3727  024676  004537  024220                  JSR     R5,$INLP
3728  024702  000000                  1$:     .WORD   0
3729  024704  025000                          .WORD   10$
3730  024706  113777  024262  154462          MOVB    $OFS,@KMAD6
3731  024714  113777  024262  154456          MOVB    $OFS,@KMAD7
3732  024722  013737  024702  024742          MOV     1$,2$
3733  024730  062737  000002  024742          ADD     #2,2$
3734  024736  004537  024220                  JSR     R5,$INLP
```

```
3735  024742  000000                  2$:     .WORD   0
3736  024744  025000                          .WORD   10$
3737  024746  113777  024262  154414          MOVB    $OFS,@KMAD3
3738  024754  152777  000340  154414          BISB    #340,@KMAD6
3739  024762  152777  000300  154410          BISB    #300,@KMAD7
3740  024770  152777  000300  154372          BISB    #300,@KMAD3
3741  024776  000205                          RTS     R5
3742  025000  000000                  10$:    .WORD   0
3743
3744          000001                  .END
```

```
ABASE = 167770            60*   282    323
ACDW1  = 000000          282    325
ACDW2  = 000000          282
ACPUOP= 000000           282    297
ADBR    015466          2244   2279*
ADCS    015462          2223   2226   2232   2235   2240   2277*
ADCS1   015464          2218   2221   2278*
ADDW0  = 000000          282
ADDW1  = 000000          282
ADDW10= 000000           282
ADDW11= 000000           282
ADDW12= 000000           282
ADDW13= 000000           282
ADDW14= 000000           282
ADDW15= 000000           282
ADDW2  = 000000          282
ADDW3  = 000000          282
ADDW4  = 000000          282
ADDW5  = 000000          282
ADDW6  = 000000          282
ADDW7  = 000000          282
ADDW8  = 000000          282
ADDW9  = 000000          282
ADEVCT= 000000           282    288
ADEVM  = 000000          282    324
AENV   = 000000          282    293
AENVM  = 000000          282    294
AFATAL= 000000           282    285
AMADR1= 000000           282    310
AMADR2= 000000           282    314
AMADR3= 000000           282    317
AMADR4= 000000           282    320
AMAMS1= 000000           282    304
AMAMS2= 000000           282    312
AMAMS3= 000000           282    315
AMAMS4= 000000           282    318
AMSGAD= 000000           282    290
AMSGLG= 000000           282    291
AMSGTY= 000000           282    284
AMTYP1= 000000           282    305
AMTYP2= 000000           282    313
AMTYP3= 000000           282    316
AMTYP4= 000000           282    319
APASS  = 000000          282    287
APRIOR= 000000           282
APTCSU= 000040          2791   3104*
APTENV= 000001          2560   2784   3060   3102*
APTSIZ= 000200           520   3101*
APTSPO= 000100          2786   3062   3103*
ASWREG= 000000           282    295
ATESTN= 000000           282    286
AUNIT  = 000000          282    289
AUSWR  = 000000          282    296
AVECT1= 100300            61*   282    321    423    424
```

```
AVECT2= 000000  282     322
BASEBA  001452  432#    540*    546     711     1847
BASEBR  001456  434#    543*    544*    741
BASEIV  001454  433#    541*    542*    712     1848
BEGIN   001546  195     470#    645     2003
BEGIN1  001602  196     479#
BIT0  = 000001  157#    842     863     1194    1238    1276    1719    1789
BIT00 = 000001  147#    157
BIT01 = 000002  146#    156
BIT02 = 000004  145#    155
BIT03 = 000010  144#    154
BIT04 = 000020  143#    153
BIT05 = 000040  142#    152
BIT06 = 000100  141#    151
BIT07 = 000200  140#    150
BIT08 = 000400  139#    149     2498
BIT09 = 001000  138#    148     2506    2571
BIT1  = 000002  156#   3264
BIT10 = 002000  137#   1056     1715    1785
BIT11 = 004000  136#   2513     3204
BIT12 = 010000  135#    638     1147
BIT13 = 020000  134#   2223     2555
BIT14 = 040000  133#   1012     1964    2484    3196
BIT15 = 100000  132#    996     1693    3204
BIT2  = 000004  155#
BIT3  = 000010  154#
BIT4  = 000020  153#   3465
BIT5  = 000040  152#   3208     3421    3438
BIT6  = 000100  151#   1041
BIT7  = 000200  150#   1025     1674    1721
BIT8  = 000400  149#
BIT9  = 001000  148#
BPTVEC= 000014  164#
BRLEV1  001526  458#
BRLEV2  001530  459#   1194*    1197    1225    1230*
BRLEV3  001532  460#    652     1195*   1196*   1207    1223    1276*   1294    1291    1307    1310*   1719*   1723
                1736    1741    1746    1765    1770    1779*   1789*   1793    1806    1809    1814    1832*   2460
BUFFER  022522 2013     2038    3149#
BUSTRP  015647  560     2001    2300#
BYPASS  013134 1839#
BYPAS1  013222 1842     1846    1851#   1888
CKSWR = 104407 2483     2546    2570    3144#
CLKINT  024466 3644#
COULTJ  001556  199     473#
COULTR  014036  198    2005#
CPU     001460  435#   2021
CR    = 000015   72#   2830    2840
CRLF  = 000200   73#    602    2801    2840
DDISP = 177570   79#    264     508
DEPCNT  016413  700    2367#
DF0     017410  348     354     360     366     372     378     384     390     396    2463#
DH1     017055  346    2422#
DH10    017266  388     394    2447#
DH2     017125  352    2429#
```

```
DH3     017173             358    2436#
DH4     017241             364     370     376     382    2443#
DIOBRL  001524             457#    741#
DISPLA  001142             264#    508#    516#   2528#   2550#
DISPRE  000174             192#    516
DRADD   001466             440#    711#    726     727     729     731    1843#   1847#   2456    2458    2460
DRIV    001470             441#    683#    712#    733     734     736     738    1844#   1848#
DRLPX2= ******  G           14#   3194
DRT21   012250            1059    1710#
DSWR  = 177570              78#    263     507
DT1     017346             347     353     359    2456#
DT10    017372             389     395    2460#
DT4     017362             365     371     377     383    2458#
EMTVEC= 000030             167#    491#    492#
EM1     016435             345    2371#
EM10    016715             387    2404#
EM11    017000             393    2413#
EM2     016466             351    2376#
EM3     016516             357    2380#
EM4     016547             363    2385#
EM5     016601             369    2390#
EM6     016634             375    2395#
EM7     016661             381    2399#
ERRVEC= 000004             160#    505     506#    517#    545#    559#    570#   2489   2490#   2492#   2495#
EXTCNT  013760            1964#   1986#   1991#
EXTTMP  013756            1974    1976    1984    1986    1990#   2020#   2024#   2044#   2054#
EXTTST  013600             197    1962#
GNS   = ****** U           191     580     601     608     670     676    3136    3137    3138    3139    3140    3142    3144
                           3145    3146    3147    3187    3241    3245    3249    3477    3663    3687
GRBHIO  001506             450#    731#    732#
GRDAI   001502             448#    727#    728#    760     996    1067    1073    1088    1094    1109    1115    1130    1136
                           1157    1164    1174    1181    1207    1223    1245    1258    1265    1268    1283    1300    1307
                           1310    1327    1338    1341    1352    1355    1366    1369    1380    1383    1394    1397    1408
                           1411    1422    1425    1436    1439    1450    1453    1464    1467    1478    1481    1506    1515
                           1518    1527    1530    1539    1542    1551    1554    1563    1566    1575    1578    1587    1590
                           1599    1602    1611    1614    1623    1626    1635    1638    1658    1665    1702    1731    1770
                           1773    1801    1974    1976    1984    1986    2028    2033
GRDIO   001504             449#    729#    730#    762     771     773     785     788     800     802     814     816     830
                            832     847     850     852     868     871     873     888     890     893     895     898     900
                            903     905     908     910     913     915     918     920     923     925     928     930     940
                            942     945     947     950     952     955     957     960     962     965     967     970     972
                            975     977     980     982     993    1064    1071    1085    1092    1106    1113    1127    1134
                           1154    1159    1162    1176    1179    1205    12 3    1221    1242    1253    1256    1280    1294
                           1297    1324    1330    1333    1336    1344    1347    1350    1358    1361    1364    1372    1375
                           1378    1386    1389    1392    1400    1403    1406    1414    1417    1420    1428    1431    1434
                           1442    1445    1448    1456    1459    1462    1470    1473    1476    1503    1510    1513    1522
                           1525    1534    1537    1546    1549    1558    1561    1570    1573    1582    1585    1594    1597
                           1606    1609    1618    1621    1630    1633    1655    1661    1680    1699    1728    1736    1739
                           1741    1744    1746    1749    1765    1768    1798    1806    1809    1812    1814    1817    1819
                           1969    1972    1979    1982
GRIVA   001514             453#    733#    745#
GRIVB   001520             455#    736#    737#    747#
GRIVSA  001516             454#    734#    735#    746#
GRIVSB  001522             456#    738#    739#    748#
```

```
GRSTAT  001500           447*   726*   758    999    1001   1012   1015   1017   1028   1030   1041   1044   1046
                        1678   1683   1697   1705   1734   1750*  1753   1756   1774*  1775   1804   1825   2011
                        2018
GTSWR = 104406           596   3142*
HT    = 000011            70*  2799   2840
H322J   001570           200    476*
INADR   014432          2063   2066   2069   2087*  2137   2140   2143
INADRH  015531          2090   2286*
INDADR  015574          2093   2292*
INTBIT  001512           452*   613*   620*   629    1723   1793
IOTEST  003332           576    616    623    653     710*  2747
IOTST1  003412           723*
IOTTS1  003552           725    741*
IOTVEC= 000020           165*   489*   490*
JUMPER  001542           464*   471*   473*   476*   610    617
KMAD0   001362           406*   530   3196*  3204*  3208   3264   3324*  3326*  3328*  3329*  3336*  3341*  3421
                        3438   3465
KMAD1   001364           409*   531
KMAD2   001366           411*  3213*  3226   3266*  3280   3294*  3307   3375*  3380*  3384*  3406*  3423*  3440*
                        3459
KMAD3   001370           413*  3737*  3740*
KMAD4   001372           415*  3228   3233   3257*  3259*  3261*  3282   3325*  3330*  3333*  3342*  3374*  3378*
                        3383*  3405*  3425   3442
KMAD5   001374           417*
KMAD6   001376           419*  3327*  3337   3730*  3738*
KMAD7   001400           421*   535   3731*  3739*
LF    = 000012            71*  2834   2840
LOCA    014364          2065   2068   2071   2073*  2085
LOCBOX  014312           201   2060*
LOCHDR  015470          2062   2280*
LOC1    001472           444*  2064   2076   2138   2152   2156   2160   2164   2168
LOC1Y   014616          2064   2073   2129*  2138   2147
LOC2    001474           445*  2067   2080   2141   2174   2178   2182   2186   2190
LOC2Y   014620          2067   2077   2130*  2141   2169
LOC3    001476           446*  2070   2084   2144   2196   2200   2204   2208   2212
LOC3Y   014622          2070   2081   2131*  2144   2191
LOUP1   014544          2075   2079   2083   2112*
LPADH   001374           416*
LPADL   001372           414*
LPCI    001362           405*
LPCO    001366           410*
LPMR    001364           408*
LPMS1   001376           418*
LPMS2   001400           420*
LPSO    001370           412*
MINSIN  001536           462*   614*   621*   632    639    743   1169
MSGRUS  015730          2042*  2056*  2057   2309*
MSPNT1  015737          2043   2311*
NBEXT   001464           438*   713*  1839   1845*  1849*
NMBEXT  001462           437*   568*   603    713    1849
NOTLCH  001510           451*   612*   619*   626    704   1150   1195   1200   1246   1284   1290   1318   1498
ODDJMP  001534           461*   742*   743*   744*  1073   1089   1094   1110   1115   1131   1136   1145   1148
                        1198   1248   1286   1319   1497   1665
PC    =%000007            91*   571*   710*   714*   723*   724*   740*   785*  1662*  1869*  1872*  1882*  1887
```

```
SWINTB  016036            628    2326#
SWNLB   015740            625     692    2315#
SWPOSB  016134            631    2337#
SWR     001140            263#    485     507*    509     515*    522*    558     561     594     666     679     693     697*
                          702    1056    1715    1785    1841    2484    2498    2500    2506    2513    2555    2567    2571
                         2722    2735*   2888    2925*
SWREG   000176            193#    515     594     693    2868    2901
SWTRAB  016230            634    2347#
SW0    = 000001           129#
SW00   = 000001           119#    129
SW01   = 000002           118#    128
SW02   = 000004           117#    127
SW03   = 000010           116#    126
SW04   = 000020           115#    125
SW05   = 000040           114#    124
SW06   = 000100           113#    123
SW07   = 000200           112#    122
SW08   = 000400           111#    121
SW09   = 001000           110#    120
SW1    = 000002           128#
SW10   = 002000           109#
SW11   = 004000           108#
SW12   = 010000           107#   1841
SW13   = 020000           106#    558     666
SW14   = 040000           105#
SW15   = 100000           104#    561     679
SW2    = 000004           127#
SW3    = 000010           126#
SW4    = 000020           125#
SW5    = 000040           124#
SW6    = 000100           123#
SW7    = 000200           122#
SW8    = 000400           121#
SW9    = 001000           120#
TALK    003246            624     627     630     633     650     689#
TBITVE= 000014            162#
TIME    024464           3630*   3633    3638*   3642#   3644*
TKVEC  = 000060            169#
TPVEC  = 000064            170#
TRANST  001540            463#    615*    622*    635     641     744
TRAPVE= 000034            168#    493*    494*   1994*   1995*
TRTVEC= 000014            163#
TSTNUM  017426           2456    2458    2460    2466#   2547*
TST1    003630            752#
TST10   004336            861#
TST11   004444            884#
TST12   004746            931     936#
TST13   005250            983     989#
TST14   005346           1002    1008#
TST15   005426           1018    1024#
TST16   005470           1031    1037#
TST17   005550           1047    1055#
TST2    003704            766#
TST20   005702           1075    1081#
```

```
TST21    006014          1096    1102#
TST22    006126          1117    1123#
TST23    006240          1138    1143#
TST24    006506          1146    1191#
TST25    006714          1236#
TST26    007112          1274#
TST27    007332          1315#
TST3     003754           774     779#
TST30    010612          1488    1494#
TST31    011742          1645    1650#
TST32    012062          1667    1673#
TST33    012150          1684    1692#
TST34    012250          1706    1714#
TST35    012634          1716    1784#
TST36    013124          1786    1837#
TST4     004032           789     795#
TST5     004102           803     809#
TST6     004152           817     823#
TST7     004234           840#
TYPDS  = 104405          1877    3140#
TYPE   = 104401           560     578     588     606     668     674     691     695     699    1875    1878    1951    2001
                         2012    2057    2062    2090    2093    2136    2558    2592    2609    2611    2614    2616    2620
                         2627    2692    2744    2804    2899    2900    2903    2916    2927    2946    2999    3005    3010
                         3014    3019    3020    3022    3025    3029    3136#   3185    3239    3243    3247    3475    3661
                         3668    3685
TYPOC  = 104402           673    2092    2600    2624    2902    3137#   3680
TYPON  = 104404          3139#
TYPOS  = 104403           604    3138#
UNEXPT   003326           707#    745     747
VECTOR   001402           423#
VECTPS   001404           424#
VECTRP   003022           569     656#
VERSN    001406           426#
W1       023652          3373#   3388#
W2       023654          3377#   3389#
W3       023656          3382#   3390#
XLOADJ   015152          2149    2153    2157    2161    2165    2171    2175    2179    2183    2187    2193    2197    2201
                         2205    2209    2215#
XLOHDR   015612          2136    2295#
XLO1AD   014624           202    2134#
XLO10    014676          2139    2142    2145    2147#   2213
$AERR    023402           551    3192#   3285#   3311#   3601
$APTHD   001000           223     229#
$ASTAT=  ****** U        3082    3097
$ATYC    022216          3053    3055#
$ATY1    022172          3051#
$ATY3    022200          2789    3052#
$ATY4    022210          2563    3054#
$AUTOB   001134           260#    598#   2896    3045
$BASE    001244           323#    540
$BDADR   001122           255#
$BDDAT   001126           257#    773     788     802     816     832     852     873#    874     930     982    1001    1017
                         1030    1046    1073#   1074    1094#   1095    1115#   1116    1136#   1137    1164    1181    1207#
                         1208    1223#   1225    1258    1300    1483#   1487    1640#   1644    1665#   1666    1683    1705
```

```
$ICNT   001104   247#  2517*  2518   2520*  2531
$ILLUP  020630  2714   2730   2749#
$INLP   024220   551    773    788    802    816    832    852    873    890    895    900    905    910
                 915    920    925    930    942    947    952    957    962    967    972    977    982
                1001   1017   1030   1046   1073   1094   1115   1136   1159   1164   1176   1181   1207
                1218   1223   1258   1265   1300   1307   1338   1352   1366   1380   1394   1408   1422
                1436   1450   1464   1478   1515   1527   1539   1551   1563   1575   1587   1599   1611
                1623   1635   1665   1683   1702   1705   1736   1741   1746   1756   1765   1770   1806
                1814   1825   1974   1984   2018   2028   2218   2223   2232   2240   2244   3538#  3601
                3675   3727   3734
$INTAG  001135   261#  2928   3045
$ITEMB  001114   251#  2554*  2562   2583   2595
$LF     001166   277#  2583   2840   3029   3039
$LFLG   022435  3092*  3098#
$LOAD   023404  3193   3321#
$LPADR  001106   248#   501*  2508*  2524*  2529   2531
$LPAI   022524  3167#  3519   3564
$LPERR  001110   249#   502*   825*   841*   862*  1192*  1237*  1275*  1718*  1788*  2508   2525*  2531
                2573
$LPW    024026  3372   3376   3381   3385   3404   3407   3424   3441   3457#
$MADR1  001222   310#
$MADR2  001226   314#
$MADR3  001232   317#
$MADR4  001236   320#
$MAIL   001170   231    235    283#   519    592   2523   2560   2784
$MAMS1  001220   304#
$MAMS2  001224   312#
$MAMS3  001230   315#
$MAMS4  001234   318#
$MBADR  001002   231#
$MFLG   022434  3052*  3058   3093*  3097#
$MNEW   022160  2903   3043#
$MSGAD  001204   290#  3068*  3071
$MSGLG  001206   291#  3073*
$MSGTY  001170   284#  3066   3074*  3086   3090*
$MSWR   022147   695   2900   3041#
$MTYP1  001221   305#
$MTYP2  001225   313#
$MTYP3  001231   316#
$MTYP4  001235   319#
$MXCNT  017706  2521   2531#
$NULL   001154   269#  2811   2840
$NWTST= 000001   749#   763#   776#   792#   806#   820#   837#   858#   881#   933#   986#  1005#  1021#
                1034#  1052#  1078#  1099#  1120#  1140#  1188#  1233#  1271#  1312#  1491#  1647#  1670#
                1689#  1711#  1781#  1834#
$OCNT   020460  2664*  2693*  2706#
$OF8    024262  3554#  3730   3731   3737
$OMODE  020462  2659*  2663*  2668   2671*  2682*  2708#
$OUTLP  024122   758    760    762    771    785    800    814    830    847    850    868    871    888
                 893    898    903    908    913    918    923    928    940    945    950    955    960
                 965    970    975    980    993    996    999   1012   1015   1028   1041   1044   1064
                1067   1071   1085   1088   1092   1106   1109   1113   1127   1130   1134   1154   1157
                1162   1174   1179   1205   1221   1242   1245   1253   1256   1268   1280   1283   1294
                1297   1310   1324   1327   1330   1333   1336   1341   1344   1347   1350   1355   1358
```

```
                        980    990*   993*   996   1009*  1012   1038*  1041   1061*  1064*  1067   1068*  1071
                       1082*  1085*  1088   1103*  1106*  1109   1124*  1127*  1130   1154*  1157   1159*  1162
                       1171*  1174   1176*  1179   1218*  1221   1239*  1242*  1245   1253*  1256   1265*  1268
                       1277*  1280*  1283   1294*  1297   1307*  1310   1321*  1324*  1327   1330*  1333*  1336
                       1338   1341   1344*  1347*  1350   1352   1355   1358*  1361*  1364   1366   1369   1372*
                       1375*  1378   1380   1383   1386*  1389*  1392   1394   1397   1400*  1403*  1406   1408
                       1411   1414*  1417*  1420   1422   1425   1428*  1431*  1434   1436   1439   1442*  1445*
                       1448   1450   1453   1456*  1459*  1462   1464   1467   1470*  1473*  1476   1478   1481
                       1500*  1503*  1506   1507*  1510*  1513   1515   1518   1519*  1522*  1525   1527   1530
                       1531*  1534*  1537   1539   1542   1543*  1546*  1549   1551   1554   1555*  1558*  1561
                       1563   1566   1567*  1570*  1573   1575   1578   1579*  1582*  1585   1587   1590   1591*
                       1594*  1597   1599   1602   1603*  1606*  1609   1611   1614   1615*  1618*  1621   1623
                       1626   1627*  1630*  1633   1635   1638   1652*  1655*  1658*  1661   1675*  1678   1680
                       1694*  1697   1699   1702   1725*  1728*  1731*  1734   1736*  1739   1741*  1744   1746*
                       1749   1753   1756   1765*  1768   1770*  1773   1795*  1798*  1801*  1804   1806*  1809*
                       1812   1814*  1817   1819*  1822   1825   1966*  1969*  1972   1976*  1979*  1982   2008*
                       2011   2018   2028   2030*  2033   2218*  2221   2223*  2226   2232*  2235   2240   2244
$TN     = 000037        30*    59     749    753*   763    767*   774    776    780*   789    792    796*   803
                        806    810*   817    820    824*   837    841*   858    862*   881    885*   931    933
                        937*   983    986    990*  1002   1005   1009*  1018   1021   1025*  1031   1034   1038*
                       1047   1052   1056*  1075   1078   1082*  1096   1099   1103*  1117   1120   1124*  1138
                       1140   1144*  1146   1188   1192*  1233   1237*  1271   1275*  1312   1316*  1488   1491
                       1495*  1645   1647   1651*  1667   1670   1674*  1684   1689   1693*  1706   1711   1715*
                       1716   1781   1785*  1786   1834   1838*
$TOUt   024314        3215   3269   3296   3410   3427   3576*
$TPB    001152         268*  2829*  2840
$TPFLG  001157         272*  2778   2840
$TPS    001150         267*  2827   2840
$TRAP   022440         493   1994   3113*
$TRAP2  022462        3124*  3135
$TRP    = 000013      3128*  3137*  3138*  3139*  3140*  3141*  3142   3143*  3144   3145*  3146*  3147*  3148*
$TRPAD  022474        3118   3135*
$TSTM   001004         232*
$TSINM  001102         245*   754*  1058*  1865*  2473   2500   2522*  2523   2528   2532   2547   2550   2583
$TTYIN  022126        2989   2990   3002   3020   3034   3038*
$TYpBN= ****** U      3141
$TYPOS  013354        1905*  3140
$TYPE   020704        2778*  3079   3128   3136
$TYPEC  021116        2808   2815   2822   2827*  2828   2932
$TYPEX  021164        2833   2835   2838*
$TYpOC  020262        2662*  3137
$TYPON  020276        2661   2664*  3139
$TYPOS  020236        2657*  3138
$UNIT   001202         289*
$UNITM  001010         234*
$USWR   001214         296*
$UTK    024476        3658*
$VECT1  001240         321*   541    543
$VECT2  001242         322*
$XTSTR  017442        2487*
$$GET4= 000000        1881*
$OFILL  020461        2658*  2662*  2672   2707*
$40CAT= ****** U      2484   2557
.       = 025002       187*   191*   207    208*   210*   212*   213*   219    220*   222*   224*   242*   278
```

```
                          429#    486     501     502     581#    602#    671#    1889    1893    1959#   2455#   2531    2532
                         2583    2630#   2726    2750    2759#   2840    2881    3038#   3039    3045#   3100#   3554    3664#
.DVLS   001410            428#    537*    3499    3520    3541    3565    3659*
.$ASTA= ****** U         3052    3055
.$X   = 001000            219#    224
```

.$POWE	30#	2710
.$RDOC	30#	2840
.$READ	30#	2878
.$SAVE	30#	
.$SCOP	30#	2468
.$SPAC	30#	
.$SWDO	30#	
.$TLKW	26#	3362
.$TOUT	431#	3567
.$TRAP	30#	3105
.$TYPD	30#	1893
.$TYPE	30#	2761
.$TYPO	30#	2632

	2888	2894	2914	2921	2933	2935	2965	2971	2974	2976	2990	3002	3190	3254	3331
	3337	3339	3503	3546	3576										
CMPB	592	2500	2504	2560	2784	2799	2801	2809	2830	2834	2896	2928	2994	3012	3016
	3026	3060	3226	3228	3233	3280	3307	3459	3672						
COM	873	890	895	900	905	910	915	920	925	942	947	952	957	962	967
	972	977	1196	1341	1355	1369	1383	1397	1411	1425	1439	1453	1467	1481	1518
	1530	1542	1554	1566	1578	1590	1602	1614	1626	1638					
DEC	567	1845	1869	1986	2022	2024	2054	2266	2602	3001	3644				
DECB	2682	2693	2813	2816											
EMT	66														
HALT	191	563	658	662	681	701	2002	2569	2580	2725	2749	2780	3353	3480	
INC	533	554	584	835	1867	1923	2104	2260	2517	2551	2688	2696	2733	2943	3090
	3184	3202	3235	3237	3285	3330	3342	3461	3467	3505	3548	3581			
INCB	2232	2522	2548	2836	3205	3256	3351	3638							
IOT	67														
JMP	195	196	197	198	199	200	201	202	576	616	623	645	653	684	1059
	1850	1852	1887	2003	2213										
JSR	551	571	624	627	630	633	650	710	714	723	724	758	760	762	771
	773	785	788	800	802	814	816	830	832	847	850	852	868	871	873
	888	890	893	895	898	900	903	905	908	910	913	915	918	920	923
	925	928	930	940	942	945	947	950	952	955	957	960	962	965	967
	970	972	975	977	980	982	993	996	999	1001	1012	1015	1017	1028	1030
	1041	1044	1046	1064	1067	1071	1073	1085	1088	1092	1094	1106	1109	1113	1115
	1127	1130	1134	1136	1154	1157	1159	1162	1164	1174	1176	1179	1181	1205	1207
	1218	1221	1223	1242	1245	1253	1256	1258	1265	1268	1280	1283	1294	1297	1300
	1307	1310	1324	1327	1330	1333	1336	1338	1341	1344	1347	1350	1352	1355	1358
	1361	1364	1366	1369	1372	1375	1378	1380	1383	1386	1389	1392	1394	1397	1400
	1403	1406	1408	1411	1414	1417	1420	1422	1425	1428	1431	1434	1436	1439	1442
	1445	1448	450	1453	1456	1459	1462	1464	1467	1470	1473	1476	1478	1481	1503
	1506	1510	1513	1515	1518	1522	1525	1527	1530	1534	1537	1539	1542	1546	1549
	1551	1554	1558	1561	1563	1566	1570	1573	1575	1578	1582	1585	1587	1590	1594
	1597	1599	1602	1606	1609	1611	1614	1618	1621	1623	1626	1630	1633	1635	1638
	1655	1658	1661	1662	1665	1678	1680	1683	1697	1699	1702	1705	1728	1731	1734
	1736	1739	1741	1744	1746	1749	1753	1756	1765	1768	1770	1773	1798	1801	1804
	1806	1809	1812	1814	1817	1819	1822	1825	1882	1963	1969	1972	1974	1976	1979
	1982	1984	1986	2006	2007	2011	2018	2028	2033	2037	2061	2063	2066	2069	2075
	2079	2083	2135	2137	2140	2143	2149	2153	2157	2161	2165	2171	2175	2179	2183
	2187	2193	2197	2201	2205	2209	2218	2221	2223	2226	2232	2235	2240	2244	2557
	2563	2789	2808	2815	2822	2932	3079	3193	3215	3258	3260	3262	3269	3296	3372
	3376	3381	3385	3404	3407	3410	3424	3427	3441	3511	3519	3553	3564	3601	3675
	3692	3727	3734												
MOV	473	476	479	483	487	489	490	491	492	493	494	495	496	497	501
	502	505	506	507	508	513	515	516	517	522	528	529	530	531	534
	538	539	540	541	545	546	548	568	569	570	603	613	619	620	622
	638	664	672	683	689	690	697	702	711	712	713	715	716	717	718
	726	727	729	731	733	734	736	738	741	745	746	747	748	753	754
	755	767	768	780	782	796	797	810	811	824	825	841	842	844	862
	863	865	885	937	990	993	996	1009	1012	1025	1038	1041	1061	1064	1068
	1082	1085	1088	1103	1106	1109	1124	1127	1130	1144	1147	1154	1171	1192	1194
	1195	1197	1237	1238	1239	1242	1253	1275	1276	1277	1280	1289	1294	1317	1320
	1321	1324	1330	1333	1338	1344	1347	1352	1358	1361	1366	1372	1375	1380	1386
	1389	1394	1400	1403	1408	1414	1417	1422	1428	1431	1436	1442	1445	1450	1456
	1459	1464	1470	1473	1478	1483	1496	1499	1500	1503	1507	1510	1515	1519	1522
	1527	1531	1534	1539	1543	1546	1551	1555	1558	1563	1567	1570	1575	1579	1582

```
        1587  1591  1594  1599  1603  1606  1611  1615  1618  1623  1627  1630  1635  1640  1651
        1652  1655  1658  1674  1675  1693  1694  1702  1718  1719  1721  1725  1728  1731  1788
        1789  1790  1795  1798  1801  1819  1838  1847  1848  1849  1851  1872  1876  1879  1906
        1907  1908  1909  191C  1911  1912  1917  1920  1940  1946  1947  1948  1949  1950  1952
        1953  1962  1964  1966  1969  1976  1979  1994  1995  1996  1997  1998  2005  2008  2013
        2020  2021  2028  2029  2030  2035  2036  2042  2043  2044  2050  2060  2087  2088  2091
        2097  2112  2115  2117  2119  2120  2124  2134  2227  2228  2244  2258  2262  2264  2489
        2490  2492  2495  2508  2520  2521  2524  2525  2528  2529  2550  2552  2573  2576  2593
        2598  2607  2612  2617  2619  2623  2657  2665  2666  2667  2673  2680  2698  2699  2700
        2701  2702  2714  2715  2716  2717  2718  2719  2720  2721  2722  2723  2724  2730  2731
        2735  2736  2737  2738  2739  2740  2741  2742  2743  2746  2782  2783  2788  2796  2811
        2850  2851  2852  2853  2854  2856  2871  2872  2873  2874  2875  2901  2925  2930  2959
        2960  2987  2989  3000  3031  3032  3033  3034  3056  3057  3064  3068  3073  3074  3076
        3078  3088  3094  3095  3113  3114  3118  3124  3125  3168  3183  3191  3200  3204  3213
        3253  3288  3321  3322  3323  3327  3334  3345  3346  3369  3370  3373  3374  3377  3386
        3401  3402  3408  3443  3457  3471  3496  3497  3499  3508  3510  3514  3515  3517  3538
        3539  3541  3551  3557  3558  3559  3562  3578  3630  3671  3679  3690  3726  3732
MOVB     500   543   598  1058  1775  1915  1918  1932  1935  1944  2053  2218  2226  2523  2527
        2547  2554  2562  2658  2659  2662  2663  2664  2668  2671  2672  2691  2793  2821  2829
        2859  2892  2909  2963  2969  2993  2998  3004  3009  3024  3051  3052  3054  3116  3252
        3257  3259  3261  3266  3294  3375  3378  3380  3382  3383  3384  3405  3406  3423  3425
        3440  3442  3667  3730  3731  3737
NEG     1914  2669
NOP      480  1484  1485  1486  1641  1642  1643  1883  1884  1885  2095  2096
RESET   1881  3598
ROL      647  1186  2675  2677  2678  2679  2681  2862  2864  2866
ROR     2046  2047  2048  2049
RTI      514   708  1954  2530  2582  2703  2748  2798  2876  2931  2979  3035  3126  3645
RTS      703   740  1999  2058  2106  2114  2125  2272  2621  2838  3096  3119  3289  3309  3347
        3387  3444  3472  3516  3560  3584  3610  3637  3640  3741
SUB      553   665  1843  1921  2245  2247  2249  2251  2253  2255  2553  3071
SWAB    2056
TRAP    3128  3137  3138  3139  3140  3142  3144  3145  3146  3147
TST      551   556   572   574   590   788  1145  1666  1705  1839  1926  1936  2073  2077  2081
        2105  2147  2169  2191  2491  2515  2567  2574  2625  2686  2795  2803  2825  2870  2923
        2938  2996  3007  3030  3066  3084  3086  3115  3350  3501  3509  3543  3552  3601  3628
        3633
TSTB    1756  1825  1928  1942  2018  2240  2502  2778  2827  2890  2906  2961  2967  3058  3069
        3082  3282  3665
.ASCII   275   276  2309
.ASCIZ   277   581   602   609   671   677  1890  2280  2286  2292  2295  2300  2306  2315  2326
        2337  2347  2358  2367  2371  2376  2380  2385  2390  2395  2399  2404  2413  2422  2429
        2436  2443  2447  2629  2752  3039  3040  3041  3043  3188  3242  3246  3250  3478  3664
        3688  3699
.ASECT    14
.BLKB   3038
.BLKW    429  1959
.BYTE    245   246   251   252   260   261   269   270   271   272   293   294   304   305   312
         313   315   316   318   319  1889  2150  2154  2158  2162  2166  2172  2176  2180  2184
        2188  2194  2198  2202  2206  2210  2311  2564  2565  2704  2705  2706  2707  3036  3037
        3097  3098  3099
.DSABL  2948
.ENABL    30  2881
.END    3744
```

```
                1055  1056  1076  1079  1080  1081  1082  1097  1100  1101  1102  1103  1118  1121  1122
                1123  1124  1139  1141  1142  1143  1144  1145  1147  1189  1190  1191  1192  1209  1226
                1234  1235  1236  1237  1259  1272  1273  1274  1275  1301  1313  1314  1315  1316  1489
                1492  1493  1494  1495  1646  1648  1649  1650  1651  1668  1671  1672  1673  1674  1685
                1690  1691  1692  1693  1707  1712  1713  1714  1715  1717  1724  1757  1776  1782  1783
                1784  1785  1787  1794  1826  1835  1836  1837  1838  1839  1857  1861  1865  1870  1873
                1889  1896  2471  2497  2500  2501  2504  2531  2535  2537  2552  2578  2583  2587  2602
                2631  2635  2713  2729  2744  2764  2843  2881  2884  2952  2954  2959  2980  2981  2990
                3022  3038  3051  3108  3114  3217  3223  3271  3277  3298  3304  3412  3418  3429  3435
.IFT             581   602   609   671   677  2512  2558  2861  2877  2878  2954  2959  3188  3242  3246
                3250  3478  3664  3688
.IFTF            581   602   609   671   677  2510  2557  2857  2861  2877  2899  2952  2955  3188  3242
                3246  3250  3478  3664  3688
.IIF              49    54    59   176   177   178   180   183   184   191   278   282   488   491   497
                 498   499   501   502   586  1859  1865  1866  1877  1889  1893  2474  2475  2476  2477
                2478  2479  2483  2511  2512  2528  2531  2532  2538  2539  2540  2541  2546  2570  2578
                2583  2599  2624  2840  2881  2902  3030  3039  3045  3136  3137  3138  3139  3140  3142
                3144  3145  3146  3147  3680
.IRP             380   749   763   776   792   806   820   837   858   881   933   986  1005  1021  1034
                1052  1078  1099  1120  1140  1188  1233  1271  1312  1491  1647  1670  1689  1711  1781
                1834  1906  1946  2547  2716  2722  2735  2736  2852  2873  3056  3057  3078  3094  3095
.LIST              14    30   172   183   191   273   279   282   380   503   551   581   586   589   602
                 609   671   677   749   753   758   760   762   763   767   771   773   776   780   785
                 788   792   796   800   802   806   810   814   816   820   824   830   832   837   841
                 847   850   852   858   862   868   871   873   881   885   888   890   893   895   898
                 900   903   905   908   910   913   915   918   920   923   925   928   930   933   937
                 940   942   945   947   950   952   955   957   960   962   965   967   970   972   975
                 977   980   982   986   990   993   996   999  1001  1005  1009  1012  1015  1017  1021
                1025  1028  1030  1034  1038  1041  1044  1046  1052  1056  1064  1067  1071  1073  1078
                1082  1085  1088  1092  1094  1099  1103  1106  1109  1113  1115  1120  1124  1127  1130
                1134  1136  1140  1144  1154  1157  1159  1162  1164  1174  1176  1179  1181  1188  1192
                1205  1207  1218  1221  1223  1233  1237  1242  1245  1253  1256  1258  1265  1268  1271
                1275  1280  1283  1294  1297  1300  1307  1310  1312  1316  1324  1327  1330  1333  1336
                1338  1341  1344  1347  1350  1352  1355  1358  1361  1364  1366  1369  1372  1375  1378
                1380  1383  1386  1389  1392  1394  1397  1400  1403  1406  1408  1411  1414  1417  1420
                1422  1425  1428  1431  1434  1436  1439  1442  1445  1448  1450  1453  1456  1459  1462
                1464  1467  1470  1473  1476  1478  1481  1491  1495  1503  1506  1510  1513  1515  1518
                1522  1525  1527  1530  1534  1537  1539  1542  1546  1549  1551  1554  1558  1561  1563
                1566  1570  1573  1575  1578  1582  1585  1587  1590  1594  1597  1599  1602  1606  1609
                1611  1614  1618  1621  1623  1626  1630  1633  1635  1638  1647  1651  1655  1658  1661
                1665  1670  1674  1678  1680  1683  1689  1693  1697  1699  1702  1705  1711  1715  1728
                1731  1734  1736  1739  1741  1744  1746  1749  1753  1756  1765  1768  1770  1773  1781
                1785  1798  1801  1804  1806  1809  1812  1814  1817  1819  1822  1825  1834  1838  1865
                1881  1969  1972  1974  1976  1979  1982  1984  1986  2011  2018  2028  2033  2218  2221
                2223  2226  2232  2235  2240  2244  2478  2578  2980  3128  3136  3137  3138  3139  3140
                3141  3142  3143  3144  3145  3146  3147  3148  3188  3242  3246  3250  3478  3601  3664
                3688
.MACRO            17    18    19    20    22    24    25    26    27    28    29    30   184   236   431
                 467   519  1050  1051  2147  3128
.MCALL            30   172   279   503   589
.MEXIT           327
.NLIST            14    30   172   183   191   273   279   282   380   503   551   581   586   589   602
                 609   671   677   749   753   758   760   762   763   767   771   773   776   780   785
                 788   792   796   800   802   806   810   814   816   820   824   830   832   837   841
```

```
                847   850   852   858   862   868   871   873   881   885   888   890   893   895   898
                900   903   905   908   910   913   915   918   920   923   925   928   930   933   937
                940   942   945   947   950   952   955   957   960   962   965   967   970   972   975
                977   980   982   986   990   993   996   999  1001  1005  1009  1012  1015  1017  1021
               1025  1028  1030  1034  1038  1041  1044  1046  1052  1056  1064  1067  1071  1073  1078
               1082  1085  1088  1092  1094  1099  1103  1106  1109  1113  1115  1120  1124  1127  1130
               1134  1136  1140  1144  1154  1157  1159  1162  1164  1174  1176  1179  1181  1188  1192
               1205  1207  1218  1221  1223  1233  1237  1242  1245  1253  1256  1258  1265  1268  1271
               1275  1280  1283  1294  1297  1300  1307  1310  1312  1316  1324  1327  1330  1333  1336
               1338  1341  1344  1347  1350  1352  1355  1358  1361  1364  1366  1369  1372  1375  1378
               1380  1383  1386  1389  1392  1394  1397  1400  1403  1406  1408  1411  1414  1417  1420
               1422  1425  1428  1431  1434  1436  1439  1442  1445  1448  1450  1453  1456  1459  1462
               1464  1467  1470  1473  1476  1478  1481  1491  1495  1503  1506  1510  1513  1515  1518
               1522  1525  1527  1530  1534  1537  1539  1542  1546  1549  1551  1554  1558  1561  1563
               1566  1570  1573  1575  1578  1582  1585  1587  1590  1594  1597  1599  1602  1606  1609
               1611  1614  1618  1621  1623  1626  1630  1633  1635  1638  1647  1651  1655  1658  1661
               1665  1670  1674  1678  1680  1683  1689  1693  1697  1699  1702  1705  1711  1715  1728
               1731  1734  1736  1739  1741  1744  1746  1749  1753  1756  1765  1768  1770  1773  1781
               1785  1798  1801  1804  1806  1809  1812  1814  1817  1819  1822  1825  1834  1838  1865
               1881  1969  1972  1974  1976  1979  1982  1984  1986  2011  2018  2028  2033  2218  2221
               2223  2226  2232  2235  2240  2244  2478  2578  2980  3128  3136  3137  3138  3139  3140
               3141  3142  3143  3144  3145  3146  3147  3148  3188  3242  3246  3250  3478  3601  3664
               3688
.PAGE            30   172   236   328
.PSECT           14
.REM              1    14    30
.REPT           191   888   940  1328  1507
.SBTTL           62   172   185   194   203   214   236   279   328   481   582   589   637   749   763
                776   792   806   820   837   858   881   933   986  1005  1021  1034  1052  1078  1099
               1120  1140  1188  1233  1271  1312  1491  1647  1670  1689  1711  1781  1834  1854  1893
               1961  2004  2059  2133  2468  2532  2584  2632  2710  2761  2840  2878  3048  3105  3128
.TITLE           49
.WORD           191   192   193   211   230   231   232   233   234   235   244   247   248   249   250
                253   254   255   256   257   258   259   262   263   264   284   285   286   287   288
                289   290   291   295   296   29.   310   314   317   320   321   322   323   324   325
                406   409   411   413   415   417   419   421   423   424   426   428   551   605   758
                760   762   771   773   785   788   800   802   814   816   830   832   847   850   852
                868   871   873   888   890   893   895   898   900   903   905   908   910   913   915
                918   920   923   925   928   930   940   942   945   947   950   952   955   957   960
                962   965   967   970   972   975   977   980   982   993   996   999  1001  1012  1015
               1017  1028  1030  1041  1044  1046  1064  1067  1071  1073  1085  1088  1092  1094  1106
               1109  1113  1115  1127  1130  1134  1136  1154  1157  1159  1162  1164  1174  1176  1179
               1181  1205  1207  1218  1221  1223  1242  1245  1253  1256  1258  1265  1268  1280  1283
               1294  1297  1300  1307  1310  1324  1327  1330  1333  1336  1338  1341  1344  1347  1350
               1352  1355  1358  1361  1364  1366  1369  1372  1375  1378  1380  1383  1386  1389  1392
               1394  1397  1400  1403  1406  1408  1411  1414  1417  1420  1422  1425  1428  1431  1434
               1436  1439  1442  1445  1448  1450  1453  1456  1459  1462  1464  1467  1470  1473  1476
               1478  1481  1503  1506  1510  1513  1515  1518  1522  1525  1527  1530  1534  1537  1539
               1542  1546  1549  1551  1554  1558  1561  1563  1566  1570  1573  1575  1578  1582  1585
               1587  1590  1594  1597  1599  1602  1606  1609  1611  1614  1618  1621  1623  1626  1630
               1633  1635  1638  1655  1658  1661  1665  1678  1680  1683  1697  1699  1702  1705  1728
               1731  1734  1736  1739  1741  1744  1746  1749  1753  1756  1765  1768  1770  1773  1798
               1801  1804  1806  1809  1812  1814  1817  1819  1822  1825  1870  1873  1888  1969  1972
               1974  1976  1979  1982  1984  1986  2011  2018  2028  2033  2218  2221  2223  2226  2232
```

2235	2240	2244	2610	2615	2708	2745	2747	2790	2837	2877	3080	3135	3194	3291
3311	3446	3512	3513	3520	3555	3565	3586	3587	3601	3611	3612	3642	3646	3676
3677	3682	3693	3694	3697	3698	3728	3729	3735	3736	3742				

000000

ERRORS DETECTED: 0

```
*DRLPF,DRLPF/SOL/CRF=DRLPA.MAC,DRLPF
RUN-TIME:  26 17 2 SECONDS
CORE USED:  40K
```