DR11-K

DIGITAL I/O TEST

MD-11-DZDRG-D

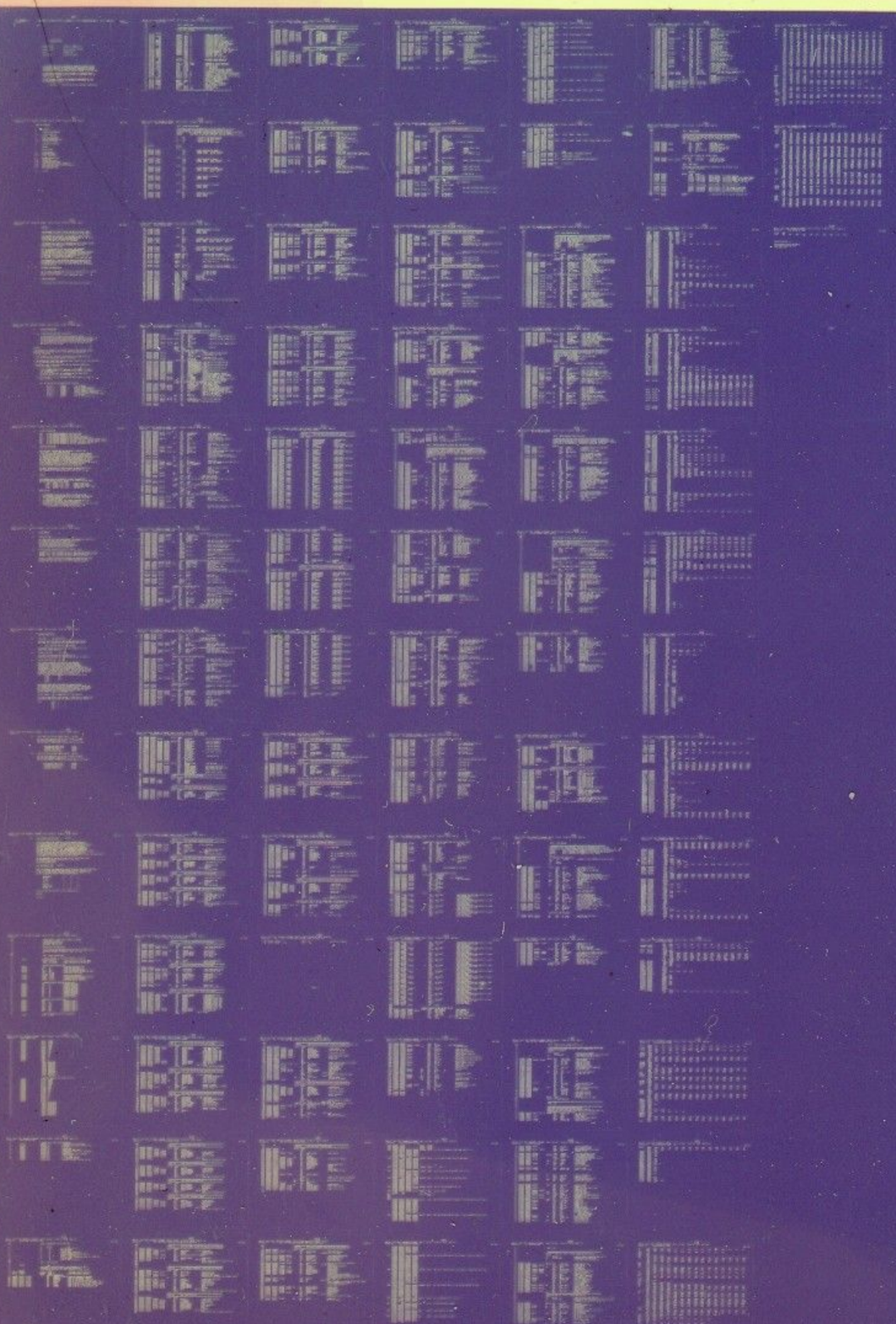
EP DZDRG D-DL B

FICHE 1 OF 1

MAR 1977

digital

MADE IN USA



.REM%

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZDRG-D-D

PRODUCT NAME: DR11-K DIGITAL I/O TEST

DATE: JANUARY 1977

MAINTAINER: DIAGNOSTIC GROUP

TABLE OF CONTENTS

1. ABSTRACT

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

THIS PROGRAM ALSO INCLUDES TWO ROUTINES TO VERIFY PROPER
CONNECTION'S TO THE H321 LOC-BOX AND ADJUSTMENT OF THE XLO1'S.
THE ROUTINES SERVE THE SAME FUNCTION AS THE
"PDLD #3 AND PDLD #4" ONLINE TEST PROGRAMS.

THE PROGRAM ALSO SUPPLY'S A SOFTWARE ROUTINE TO VERIFY PROPER
CONNECTION'S TO DIGITAL DATA INSTRUMENT'S LIKE THE
COULTER MODEL "S" COUNTER.

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 CONFIGERATIONS 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 "BASEBA" AND "BASEIV".
FOR NORMAL FACTORY CONFIG., ALL QUESTIONS SHOULD BE ANSWERED
WITH A VALUE OF 0.

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERMINAL AND 4K OF MEMORY
DR11-K OPTION INSTALLED
BC08R-1 ONE FOOT OUTPUT TO INPUT WRAPAROUND CABLE

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 BACK 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.
214 IS THE STARTING ADDRESS OF THE COULTER INTERFACE LOOP
220 IS THE STARTING ADDRESS OF THE LOGIC TEST WITH "COULTER JUMPER" CONFIG.
224 IS THE STARTING ADDRESS OF THE LOGIC TEST WITH "LOC BOX JUMPER" CONFIG.
230 IS THS STARTING ADDRESS OF THE H321 "LOC-BOX" FUNCTION LOOP
234 IS THE STARTING ADDRESS OF THE XL01 ADJUSTMENT 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.

WORD	$PASS: (LOC. 1100)	CURRENT PASS COUNT
BYTE	$TSTNM: (LOC. 1102)	CURRENT TEST NUMBER
BYTE	$ITEMB: (LOC. 1114)	ITEM #N ERROR TABLE INDEX
WORD	$ERRPC: (LOC. 1116)	ERRORING P.C.

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.

8. MISCELANEOUS

8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 60 SECONDS FOR COMPLETION
ON A PDP11/05 TYPE AND WILL TYPE 'END PASS NNNN.'.
THE CONTROL LINE LOOP WILL NEVER EXIT.
THE COULTER INTERFACE LOOP WILL NEVER EXIT.
THE H321 LOC-BOX FUNCTION LOOP WILL NEVER EXIT.
THE XL01 ADJUSTMENT LOOP WILL NEVER EXIT.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS (AT APPROX. 1300)

"BASEBA" (LOC. 1306) CONTAINS THE DR11-K BASE DEVICE ADDRESS <767770>
"BASEIV" (LOC. 1310) CONTAINS THE DR11-K BASE INTERRUPT VECTOR <300>
"BASEBR" (LOC. 1312) CONTAINS 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.

9. PROGRAM DESCRIPTION

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 CAN INTERRUPT AND THAT "RESET" WILL WORK CORRECTLY.

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.

9.3 COULTER FUNCTION LOOP <SA 214>

THE COULTER INTERFACE LOOP PROVIDES THE OPERATOR WITH SOFTWARE INTERFACE WITH THE COULTER HARDWARE. "BASEBA" (LOC. 1306) CONTAINS THE DR11K BUS ADDRESS. THE OPERATOR IS INSTRUCTED TO RUN A SAMPLE ON THE COULTER. THE PROGRAM WILL WAIT FOR AN INTERRUPT FROM THE DR11K. UPON AN INTERRUPT, THE PROGRAM DELAYS 70 MSEC. TO ALLOW THE DATA LINES TO SETTLE. THE 70 MSEC. DELAY IS A FACTOR OF CPU TYPE AND MAY REQUIRE THE OPERATOR TO MODIFY THE VALUE OF "CPU" (LOC. 1314) FOR THE PROPER DELAY ON A DIFFERENT CPU TYPE:

11/05	240
11/34	?
11/40	620
11/45	?
11/70	?

THE COULTER INPUT DATA IS STORED IN A MEMORY BUFFER. THE HIGHER FOUR BITS ARE THE TEST NUMBERS WITH ZERO BEING THE RUN TERMINATOR. UPON COMPLETION OF A RUN (SAMPLE), THE PROGRAM WILL TYPE THE RESULTS IN THE FOLLOWING FORMAT AND THEN RESTART THE RUN SEQUENCE. THE TYPED RESULTS CAN BE COMPARED TO THE COULTER TYPOUT TO VERIFY A SUCCESSFUL RUN.

"N-XXX" WHERE N = TEST NUMBER AND XXX = TEST RESULTS

THE RESULTS ARE BINARY CODED DECIMAL. THERE ARE BIT COMBINATIONS WHICH ARE ERRORS AND SHOULD NOT APPEAR ON THE COULTER OUTPUT.

CODE:			
	1010	=	:
	1011	=	;
	1100	=	<
	1101	=	=
	1110	=	>
	1111	=	?

9.4 LOGIC TEST WITH COULTER JUMPER CONFIG. <SA 220>

RUN THE LOGIC TEST WITH THE COULTER INTERFACE JUMPERS
AND WITH AN INTERRUPT LEVEL OF 5.
COULTER DEFAULT JUMPER/SWITCH VALUES USED IN THIS MODE:

NON-LATCHING INPUT BITS	177777
INTERRUPTING INPUT BITS	170000
POSITIVE INPUT BITS	000000
TRANSITION INPUT BITS	170000

9.5 LOGIC TEST WITH LOC-BOX JUMPER CONFIG. <SA 224>

RUN THE LOGIC TEST WITH THE LOC-BOX INTERFACE JUMPERS
AND WITH AN INTERRUPT LEVEL OF 5.
H321 LOC-BOX DEFAULT JUMPER/SWITCH VALUES USED IN THIS MODE:

NON-LATHING INPUT BITS	000000
INTERRUPTING INPUT BITS	177777
POSITIVE INPUT BITS	000000
TRANSITION INPUT BITS	000000

9.6 H321 LOC-BOX FUNCTION LOOP (SA 230)

THIS LOOP ENABLES THE OPERATOR TO VERIFY PROPER OPERATION
OF THE H321 LOC-BOX LAMP'S, SWITCHES AND THE WIRE CONNECTION.
THIS LOOP SERVES THE SAME FUNCTION AS THE "PDL DIAG #3 -- PDLD3"
THE OPERATOR SELECTS THE LOC-BOX'S, VIA DR11K BUS ADDRESS,
AND THEN DEPRESSES EACH SWITCH. THE PROGRAM WILL COMPLEMENT
THE LAMP ABOVE THE SWITCH. THE OPERATOR SHOULD DEPRESS THE
SWITCHES SLOWLY TO VERIFY THAT ONLY DEPRESSING A SWITCH WILL CAUSE
THE LAMP TO CHANGE AND THAT RELEASING THE SWITCH DOES NOT.
IF THE LATER IS DETECTED THE DR11-K JUMPER CONFIGURATION SHOULD BE QUESTIONED?
THE PROGRAM MAY OCCASIONLY MISS A SWITCH CLOSURE.

THE BUS ADDRESSES OF THE DR11K'S ARE 167770, 167760 AND 167750.
THE OPERATOR MAY CHANGE LOCATIONS LOC1, LOC2, LOC3
(LOC. 1326, 1330, 1332)
TO RUN ON OTHER DR11K'S.

9.7 XLO1 ADJUSTMENT LOOP (SA 234)

THIS LOOP ENABLES THE OPERATOR TO VERIFY OR ADJUST THE XLO1/CC55A/CC55C
POT'S. THIS LOOP SERVES THE SAME FUNCTION AS "PDL DIAGNOSTIC #4
-- PDLD4" THE H321 LAMP'S OPERATE AS SHOWN BELOW. REFER TO THE
PDL-11 INSTALLATION MANUAL FOR ADJUSTMENT DETAILS.

APPROCHING	0	0	1
AN END	0	1	1
---------	1	1	1
CENTER	1	1	1
	1	1	1
REGION	1	1	1
----------	1	1	1
APPROACHING	1	1	0
AN END	1	0	0

THIS LOOP USES THE SAME DR11K'S BUS ADDRESS AS 9.6

10. LISTING

ATTACHED.

%

```
388                             .TITLE  DR11-K  LOGIC TEST  MAINDEC-11-DZDRGD
(1)                             ;*COPYRIGHT (C) 1976
(1)                             ;*DIGITAL EQUIPMENT CORP.
(1)                             ;*MAYNARD, MASS. 01754
(1)                             ;*
(1)                             ;*PROGRAM BY RAYMOND SHOOP
(1)                             ;*
(1)                             ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)                             ;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
(1)                             ;*
389
390                             .SBTTL  BASIC DEFINITIONS
(1)
(1)                             ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)             001100          STACK=  1100
(1)                             .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
(1)                             .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)                             ;*MISCELLANEOUS DEFINITIONS
(1)             000011          HT=     11              ;;CODE FOR HORIZONTAL TAB
(1)             000012          LF=     12              ;;CODE FOR LINE FEED
(1)             000015          CR=     15              ;;CODE FOR CARRIAGE RETURN
(1)             000200          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)             177776          PS=     177776          ;;PROCESSOR STATUS WORD
(1)                             .EQUIV  PS,PSW
(1)             177774          STKLMT= 177774          ;;STACK LIMIT REGISTER
(1)             177772          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)             177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
(1)             177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
(1)
(1)                             ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)             000000          R0=     %0              ;;GENERAL REGISTER
(1)             000001          R1=     %1              ;;GENERAL REGISTER
(1)             000002          R2=     %2              ;;GENERAL REGISTER
(1)             000003          R3=     %3              ;;GENERAL REGISTER
(1)             000004          R4=     %4              ;;GENERAL REGISTER
(1)             000005          R5=     %5              ;;GENERAL REGISTER
(1)             000006          R6=     %6              ;;GENERAL REGISTER
(1)             000007          R7=     %7              ;;GENERAL REGISTER
(1)             000006          SP=     %6              ;;STACK POINTER
(1)             000007          PC=     %7              ;;PROGRAM COUNTER
(1)
(1)                             ;*PRIORITY LEVEL DEFINITIONS
(1)             000000          PR0=    0               ;;PRIORITY LEVEL 0
(1)             000040          PR1=    40              ;;PRIORITY LEVEL 1
(1)             000100          PR2=    100             ;;PRIORITY LEVEL 2
(1)             000140          PR3=    140             ;;PRIORITY LEVEL 3
(1)             000200          PR4=    200             ;;PRIORITY LEVEL 4
(1)             000240          PR5=    240             ;;PRIORITY LEVEL 5
(1)             000300          PR6=    300             ;;PRIORITY LEVEL 6
(1)             000340          PR7=    340             ;;PRIORITY LEVEL 7
(1)
(1)                             ;*"SWITCH REGISTER" SWITCH DEFINITIONS
(1)             100000          SW15=   100000
```

```
(1)             040000                  SW14=   40000
(1)             020000                  SW13=   20000
(1)             010000                  SW12=   10000
(1)             004000                  SW11=   4000
(1)             002000                  SW10=   2000
(1)             001000                  SW09=   1000
(1)             000400                  SW08=   400
(1)             000200                  SW07=   200
(1)             000100                  SW06=   100
(1)             000040                  SW05=   40
(1)             000020                  SW04=   20
(1)             000010                  SW03=   10
(1)             000004                  SW02=   4
(1)             000002                  SW01=   2
(1)             000001                  SW00=   1
(1)                                     .EQUIV  SW09,SW9
(1)                                     .EQUIV  SW08,SW8
(1)                                     .EQUIV  SW07,SW7
(1)                                     .EQUIV  SW06,SW6
(1)                                     .EQUIV  SW05,SW5
(1)                                     .EQUIV  SW04,SW4
(1)                                     .EQUIV  SW03,SW3
(1)                                     .EQUIV  SW02,SW2
(1)                                     .EQUIV  SW01,SW1
(1)                                     .EQUIV  SW00,SW0
(1)
(1)                                     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)             100000                  BIT15=  100000
(1)             040000                  BIT14=  40000
(1)             020000                  BIT13=  20000
(1)             010000                  BIT12=  10000
(1)             004000                  BIT11=  4000
(1)             002000                  BIT10=  2000
(1)             001000                  BIT09=  1000
(1)             000400                  BIT08=  400
(1)             000200                  BIT07=  200
(1)             000100                  BIT06=  100
(1)             000040                  BIT05=  40
(1)             000020                  BIT04=  20
(1)             000010                  BIT03=  10
(1)             000004                  BIT02=  4
(1)             000002                  BIT01=  2
(1)             000001                  BIT00=  1
(1)                                     .EQUIV  BIT09,BIT9
(1)                                     .EQUIV  BIT08,BIT8
(1)                                     .EQUIV  BIT07,BIT7
(1)                                     .EQUIV  BIT06,BIT6
(1)                                     .EQUIV  BIT05,BIT5
(1)                                     .EQUIV  BIT04,BIT4
(1)                                     .EQUIV  BIT03,BIT3
(1)                                     .EQUIV  BIT02,BIT2
(1)                                     .EQUIV  BIT01,BIT1
(1)                                     .EQUIV  BIT00,BIT0
(1)
```

```
(1)                         ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)          000004         ERRVEC= 4                ;;TIME OUT AND OTHER ERRORS
(1)          000010         RESVEC= 10               ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)          000014         TBITVEC=14               ;;"T" BIT
(1)          000014         TRTVEC= 14               ;;TRACE TRAP
(1)          000014         BPTVEC= 14               ;;BREAKPOINT TRAP (BPT)
(1)          000020         IOTVEC= 20               ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)          000024         PWRVEC= 24               ;;POWER FAIL
(1)          000030         EMTVEC= 30               ;;EMULATOR TRAP (EMT) **ERROR**
(1)          000034         TRAPVEC=34               ;;"TRAP" TRAP
(1)          000060         TKVEC=  60               ;;TTY KEYBOARD VECTOR
(1)          000064         TPVEC=  64               ;;TTY PRINTER VECTOR
(1)          000240         PIRQVEC=240              ;;PROGRAM INTERRUPT REQUEST VECTOR
```

```
  392                                   .SBTTL  OPERATIONAL SWITCH SETTINGS
  (1)                                   ;*
  (1)                                   ;*      SWITCH                  USE
  (1)                                   ;*      ------          --------------------
  (1)                                   ;*        15            HALT ON ERROR
  (1)                                   ;*        14            LOOP ON TEST
  (1)                                   ;*        13            INHIBIT ERROR TYPEOUTS
  (1)                                   ;*        12            LOOP ON CURRENTLY SELECTED DR11-K
  (1)                                   ;*        11            INHIBIT ITERATIONS
  (1)                                   ;*        10            OUTPUT TO INPUT WRAPAROUND CABLE NOT CONNECTED
  (1)                                   ;*         9            LOOP ON ERROR
  (1)                                   ;*         8            LOOP ON TEST IN SWR<7:0>
  393
  394                                   .SBTTL  TRAP CATCHER
  (1)
  (1)             000000                        .=0
  (1)                                   ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
  (1)                                   ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
  (1)                                   ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
  (1)             000174                        .=174
  (1)   000174    000000          DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
  (1)   000176    000000          SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
  (1)                                   .SBTTL  STARTING ADDRESS(ES)
  (1)   000200    000137  001400        JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
  395   000204    000137  001434        JMP     @#BEGIN1        ;JUMP TO THE RESTART ADDRESS
  396   000210    000137  011454        JMP     @#EXTTST        ;JUMP TO THE CONTROL LINES LOOP
  397   000214    000137  011614        JMP     @#COULTR        ;JUMP TO THE COULTER FUNCTION LOOP
  398   000220    000137  001410        JMP     COULTJ          ;JUMP TO SA WITH THE COULTER JUMPER CONFIG.
  399   000224    000137  001422        JMP     H322J           ;JUMP TO SA WITH THE LOC-BOX JUMPER CONFIG.
  400   000230    000137  012054        JMP     LOCBOX          ;JUMP TO SA OF LOC-BOX FUNCTION LOOP
  401   000234    000137  012340        JMP     XLO1AD          ;JUMP TO SA OF XLO1 ADJUSTMENT LOOP
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REGO) WAS OBTAINED
001162  000000          $REGO:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001170  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001172  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001173  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001174  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
```

```
(1)                                     .SBTTL  ERROR POINTER TABLE
(1)
(1)                                     ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1)                                     ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1)                                     ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1)                                     ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1)                                     ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1)                                     ;*      EM              ;;POINTS TO THE ERROR MESSAGE
(1)                                     ;*      DH              ;;POINTS TO THE DATA HEADER
(1)                                     ;*      DT              ;;POINTS TO THE DATA
(1)                                     ;*      DF              ;;POINTS TO THE DATA FORMAT
(1)
(1)
(1)  001176                             $ERRTB:
403
404                                     ;ITEM   1
405  001176  014027                             EM1             ;STATUS REGISTER IN ERROR
406  001200  014447                             DH1             ;ERRPC  DRADD   STATUS EXPECTED
407  001202  014740                             DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
408  001204  015002                             DF0
409
410                                     ;ITEM   2
411  001206  014060                             EM2             ;INPUT REGISTER IN ERROR
412  001210  014517                             DH2             ;ERRPC  DRADD   INPUT EXPECTED
413  001212  014740                             DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
414  001214  015002                             DF0
415
416                                     ;ITEM 3
417  001216  014110                             EM3             ;OUTPUT REGISTER IN ERROR
418  001220  014565                             DH3             ;ERRPC  DRADD   OUTPUT EXPECTED
419  001222  014740                             DT1             ;$ERRPC DRADD   $BDDAT $GDDAT
420  001224  015002                             DF0
421
422                                     ;ITEM   4
423  001226  014141                             EM4             ;INPUT FAILED TO INTERRUPT
424  001230  014633                             DH4             ;ERRPC  DRADD
425  001232  014754                             DT4             ;$ERRPC DRADD
426  001234  015002                             DF0
427
428                                     ;ITEM   5
429  001236  014173                             EM5             ;OUTPUT FAILED TO INTERRUPT
430  001240  014633                             DH4             ;ERRPC  DRADD
431  001242  014754                             DT4             ;$ERRPC DRADD
432  001244  015002                             DF0
433
434                                     ;ITEM   6
435  001246  014226                             EM6             ;UNEXPECTED INTERRUPT
436  001250  014633                             DH4             ;ERRPC  DRADD
437  001252  014754                             DT4             ;$ERRPC DRADD
438  001254  015002                             DF0
439
```

```
                                ;ITEM   7
001256  014253                          EM7             ;OPERATOR INTERVENTION ERROR
001260  014633                          DH4             ;ERRPC  DRADD
001262  014754                          DT4             ;$ERRPC DRADD
001264  015002                          DF0

                                ;ITEM   10
001266  014307                          EM10            ;INTERRUPT INPUT BIT FAILED TO SET INPUT READY
001270  014660                          DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
001272  014764                          DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
001274  015002                          DF0

                                ;ITEM   11
001276  014372                          EM11            ;NON-INTERRUPTING INPUT BIT SET INPUT READY
001300  014660                          DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
001302  014764                          DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
001304  015002                          DF0

001306  167770          BASEBA: 167770                  ;STARTING BASE BUS ADDRESS
001310  000300          BASEIV: 300                     ;STARTING BASE INTERRUPT ADDRESS
001312  000200          BASEBR: 200
001314  000240          CPU:    240                     ;1 MS. CPU DELAY FACTOR 240 FOR 11/05
                                                        ;                       620 FOR 11/40
001316  000000          NMBEXT: 0                       ;ADDITIONAL DR-11-K
001320  000000          NBEXT:  0

001322  167770          DRADD:  167770                  ;CURRENT DR11-K STARTING ADDRESS
001324  000300          DRIV:   300                     ;CURRENT DR11-K STARTING INTERRUPT VECTOR


001326  167770          LOC1:   167770                  ;LOC BOX #1 ADDR
001330  167760          LOC2:   167760                  ;LOC BOX #2 ADDR
001332  167750          LOC3:   167750                  ;LOC BOX #3 ADDR
001334  167770          GRSTAT: 167770          ;DR STATUS
001336  167772          GRDAI:  167772          ;INPUT REG.
001340  167774          GRDIO:  167774          ;OUTPUT REG.
001342  167775          GRBHIO: 167775          ;OUTPUT REG HIGH BYTE
001344  000000          NOTLCH: 0
001346  000000          INTBIT: 0
001350  000300          GRIVA:  300
001352  000302          GRIVSA: 302
001354  000304          GRIVB:  304
001356  000306          GRIVSB: 306
001360  000200          DIOBRL: 200
001362  000000          BRLEV1: 0
001364  000000          BRLEV2: 0
001366  000000          BRLEV3: 0
001370  000000          ODDJMP: 0
001372  000000          MINSIN: 0
001374  000000          TRANST: 0
001376  000000          JUMPER: 0       ;1 IF RUNNING H322 JUMPER CONFIG ;2 IF COULTER JUMPER CONFIG
```

```
                                        ;;*******************************************************
                                        ;       DIGITAL I-O LOGIC TEST
                                        ;*******************************************************
001400  005000                  BEGIN:  CLR     R0                      ;CLEAR R0
001402  005037  001376                  CLR     JUMPER                  ;INDICATE NORMAL FACTORY BUILD
001406  000414                          BR      RBEG
001410  012737  000002  001376  COULTJ: MOV     #2,JUMPER               ;INDICATE COULTER JUMPER CONFIG
001416  005000                          CLR     R0
001420  000407                          BR      RBEG
001422  012737  000001  001376  H322J:  MOV     #1,JUMPER               ;INDICATE LOC BOX JUMPER CONFIG
001430  005000                          CLR     R0
001432  000402                          BR      RBEG
001434  012700  177777          BEGIN1: MOV     #-1,R0                  ;LOAD R0
001440  000005                  RBEG:   RESET
                                .SBTTL  INITIALIZE THE COMMON TAGS
(1)                             ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1)     001442  012706  001100          MOV     #$CMTAG,R5      ;;FIRST LOCATION TO BE CLEARED
(1)     001446  005026                  CLR     (R5)+           ;;CLEAR MEMORY LOCATION
(1)     001450  022706  001140          CMP     #SWR,R5 ;;DONE?
(1)     001454  001374                  BNE     .-6             ;;LOOP BACK IF NO
(1)     001456  012706  001100          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
(1)                             ;;INITIALIZE A FEW VECTORS
(1)     001462  012737  015022  000020  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1)     001470  012737  000340  000022  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
(1)     001476  012737  015274  000030  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1)     001504  012737  000340  000032  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
(1)     001512  012737  017466  000034  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1)     001520  012737  000340  000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
(1)     001526  012737  016022  000024  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1)     001534  012737  000340  000026  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
(1)     001542  013737  011142  011134  MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
(1)     001550  005037  001166          CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
(1)     001554  005037  001170          CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1)     001560  112737  000001  001115  MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
(1)     001566  012737  001566  001106  MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1)     001574  012737  001574  001110  MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
(2)                             ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)                             ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
(2)     001602  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
(2)     001606  012737  001642  000004  MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
(2)     001614  012737  177570  001140  MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
(2)     001622  012737  177570  001142  MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
(2)     001630  022777  177777  177302  CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
(2)     001636  001012                  BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                                                             ;;AND  THE HARDWARE SWR IS NOT = -1
(2)     001640  000403                  BR      65$             ;;BRANCH IF NO TIMEOUT
(2)     001642  012716  001650  64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
(2)     001646  000002                  RTI
(2)     001650  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
(2)     001656  012737  000174  001142          MOV     #DISPREG,DISPLAY
(2)     001664  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
(1)
516     001670  000240                  NOP
517     001672  000240                  NOP
```

```
                                                    NOP
                                                    MOV     #1$,@#ERRVEC            ;LOAD BUS ERROR
                                                    MOV     BASEBA,R2               ;LOAD STARTING ADDRESS
                                                    CLR     R3                      ;CLEAR COUNT
                                            2$:     TST     (R2)                    ;TEST IF EXISTENT
                                                    SUB     #10,R2                  ;EXIST, UPDATE TEST ADDRESS
                                                    INC     R3                      ;UPDATE # OF DR11'S
                                                    BR      2$
                                            1$:     CMP     (SP)+,(SP)+             ;POP STACK
                                                    TST     R3                      ;TEST IF FIRST DOES EXIST
                                                    BNE     3$                      ;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$:     DEC     R3                      ;ADJUST R3
                                                    MOV     R3,NMBEXT               ;SAVE THE NUMBER OF ADDITIONAL DR11-K
                                                    MOV     #VECTRP,@#ERRVEC        ;RESET BUS ERROR
                                                    MOV     #340,@#ERRVEC+2
                                            RBEG1:  RESET
                                                    TST     @#42                    ;TEST IF UNDER A MONITOR
                                                    BNE     4$                      ;BR IF
                                                    TST     R0                      ;TEST R0
                                                    BEQ     2$                      ;BR IF CLEARED
                                            4$:     JMP     IOTEST                  ;JUMP IF SET
                                            2$:
(1)                                                 TYPE    ,65$                ;;TYPE ASCIZ STRING
(1)                                                 BR      64$                 ;;GET OVER THE ASCIZ
(1)                                         ;;65$:  .ASCIZ  <15><12><15><12>/DR11-K DIGITAL INPUT OUTPUT LOGIC TEST/
(1)                                         64$:
                                                    TYPE    ,67$                ;;TYPE ASCIZ STRING
(1)                                                 BR      66$                 ;;GET OVER THE ASCIZ
(1)                                         ;;67$:  .ASCIZ  <15><12>/MAINDEC-11-DZDRGD/<15><12>
(1)                                         66$:
                                                    MOV     NMBEXT,-(SP)
                                                    TYPOS
                                                    .WORD   2
                                                    TYPE    ,69$                ;;TYPE ASCIZ STRING
(1)                                                 BR      68$                 ;;GET OVER THE ASCIZ
(1)                                         ;;69$:  .ASCIZ  /(8) ADDITIONAL DR11-K'S CONNECTED/<15><12>
(1)                                         68$:
                                                    CMP     #1,JUMPER               ;TEST IF LOC-BOX CONFIG.
                                                    BNE     1$                      ;BR IF NOT
                                                    CLR     NOTLCH                  ;LOAD LOC-BOX JUMPER CONFIG
                                                    MOV     #-1,INTBIT              ;
                                                    CLR     MINSIN
                                                    CLR     TRANST
                                                    JMP     IOTEST                  ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
                                            1$:     CMP     #2,JUMPER               ;TEST IF COULTER CONFIG.
```

```
                                        BNE     3$              ;BR IF NOT
                                        MOV     #-1,NOTLCH              ;LOAD COULTER JUMPER CONFIG.
                                        MOV     #170000,INTBIT
                                        CLR     MINSIN
                                        MOV     #170000,TRANST
                                        JMP     IOTEST                  ;RUN THE LOGIC TEST WITH COULTER JUMPERS
                                3$:     JSR     R5,TALK                 ;TALK TO THE ANIMALS
                                        SWNLB
                                        NOTLCH                          ;ABOUT NON LATCH INPUT BITS
                                        JSR     R5,TALK                 ;TALK TO THE ANIMAL AGAIN
                                        SWINTB
                                        INTBIT                          ;ABOUT THE INTERRUPTING BITS
                                        JSR     R5,TALK                 ;ISN'T THIS BOARING
                                        SWPOSB
                                        MINSIN                          ;AGAIN-ABOUT THE MINUS INPUT BITS
                                        JSR     R5,TALK                 ;HO-HUM
                                        SWTRAB
                                        TRANST                          ;AGAIN-ABOUT THE TRANSITION BITS

                                        .SBTTL  DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
                                        MOV     #BIT12,$GDDAT           ;LOAD TEST BIT
                                10$:    BIT     $GDDAT,MINSIN           ;TEST IF NEG INPUT BIT
                                        BEQ     11$                     ;BR IF NOT
                                        BIT     $GDDAT,TRANST           ;TEST IF TRANS. INPUT
                                        BEQ     11$                     ;BR IF NOT
                                        ERROR   7                       ;LOGICAL OPERATOR ERROR
                                                                        ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
                                        JMP     BEGIN                   ;

                                11$:    ROL     $GDDAT                  ;MOVE LEFT
                                        BCC     10$                     ;BR UNTIL DONE

                                        JSR     R5,TALK                 ;ASK THE OPERATOR
                                        SWDPOB
                                        BRLEV3                          ;ABOUT PROGRAM OPTIONS
                                        JMP     IOTEST

                                ;INTERRUPT TO UNEXPECTED VECTOR
                                VECTRP: CMP     (SP),#1000              ;TEST IF IN PROGRAM CODE
                                        BLO     1$                      ;BR IF NOT
                                70$:    HALT                            ;FATAL BUS TRAP IN PROGRAM AREA
                                        BR      70$
                                1$:     CMP     (SP),#200               ;TEST IF IN SACRED VECTOR AREA
                                        BHI     2$                      ;BR IF NOT
                                71$:    HALT                            ;FATAL VECTOR TRAP
                                        BR      71$
                                2$:     MOV     (SP)+,72$               ;GET ADDR THE TRAP OCCURRED TO
                                        SUB     #4,72$                  ;MAKE REAL ADDRESS
                                        BIT     #SW13,@SWR              ;TEST IF TYPE ERROR INHIBIT
                                        BNE     3$                      ;BR IF INHIBIT
                                        TYPE    ,65$                ;;TYPE ASCIZ STRING
                                        BR      64$                 ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/DR11K INTERRUPTED TO LOC. /
                                64$:
```

```
                                        MOV     72$,-(SP)               ;LOAD VALUE TO BE TYPED
                                        TYPOC
                                        TYPE    ,67$                ;;TYPE ASCIZ STRING
                                        BR      66$                 ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  / AND WILL NOW USE THAT VECTOR/<15><12>
                                66$:
                                3$:     BIC     #7,72$                  ;MASK OFF LOWER 3 BITS
                                        BIT     #SW15,@SWR              ;TEST IF HALT ON ERROR
                                        BEQ     4$                      ;BR IF NOT
                                        HALT                ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
                                4$:     CMP     (SP)+,(SP)+             ;CLEAN THE STACK
                                        MOV     72$,DRIV                ;LOAD NEW VECTOR ADDRESS
                                        JMP     RBEG2                   ;TEST THE DR11K AT THE NEW VECTOR
                                72$:    0

                                ;OPERATOR QUESTIONS AND ANSWER ROUTINE

                                TALK:   MOV     (R5)+,10$               ;GET ASCII POINTER
                                        MOV     (R5)+,11$               ;GET POINTER TO DATA FROM OPERATOR
                                        TYPE
                                10$:    SWNLB                           ;TELL OPERATOR TO LOAD SWITCHES
                                        CMP     SWR,#SWREG              ;TEST IF SWITCH REG. EXISTS
                                        BNE     1$                      ;BR IF NO
                                        TYPE    ,$MSWR                  ;TYPE "SWR ="
                                        RDOCT                           ;READ OCTAL
                                        MOV     (SP)+,@SWR              ;SAVE THE SWITCHES
                                        BR      2$
                                1$:     TYPE
                                        DEPCNT                          ;TELL OPERATOR TO DEPRESS CONT
                                        HALT                            ;WAIT FOR OPERATOR
                                2$:     MOV     @SWR,@11$               ;LOAD SWITCH VALUE
                                        RTS     R5                      ;EXIT
                                11$:    NOTLCH                          ;POINTER TO DATA TO BE LOADED

                                ;UNEXPECTED INTERRUPT
                                UNEXPT: ERROR   6                       ;UNEXPECTED INTERRUPT DURING A SUB-TEST
                                        RTI                             ;EXIT

                                IOTEST: RESET
                                        MOV     BASEBA,DRADD            ;LOAD INITIAL STARTING ADDRESS
                                        MOV     BASEIV,DRIV             ;LOAD INITIAL STARTING VECTOR
                                        MOV     NMBEXT,NBEXT            ;RELOAD # AVAILABLE
                                RBEG2:  RESET
                                        MOV     #240,R1                 ;LOAD R1
                                        MOV     #242,R2
                                5$:     MOV     R2,(R1)+                ;SAVE THE ADDRESS
                                        MOV     #4700,(R1)+             ;LOAD "JSR       PC,RC"
                                        CMP     (R2)+,(R2)+             ;BUMP R2
                                        CMP     R2,#$CMTAG-2            ;TEST FOR LAST
                                        BNE     5$                      ;BR UNTIL DONE
```

```
                                IOTST1: RESET
003026  000005
003030  004737  003036                  JSR     PC,SETADD               ;SET UP BUS ADDRESS AND VECTOR
003034  000453                          BR      IOTTS1
003036  013737  001322  001334  SETADD: MOV     DRADD,GRSTAT            ;LOAD 1ST ADDRESS
003044  013737  001322  001336          MOV     DRADD,GRDAI             ;LOAD 2ND ADDRESS
003052  062737  000002  001336          ADD     #2,GRDAI
003060  013737  001322  001340          MOV     DRADD,GRDIO             ;LOAD 3RD ADDRESS
003066  062737  000004  001340          ADD     #4,GRDIO
003074  013737  001322  001342          MOV     DRADD,GRBHIO            ;LOAD 4TH ADDRESS
003102  062737  000005  001342          ADD     #5,GRBHIO
003110  013737  001324  001350          MOV     DRIV,GRIVA              ;LOAD FIRST VECTOR
003116  013737  001324  001352          MOV     DRIV,GRIVSA
003124  062737  000002  001352          ADD     #2,GRIVSA
003132  013737  001324  001354          MOV     DRIV,GRIVB              ;LOAD 2ND VECTOR
003140  062737  000004  001354          ADD     #4,GRIVB
003146  013737  001324  001356          MOV     DRIV,GRIVSB
003154  062737  000006  001356          ADD     #6,GRIVSB
003162  000207                          RTS     PC
003164  013737  001312  001360  IOTTS1: MOV     BASEBR,DIOBRL           ;LOAD BR LEVEL
003172  005037  001370                  CLR     ODDJMP
003176  053737  001372  001370          BIS     MINSIN,ODDJMP           ;SET MINUS INPUT BITS
003204  053737  001374  001370          BIS     TRANST,ODDJMP           ;SET TRANSITION INPUT BITS
003212  012777  002746  176130          MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
(1) 003220  012777  000340  176124      MOV     #340,@GRIVSA
(1) 003226  012777  002746  176120      MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
(1) 003234  012777  000340  176114      MOV     #340,@GRIVSB
                        ;;**************************************************************
                        ;*TEST 1         TEST FOR NO BUS ERRORS
                        ;;**************************************************************
003242  000004          TST1:   SCOPE
003244  005077  176064          CLR     @GRSTAT
003250  005077  176062          CLR     @GRDAI
003254  005077  176060          CLR     @GRDIO

                        ;;**************************************************************
                        ;*TEST 2         TEST THAT OUTPUT REG. CAN HOLD #-1
                        ;;**************************************************************
003260  000004          TST2:   SCOPE
003262  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD EXPECTED
003270  012777  177777  176042          MOV     #-1,@GRDIO              ;ALL ONES TO REGISTER
003276  017737  176036  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003304  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003312  001401                          BEQ     TST3                ;;BR IF EQUAL
003314  104003                          ERROR   3                       ;REG WILL NOT HOLD ONES
```

```
                                ;;*****************************************************************
                                ;*TEST 3         TEST THAT RESET CLEARS OUTPUT REG.
                                ;;*****************************************************************
003316  000004                  TST3:   SCOPE
003320  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
003326  005037  001124          CLR     $GDDAT
003332  012777  177777  176000          MOV     #-1,@GRDIO
003340  000005                          RESET                   ;SET DATA TO ALL ONES
003342  017737  175772  001126          MOV     @GRDIO,$BDDAT   ;READ OUTPUT REG.
003350  001401                          BEQ     TST4            ;;BR IF EQUAL
003352  104003                          ERROR   3               ;REG FAILED TO CLEAR

                                ;;*****************************************************************
                                ;*TEST 4         TEST THAT OUTPUT REG. CAN HOLD #52525
                                ;;*****************************************************************
003354  000004                  TST4:   SCOPE
003356  012737  052525  001124          MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
003364  012777  052525  175746          MOV     #52525,@GRDIO
003372  017737  175742  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003400  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003406  001401                          BEQ     TST5            ;;BR IF EQUAL
003410  104003                          ERROR   3               ;DATA NOT=52525

                                ;;*****************************************************************
                                ;*TEST 5         TEST THAT OUTPUT REG. CAN HOLD #125252
                                ;;*****************************************************************
003412  000004                  TST5:   SCOPE
003414  012737  125252  001124          MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
003422  012777  125252  175710          MOV     #125252,@GRDIO
003430  017737  175704  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003436  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003444  001401                          BEQ     TST6            ;;BR IF EQUAL
003446  104003                          ERROR   3               ;DATA NOT=125252

                                ;;*****************************************************************
                                ;*TEST 6         TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
                                ;;*****************************************************************
003450  000004                  TST6:   SCOPE
003452  012737  000004  001166          MOV     #4,$TIMES       ;;DO 4 ITERATIONS
003460  012737  003472  001110          MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
003466  005037  001124          CLR     $GDDAT                  ;CLEAR PATTERN
003472  013777  001124  175640  2$:     MOV     $GDDAT,@GRDIO           ;LOAD THE OUTPUT REG.
003500  017737  175634  001126          MOV     @GRDIO,$BDDAT           ;READ THE REG.
003506  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
003514  001401                          BEQ     1$              ;;BR IF EQUAL
003516  104003                          ERROR   3               ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
003520  005237  001124          1$:     INC     $GDDAT                  ;UPDATE THE PATTERN
003524  001362                          BNE     2$                      ;TRY AGAIN
```

```
                                        ;;*****************************************************************
                                        ;*TEST 7         FLOAT A 1 ACROSS THE OUTPUT REGISTER
                                        ;;*****************************************************************
003526  000004                          TST7:   SCOPE
003530  012737  003544  001110                  MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
003536  012737  000001  001124                  MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE

003544  005077  175570                  1$:     CLR     @GRDIO                  ;CLEAR OUTPUT
003550  053777  001124  175562                  BIS     $GDDAT,@GRDIO           ;SET THAT BIT
003556  017737  175556  001126                  MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003564  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;TEST RESULTS
003572  001401                                  BEQ     2$                      ;BR IF EQUAL ?
003574  104003                                  ERROR   3

003576  006337  001124                  2$:     ASL     $GDDAT                  ;SHIFT EXPECTED DATA
003602  001360                                  BNE     1$                      ;BR UNTIL DONE
                                        ;;*****************************************************************
                                        ;*TEST 10        FLOAT A 0 ACROSS THE OUTPUT REGISTER
                                        ;;*****************************************************************
003604  000004                          TST10:  SCOPE
003606  012737  003622  001110                  MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
003614  012737  000001  001124                  MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE

003622  012777  177777  175510  1$:     MOV     #-1,@GRDIO              ;LOAD OUTPUT TO A ONE
003630  043777  001124  175502          BIC     $GDDAT,@GRDIO           ;CLEAR A BIT
003636  017737  175476  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003644  005137  001126                  COM     $BDDAT                  ;COMPLEMENT IT
003650  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;EQUAL ?
003656  001401                          BEQ     2$                      ;BR IF EQUAL
003660  104003                          ERROR   3

003662  006337  001124          2$:     ASL     $GDDAT                  ;SHIFT LEFT
003666  001355                          BNE     1$                      ;BRANCH UNTIL DONE

                                        ;;*****************************************************************
                                        ;*TEST 11        TEST FOR SLOW OUTPUT GATES WITH #125252
                                        ;;*****************************************************************
003670  000004                          TST11:  SCOPE
003672  012737  125252  001124                  MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
003700  013777  001124  175432                  MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
003706  005177  175426                          COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1)     003712  005177  175422                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1)     003716  005177  175416                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1)     003722  005177  175412                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1)     003726  005177  175406                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1)     003732  005177  175402                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1)     003736  005177  175376                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1)     003742  005177  175372                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
003746  017737  175366  001126                  MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003754  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;TEST REGISTER
003762  001401                                  BEQ     TST12           ;;BR IF CORRECT
003764  104003                                  ERROR   3
```

```
                                ;;*****************************************************************
                                ;*TEST 12        TEST FOR SLOW OUTPUT GATES WITH #52525
                                ;;*****************************************************************
004004  000004                  TST12:  SCOPE
003770  012737  052525  001124          MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
003776  013777  001124  175334          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT REG.
004004  005177  175330                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004010  005177  175324                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004014  005177  175320                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004020  005177  175314                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004024  005177  175310                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004030  005177  175304                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004034  005177  175300                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004040  005177  175274                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004044  017737  175270  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
004052  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST PATTERN
004060  001401                        BEQ     TST13           ;;BR IF EQUAL
004062  104003                        ERROR   3                       ;OUTPUT REGISTER IN ERROR
                                ;;*****************************************************************
                                ;*TEST 13        TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN
                                ;;*****************************************************************
004064  000004                  TST13:  SCOPE
004066  012737  000400  001166          MOV     #400,$TIMES     ;;DO 400 ITERATIONS
004074  012737  004116  001110          MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
004102  012737  052400  001124          MOV     #52400,$GDDAT           ;LOAD EXPECTED DATA
004110  013777  001124  175222          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT REG.
004116  113777  001124  175214  2$:     MOVB    $GDDAT,@GRDIO           ;LOAD THE OUTPUT
004124  017737  175210  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
004132  023737  001124  001126          CMP     $GDDAT,$BDDAT
004140  001401                        BEQ     1$                      ;BRANCH IF EQUAL
004142  104003                        ERROR   3                       ;ERROR, OUTPUT REG. IN ERROR
004144  105237  001124          1$:     INCB    $GDDAT                  ;UPDATE PATTERN
004150  001362                        BNE     2$

                                ;;*****************************************************************
                                ;*TEST 14        TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN
                                ;;*****************************************************************
004152  000004                  TST14:  SCOPE
004154  012737  000400  001166          MOV     #400,$TIMES     ;;DO 400 ITERATIONS
004162  012737  004204  001110          MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
004170  012737  000252  001124          MOV     #252,$GDDAT             ;LOAD EXPECTED
004176  013777  001124  175134          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT REG.
004204  113777  001125  175130  2$:     MOVB    $GDDAT+1,@GRDIO         ;LOAD THE HIGH BYTE
004212  017737  175122  001126          MOV     @GRDIO,$BDDAT           ;READ  OUTPUT REG.
004220  023737  001124  001126          CMP     $GDDAT,$BDDAT
004226  001401                        BEQ     1$                      ;BRANCH IF EQUAL
004230  104003                        ERROR   3                       ;ERROR, OUTPUT REG. IN ERROR
004232  105237  001125          1$:     INCB    $GDDAT+1
004236  001362                        BNE     2$
```

```
                                        ;;*****************************************************************
                                        ;*TEST 15        TEST OUTPUT DATA ACCEPT FLAG
                                        ;;*****************************************************************
004240  000004                          TST15:  SCOPE
004242  005077  175072                          CLR     @GRDIO
004246  012777  177777  175062                  MOV     #-1,@GRDAI
004254  012737  100000  001124                  MOV     #BIT15,$GDDAT           ;LOAD EXPECTED
004262  012777  100000  175044                  MOV     #BIT15,@GRSTAT          ;SET BIT 15
004270  017737  175040  001126                  MOV     @GRSTAT,$BDDAT          ;READ STATUS
004276  033737  001124  001126                  BIT     $GDDAT,$BDDAT
004304  001001                                  BNE     TST16           ;;BR IF SET
004306  104001                                  ERROR   1                       ;ERROR, BIT 15 FAILED TO SET

                                        ;;*****************************************************************
                                        ;*TEST 16        TEST OUTPUT INTERRUPT ENABLE
                                        ;;*****************************************************************
004310  000004                          TST16:  SCOPE
004312  005077  175016                          CLR     @GRSTAT                 ;CLEAR STATUS
004316  012737  040000  001124                  MOV     #BIT14,$GDDAT           ;LOAD EXPECTED
004324  012777  040000  175002                  MOV     #BIT14,@GRSTAT          ;LOAD BIT 14
004332  017737  174776  001126                  MOV     @GRSTAT,$BDDAT          ;READ STATUS
004340  033737  001124  001126                  BIT     $GDDAT,$BDDAT
004346  001001                                  BNE     TST17           ;;BR IF SET
004350  104001                                  ERROR   1                       ;ERROR BIT 14 FAILED TO SET

                                        ;;*****************************************************************
                                        ;*TEST 17        TEST INPUT DATA READY FLAG
                                        ;;*****************************************************************
004352  000004                          TST17:  SCOPE
004354  012737  000200  001124                  MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
004362  012777  000200  174744                  MOV     #BIT7,@GRSTAT           ;SET BIT 7
004370  017737  174740  001126                  MOV     @GRSTAT,$BDDAT          ;READ RESULT
004376  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
004404  001401                                  BEQ     TST20           ;;BR IF EQUAL
004406  104001                                  ERROR   1                       ;ERROR, BIT 7 FAILED TO SET

                                        ;;*****************************************************************
                                        ;*TEST 20        TEST INPUT INTERRUPT ENABLE
                                        ;;*****************************************************************
004410  000004                          TST20:  SCOPE
004412  005077  174716                          CLR     @GRSTAT                 ;CLEAR STATUS
004416  012737  000100  001124                  MOV     #BIT6,$GDDAT            ;LOAD EXPECTED
004424  012777  000100  174702                  MOV     #BIT6,@GRSTAT           ;SET BIT 6
004432  017737  174676  001126                  MOV     @GRSTAT,$BDDAT          ;READ RESULT
004440  023737  001124  001126                  CMP     $GDDAT,$BDDAT           ;COMPARE
004446  001401                                  BEQ     TST21           ;;BR IF EQUAL
004450  104001                                  ERROR   1                       ;ERROR, BIT 6 FAILED TO SET
```

```
                        ;;*****************************************************************
                        ;*TEST 21       TEST THAT RESET CLEARS THE DIGITAL STATUS REGISTER
                        ;;*****************************************************************
004452  000004         TST21:  SCOPE
004454  012737  000040  001166        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
004462  005077  174646              CLR     @GRSTAT                 ;CLEAR STATUS
004466  005037  001124              CLR     $GDDAT                  ;LOAD EXPECTED
004472  012777  140300  174634      MOV     #BIT15!BIT14!BIT7!BIT6,@GRSTAT
004500  000005                      RESET
004502  017737  174626  001126      MOV     @GRSTAT,$BDDAT          ;READ RESULT
004510  001401                      BEQ     TST22           ;;BR IF EQUAL
004512  104001                      ERROR   1                       ;ERROR, RESET FAILED TO CLEAR DIGITAL
                                                                    ;STATUS REGISTER


                        ;;*****************************************************************
                        ;*TEST 22       TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
                        ;;*****************************************************************
004514  000004         TST22:  SCOPE
004516  032777  002000  174414      BIT     #BIT10,@SWR             ;TEST SWITCH BIT
004524  001405                      BEQ     1$                      ;BRANCH IF DOWN
004526  112737  000036  001102      MOVB    #36,$TSTNM              ;LOAD NEW TEST NUMBER
004534  000137  007050              JMP     DRT21                   ;BYPASS SOME TEST USING THE EXTERNAL CABLE
004540                          1$:
004540  005077  174574              CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
004544  012777  177777  174564      MOV     #-1,@GRDAI              ;CLEAR INPUT
004552  005037  001124              CLR     $GDDAT                  ;CLEAR EXPECTED
004556  012777  000000  174554      MOV     #0,@GRDIO               ;LOAD THE OUTPUT
004564  017737  174546  001126      MOV     @GRDAI,$BDDAT           ;READ THE INPUT
004572  043737  001370  001126      BIC     ODDJMP,$BDDAT           ;MASK
004600  023737  001124  001126      CMP     $GDDAT,$BDDAT           ;COMPARE
004606  001401                      BEQ     TST23           ;;BR IF EQUAL
004610  104002                      ERROR   2                       ;ERROR, INPUT DID NOT EQUAL THE OUTPUT REG.


                        ;;*****************************************************************
                        ;*TEST 23       TEST INPUT WITH #-1
                        ;;*****************************************************************
004612  000004         TST23:  SCOPE
004614  005077  174520              CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
004620  012777  177777  174510      MOV     #-1,@GRDAI              ;CLEAR INPUT
004626  012737  177777  001124      MOV     #-1,$GDDAT              ;LOAD EXPECTED
004634  043737  001370  001124      BIC     ODDJMP,$GDDAT           ;MASK
004642  013777  001124  174470      MOV     $GDDAT,@GRDIO           ;LOAD THE OUTPUT
004650  017737  174462  001126      MOV     @GRDAI,$BDDAT           ;READ INPUT
004656  043737  001370  001126      BIC     ODDJMP,$BDDAT           ;MASK
004664  023737  001124  001126      CMP     $GDDAT,$BDDAT           ;COMPARE
004672  001401                      BEQ     TST24           ;;BR IF EQUAL
004674  104002                      ERROR   2                       ;ERROR, INPUT DID NOT EQUAL THE OUTPUT
```

```
888                                     ;;*****************************************************************
(3)                                     ;*TEST 24       TEST INPUT WITH #52525
(3)                                     ;;*****************************************************************
(2)  004676  000004                     TST24:  SCOPE
889  004700  005077  174434                        CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
890  004704  012777  177777  174424               MOV     #-1,@GRDAI              ;CLEAR INPUT
891  004712  012737  052525  001124               MOV     #52525,$GDDAT           ;LOAD EXPECTED
892  004720  043737  001370  001124               BIC     ODDJMP,$GDDAT           ;MASK
893  004726  013777  001124  174404               MOV     $GDDAT,@GRDIO           ;LOAD THE OUTPUT
894  004734  017737  174376  001126               MOV     @GRDAI,$BDDAT           ;READ INPUT
895  004742  043737  001370  001126               BIC     ODDJMP,$BDDAT           ;MASK
896  004750  023737  001124  001126               CMP     $GDDAT,$BDDAT           ;COMPARE
897  004756  001401                               BEQ     TST25           ;;BR IF EQUAL
898  004760  104002                               ERROR   2                       ;ERROR, INPUT DID NOT EQUAL OUTPUT
899
900                                     ;;*****************************************************************
(3)                                     ;*TEST 25       TEST INPUT WITH #125252
(3)                                     ;;*****************************************************************
(2)  004762  000004                     TST25:  SCOPE
901  004764  005077  174350                        CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
902  004770  012777  177777  174340               MOV     #-1,@GRDAI              ;CLEAR INPUT
903  004776  012737  125252  001124               MOV     #125252,$GDDAT          ;LOAD EXPECTED
904  005004  043737  001370  001124               BIC     ODDJMP,$GDDAT           ;MASK
905  005012  013777  001124  174320               MOV     $GDDAT,@GRDIO           ;LOAD THE OUTPUT
906  005020  017737  174312  001126               MOV     @GRDAI,$BDDAT           ;READ INPUT
907  005026  043737  001370  001126               BIC     ODDJMP,$BDDAT           ;MASK
908  005034  023737  001124  001126               CMP     $GDDAT,$BDDAT           ;COMPARE
909  005042  001401                               BEQ     TST26           ;;BR IF EQUAL
910  005044  104002                               ERROR   2                       ;ERROR, INPUT DID NOT EQUAL OUTPUT
```

```
                                ;;*****************************************************************
                                ;*TEST 26       TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
                                ;;*****************************************************************
005046  000004                TST26:  SCOPE
005050  012737  000001  001166        MOV     #1,$TIMES       ;;DO 1 ITERATION
005056  005737  001370                TST     ODDJMP                  ;TEST IF ANY ODD JUMPERS
005062  001461                        BEQ     TST27           ;;BR IF NONE
005064  012737  010000  001124        MOV     #BIT12,$GDDAT           ;LOAD TEST BIT
005072  033737  001370  001124  1$:   BIT     ODDJMP,$GDDAT           ;TEST IF ODD JUMPER BIT
005100  001447                        BEQ     2$                      ;BR IF NOT
005102  033737  001344  001124        BIT     NOTLCH,$GDDAT           ;TEST IF LATCHING INPUT BIT
005110  001043                        BNE     2$                      ;BR IF NOT
005112  013777  001124  174220        MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
005120  012777  177777  174210        MOV     #-1,@GRDAI              ;CLEAR INPUT
005126  043777  001124  174204        BIC     $GDDAT,@GRDIO           ;CLEAR OUTPUT BIT
005134  017737  174176  001126        MOV     @GRDAI,$BDDAT           ;READ INPUT REG
005142  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
005150  001401                        BEQ     3$                      ;BR IF EQUAL
005152  104002                        ERROR   2                       ;ERROR, NEG. INPUT OR NEG. TRANSITION
                                                                        ; INPUT BIT FAILED TO SET INPUT REGISTER
005154  033737  001124  001372  3$:   BIT     $GDDAT,MINSIN           ;TEST IF NEG. INPUT BIT
005162  001016                        BNE     2$                      ;BR IF
005164  012777  177777  174144        MOV     #-1,@GRDAI              ;CLEAR INPUT
005172  053777  001124  174140        BIS     $GDDAT,@GRDIO           ;LOAD OUTPUT
005200  017737  174132  001126        MOV     @GRDAI,$BDDAT           ;READ INPUT
005206  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
005214  001401                        BEQ     2$                      ;BR IF EQUAL
005216  104002                        ERROR   2                       ;ERROR, POSITIVE INPUT TRANSITION
                                                                        ;LOGIC FAILED TO SET INPUT REGISTER BIT
005220  006137  001124          2$:   ROL     $GDDAT                  ;CHANGE DATA PATTERN
005224  103322                        BCC     1$                      ;BR IF MORE DATA
```

```
                                        ;;*****************************************************************
                                        ;*TEST 27       FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
                                        ;;*****************************************************************
005226  000004                  TST27:  SCOPE
005230  012737  005256  001110          MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN

005236  012737  000001  001364          MOV     #BIT0,BRLEV2            ;LOAD EXPECTED
005244  013737  001344  001366          MOV     NOTLCH,BRLEV3           ;GET NON-LATCH
005252  005137  001366                  COM     BRLEV3                  ;COMPLEMENT
005256  013737  001364  001124  2$:     MOV     BRLEV2,$GDDAT           ;LOAD GOOD
005264  033737  001124  001370          BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
005272  001042                          BNE     1$                      ;BYPASS IF ODD JUMPER
005274  033737  001124  001344          BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCH
005302  001436                          BEQ     1$                      ;BR IF LATCHING

005304  013777  001124  174026          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
005312  017737  174020  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT
005320  043737  001366  001126          BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
005326  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005334  001401                          BEQ     3$              ;;BR IF EQUAL
005336  104002                          ERROR   2                       ;INPUT REGISTER IN ERROR
                                                                        ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?

                                        ;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH

005340  043777  001124  173772  3$:     BIC     $GDDAT,@GRDIO           ;CLEAR OUTPUT BIT
005346  017737  173764  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT
005354  043737  001366  001126          BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
005362  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
005366  033737  001364  001126          BIT     BRLEV2,$BDDAT           ;TEST FOR BIT
005374  001401                          BEQ     1$              ;;BR IF CLEARED
005376  104002                          ERROR   2                       ;INPUT BIT LATCHED IN ERROR
                                                                        ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?

005400  006337  001364          1$:     ASL     BRLEV2                  ;CHANGE PATTERN
005404  001324                          BNE     2$                      ;BR UNTIL DONE
```

```
                                ;;*****************************************************************
                                ;*TEST 30       FLOAT A 1 ACROSS LATCHING INPUT BITS
                                ;;*****************************************************************
005406  000004                  T$T30:  SCOPE
005410  012737  005436  001110          MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
005416  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE
005424  005077  173710                  CLR     @GRDIO                  ;CLEAR OUTPUT REG
005430  012777  177777  173700          MOV     #-1,@GRDAI              ;CLEAR INPUT

005436  033737  001124  001344  2$:     BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCHING
005444  001021                          BNE     1$                      ;BR IF NON-LATCH
005446  033737  001124  001370          BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
005454  001015                          BNE     1$                      ;BYPASS IF ODD JUMPER

005456  013777  001124  173654          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
005464  005077  173650                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
005470  017737  173642  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT REG.
005476  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005504  001401                          BEQ     1$              ;;BR IF EQUAL
005506  104002                          ERROR   2                       ;INPUT REGISTER FAILED TO LATCH DATA

005510  053777  001124  173620  1$:     BIS     $GDDAT,@GRDAI           ;CLEAR INPUT BIT
005516  006337  001124                  ASL     $GDDAT                  ;CHANGE PATTERN
005522  001345                          BNE     2$                      ;BR UNTIL DONE

                                ;;*****************************************************************
                                ;*TEST 31       FLOAT A 0 ACROSS LATCHING INPUT BITS
                                ;;*****************************************************************
005524  000004                  T$T31:  SCOPE
005526  012737  005554  001110          MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
005534  012737  000001  001366          MOV     #BIT0,BRLEV3            ;LOAD EXPECTED
005542  005077  173572                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
005546  012777  177777  173562          MOV     #-1,@GRDAI              ;CLEAR INPUT

005554  033737  001366  001344  2$:     BIT     BRLEV3,NOTLCH           ;TEST FOR LATCHING
005562  001032                          BNE     1$                      ;BR IF NOT
005564  033737  001124  001370          BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
005572  001026                          BNE     1$                      ;BYPASS IF ODD JUMPER BIT

005574  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD
005602  043737  001344  001124          BIC     NOTLCH,$GDDAT
005610  043737  001366  001124          BIC     BRLEV3,$GDDAT           ;MAKE BRLEV3
005616  013777  001124  173514          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
005624  005077  173510                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER

005630  017737  173502  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT
005636  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005644  001401                          BEQ     1$              ;;BR IF EQUAL
005646  104002                          ERROR   2                       ;INPUT REGISTER FAILED TO LATCH DATA

005650  053777  001366  173460  1$:     BIS     BRLEV3,@GRDAI           ;CLEAR INPUT BIT
005656  006337  001366                  ASL     BRLEV3                  ;CHANGE PATTERN
005662  001334                          BNE     2$                      ;BR UNTIL DONE
```

```
1025                                    ;;*****************************************************************
 (3)                                    ;*TEST 32       TEST FOR SLOW INPUT GATES WITH #125252
 (3)                                    ;;*****************************************************************
 (2)  005664  000004                    TST32:  SCOPE
1026
1027  005666  012737  125252  001124            MOV     #125252,$GDDAT          ;LOAD EXPECTED
1028  005674  043737  001344  001124            BIC     NOTLCH,$GDDAT           ;CONVERT
1029  005702  043737  001370  001124            BIC     ODDJMP,$GDDAT           ;MASK
1030  005710  013700  001124                    MOV     $GDDAT,R0               ;LOAD PATTERN
1031  005714  005077  173420                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
1032  005720  012777  177777  173410          MOV     #-1,@GRDAI              ;CLEAR INPUT
1033
1040  005726  010077  173406                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  005732  005077  173402                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  005736  017701  173374                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  005742  050177  173370                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  005746  005100                            COM     R0
 (1)  005750  010077  173364                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  005754  005077  173360                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  005760  017701  173352                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  005764  050177  173346                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  005770  005100                            COM     R0
 (1)  005772  010077  173342                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  005776  005077  173336                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006002  017701  173330                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006006  050177  173324                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006012  005100                            COM     R0
 (1)  006014  010077  173320                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006020  005077  173314                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006024  017701  173306                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006030  050177  173302                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006034  005100                            COM     R0
 (1)  006036  010077  173276                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006042  005077  173272                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006046  017701  173264                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006052  050177  173260                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006056  005100                            COM     R0
 (1)  006060  010077  173254                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006064  005077  173250                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006070  017701  173242                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006074  050177  173236                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006100  005100                            COM     R0
 (1)  006102  010077  173232                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006106  005077  173226                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006112  017701  173220                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006116  050177  173214                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006122  005100                            COM     R0
 (1)  006124  010077  173210                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006130  005077  173204                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006134  017701  173176                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006140  050177  173172                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006144  005100                            COM     R0
 (1)  006146  010077  173166                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006152  005077  173162                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
```

```
   (1)  006156  017701  173154                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006162  050177  173150                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006166  005100                        COM     R0
   (1)  006170  010077  173144                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006174  005077  173140                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006200  017701  173132                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006204  050177  173126                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006210  005100                        COM     R0
   (1)  006212  010077  173122                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006216  005077  173116                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006222  017701  173110                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006226  050177  173104                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006232  005100                        COM     R0
 1041
 1042  006234  010137  001126                MOV     R1,$BDDAT               ;LOAD READ
 1043  006240  000240                        NOP
 1044  006242  000240                        NOP
 1045  006244  000240                        NOP
 1046  006246  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
 1047  006254  001401                        BEQ     TST33           ;;BR IF EQUAL
 1048  006256  104002                        ERROR   2                       ;INPUT GATE SLOW
 1049
 1050                                ;;*****************************************************************
  (3)                                ;*TEST 33       TEST FOR SLOW INPUT GATES WITH #52525
  (3)                                ;;*****************************************************************
  (2)  006260  000004                TST33:  SCOPE
 1051
 1052  006262  012737  052525  001124        MOV     #52525,$GDDAT           ;SETUP EXPECTED
 1053  006270  043737  001370  001124        BIC     ODDJMP,$GDDAT           ;MASK ODD JUMPER BITS
 1054  006276  043737  001344  001124        BIC     NOTLCH,$GDDAT           ;CONVERT
 1055  006304  013700  001124                MOV     $GDDAT,R0
 1056  006310  005077  173024                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 1057  006314  012777  177777  173014        MOV     #-1,@GRDAI              ;CLEAR INPUT
 1058
 1065  006322  010077  173012                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006326  005077  173006                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006332  017701  173000                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006336  050177  172774                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006342  005100                        COM     R0
   (1)  006344  010077  172770                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006350  005077  172764                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006354  017701  172756                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006360  050177  172752                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006364  005100                        COM     R0
   (1)  006366  010077  172746                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006372  005077  172742                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006376  017701  172734                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006402  050177  172730                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006406  005100                        COM     R0
   (1)  006410  010077  172724                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006414  005077  172720                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006420  017701  172712                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006424  050177  172706                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006430  005100                        COM     R0
```

```
(1)  006432  010077  172702          MOV     R0,@GRDIO               ;LOAD OUTPUT
(2)  006436  005077  172676          CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
(1)  006442  017701  172670          MOV     @GRDAI,R1               ;READ INPUT
(1)  006446  050177  172664          BIS     R1,@GRDAI               ;CLEAR INPUT
(1)  006452  005100                  COM     R0
(1)  006454  010077  172660          MOV     R0,@GRDIO               ;LOAD OUTPUT
(2)  006460  005077  172654          CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
(1)  006464  017701  172646          MOV     @GRDAI,R1               ;READ INPUT
(1)  006470  050177  172642          BIS     R1,@GRDAI               ;CLEAR INPUT
(1)  006474  005100                  COM     R0
(1)  006476  010077  172636          MOV     R0,@GRDIO               ;LOAD OUTPUT
(2)  006502  005077  172632          CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
(1)  006506  017701  172624          MOV     @GRDAI,R1               ;READ INPUT
(1)  006512  050177  172620          BIS     R1,@GRDAI               ;CLEAR INPUT
(1)  006516  005100                  COM     R0
(1)  006520  010077  172614          MOV     R0,@GRDIO               ;LOAD OUTPUT
(2)  006524  005077  172610          CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
(1)  006530  017701  172602          MOV     @GRDAI,R1               ;READ INPUT
(1)  006534  050177  172576          BIS     R1,@GRDAI               ;CLEAR INPUT
(1)  006540  005100                  COM     R0
(1)  006542  010077  172572          MOV     R0,@GRDIO               ;LOAD OUTPUT
(2)  006546  005077  172566          CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
(1)  006552  017701  172560          MOV     @GRDAI,R1               ;READ INPUT
(1)  006556  050177  172554          BIS     R1,@GRDAI               ;CLEAR INPUT
(1)  006562  005100                  COM     R0
(1)  006564  010077  172550          MOV     R0,@GRDIO               ;LOAD OUTPUT
(2)  006570  005077  172544          CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
(1)  006574  017701  172536          MOV     @GRDAI,R1               ;READ INPUT
(1)  006600  050177  172532          BIS     R1,@GRDAI               ;CLEAR INPUT
(1)  006604  005100                  COM     R0
(1)  006606  010077  172526          MOV     R0,@GRDIO               ;LOAD OUTPUT
(2)  006612  005077  172522          CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
(1)  006616  017701  172514          MOV     @GRDAI,R1               ;READ INPUT
(1)  006622  050177  172510          BIS     R1,@GRDAI               ;CLEAR INPUT
(1)  006626  005100                  COM     R0
1066
1067  006630  010137  001126          MOV     R1,$BDDAT               ;LOAD VALUE READ
1068  006634  000240                  NOP
1069  006636  000240                  NOP
1070  006640  000240                  NOP
1071  006642  023737  001124  001126  CMP     $GDDAT,$BDDAT           ;COMPARE
1072  006650  001401                  BEQ     TST34           ;;BR IF EQUAL
1073  006652  104002                  ERROR   2
```

```
1075                                    ;;*****************************************************************
 (3)                                    ;*TEST 34       TEST THAT RESET CLEARS INPUT REGISTER BITS
 (3)                                    ;;*****************************************************************
 (2) 006654 000004                      TST34:  SCOPE
 (1) 006656 012737 000040 001166               MOV     #40,$TIMES      ;;DO 40 ITERATIONS
1076 006664 005077 172450                      CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
1077 006670 012777 177777 172440               MOV     #-1,@GRDAI              ;CLEAR INPUT
1078 006676 012777 177777 172434               MOV     #-1,@GRDIO              ;LOAD OUTPUT
1079 006704 005037 001124                      CLR     $GDDAT                  ;LOAD EXPECTED
1080 006710 000005                             RESET
1081 006712 017737 172420 001126               MOV     @GRDAI,$BDDAT           ;READ INPUT REG.
1082 006720 043737 001370 001126               BIC     ODDJMP,$BDDAT           ;MASK ODD JUMPERS
1083 006726 001401                             BEQ     TST35           ;;BR IF ALL BITS CLEARED
1084 006730 104002                             ERROR   2                       ;INPUT REG. FAILED TO CLEAR UPON RESET INST.
1085
1086                                    ;;*****************************************************************
 (3)                                    ;*TEST 35       TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
 (3)                                    ;;*****************************************************************
 (2) 006732 000004                      TST35:  SCOPE
1087 006734 012737 000200 001124               MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
1088 006742 005077 172366                      CLR     @GRSTAT
1089 006746 012777 000000 172364               MOV     #0,@GRDIO               ;OUTPUT 0
1090 006754 022727 000000 000000               CMP     #0,#0                   ;DELAY
1091 006762 017737 172346 001126               MOV     @GRSTAT,$BDDAT          ;READ RESULTS
1092 006770 023737 001124 001126               CMP     $GDDAT,$BDDAT           ;COMPARE
1093 006776 001401                             BEQ     TST36           ;;BR IF EQUAL
1094 007000 104001                             ERROR   1                       ;INPUT DATA READY FLAG FAILED TO SET
1095
1096                                    ;TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
1097
1098                                    ;;*****************************************************************
 (3)                                    ;*TEST 36       TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
 (3)                                    ;;*****************************************************************
 (2) 007002 000004                      TST36:  SCOPE
1099 007004 012737 100000 001124               MOV     #BIT15,$GDDAT           ;LOAD EXPECTED
1100 007012 005077 172316                      CLR     @GRSTAT
1101 007016 012777 000000 172314               MOV     #0,@GRDIO
1102 007024 022727 000000 000000               CMP     #0,#0
1103 007032 017700 172300                      MOV     @GRDAI,R0               ;READ INPUT
1104 007036 017737 172272 001126               MOV     @GRSTAT,$BDDAT          ;READ RESULTS
1105 007044 100401                             BMI     TST37           ;;BR IF SET
1106 007046 104001                             ERROR   1                       ;INPUT DATA READY FLAG FAILED TO SET
1107
```

```
 1109  007050                                 DRT21:
  (4)                                         ;;*****************************************************************
  (3)                                         ;*TEST 37       TEST FOR UNEXPECTED INTERRUPT
  (3)                                         ;;*****************************************************************
  (2)  007050  000004                         TST37:  SCOPE
  (1)  007052  012737  000040  001166                MOV     #40,$TIMES      ;;DO 40 ITERATIONS
 1110  007060  000005                                RESET
 1111  007062  005077  172246                        CLR     @GRSTAT
 1112  007066  005037  177776                        CLR     PSW
 1113  007072  012777  007152  172250                MOV     #1$,@GRIVA              ;SET UP INTERRUPT INPUT VECTOR
 1114  007100  012777  000340  172244                MOV     #340,@GRIVSA
 1115  007106  012777  007156  172240                MOV     #2$,@GRIVB              ;SET UP INTERRUPT OUTPUT VECTOR
 1116  007114  012777  000340  172234                MOV     #340,@GRIVSB
 1117  007122  012777  000000  172210                MOV     #0,@GRDIO               ;OUTPUT
 1118  007130  017700  172202                        MOV     @GRDAI,R0               ;INPUT
 1119  007134  000240                                NOP
 1120  007136  000240                                NOP
 1121  007140  000240                                NOP
 1122  007142  000240                                NOP
 1123  007144  005077  172164                        CLR     @GRSTAT                 ;CLEAR STATUS
 1124  007150  000404                                BR      TST40           ;;
 1125
 1126  007152  104006                        1$:     ERROR   6                       ;ERROR, DIGITAL INPUT INTERRUPTED
 1127  007154  000002                                RTI
 1128
 1129  007156  104006                        2$:     ERROR   6                       ;ERROR, DIGITAL OUTPUT INTERRUPTED
 1130  007160  000002                                RTI
 1131                                          ;;*****************************************************************
  (3)                                         ;*TEST 40       TEST THAT THE INPUT CAN INTR. USING THE MAINT. BIT
  (3)                                         ;;*****************************************************************
  (2)  007162  000004                         TST40:  SCOPE
 1132  007164  005077  172144                        CLR     @GRSTAT                 ;CLEAR STATUS
 1133  007170  012777  007242  172152                MOV     #1$,@GRIVA              ;LOAD RETURN VECTOR
 1134  007176  012777  000340  172146                MOV     #340,@GRIVSA            ;
 1135  007204  012777  007264  172142                MOV     #2$,@GRIVB              ;LOAD OUTPUT VECTOR
 1136  007212  012777  000340  172136                MOV     #340,@GRIVSB
 1137  007220  052777  000040  172106                BIS     #BIT5,@GRSTAT           ;MAINT. INT C
 1138  007226  005037  177776                        CLR     PSW                     ;LOWER PSW
 1139  007232  000240                                NOP
 1140  007234  000240                                NOP
 1141  007236  104004                                ERROR   4                       ;ERROR, INPUT FAILED TO INTERRUPT
 1142  007240  000415                                BR      TST41           ;;BR TO NEXT TEST
 1143
 1144                                          ;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN NO INTERRUPT WILL OCCUR
 1145                                          ; IF INTERRUPT OCCURS 'INT DONE C' FAILED TO CLEAR INT C FLOP
 1146
 1147  007242  012777  007264  172100  1$:     MOV     #2$,@GRIVA              ;LOAD INPUT VECTOR
 1148  007250  022626                                CMP     (SP)+,(SP)+             ;POP THE STACK *4
 1149  007252  005037  177776                        CLR     PSW                     ;LOWER PRIOR.
 1150  007256  000240                                NOP
 1151  007260  000240                                NOP
 1152  007262  000404                                BR      TST41           ;;<NEXT TEST>
 1153
 1154  007264  022626                        2$:     CMP     (SP)+,(SP)+             ;POP THE STACK
```

```
 1155  007266  104006                    ERROR   6                       ;UNEXPECTED INTERRUPT
 1156  007270  005037  177776            CLR     PSW
```

```
1158                                    ;;*****************************************************************
 (3)                                    ;*TEST 41       TEST THAT THE INPUT INTR. CLEARS INT. ENABLE VIA MAINT. BIT
 (3)                                    ;*****************************************************************
 (2)  007274  000004                    TST41:  SCOPE
1159  007276  005077  172032                      CLR     @GRSTAT                 ;CLEAR STATUS
1160  007302  012777  007360  172040              MOV     #1$,@GRIVA              ;LOAD RETURN VECTOR
1161  007310  012777  000340  172034              MOV     #340,@GRIVSA            ;
1162  007316  012777  007422  172030              MOV     #2$,@GRIVB              ;LOAD OUTPUT VECTOR
1163  007324  012777  000340  172024              MOV     #340,@GRIVSB
1164  007332  012777  000100  171774              MOV     #BIT6,@GRSTAT           ;LOAD INPUT INTR. ENABLE
1165  007340  052777  000040  171766              BIS     #BIT5,@GRSTAT           ;MAINT. INT C
1166  007346  005037  177776                      CLR     PSW                     ;LOWER PSW
1167  007352  000240                              NOP
1168  007354  000240                              NOP
1169  007356  104004                              ERROR   4                       ;ERROR, INPUT FAILED TO INTERRUPT
1170  007360  012777  007422  171762  1$:         MOV     #2$,@GRIVA              ;LOAD INPUT VECTOR
1171  007366  022626                              CMP     (SP)+,(SP)+             ;POP THE STACK *4
1172  007370  005037  177776                      CLR     PSW                     ;LOWER PRIOR.
1173  007374  005037  001124                      CLR     $GDDAT                  ;CLEAR EXPECTED
1174  007400  017737  171730  001126              MOV     @GRSTAT,$BDDAT          ;READ STATUS
1175  007406  032737  000100  001126              BIT     #BIT6,$BDDAT            ;TEST BIT 6
1176  007414  001406                              BEQ     TST42           ;;BR IF CLEARED
1177  007416  104001                              ERROR   1                       ;INPUT INTR. FAILED TO CLEAR
1178  007420  000404                              BR      TST42           ;;<NEXT TEST>
1179  007422  022626                      2$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1180  007424  104006                              ERROR   6                       ;UNEXPECTED INTERRUPT
1181  007426  005037  177776                      CLR     PSW
1182                                    ;;*****************************************************************
 (3)                                    ;*TEST 42       TEST THAT THE OUTPUT CAN INTR. USING THE MAINT. BIT
 (3)                                    ;*****************************************************************
 (2)  007432  000004                    TST42:  SCOPE
1183  007434  005077  171674                      CLR     @GRSTAT                 ;CLEAR STATUS
1184  007440  012777  007534  171702              MOV     #2$,@GRIVA              ;LOAD INPUT VECTOR
1185  007446  012777  000340  171676              MOV     #340,@GRIVSA
1186  007454  012777  007512  171672              MOV     #1$,@GRIVB              ;LOAD OUTPUT VECTOR
1187  007462  012777  000340  171666              MOV     #340,@GRIVSB
1188  007470  052777  020000  171636              BIS     #BIT13,@GRSTAT          ;MAINT. INTERRUPT
1189  007476  005037  177776                      CLR     PSW                     ;LOWER PSW
1190  007502  000240                              NOP
1191  007504  000240                              NOP
1192  007506  104005                              ERROR   5                       ;OUTPUT FAILED TO INTERRUPT
1193  007510  000415                              BR      TST43           ;;BR TO NEXT TEST
1194                                  ;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN, NO INTERRUPT WILL OCCUR
1195                                  ; IF IT DOES, 'INT DONE D HIGH' FAILED TO CLEAR 'INT D' FLOP
1196  007512  012777  007534  171634  1$:         MOV     #2$,@GRIVB              ;LOAD OUTPUT VECTOR
1197  007520  022626                              CMP     (SP)+,(SP)+             ;POP THE STACK *4
1198  007522  005037  177776                      CLR     PSW
1199  007526  000240                              NOP
1200  007530  000240                              NOP
1201  007532  000404                              BR      TST43           ;;<NEXT TEST>
1202  007534  022626                      2$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1203  007536  104006                              ERROR   6                       ;UNEXPECTED INTERRUPT
1204  007540  005037  177776                      CLR     PSW
1205
```

```
1207                                    ;;*****************************************************************
 (3)                                    ;*TEST 43       TEST FOR INTR. FROM DRA INPUT
 (3)                                    ;;*****************************************************************
 (2)  007544  000004                    TST43:  SCOPE
1208  007546  000240                            NOP
1209  007550  000240                            NOP
1210  007552  032777  002000  171360            BIT     #BIT10,@SWR             ;TEST SWITCH
1211  007560  001401                            BEQ     1$
1212  007562  000463                            BR      TST44           ;;
1213  007564  005077  171544            1$:     CLR     @GRSTAT
1214  007570  012777  007670  171552            MOV     #2$,@GRIVA              ;IN CASE OF INTERRUPT
1215  007576  012777  000340  171546            MOV     #340,@GRIVSA
1216  007604  012777  007666  171542            MOV     #3$,@GRIVB
1217  007612  012777  000340  171536            MOV     #340,@GRIVSB
1218  007620  012777  000100  171506            MOV     #BIT6,@GRSTAT           ;ENABLE INPUT INTR.
1219  007626  012777  000000  171504            MOV     #0,@GRDIO               ;OUTPUT
1220  007634  017700  171476                    MOV     @GRDAI,R0               ;INPUT
1221  007640  005037  177776                    CLR     PSW                     ;LOWER PSW
1222  007644  000240                            NOP                             ;LET INTERRUPT OCCUR
1223  007646  000240                            NOP
1224  007650  000240                            NOP
1225  007652  042777  000100  171454            BIC     #BIT6,@GRSTAT
1226  007660  000240                            NOP
1227  007662  104004                            ERROR   4                       ;ERROR, INPUT FAILED TO INTERRUPT
1228  007664  000404                            BR      4$              ;;
1229  007666  104006                    3$:     ERROR   6                       ;UNEXPECTED OUTPUT INTERRUPT
1230  007670  022626                    2$:     CMP     (SP)+,(SP)+
1231  007672  005037  177776                    CLR     PSW
1232  007676  005077  171432            4$:     CLR     @GRSTAT                 ;CLEAR STATUS
1233  007702  012777  002746  171440            MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
 (1)  007710  012777  000340  171434            MOV     #340,@GRIVSA
 (1)  007716  012777  002746  171430            MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
 (1)  007724  012777  000340  171424            MOV     #340,@GRIVSB
1234
```

```
                                ;;*****************************************************************
                                ;*TEST 44       TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
                                ;;*****************************************************************
                        TST44:  SCOPE
007734  032777  002000  171176          BIT     #BIT10,@SWR             ;TEST CABLE SWITCH
007742  001067                          BNE     TST45           ;;BYPASS IF NO I/O CABLE

007744  012737  007760  001110          MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
007752  012737  000001  001366          MOV     #BIT0,BRLEV3            ;LOAD INTERRUPT BIT
007760  005037  001126          1$:     CLR     $BDDAT                  ;CLEAR BAD DATA
007764  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD GOOD DATA

007772  033737  001366  001346          BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
010000  001445                          BEQ     3$              ;;NO TRY NEXT BIT
010002  005077  171332                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
010006  012777  177777  171322          MOV     #-1,@GRDAI              ;CLEAR INPUT
010014  005077  171314                  CLR     @GRSTAT                 ;CLEAR STATUS
010020  053777  001366  171312          BIS     BRLEV3,@GRDIO           ;LOAD OUTPUT
010026  043777  001366  171304          BIC     BRLEV3,@GRDIO           ;CLEAR OUTPUT
010034  053777  001366  171276          BIS     BRLEV3,@GRDIO           ;LOAD OUTPUT
010042  005077  171266                  CLR     @GRSTAT                 ;CLEAR FLAG FROM DATA READY
                                                                        ;SHOULD REMAIN SET VIA DIRECT SET SIDE
                                                                        ;IF INTERRUPT INPUT SWITCH IS ON
010046  105777  171262                  TSTB    @GRSTAT                 ;TEST READY BIT
010052  100401                          BMI     2$              ;;BR IF SET
010054  104010                          ERROR   10                      ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
                                                                        ;?? DID OPERATOR GIVE CORRECT
                                                                        ;INPUT INTERRUPT BITS ??

010056  043777  001366  171254  2$:     BIC     BRLEV3,@GRDIO           ;CLEAR OUTPUT
010064  053777  001366  171244          BIS     BRLEV3,@GRDAI           ;CLEAR INPUT
010072  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
010076  005077  171232                  CLR     @GRSTAT                 ;CLEAR STATUS
010102  117737  171226  001126          MOVB    @GRSTAT,$BDDAT          ;READ STATUS
010110  100001                          BPL     3$              ;;BR IF CLEARED
010112  104001                          ERROR   1                       ;INPUT READY FAILED TO CLEAR

010114  006337  001366          3$:     ASL     BRLEV3                  ;CHANGE BIT
010120  001317                          BNE     1$                      ;BR IF NOT DONE
```

```
                                        ;;*****************************************************************
                                        ;*TEST 45       TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
                                        ;;*****************************************************************
010122  000004                  TST45:  SCOPE
010124  032777  002000  171006          BIT     #BIT10,@SWR             ;TEST CABLE SWITCH
010132  001050                        BNE     TST46           ;;BYPASS IF NO I/O CABLE

010134  012737  010150  001110          MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
010142  012737  000001  001366          MOV     #BIT0,BRLEV3            ;LOAD NON-INTERRUPT BIT
010150  012737  000200  001126  1$:     MOV     #200,$BDDAT             ;LOAD BAD DATA
010156  005037  001124                  CLR     $GDDAT                  ;CLEAR GOOD DATA

010162  033737  001366  001346          BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
010170  001026                        BNE     3$              ;;YES SKIP AND TRY NEXT BIT
010172  005077  171142                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
010176  012777  177777  171132          MOV     #-1,@GRDAI              ;CLEAR INPUT
010204  005077  171124                  CLR     @GRSTAT                 ;CLEAR STATUS
010210  053777  001366  171122          BIS     BRLEV3,@GRDIO           ;LOAD OUTPUT
010216  043777  001366  171114          BIC     BRLEV3,@GRDIO           ;CLEAR OUTPUT
010224  053777  001366  171106          BIS     BRLEV3,@GRDIO           ;SET OUTPUT
010232  005077  171076                  CLR     @GRSTAT                 ;CLEAR FLAG FROM DATA READY
                                                                        ;SHOULD REMAIN SET VIA DIRECT SET SIDE
010236  105777  171072                  TSTB    @GRSTAT                 ;TEST READY BIT
010242  100001                        BPL     3$              ;;BR IF CLEAR
010244  104011                        ERROR   11                      ;INPUT NON-INTERRUPT BIT SET INPUT READY
                                                                        ;?? DID OPERATOR GIVE CORRECT
                                                                        ;INPUT INTERRUPT BITS ??

010246  006337  001366          3$:     ASL     BRLEV3                  ;CHANGE BIT
010252  001336                        BNE     1$                      ;BR IF NOT DONE
```

```
1302                                    ;;*****************************************************************
 (3)                                    ;*TEST 46       TEST FOR INTR. FROM DRA OUTPUT
 (3)                                    ;;*****************************************************************
 (2) 010254 000004                      TST46:  SCOPE
1303 010256 000240                              NOP
1304 010260 000240                              NOP
1305 010262 032777 002000 170650                BIT     #BIT10,@SWR             ;TEST SWITCH
1306 010270 001401                              BEQ     1$
1307 010272 000461                              BR      TST47           ;;
1308 010274 005077 171034               1$:     CLR     @GRSTAT
1309 010300 012777 010374 171046                MOV     #2$,@GRIVB      ;IN CASE OF INTERRUPT
1310 010306 012777 000340 171042                MOV     #340,@GRIVSB
1311 010314 013777 001352 171026                MOV     GRIVSA,@GRIVA
1312 010322 005077 171024                       CLR     @GRIVSA
1313 010326 012777 040000 171000                MOV     #BIT14,@GRSTAT          ;ENABLE OUTPUT INTR.
1314 010334 012777 000000 170776                MOV     #0,@GRDIO               ;OUTPUT
1315 010342 017700 170770                       MOV     @GRDAI,R0               ;INPUT
1316 010346 005037 177776                       CLR     PSW                     ;LOWER PSW
1317 010352 000240                              NOP                             ;LET INTERRUPT OCCUR
1318 010354 000240                              NOP
1319 010356 000240                              NOP
1320 010360 042777 040000 170746                BIC     #BIT14,@GRSTAT
1321 010366 000240                              NOP
1322 010370 104005                              ERROR   5                       ;ERROR, OUTPUT FAILED TO INTERRUPT
1323 010372 000401                              BR      3$              ;;
1324 010374 022626                      2$:     CMP     (SP)+,(SP)+
1325 010376                             3$:
 (1) 010376 012777 002746 170744                MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
 (1) 010404 012777 000340 170740                MOV     #340,@GRIVSA
 (1) 010412 012777 002746 170734                MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
 (1) 010420 012777 000340 170730                MOV     #340,@GRIVSB
1326 010426 005037 177776                       CLR     PSW
1327 010432 005077 170676                       CLR     @GRSTAT                 ;CLEAR STATUS
1328
1329                                    ;;*****************************************************************
 (3)                                    ;*TEST 47       PRE-INTERRUPT SETUP
 (3)                                    ;;*****************************************************************
 (2) 010436 000004                      TST47:  SCOPE
 (1) 010440 012737 000001 001166                MOV     #1,$TIMES       ;;DO 1 ITERATION
1330 010446 000005                              RESET
1331 010450 042737 177437 001360                BIC     #177437,DIOBRL
1332 010456 001001                              BNE     1$
1333 010460 104007                              ERROR   7                       ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 0
1334 010462 022737 000340 001360        1$:     CMP     #340,DIOBRL
1335 010470 001001                              BNE     2$
1336 010472 104007                              ERROR   7                       ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 7
1337 010474 013737 001360 001362        2$:     MOV     DIOBRL,BRLEV1
1338 010502 162737 000040 001362                SUB     #40,BRLEV1
1339 010510 013737 001360 001364                MOV     DIOBRL,BRLEV2
```

```
1341                                    ;;*****************************************************************
 (3)                                    ;*TEST 50       TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT
 (3)                                    ;;*****************************************************************
 (2)  010516  000004                    TST50:  SCOPE
1342  010520  005077  170610                    CLR     @GRSTAT                 ;CLEAR ENABLES
1343  010524  012777  010576  170616            MOV     #1$,@GRIVA              ;SET UP VECTORS
1344  010532  013737  001362  177776            MOV     BRLEV1,PSW              ;CHANGE PSW
1345  010540  005737  001376                    TST     JUMPER                  ;TEST IF FACTORY MODE
1346  010544  001403                            BEQ     2$                      ;BR IF YES
1347  010546  012737  000200  177776            MOV     #200,PSW                ;LOAD BR LEVEL #5-1 IF LOC-BOX OR COULTER MODE
1348  010554  052777  000040  170552  2$:       BIS     #BIT5,@GRSTAT           ;GENERATE INPUT MAINT. INTERRUPT
1349  010562  000240                            NOP
1350  010564  000240                            NOP
1351  010566  000240                            NOP
1352  010570  000240                            NOP
1353  010572  104004                            ERROR   4                       ;ERROR, NO INTERRUPT FROM DIGITAL I/O INPUT
1354  010574  000401                            BR      3$              ;;BR TO NEXT TEST
1355  010576  022626                    1$:     CMP     (SP)+,(SP)+
1356  010600                            3$:
 (1)  010600  012777  002746  170542            MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
 (1)  010606  012777  000340  170536            MOV     #340,@GRIVSA
 (1)  010614  012777  002746  170532            MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOP
 (1)  010622  012777  000340  170526            MOV     #340,@GRIVSB
1357  010630  005037  177776                    CLR     PSW                     ;LOWER PSW
1358                                    ;;*****************************************************************
 (3)                                    ;*TEST 51       TEST FOR NO INTR. FROM DRA INPUT ON LEVEL INDICATED  VIA MAINT. INT
 (3)                                    ;;*****************************************************************
 (2)  010634  000004                    TST51:  SCOPE
1359  010636  005077  170472                    CLR     @GRSTAT                 ;CLEAR ENABLES
1360  010642  012777  010730  170500            MOV     #1$,@GRIVA              ;SET UP VECTORS
1361  010650  013737  001364  177776            MOV     BRLEV2,PSW              ;CHANGE PSW
1362  010656  005737  001376                    TST     JUMPER                  ;TEST IF FACTORY MODE
1363  010662  001403                            BEQ     4$                      ;BR IF YES
1364  010664  012737  000240  177776            MOV     #240,PSW                ;LOAD BR LEVEL #5 IF LOC-BOX OR COULTER MODE
1365  010672  052777  000040  170434  4$:       BIS     #BIT5,@GRSTAT           ;GENERATE INPUT MAINT. INTERRUPT
1366  010700  000240                            NOP
1367  010702  000240                            NOP
1368  010704  000240                            NOP
1369                                    ;SUB-TEST, NOW LOWER THE PSW AND ALLOW THE INTERRUPT
1370  010706  012777  010742  170434            MOV     #2$,@GRIVA              ;RESET VECTOR
1371  010714  005037  177776                    CLR     PSW                     ;LOWER PSW
1372  010720  000240                            NOP
1373  010722  000240                            NOP
1374  010724  000240                            NOP
1375  010726  000403                            BR      3$                      ;ERROR
1376  010730  022626                    1$:     CMP     (SP)+,(SP)+
1377  010732  005037  177776                    CLR     PSW
1378  010736  104006                    3$:     ERROR   6                       ;UNEXPECTED INTERRUPT <IS BR LEVEL PLUG CORRECT>
1379  010740  000401                            BR      5$                      ;;
1380  010742  022626                    2$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1381  010744                            5$:
 (1)  010744  012777  002746  170376            MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
 (1)  010752  012777  000340  170372            MOV     #340,@GRIVSA
 (1)  010760  012777  002746  170366            MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
```

```
(1)  010766  012777  000340  170362          MOV     #340,@GRIVSB
1382 010774  005037  177776                  CLR     PSW
1383
1384                                ;;*****************************************************************
(2)                                 ;*TEST 52       DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
(3)                                 ;;*****************************************************************
(2)  011000  000004                 TST52:  SCOPE
(1)  011002  012737  000001  001166          MOV     #1,$TIMES       ;;DO 1 ITERATION
1385 011010  005737  001320         BYPASS: TST     NBEXT                   ;TEST IF ANY
1386 011014  001415                         BEQ     1$                      ;BR IF NONE
1387 011016  032777  010000  170114         BIT     #SW12,@SWR              ;TEST BIT12 OF SWR
1388 011024  001024                         BNE     BYPAS1                  ;INHIBIT TESTING NEXT DR11-K
1389 011026  162737  000010  001322         SUB     #10,DRADD               ;UPDATE DEVICE ADDRESS
1390 011034  062737  000010  001324         ADD     #10,DRIV                ;UPDATE DEVICE VECTOR
1391 011042  005337  001320                 DEC     NBEXT                   ;ANOTHER ONE ?
1392 011046  000413                         BR      BYPAS1                  ;BR IF ANOTHER
1393 011050  013737  001306  001322  1$:    MOV     BASEBA,DRADD            ;RELOAD ADDRESS
1394 011056  013737  001310  001324         MOV     BASEIV,DRIV             ;RELOAD VECTOR
1395 011064  013737  001316  001320         MOV     NMBEXT,NBEXT            ;RELOAD NUMBER
1396 011072  000137  011106                 JMP     $EOP                    ;DONE
1397 011076  012700  177777  BYPAS1: MOV     #-1,R0
1398 011102  000137  002776                 JMP     RBEG2                   ;TEST ANOTHER UNIT
1399
1400                                .SBTTL  END OF PASS ROUTINE
(1)
(2)                                 ;;*****************************************************************
(1)                                 ;*INCREMENT THE PASS NUMBER ($PASS)
(1)                                 ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
(1)                                 ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
(1)                                 ;*IF THERES A MONITOR GO TO IT
(1)                                 ;*IF THERE ISN'T JUMP TO BYPAS1
(1)
(1)  011106                         $EOP:
(1)  011106  000004                         SCOPE
(1)  011110  005037  001102                 CLR     $TSTNM          ;;ZERO THE TEST NUMBER
(1)  011114  005037  001166                 CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
(1)  011120  005237  001100                 INC     $PASS           ;;INCREMENT THE PASS NUMBER
(1)  011124  042737  100000  001100         BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
(1)  011132  005327                         DEC     (PC)+           ;;LOOP?
(1)  011134  000001                 $EOPCT: .WORD   1
(1)  011136  003022                         BGT     $DOAGN          ;;YES
(1)  011140  012737                         MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
(1)  011142  000001                 $ENDCT: .WORD   1
(1)  011144  011134                         $EOPCT
(1)  011146  104401  011213                 TYPE    ,$ENDMG         ;;TYPE "END PASS #"
(2)  011152  013746  001100                 MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
(2)  011156  104405                         TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1)  011160  104401  011210                 TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
(1)  011164  013700  000042         $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
(1)  011170  001405                         BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
(1)  011172  000005                         RESET                   ;;CLEAR THE WORLD
(1)  011174  004710                 $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
(1)  011176  000240                         NOP                     ;;SAVE ROOM
(1)  011200  000240                         NOP                     ;;FOR
```

```
(1)  011202  000240                          NOP                     ;;ACT11
(1)  011204                          $DOAGN:
(1)  011204  000137                          JMP     @(PC)+          ;;RETURN
(1)  011206  011076                  $RTNAD: .WORD   BYPAS1
(1)  011210     377      377      000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
(1)  011213     015   042412   042116 $ENDMG: .ASCIZ  <15><12>/END PASS #/
(1)  011220  050040   051501   020123
(1)  011226  000043
1401                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)
(1)                                   ;;*****************************************************************
(1)                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)                                   ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)                                   ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)                                   ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)                                   ;*REPLACED WITH SPACES.
(1)                                   ;*CALL:
(1)                                   ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
(1)                                   ;*      TYPDS                   ;;GO TO THE ROUTINE
(1)
(2)  011230                           $TYPDS:
(3)  011230  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  011232  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  011234  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  011236  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  011240  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  011242  012746  020200                   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  011246  016605  000020                   MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  011252  100004                           BPL     1$              ;;BR IF INPUT IS POS.
(1)  011254  005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  011256  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  011264  005000                   1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  011266  012703  011444                   MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  011272  112723  000040                   MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  011276  005002                   2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  011300  016001  011434                   MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  011304  160105                   3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
(1)  011306  002402                           BLT     4$              ;;BR IF DONE
(1)  011310  005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
(1)  011312  000774                           BR      3$
(1)  011314  060105                   4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  011316  005702                           TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  011320  001002                           BNE     5$              ;;FALL THROUGH IF 0
(1)  011322  105716                           TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  011324  100407                           BMI     7$              ;;BR IF YES
(1)  011326  106316                   5$:     ASLB    (SP)            ;;MSD?
(1)  011330  103003                           BCC     6$              ;;BR IF NO
(1)  011332  116663  000001  177777           MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  011340  052702  000060                   BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  011344  052702  000040                   BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  011350  110223                           MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  011352  005720                           TST     (R0)+           ;;JUST INCREMENTING
(1)  011354  020027  000010                   CMP     R0,#10          ;;CHECK THE TABLE INDEX
(1)  011360  002746                           BLT     2$              ;;GO DO THE NEXT DIGIT
```

```
(1)  011362  003002                        BGT     8$              ;;GO TO EXIT
(1)  011364  010502                        MOV     R5,R2           ;;GET THE LSD
(1)  011366  000764                        BR      6$              ;;GO CHANGE TO ASCII
(1)  011370  105726                 8$:    TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  011372  100003                        BPL     9$              ;;BR IF NO
(1)  011374  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  011402  105013                 9$:    CLRB    (R3)            ;;SET THE TERMINATOR
(3)  011404  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  011406  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  011410  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  011412  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  011414  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  011416  104401  011444                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  011422  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  011430  012616                        MOV     (SP)+,(SP)
(1)  011432  000002                        RTI                     ;;RETURN TO USER
(1)  011434  023420                 $DTBL: 10000.
(1)  011436  001750                        1000.
(1)  011440  000144                        100.
(1)  011442  000012                        10.
(1)  011444  000004                 $DBLK: .BLKW   4
1402
1403                                        .SBTTL  MISC. EXTERNAL LOGIC TEST
1404 011454  012706  001100         EXTTST: MOV     #STACK,SP
1405 011460  004737  003036                 JSR     PC,SETADD               ;SET UP ADDRESS AND VECTOR
1406 011464  012737  040000  011560  2$:    MOV     #BIT14,EXTCNT           ;LOAD COUNT
1407 011472                         1$:
 (1) 011472  005077  167642                 CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
1408 011476  012777  125252  167634        MOV     #125252,@GRDIO          ;LOAD OUTPUT
1409 011504  017737  167626  011556        MOV     @GRDAI,EXTTMP           ;READ INPUT
1410 011512  013777  011556  167 16        MOV     EXTTMP,@GRDAI           ;CLEAR INPUT
1411 011520  005077  167614                 CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
1412 011524  012777  052525  167606        MOV     #52525,@GRDIO           ;LOAD OUTPUT
1413 011532  017737  167600  011556        MOV     @GRDAI,EXTTMP           ;READ INPUT
1414 011540  013777  011556  167570        MOV     EXTTMP,@GRDAI           ;CLEAR INPUT
1415 011546  005337  011560                 DEC     EXTCNT                  ;FINISHED COUNT
1416 011552  001347                        BNE     1$
1417 011554  000743                        BR      2$                      ;LOOP
1418
1419 011556  000000                 EXTTMP: 0
1420 011560  000000                 EXTCNT: 0
1421
1422                                        ;SUBROUTINE TO LOAD TRAP AND POWER FAIL VECTORS
1423 011562  012737  017466  000034 SETTRP: MOV     #$TRAP,@#TRAPVEC        ;LOAD TRAP VECTOR
1424 011570  012737  000340  000036        MOV     #340,@#TRAPVEC+2
1425 011576  012737  016022  000024        MOV     #$PWRDN,@#PWRVEC        ;LOAD POWER FAIL VECTOR
1426 011604  012737  000340  000026        MOV     #340,@#PWRVEC+2
1427 011612  000207                        R       PC                      ;EXIT
```

```
1429                                            .SBTTL  COULTER INTERFACE TEST
1430   011614  012706  001100          COULTR: MOV     #STACK,SP
1431   011620  004737  011562                  JSR     PC,SET                  ;LOAD TRAP AND POWER FAIL VECTORS
1432   011624  004737  003036                  JSR     PC,SETA                 ;SETUP BUS ADDRESS AND VECTOR
1433   011630  005077  167500                  CLR     @GRSTAT                 ;CLEAR INPUT STATUR REG
1434   011634  012777  011676  167506          MOV     #20$,@GRIVA             ;LOAD INPUT VECTOR
1435   011642  012777  000340  167502          MOV     #340,@GRIVSA
1436   011650  104401  013302          1$:     TYPE,RUNMSG                     ;TELL OPERATOR TO "RUN SAMPLE"
1437   011654  012701  017550                  MOV     #BUFFER,R1              ;LOAD POINTER FOR RESULTS
1438   011660  012777  000100  167446  4$:     MOV     #BIT6,@GRSTAT           ;ENABLE INTERRUPT
1439   011666  005037  177776                  CLR     PS                      ;LOWER PSW
1440   011672  000001                          WAIT                            ;WAIT FOR OPERATOR
1441   011674  000771                          BR      4$
1442   011676  022626          20$:    CMP     (SP)+,(SP)+             ;POP STACK
1443   011700  012737  000106  011556          MOV     #70.,EXTTMP             ;LOAD # OF 1 MSEC DELAYS
1444   011706  013700  001314          6$:     MOV     CPU,R0  ;LOAD 1 MSEC. DELAY WEIGHT ;240 FOR 11/05  ;620 FOR 11/40
1445   011712  005300                  3$:     DEC     R0                      ;DELAY
1446   011714  001376                          BNE     3$
1447   011716  005337  011556                  DEC     EXTTMP                  ;FINISHED ALL MSEC. DELAY ?
1448   011722  001371                          BNE     6$                      ;BR IF NOT
1449   011724  017700  167406                  MOV     @GRDAI,R0               ;SAVE RESULTS
1450   011730  010021                          MOV     R0,(R1)+                ;SAVE IN BUFFER
1451   011732  012777  177777  167376          MOV     #-1,@GRDAI              ;CLEAR INPUT
1452   011740  042700  0077??                  BIC     #7777,R0                ;MASK
1453   011744  001345                          BNE     4$                      ;BR IF NOT LAST SAMPLE
1454   011746  010103                          MOV     R1,R3                   ;COPY R1
1455   011750  014300          5$:     MOV     -(R3),R0                ;GET RESULTS
1456   011752  004737  011766                  JSR     PC,40$                  ;CONVERT AND TYPE
1457   011756  022703  017550                  CMP     #BUFFER,R3              ;FINISHED ?
1458   011762  001732                          BEQ     1$                      ;BR IF YES
1459   011764  000771                          BR      5$                      ;CONT. TYPING UNTIL DONE
1460                          ;SUBROUTINE FOR THE COULTER TEST
1461   011766  012737  000055  013324  40$:     MOV     #55,MSGRUS+2            ;FIX ASCII MESSAGE
1462   011774  012702  013331                  MOV     #MSPNT1,R2              ;LOAD DEST. POINTER
1463   012000  012737  000004  011556          MOV     #4,EXTTMP               ;LOAD COUNT
1464   012006  000404                          BR      12$
1465   012010  006000          10$:    ROR     R0
1466   012012  006000                  ROR     R0
1467   012014  006000                  ROR     R0
1468   012016  006000                  ROR     R0
1469   012020  010001          12$:    MOV     R0,R1                   ;LOAD R1
1470   012022  042701  177760                  BIC     #177760,R1              ;MASK
1471   012026  052701  000060                  BIS     #60,R1                  ;MAKE DIGIT
1472   012032  110142                          MOVB    R1,-(R2)                ;SAVE DIGIT
1473   012034  005337  011556                  DEC     EXTTMP                  ;FINISHED ?
1474   012040  001363                          BNE     10$                     ;BR IF NOT
1475   012042  000337  013324                  SWAB    MSGRUS+2                ;ADJUST MESSAGE
1476   012046  104401  013322                  TYPE,   MSGRUS
1477   012052  000207                          RTS     PC                      ;EXIT
```

```
                                        .SBTTL  LOC-BOX LAMP AND SWITCH LOOP
012054  012706  001100          LOCBOX: MOV     #STACK,SP
012060  004737  011562                  JSR     PC,SETTRP
012064  104401  013062                  TYPE    ,LOCHDR                 ;TELL OPERATOR WHAT THIS IS
012070  004537  012152                  JSR     R5,INADR                ;GET THE ADDRESSES
012074  001326  012332                  LOC1,LOC1Y
012100  000412                          BR      LOCA                    ;BR IF DONE
012102  004537  012152                  JSR     R5,INADR                ;GET NEXT ADDRESS
012106  001330  012334                  LOC2,LOC2Y
012112  000405                          BR      LOCA                    ;BR IF DONE
012114  004537  012152                  JSR     R5,INADR                ;GET NEXT ADDRESS
012120  001332  012336                  LOC3,LOC3Y
012124  000400                          BR      LOCA

012126  004537  012260          LOCA:   JSR     R5,LOUP1                ;TEST FOR SWITCHES AND LOAD LAMPS
012132  001326                          LOC1
012134  004537  012260                  JSR     R5,LOUP1                ;LOAD NEXT SET
012140  001330                          LOC2
012142  004537  012260                  JSR     R5,LOUP1                ;LOAD NEXT SET
012146  001332                          LOC3

012150  000766                          BR      LOCA                    ;LOOP BACK

012152  013537  012252          INADR:  MOV     @(R5)+,10$              ;GET BUS ADDRESS
012156  013537  012256                  MOV     @(R5)+,12$              ;GET INDICATOR
012162  104401  013123          1$:     TYPE,   INADRH                  ;ASK FOR INPUT
012166  013746  012252                  MOV     10$,-(SP)               ;TYPE CURRENT ADDRESS
012172  104402                          TYPOC
012174  104401  013166                  TYPE,   INDADR                  ;ADD YES NO
012200  104411                          RDLIN                           ;READ CHAR. AND ECHO
012202  000240                          NOP
012204  000240                          NOP
012206  013637  012254                  MOV     @(SP)+,11$              ;GET THE ANSWER
012212  042737  177640  012254        BIC     #177640,11$             ;MASK OFF BITS
012220  001413                          BEQ     3$                      ;BR IF CR
012222  022737  000116  012254        CMP     #'N,11$                 ;TEST IF NO
012230  001406                          BEQ     2$                      ;BR IF NO
012232  022737  000131  012254        CMP     #'Y,11$                 ;TEST IF YES
012240  001350                          BNE     1$                      ;BR IF OTHER
012242  005277  000010                  INC     @12$                    ;SET YES FLAG
012246  005725                  2$:     TST     (R5)+
012250  000205                  3$:     RTS     R5                      ;EXIT
012252  000000                  10$:    0
012254  000000                  11$:    0
012256  000000                  12$:    0
```

```
1525
1526                                    ;SUBROUTINE TO LOAD THE LAMPS FROM THE LOC BOX SWITCHES
1527  012260  013537  012326  LOUP1:   MOV     @(R5)+,40$              ;GET ARG.
1528  012264  001001                   BNE     1$                      ;BR IF YES
1529  012266  000205                   RTS     R5                      ;EXIT
1530  012270  013702  012326  1$:      MOV     40$,R2
1531  012274  062702  000002          ADD     #2,R2
1532  012300  010203                   MOV     R2,R3
1533  012302  062703  000002          ADD     #2,R3
1534  012306  011200                   MOV     (R2),R0
1535  012310  011301                   MOV     (R3),R1
1536  012312  040100                   BIC     R1,R0                   ;MASK OFF BITS
1537  012314  041213                   BIC     (R2),(R3)               ;LOAD OUTPUT BITS
1538  012316  060013                   ADD     R0,(R3)         ;LOAD
1539  012320  012712  177777          MOV     #-1,(R2)                ;CLEAR INPUT
1540  012324  000205                   RTS     R5                      ;EXIT
1541
1542  012326  000000          40$:     0
1543  012330  000000          41$:     0
1544  012332  000000          LOC1Y:   0               ;YES/NO ANSWER TO LOC BOX #1
1545  012334  000000          LOC2Y:   0
1546  012336  000000          LOC3Y:   0
1547
1548                                    .SBTTL  XLO1 ADJUSTMENT ROUTINE
1549  012340  012706  001100  XLO1AD: MOV     #STACK,SP
1550  012344  004737  011562          JSR     PC,SETTRP
1551  012350  104401  013204          TYPE    ,XLOHDR         ;TELL OPERATOR
1552  012354  004537  012152          JSR     R5,INADR                ;GET BUS ADDRESS
1553  012360  001326  012332          LOC1,LOC1Y
1554  012364  000412                   BR      XLO10
1555  012366  004537  012152          JSR     R5,INADR                ;
1556  012372  001330  012334          LOC2,LOC2Y
1557  012376  000405                   BR      XLO10
1558  012400  004537  012152          JSR     R5,INADR
1559  012404  001332  012336          LOC3,LOC3Y
1560  012410  000400                   BR      XLO10
1561
1568  012412                  XLO10:
 (1)  012412  004537  012644          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012416     000     000          .BYTE   0,0                     ;LOAD CH# AND SHIFT COUNT
 (1)  012420  000007                   7                               ;LOAD DATA MASK
 (1)  012422  001326                   LOC1                            ;DR11K BUS ADDRESS
1569  012424  004537  012644          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012430     001     003          .BYTE   1,3                     ;LOAD CH# AND SHIFT COUNT
 (1)  012432  000070                   70                              ;LOAD DATA MASK
 (1)  012434  001326                   LOC1                            ;DR11K BUS ADDRESS
1570  012436  004537  012644          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012442     002     006          .BYTE   2,6                     ;LOAD CH# AND SHIFT COUNT
 (1)  012444  000700                   700                             ;LOAD DATA MASK
 (1)  012446  001326                   LOC1                            ;DR11K BUS ADDRESS
1571  012450  004537  012644          JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012454     003     011          .BYTE   3,11                    ;LOAD CH# AND SHIFT COUNT
 (1)  012456  007000                   7000                            ;LOAD DATA MASK
 (1)  012460  001326                   LOC1                            ;DR11K BUS ADDRESS
```

```
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   4,14            ;LOAD CH# AND SHIFT COUNT
                                    70000                   ;LOAD DATA MASK
                                    LOC1                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   5,0             ;LOAD CH# AND SHIFT COUNT
                                    7                       ;LOAD DATA MASK
                                    LOC2                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   6,3             ;LOAD CH# AND SHIFT COUNT
                                    70                      ;LOAD DATA MASK
                                    LOC2                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   7,6             ;LOAD CH# AND SHIFT COUNT
                                    700                     ;LOAD DATA MASK
                                    LOC2                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   10,11           ;LOAD CH# AND SHIFT COUNT
                                    7000                    ;LOAD DATA MASK
                                    LOC2                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   11,14           ;LOAD CH# AND SHIFT COUNT
                                    70000                   ;LOAD DATA MASK
                                    LOC2                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   12,0            ;LOAD CH# AND SHIFT COUNT
                                    7                       ;LOAD DATA MASK
                                    LOC3                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   13,3            ;LOAD CH# AND SHIFT COUNT
                                    70                      ;LOAD DATA MASK
                                    LOC3                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   14,6            ;LOAD CH# AND SHIFT COUNT
                                    700                     ;LOAD DATA MASK
                                    LOC3                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   15,11           ;LOAD CH# AND SHIFT COUNT
                                    7000                    ;LOAD DATA MASK
                                    LOC3                    ;DR11K BUS ADDRESS
                                    JSR     R5,XLOADJ       ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
                                    .BYTE   16,14           ;LOAD CH# AND SHIFT COUNT
                                    70000                   ;LOAD DATA MASK
                                    LOC3                    ;DR11K BUS ADDRESS
                                    JMP     XL010

                            ;ANALOG CONVERSION AND LOC BOX DRIVER
                            XLOADJ: MOVB    (R5)+,@ADCS1            ;LOAD CHANNEL #
                                    BIS     #BIT13,@ADCS            ;SET UNIPOLAR
                                    MOVB    (R5)+,40$               ;GET SHIFT COUNT
                                    MOV     (R5)+,42$               ;GET DATA MASK
                                    MOV     @(R5)+,41$              ;GET DR11K BUS ADDRESS
                                    BEQ     70$             ;BR IF NONE
                                    INCB    @ADCS                   ;CONVERT THE DATA
```

```
1593  012700  005001                        CLR     R1
1594  012702  062737  000004  013050        ADD     #4,41$          ;GET LAMP ADDRESS
1595  012710  105777  000140          1$:   TSTB    @ADCS                   ;WAIT FOR DONE
1596  012714  100375                        BPL     1$
1597  012716  017700  000136                MOV     @ADBR,R0                ;READ THE DATA
1598  012722  162700  000004                SUB     #4,R0                   ;TEST IF AT UPPER END
1599  012726  100432                        BMI     7$                      ;BR IF YES
1600  012730  162700  000014                SUB     #14,R0                  ;TEST IF NEAR UPPER END
1601  012734  100425                        BMI     6$                      ;BR IF YES
1602  012736  162700  000060                SUB     #60,R0                  ;TEST IF GETTING NEAR UPPER END
1603  012742  100420                        BMI     5$                      ;BR IF YES
1604  012744  162700  001530                SUB     #1530,R0                ;TEST IF GETTING NEAR LOWER END
1605  012750  100407                        BMI     2$                      ;BR IF YES
1606  012752  162700  000060                SUB     #60,R0                  ;TEST IF NEAR LOWER END
1607  012756  100406                        BMI     3$                      ;BR IF YES
1608  012760  162700  000014                SUB     #14,R0                  ;TEST IF AT LOWER END
1609  012764  100405                        BMI     4$                      ;BR IF YES
1610  012766  000412                        BR      7$                      ;BR IF AT STOP
1611  012770  012701  000004          2$:   MOV     #4,R1                   ;LOAD #
1612  012774  062701  000002          3$:   ADD     #2,R1
1613  013000  005201                  4$:   INC     R1
1614  013002  000404                        BR      7$                      ;BR AND LOAD LAMP'S
1615  013004  012701  000002          5$:   MOV     #2,R1                   ;LOAD #
1616  013010  062701  000004          6$:   ADD     #4,R1
1617  013014  013700  013046          7$:   MOV     40$,R0                  ;LOAD SHIFT COUNT
1618  013020  006301                  10$:  ASL     R1                      ;SHUFFEL THE DATA
1619  013022  005300                        DEC     R0
1620  013024  100375                        BPL     10$                     ;BR IF NOT DONE
1621  013026  006201                        ASR     R1                      ;RE ADJUST
1622  013030  043777  013052  000012        BIC     42$,@41$                ;CLEAR OLD DATA IN LAMPS
1623  013036  050177  000006                BIS     R1,@41$                 ;LOAD THE LAMPS
1624  013042  005001                        CLR     R1
1625  013044  000205                  70$:  RTS     R5                      ;EXIT
1626
1627  013046  000000                  40$:  0
1628  013050  000000                  41$:  0
1629  013052  000000                  42$:  0
```

```
                                                ADCS:   170400
                                                ADCS1:  170401
                                                ADBR:   170402
1634  013062  005015  047514  026503  LOCHDR: .ASCIZ  <15><12>/LOC-BOX SWITCH AND LAMP LOOP/<15><12>
      013070  047502  020130  053523
      013076  052111  044103  040440
      013104  042116  046040  046501
      013112  020120  047514  050117
      013120  005015     000
1635  013123     015  052412  042523  INADRH: .ASCIZ  <15><12>/USE LOC-BOX <DR11K AT BUS ADDR. /
      013130  046040  041517  041055
      013136  054117  036040  051104
      013144  030461  020113  052101
      013152  041040  051525  040440
      013160  042104  027122  000040
1636  013166  037040  020040  020131  INDADR: .ASCIZ  / >  Y OR N ? /
      013174  051117  047040  037440
      013202  000040
1637  013204  005015  046130  030460  XLOHDR: .ASCIZ  <15><12>/XLO1 POT ADJUSTMENT LOOP/<15><12>
      013212  050040  052117  040440
      013220  045104  051525  046524
      013226  047105  020124  047514
      013234  050117  005015     000
1638  013241     015  041012  051525  BUSTRP: .ASCIZ  <15><12>/BUS TIME-OUT ON SELECTED DR11K/
      013246  052040  046511  026505
      013254  052517  020124  047117
      013262  051440  046105  041505
      013270  042524  020104  051104
      013276  030461  000113
1639  013302  005015  051012  047125  RUNMSG: .ASCIZ  <15><12><12>/RUN SAMPLE/<15><12>
      013310  051440  046501  046120
      013316  006505  000012
1640  013322  005015  047055  047116  MSGRUS: .ASCII  <15><12>/-NNNN/
      013330     116
1641  013331     000                  MSPNT1: .BYTE   0
1642                                         .EVEN
1643
1644
1645  013332  042523  020124  053523  SWNLB:  .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING INPUT BITS/
      013340  052111  044103  051040
      013346  043505  051511  042524
      013354  020122  044502  051524
      013362  042440  052521  046101
      013370  052040  020117  044124
      013376  020105  047516  026516
      013404  040514  041524  044510
      013412  043516  044440  050116
      013420  052125  041040  052111
      013426  000123
1646  013430  042523  020124  053523  SWINTB: .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING INPUT BITS/
      013436  052111  044103  051040
      013444  043505  051511  042524
      013452  020122  044502  051524
      013460  042440  052521  046101
```

```
                013466  052040  020117  044124
                013474  020105  047111  042524
                013502  051122  050125  044524
                013510  043516  044440  050116
                013516  052125  041040  052111
                013524  000123
1647            013526  042523  020124  053523  SWPOSB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO POSITIVE INPUT BITS/
                013534  052111  044103  051040
                013542  043505  051511  042524
                013550  020122  044502  051524
                013556  030440  026465  031061
                013564  042440  052521  046101
                013572  052040  020117  047520
                013600  044523  044524  042526
                013606  044440  050116  052125
                013614  041040  052111  000123
1648            013622  042523  020124  053523  SWTRAB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO TRANSITION INPUT BITS/
                013630  052111  044103  051040
                013636  043505  051511  042524
                013644  020122  044502  051524
                013652  030440  026465  031061
                013660  042440  052521  046101
                013666  052040  020117  051124
                013674  047101  044523  044524
                013702  047117  044440  050116
                013710  052125  041040  052111
                013716  000123
1649            013720  042523  020124  053523  SWDPOB: .ASCIZ  /SET SWITCH REGISTER WITH THE DESIRED PROGRAM OPTIONS/
                013726  052111  044103  051040
                013734  043505  051511  042524
                013742  020122  044527  044124
                013750  052040  042510  042040
                013756  051505  051111  042105
                013764  050040  047522  051107
                013772  046501  047440  052120
                014000  047511  051516     000
1650            014005     015  042012  050105  DEPCNT: .ASCIZ  <15><12>/DEPRESS CONT./<15><12>
                014012  042522  051523  041440
                014020  047117  027124  005015
                014026     000
1651            014027     123  040524  052524  EM1:    .ASCIZ  /STATUS REGISTER IN ERROR/
                014034  020123  042522  044507
                014042  052123  051105  044440
                014050  020116  051105  047522
                014056  000122
1652            014060  047111  052520  020124  EM2:    .ASCIZ  /INPUT REGISTER IN ERROR/
                014066  042522  044507  052123
                014074  051105  044440  020116
                014102  051105  047522  000122
1653            014110  052517  050124  052125  EM3:    .ASCIZ  /OUTPUT REGISTER IN ERROR/
                014116  051040  043505  051511
                014124  042524  020122  047111
                014132  042440  051122  051117
                014140     000
```

```
1654  014141     111  050116  052125  EM4:    .ASCIZ  /INPUT FAILED TO INTERRUPT/
      014146  043040  044501  042514
      014154  020104  047524  044440
      014162  052116  051105  052522
      014170  052120     000
1655  014173     117  052125  052520  EM5:    .ASCIZ  /OUTPUT FAILED TO INTERRUPT/
      014200  020124  040506  046111
      014206  042105  052040  020117
      014214  047111  042524  051122
      014222  050125  000124
1656  014226  047125  054105  042520  EM6:    .ASCIZ  /UNEXPECTED INTERRUPT/
      014234  052103  042105  044440
      014242  052116  051105  052522
      014250  052120     000
1657  014253     117  042520  040522  EM7:    .ASCIZ  /OPERATOR INTERVENTION ERROR/
      014260  047524  020122  047111
      014266  042524  053122  047105
      014274  044524  047117  042440
      014302  051122  051117     000
1658  014307     111  052116  051105  EM10:   .ASCIZ  /INTERRUPT INPUT BIT FAILED TO SET INPUT READY FLAG/
      014314  052522  052120  044440
      014322  050116  052125  041040
      014330  052111  043040  044501
      014336  042514  020104  047524
      014344  051440  052105  044440
      014352  050116  052125  051040
      014360  040505  054504  043040
      014366  040514  000107
1659  014372  047516  026516  047111  EM11:   .ASCIZ  /NON-INTERRUPT INPUT BIT SET INPUT READY FLAG/
      014400  042524  051122  050125
      014406  020124  047111  052520
      014414  020124  044502  020124
      014422  042523  020124  047111
      014430  052520  020124  042522
      014436  042101  020131  046106
      014444  043501     000
1660
1661  014447     105  051122  041520  DH1:    .ASCIZ  /ERRPC   DRADD  TSTNUM   STATUS  EXPECTED/
      014454  020040  042040  040522
      014462  042104  052011  052123
      014470  052516  020115  020040
      014476  052123  052101  051525
      014504  020040  054105  042520
      014512  052103  042105     000
1662  014517     105  051122  041520  DH2:    .ASCIZ  /ERRPC   DRADD  TSTNUM  INPUT   EXPECTED/
      014524  020040  042040  040522
      014532  042104  052011  052123
      014540  052516  004515  047111
      014546  052520  020124  020040
      014554  054105  042520  052103
      014562  042105     000
1663  014565     105  051122  041520  DH3:    .ASCIZ  /ERRPC   DRADD  TSTNUM  OUTPUT  EXPECTED/
      014572  020040  042040  040522
      014600  042104  052011  052123
```

```
        014606  052516  004515  052517
        014614  050124  052125  020040
        014622  054105  042520  052103
        014630  042105     000
 1664   014633     105  051122  041520  DH4:    .ASCIZ  /ERRPC   DRADD  TSTNUM/
        014640  020040  042040  040522
        014646  042104  052011  052123
        014654  052516  000115
 1665   014660  051105  050122  020103  DH10:   .ASCIZ  /ERRPC   DRADD  TSTNUM  STATUS  EXPECT  INPUT BIT/
        014666  020040  051104  042101
        014674  004504  051524  047124
        014702  046525  051411  040524
        014710  052524  020123  042440
        014716  050130  041505  020124
        014724  044440  050116  052125
        014732  041040  052111     000
 1666           014740                  .EVEN
 1667   014740  001116  001322  015020  DT1:    $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,0
        014746  001126  001124  000000
 1668   014754  001116  001322  015020  DT4:    $ERRPC,DRADD,TSTNUM,0
        014762  000000
 1669   014764  001116  001322  015020  DT10:   $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,BRLEV3,0
        014772  001126  001124  001366
        015000  000000
 1670   015002  000000  000000  000000  DF0:    0,0,0,0,0,0,0
        015010  000000  000000  000000
        015016  000000
 1671   015020  000000                  TSTNUM: 0
```

```
                                .SBTTL  SCOPE HANDLER ROUTINE

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

(1)  015022                     $SCOPE:
(1)  015022  104407                     CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
(1)  015024  032777  040000  164106 1$:  BIT     #BIT14,@SWR    ;;LOOP ON PRESENT TEST?
(1)  015032  001111                     BNE     $OVER           ;;YES IF SW14=1
(1)                             ;*****START OF CODE FOR THE XOR TESTER*****
(1)  015034  000416             $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
(1)                                                             ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
(1)  015036  013746  000004             MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  015042  012737  015062  000004     MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
(1)  015050  005737  177060             TST     @#177060        ;;TIME OUT ON XOR?
(1)  015054  012637  000004             MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  015060  000463                     BR      $SVLAD          ;;GO TO THE NEXT TEST
(1)  015062  022626             5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
(1)  015064  012637  000004             MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  015070  000423                     BR      7$              ;;LOOP ON THE PRESENT TEST
(1)  015072                     6$:;*****END OF CODE FOR THE XOR TESTER*****
(1)  015072  032777  000400  164040     BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
(1)  015100  001404                     BEQ     2$              ;;BR IF NO
(1)  015102  127737  164032  001102     CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
(1)  015110  001462                     BEQ     $OVER           ;;BR IF YES
(1)  015112  105737  001103     2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
(1)  015116  001421                     BEQ     3$              ;;BR IF NO
(1)  015120  123737  001115  001103     CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  015126  101015                     BHI     3$              ;;BR IF NO
(1)  015130  032777  001000  164002     BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
(1)  015136  001404                     BEQ     4$              ;;BR IF NO
(1)  015140  013737  001110  001106 7$: MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  015146  000443                     BR      $OVER
(1)  015150  105037  001103     4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
(1)  015154  005037  001166             CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  015160  000415                     BR      1$              ;;ESCAPE TO THE NEXT TEST
(1)  015162  032777  004000  163750 3$: BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
(1)  015170  001011                     BNE     1$              ;;BR IF YES
(1)  015172  005737  001100             TST     $PASS           ;;IF FIRST PASS OF PROGRAM
(1)  015176  001406                     BEQ     1$              ;;      INHIBIT ITERATIONS
(1)  015200  005237  001104             INC     $ICNT           ;;INCREMENT ITERATION COUNT
(1)  015204  023737  001166  001104     CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)  015212  002021                     BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
(1)  015214  012737  000001  001104 1$: MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
```

```
(1)  015222  013737  015272  001166          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
(1)  015230  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
(1)  015234  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1)  015240  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  015244  005037  001170                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  015250  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  015256  013777  001102  163656  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  015264  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(1)  015270  000002                          RTI                     ;;FIXES PS
(1)  015272  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
1675                                 .SBTTL  ERROR HANDLER ROUTINE
(1)
(2)                                  ;;**************************************************************
(1)                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)                                  ;*AND GO TO $ERRTYP ON ERROR
(1)                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                  ;*SW15=1        HALT ON ERROR
(1)                                  ;*SW13=1        INHIBIT ERROR TYPEOUTS
(1)                                  ;*SW09=1        LOOP ON ERROR
(1)                                  ;*CALL
(1)                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1)  015274                          $ERROR:
(1)  015274  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(2)  015276  113737  001102  015020          MOVB    $TSTNM,TSTNUM
(1)  015304  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
(1)  015310  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
(1)  015312  013777  001102  163622          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  015320  005237  001112                  INC     $ERTTL          ;;INC THE ERROR COUNT
(1)  015324  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  015330  162737  000002  001116          SUB     #2,$ERRPC
(1)  015336  117737  163554  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  015344  032777  020000  163566          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
(1)  015352  001004                          BNE     20$             ;;SKIP TYPEOUTS
(1)  015354  004737  015440                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(1)  015360  104401  001173                  TYPE    ,$CRLF
(1)  015364                          20$:
(1)  015364  005777  163550          2$:     TST     @SWR            ;;HALT ON ERROR
(1)  015370  100002                          BPL     3$              ;;SKIP IF CONTINUE
(1)  015372  000000                          HALT                    ;;HALT ON ERROR!
(1)  015374  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(1)  015376  032777  001000  163534  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
(1)  015404  001402                          BEQ     4$              ;;BR IF NO
(1)  015406  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
(1)  015412  005737  001170          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
(1)  015416  001402                          BEQ     5$              ;;BR IF NONE
(1)  015420  013716  001170                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  015424                          5$:
(1)  015424  022737  011174  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
(1)  015432  001001                          BNE     6$              ;;BRANCH IF NO
(1)  015434  000000                          HALT                    ;;YES
(1)  015436                          6$:
(1)  015436  000002                          RTI                     ;;RETURN
```

```
 1677
 1678                                   .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
  (1)                                   ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
  (1)                                   ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
  (1)
  (1)  015440                           $ERRTYP:
  (1)  015440  104401  001173                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  015444  010046                           MOV     R0,-(SP)        ;;SAVE R0
  (1)  015446  005000                           CLR     R0              ;;PICKUP THE ITEM INDEX
  (1)  015450  153700  001114                   BISB    @#$ITEMB,R0
  (1)  015454  001004                           BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
  (1)                                                                   ;;TYPE THE PC OF THE ERROR
  (2)  015456  013746  001116                   MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
  (2)                                                                   ;;ERROR ADDRESS
  (2)  015462  104402                           TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  (1)  015464  000426                           BR      6$              ;;GET OUT
  (1)  015466  005300                   1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
  (1)  015470  006300                           ASL     R0              ;;      WORK FOR THE ERROR TABLE
  (1)  015472  006300                           ASL     R0
  (1)  015474  006300                           ASL     R0
  (1)  015476  062700  001176                   ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
  (1)  015502  012037  015512                   MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
  (1)  015506  001404                           BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
  (1)  015510  104401                           TYPE                    ;;TYPE THE "ERROR MESSAGE"
  (1)  015512  000000                   2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
  (1)  015514  104401  001173                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  015520  012037  015530           3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
  (1)  015524  001404                           BEQ     5$              ;;SKIP TYPEOUT IF 0
  (1)  015526  104401                           TYPE                    ;;TYPE THE "DATA HEADER"
  (1)  015530  000000                   4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
  (1)  015532  104401  001173                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  015536  011000                   5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
  (1)  015540  001004                           BNE     7$              ;;GO TYPE THE DATA
  (1)  015542  012600                   6$:     MOV     (SP)+,R0        ;;RESTORE R0
  (1)  015544  104401  001173                   TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
  (1)  015550  000207                           RTS     PC              ;;RETURN
  (2)  015552                           7$:
  (2)  015552  013046                           MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
  (2)  015554  104402                           TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  (1)  015556  005710                           TST     (R0)            ;;IS THERE ANOTHER NUMBER?
  (1)  015560  001770                           BEQ     6$              ;;BR IF NO
  (1)  015562  104401  015570                   TYPE    ,8$             ;;TYPE TWO(2) SPACES
  (1)  015566  000771                           BR      7$              ;;LOOP
  (1)  015570  020040      000          8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
  (1)           015574                         .EVEN
```

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

                        ;;**********************************************************************
                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                        ;*      .BYTE   M               ;;M=1 OR 0
                        ;*                                      ;;1=TYPE LEADING ZEROS
                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                        ;*
                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                        ;*$TYPOS OR $TYPOC
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                        ;*
                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                        ;*CALL:
                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)   015574  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)   015600  116637  000001  016017          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)   015606  112637  016021                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)   015612  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)   015616  000406                          BR      $TYPON
(1)   015620  112737  000001  016017  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)   015626  112737  000006  016021          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)   015634  112737  000005  016016  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)   015642  010346                          MOV     R3,-(SP)        ;;SAVE R3
(1)   015644  010446                          MOV     R4,-(SP)        ;;SAVE R4
(1)   015646  010546                          MOV     R5,-(SP)        ;;SAVE R5
(1)   015650  113704  016021                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)   015654  005404                          NEG     R4
(1)   015656  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)   015662  110437  016020                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)   015666  113704  016017                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)   015672  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)   015676  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)   015700  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)   015702  000404                          BR      3$              ;;GO DO MSB
(1)   015704  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
(1)   015706  006105                          ROL     R5
(1)   015710  006105                          ROL     R5
(1)   015712  010503                          MOV     R5,R3
(1)   015714  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)   015716  105337  016020                  DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)   015722  100016                          BPL     7$              ;;BR IF NO
(1)   015724  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
```

```
(1)  015730  001002                        BNE     4$              ;;TEST FOR 0
(1)  015732  005704                        TST     R4              ;;SUPPRESS THIS 0?
(1)  015734  001403                        BEQ     5$              ;;BR IF YES
(1)  015736  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  015740  052703  000060               BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  015744  052703  000040        5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  015750  110337  016014               MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  015754  104401  016014               TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  015760  105337  016016        7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  015764  003347                        BGT     2$              ;;BR IF MORE TO DO
(1)  015766  002402                        BLT     6$              ;;BR IF DONE
(1)  015770  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  015772  000744                        BR      2$              ;;GO DO THE LAST DIGIT
(1)  015774  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  015776  012604                        MOV     (SP)+,R4        ;;RESTORE R4
(1)  016000  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  016002  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  016010  012616                        MOV     (SP)+,(SP)
(1)  016012  000002                        RTI                     ;;RETURN
(1)  016014      000               8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  016015      000                       .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  016016      000               $OCNT   .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  016017      000               $OFIL:  .BYTE   0               ;;ZERO FILL SWITCH
(1)  016020  000000                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
 1683
 1684                                   .SBTTL  POWER DOWN AND UP ROUTINES
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;POWER DOWN ROUTINE
  (1)  016022  012737  016166  000024   $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
  (1)  016030  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (3)  016036  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (3)  016040  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (3)  016042  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (3)  016044  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (3)  016046  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
  (3)  016050  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
  (3)  016052  017746  163062                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
  (1)  016056  010637  016172                  MOV     SP,$SAVR6       ;;SAVE SP
  (1)  016062  012737  016074  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
  (1)  016070  000000                          HALT
  (1)  016072  000776                          BR      .-2             ;;HANG UP
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;POWER UP ROUTINE
  (1)  016074  012737  016166  000024   $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
  (1)  016102  013706  016172                  MOV     $SAVR6,SP       ;;GET SP
  (1)  016106  005037  016172                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
  (1)  016112  005237  016172           1$:    INC     $SAVR6          ;;WAIT FOR THE INC
  (1)  016116  001375                          BNE     1$              ;;OF  WORD
  (3)  016120  012677  163014                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
  (3)  016124  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
  (3)  016126  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
  (3)  016130  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
  (3)  016132  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
  (3)  016134  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
  (3)  016136  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  (1)  016140  012737  016022  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
  (1)  016146  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (1)  016154  104401                          TYPE                    ;;REPORT THE POWER FAILURE
  (1)  016156  016174           $PWRMG: .WORD   PWRMSG                  ;;POWER FAIL MESSAGE POINTER
  (1)  016160  012716                          MOV     (PC)+,(SP)      ;;RESTART AT IOTEST
  (1)  016162  002752           $PWRAD: .WORD   IOTEST                  ;;RESTART ADDRESS
  (1)  016164  000002                          RTI
  (1)  016166  000000           $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
  (1)  016170  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
  (1)  016172  000000           $SAVR6: 0                               ;;PUT THE SP HERE
 1685  016174  005015  042522  052123 PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
       016202  051101  044524  043516
       016210  040440  052106  051105
       016216  040440  050040  053517
       016224  051105  043040  044501
       016232  052514  042522  005015
       016240     000
 1686          016242                          .EVEN
```

```
                                .SBTTL  TYPE ROUTINE
                                ;;*****************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                ;*OR
                                ;*      TYPE
                                ;*      MESADR
                                ;*
016242  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
016246  100002                         BPL     1$              ;;BR IF YES
016250  000000                         HALT                    ;;HALT HERE IF NO TERMINAL
016252  000407                         BR      3$              ;;LEAVE
016254  010046                  1$:    MOV     R0,-(SP)        ;;SAVE R0
016256  017600  000002                 MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
016262  112046                  2$:    MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
016264  001005                         BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
016266  005726                         TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
016270  012600                  60$:   MOV     (SP)+,R0        ;;RESTORE R0
016272  062716  000002          3$:    ADD     #2,(SP)         ;;ADJUST RETURN PC
016276  000002                         RTI                     ;;RETURN
016300  122716  000011          4$:    CMPB    #HT,(SP)        ;;BRANCH IF <HT>
016304  001430                         BEQ     8$
016306  122716  000200                 CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
016312  001006                         BNE     5$
016314  005726                         TST     (SP)+           ;;POP  <CR><LF> EQUIV
016316  104401                         TYPE                    ;;TYPE A CR AND LF
016320  001173                         $CRLF
016322  105037  016456                 CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
016326  000755                         BR      2$              ;;GET NEXT CHARACTER
016330  004737  016412          5$:    JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
016334  123726  001156          6$:    CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
016340  001350                         BNE     2$              ;;IF NO GO GET NEXT CHAR.
016342  013746  001154                 MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                               ;;AND THE NULL CHAR.
016346  105366  000001          7$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
016352  002770                         BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
016354  004737  016412                 JSR     PC,$TYPEC       ;;GO TYPE A NULL
016360  105337  016456                 DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
016364  000770                         BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

016366  112716  000040          8$:    MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
016372  004737  016412          9$:    JSR     PC,$TYPEC       ;;TYPE A SPACE
```

```
(1)  016376  132737  000007  016456          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  016404  001372                          BNE     9$              ;;TAB STOP
(1)  016406  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
(1)  016410  000724                          BR      2$              ;;GET NEXT CHARACTER
(1)  016412  105777  162532          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  016416  100375                          BPL     $TYPEC
(1)  016420  116677  000002  162524          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  016426  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  016434  001003                          BNE     1$              ;;BRANCH IF NO
(1)  016436  105037  016456                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  016442  000406                          BR      $TYPEX          ;;EXIT
(1)  016444  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  016452  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
(1)  016454  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  016456  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  016460  000207          $TYPEX: RTS     PC
(1)
```

```
                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

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

016462  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
016464  016666  000004  000002  MOV     4(SP),2(SP)     ;;INPUT NUMBER
016472  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
016474  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016476  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016500  104411          1$:     RDLIN                   ;;READ AN ASCIZ LINE
016502  012600                  MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
016504  005001                  CLR     R1              ;;CLEAR DATA WORD
016506  005002                  CLR     R2
016510  112046          2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
016512  001412                  BEQ     3$              ;;IF ZERO GET OUT
016514  006301                  ASL     R1              ;;*2
016516  006102                  ROL     R2
016520  006301                  ASL     R1              ;;*4
016522  006102                  ROL     R2
016524  006301                  ASL     R1              ;;*8
016526  006102                  ROL     R2
016530  042716  177770          BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
016534  062601                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
016536  000764                  BR      2$              ;;LOOP
016540  005726          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
016542  010166  000012          MOV     R1,12(SP)       ;;SAVE THE RESULT
016546  010237  016562          MOV     R2,$HIOCT
016552  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
016554  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
016556  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
016560  000002                  RTI                     ;;RETURN
016562  000000          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
                                .SBTTL  TTY INPUT ROUTINE

                                ;;**********************************************************************
                                .ENABL  LSB

                                ;;**********************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*WHEN OPERATING IN TTY FLAG MODE.
016564  022737  000176  001140  $CKSWR: CMP     #$SWREG,SWR     ;;IS THE SOFT-SWR SELECTED?
016572  001074                  BNE     15$             ;;BRANCH IF NO
016574  105777  162344          TSTB    @$TKS           ;;CHAR THERE?
016600  100071                  BPL     15$             ;;IF NO, DON'T WAIT AROUND
016602  117746  162340          MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
016606  042716  177600          BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
```

```
(1)  016612  022726  000007            CMP     #7,(SP)+        ;;IS IT A CONTROL G?
(1)  016616  001062                    BNE     15$             ;;NO  RETURN TO USER
(1)  016620  123727  001134  000001    CMPB    $AUTOB,#1       ;;AR WE RUNNING IN AUTO-MODE?
(1)  016626  001456                    BEQ     15$             ;;BRANCH IF YES
(1)
(1)  016630  104401  017436            TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (↑G)
(1)  016634  104401  017443  $GTSWR:  TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  016640  013746  000176            MOV     $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  016644  104402                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016646  104401  017454            TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  016652  005046          19$:      CLR     -(SP)           ;;CLEAR COUNTER
(1)  016654  005046                    CLR     -(SP)           ;;THE NEW SWR
(1)  016656  105777  162262  7$:       TSTB    @$TKS           ;;CHAR THERE?
(1)  016662  100375                    BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  016664  117746  162256            MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  016670  042716  177600            BIC     #↑C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1)  016674  021627  000025  9$:       CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  016700  001005                    BNE     10$             ;;BRANCH IF NOT
(1)  016702  104401  017431            TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (↑U)
(1)  016706  062706  000006  20$:      ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  016712  000757                    BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  016714  021627  000015  10$:      CMP     (SP),#15        ;;IS IT A <CR>?
(1)  016720  001022                    BNE     16$             ;;BRANCH IF NO
(1)  016722  005766  000004            TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  016726  001403                    BEQ     11$             ;;BRANCH IF YES
(1)  016730  016677  000002  162202    MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  016736  062706  000006  11$:      ADD     #6,SP           ;;CLEAR UP STACK
(1)  016742  104401  001173  14$:      TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  016746  123727  001135  000001    CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  016754  001003                    BNE     15$             ;;BRANCH IF NOT
(1)  016756  012777  000100  162160    MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  016764  000002          15$:      RTI                     ;;RETURN
(1)  016766  004737  016412  16$:      JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  016772  021627  000060            CMP     (SP),#60        ;;CHAR < 0?
(1)  016776  002420                    BLT     18$             ;;BRANCH IF YES
(1)  017000  021627  000067            CMP     (SP),#67        ;;CHAR > 7?
(1)  017004  003015                    BGT     18$             ;;BRANCH IF YES
(1)  017006  042726  000060            BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  017012  005766  000002            TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  017016  001403                    BEQ     17$             ;;BRANCH IF YES
(1)  017020  006316                    ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  017022  006316                    ASL     (SP)            ;;   CHAR OVER TO MAKE
(1)  017024  006316                    ASL     (SP)            ;;   ROOM FOR NEW ONE.
(1)  017026  005266  000002  17$:      INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  017032  056616  177776            BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  017036  000707                    BR      7$              ;;GET THE NEXT ONE
(1)  017040  104401  001172  18$:      TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  017044  000720                    BR      20$             ;;SIMULATE CONTROL-U
```

```
(1)                                     .DSABL  LSB
(1)
(1)
(2)                                     ;;*****************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                     ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                     ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                     ;
(1)
(1)  017046  011646                     $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  017050  016666  000004  000002             MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  017056  105777  162062             1$:     TSTB    @$TKS           ;;WAIT FOR
(1)  017062  100375                             BPL     1$              ;;A CHARACTER
(1)  017064  117766  162056  000004             MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  017072  042766  177600  000004             BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  017100  026627  000004  000023             CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  017106  001013                             BNE     3$              ;;BRANCH IF NO
(1)  017110  105777  162030             2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  017114  100375                             BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  017116  117746  162024                     MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  017122  042716  177600                     BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  017126  022627  000021                     CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  017132  001366                             BNE     2$              ;;IF NOT DISCARD IT
(1)  017134  000750                             BR      1$              ;;YES, RESUME
(1)  017136  026627  000004  000140     3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  017144  002407                             BLT     4$              ;;BRANCH IF YES
(1)  017146  026627  000004  000175             CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  017154  003003                             BGT     4$              ;;BRANCH IF YES
(1)  017156  042766  000040  000004             BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
(1)  017164  000002                     4$:     RTI                     ;;GO BACK TO USER
(2)                                     ;;*****************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                     ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  017166  010346                     $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  017170  005046                             CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  017172  012703  017422             1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  017176  022703  017431             2$:     CMP     #$TTYIN+7,R3    ;;BUFFER FULL?
(1)  017202  101456                             BLOS    4$              ;;BR IF YES
(1)  017204  104410                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  017206  112613                             MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  017210  122713  000177             10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  017214  001022                             BNE     5$              ;;BR IF NO
(1)  017216  005716                             TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  017220  001007                             BNE     6$              ;;BR IF NO
(1)  017222  112737  000134  017420             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  017230  104401  017420                     TYPE    ,9$
(1)  017234  012716  177777                     MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  017240  005303                     6$:     DEC     R3              ;;BACKUP BY ONE
```

```
(1)  017242  020327  017422                CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  017246  103434                        BLO     4$              ;;BR IF YES
(1)  017250  111337  017420                MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  017254  104401  017420                TYPE    ,9$             ;;GO TYPE
(1)  017260  000746                        BR      2$              ;;GO READ ANOTHER CHAR.
(1)  017262  005716                5$:     TST     (SP)            ;;RUBOUT KEY SET?
(1)  017264  001406                        BEQ     7$              ;;BR IF NO
(1)  017266  112737  000134  017420        MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  017274  104401  017420                TYPE    ,9$
(1)  017300  005016                        CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  017302  122713  000025        7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  017306  001003                        BNE     8$              ;;BR IF NO
(1)  017310  104401  017431                TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
(1)  017314  000726                        BR      1$              ;;GO START OVER
(1)  017316  122713  000022        8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "↑R"?
(1)  017322  001011                        BNE     3$              ;;BRANCH IF NO
(1)  017324  105013                        CLRB    (R3)            ;;CLEAR THE CHARACTER
(1)  017326  104401  001173                TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
(1)  017332  104401  017422                TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  017336  000717                        BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  017340  104401  001172        4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  017344  000712                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  017346  111337  017420        3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  017352  104401  017420                TYPE    ,9$
(1)  017356  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  017362  001305                        BNE     2$              ;;LOOP IF NOT RETURN
(1)  017364  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  017370  104401  001174                TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  017374  005726                        TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  017376  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  017400  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  017402  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  017410  012766  017422  000004        MOV     #$TTYIN,4(SP)
(1)  017416  000002                        RTI                     ;;RETURN
(1)  017420     000                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  017421     000                        .BYTE   0               ;;TERMINATOR
(1)  017422  000007                $TTYIN: .BLKB   7               ;;RESERVE 7 BYTES FOR TTY INPUT
(1)  017431     136  006525  000012 $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
(1)  017436  043536  005015     000 $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
(1)  017443     015  051412  051127 $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  017450  036440  000040
(1)  017454  020040  042516  020127 $MNEW:  .ASCIZ  /  NEW = /
(1)  017462  020075     000
(1)          017466                .EVEN
1693
```

```
                                        .SBTTL  TRAP DECODER

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

017466  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
017470  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
017474  005740                          TST     -(R0)           ;;BACKUP BY 2
017476  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
017500  006300                          ASL     R0              ;;POSITION FOR INDEXING
017502  016000  017522                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
017506  000200                          RTS     R0              ;;GO TO ROUTINE

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

017510  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
017512  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
017520  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
017522  017510                  $TRPAD: .WORD   $TRAP2
                                        $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
                                        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
                                        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
                                        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
                                        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

                                        $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

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

017550  000000                  BUFFER: 0                       ;10 WORD BUFFER FOR THE COULTER INTERFACE TEST
        000001                  .END
```

```
ADBR    013060          1597    1633#
ADCS    013054          1587*   1592*   1595    1631#
ADCS1   013056          1586*   1632#
BASEBA  001306           460#    520     649    1393
BASEBR  001312           462#    679
BASEIV  001310           461#    650    1394
BEGIN   001400           394     504#    589
BEGIN1  001434           395     513#
BIT0  = 000001           390#    734     747     945     979    1001    1241    1278
BIT00 = 000001           390#
BIT01 = 000002           390#
BIT02 = 000004           390#
BIT03 = 000010           390#
BIT04 = 000020           390#
BIT05 = 000040           390#
BIT06 = 000100           390#
BIT07 = 000200           390#
BIT08 = 000400           390#   1674
BIT09 = 001000           390#   1674    1675
BIT1  = 000002           390#
BIT10 = 002000           390#    861    1210    1237    1274    1305
BIT11 = 004000           390#   1674
BIT12 = 010000           390#    582     915
BIT13 = 020000           390#   1188    1587    1675
BIT14 = 040000           390#    818     819     846    1313    1320    1406    1674
BIT15 = 100000           390#    809     810     846    1099
BIT2  = 000004           390#
BIT3  = 000010           390#
BIT4  = 000020           390#
BIT5  = 000040           390#   1137    1165    1348    1365
BIT6  = 000100           390#    835     836     846    1164    1175    1218    1225    1438
BIT7  = 000200           390#    826     827     846    1087    1243
BIT8  = 000400           390#
BIT9  = 001000           390#
BPTVEC= 000014           390#
BRLEV1  001362           486#   1337*   1338*   1344
BRLEV2  001364           487#    945*    948     968     973*   1339*   1361
BRLEV3  001366           488#    596     946*    947*    956     966    1001*   1005    1012    1021    1022*   1241*   1245
                        1250    1251    1252    1262    1263   1270*   1278*   1282    1287    1288    1289    1299*   1669
BUFFER  017550          1437    1457    1698#
BUSTRP  013241           531    1638#
BYPASS  011010          1385#
BYPAS1  011076          1388    1392    1397#   1400
CKSWR = 104407          1674    1675    1696#
COULTJ  001410           398     507#
COULTR  011614           397    1430#
CPU     001314           463#   1444
CR    = 000015           390#   1689
CRLF  = 000200           390#   1689
DDISP = 177570           390#    402     515
DEPCNT  014005           638    1650#
DFO     015002           408     414     420     426     432     438     446     452     458    1670#
DH1     014447           406    1661#
DH10    014660           450     456    1665#
```

```
DH2       014517             412    1662#
DH3       014565             418    1663#
DH4       014633             424     430     436     444    1664#
DIOBRL    001360             485#    679*   1331*   1334    1337    1339
DISPLA    001142             402#    515*   1674*   1675*
DISPRE    000174             394#    515
DRADD     001322             468#    649*    664     665     667     669    1389*   1393*   1667    1668    1669
DRIV      001324             469#    621*    650*    671     672     674     676    1390*   1394*
DRT21     007050             864    1109#
DSWR   =  177570             390#    402     515
DT1       014740             407     413     419    1667#
DT10      014764             451     457    1669#
DT4       014754             425     431     437     445    1668#
FMTVEC=   000030             390#    515*
EM1       014027             405    1651#
EM10      014307             449    1658#
EM11      014372             455    1659#
EM2       014060             411    1652#
EM3       014110             417    1653#
EM4       014141             423    1654#
EM5       014173             429    1655#
EM6       014226             435    1656#
EM7       014253             443    1657#
ERRVEC=   000004             390#    515*    519*    540*    541*   1674*
EXTCNT    011560            1406*   1415*   1420#
EXTTMP    011556            1409*   1410    1413*   1414    1419#   1443*   1447*   1463*   1473*
EXTTST    011454             396    1404#
GNS    =  ****** U           394     548     549     553     612     615    1696
GRBHIO    001342             478#    669*    670*    798*
GRDAI     001336             476#    665*    666*    686*    808*    866*    869     877*    881     890*    894     902*    906
                             921*    923     931*    933     955     965     981*    990     995*   1003*   1016    1021*   1032*
                            1040*   1057*   1065*   1077*   1081    1103    1118    1220    1248*   1263*   1285*   1315    1409
                            1410*   1413    1414*   1449    1451*
GRDIO     001340             477#    667*    668*    687*    691*    692     699*    701     707*    708     715*    716     724*
                             725     736*    737*    738     749*    750*    751     762*    765*    766     773*    776*    777
                             785*    786*    787     797*    799     807*    865*    868*    876*    880*    889*    893*    901*
                             905*    920*    922*    932*    954*    964*    980*    988*    989*   1002*   1013*   1014*   1031*
                            1040*   1056*   1065*   1076*   1078*   1089*   1101*   1117*   1219*   1247*   1250*   1251*   1252*
                            1262*   1284*   1287*   1288*   1289*   1314*   1407*   1408*   1411*   1412*
GRIVA     001350             481#    671*    683*   1113*   1133*   1147*   1160*   1170*   1184*   1214*   1233*   1311*   1325*
                            1343*   1356*   1360*   1370*   1381*   1434*
GRIVB     001354             483#    674*    675*    683*   1115*   1135*   1162*   1186*   1196*   1216*   1233*   1309*   1325*
                            1356*   1381*
GRIVSA    001352             482#    672*    673*    683*   1114*   1134*   1161*   1185*   1215*   1233*   1311    1312*   1325*
                            1356*   1381*   1435*
GRIVSB    001356             484#    676*    677*    683*   1116*   1136*   1163*   1187*   1217*   1233*   1310*   1325*   1356*
                            1381*
GRSTAT    001334             475#    664*    685*    810*    811     817*    819*    820     827*    828     834*    836*    837
                             844*    846*    848    1088*   1091    1100*   1104    1111*   1123*   1132*   1137*   1159*   1164*
                            1165*   1174    1183*   1188*   1213*   1218*   1225*   1232*   1249*   1253*   1256    1265*   1266
                            1286*   1290*   1292    1308*   1313*   1320*   1327*   1342*   1348*   1359*   1365*   1433*   1438*
GTSWR  =  104406            1696#
HT     =  000011             390#   1689
H322J     001422             399     510#
```

```
INADR   012152          1483    1486    1489    1502#   1552    1555    1558
INADRH  013123          1504    1635#
INDADR  013166          1507    1636#
INTBIT  001346           480#    557*    564*    573    1245    1282
IOTEST  002752           547     560     567     597     648#   1684
IOTST1  003026           661#
IOTTS1  003164           663     679#
IOTVEC= 000020           390#    515*
JUMPER  001376           492#    505*    507*    510*    554     561    1345    1362
LF    = 000012           390#   1689
LOCA    012126          1485    1488    1491    1493#   1500
LOCBOX  012054           400    1480#
LOCHDR  013062          1482    1634#
LOC1    001326           472#   1484    1494    1553    1568    1569    1570    1571    1572
LOC1Y   012332          1484    1544#   1553
LOC2    001330           473#   1487    1496    1556    1573    1574    1575    1576    1577
LOC2Y   012334          1487    1545#   1556
LOC3    001332           474#   1490    1498    1559    1578    1579    1580    1581    1582
LOC3Y   012336          1490    1546#   1559
LOUP1   012260          1493    1495    1497    1527#
MINSIN  001372           490#    558*    565*    576     583     681     929
MSGRUS  013322          1461*   1475*   1476    1640#
MSPNT1  013331          1462    1641#
NBEXT   001320           466#    651*   1385    1391*   1395*
NMBEXT  001316           465#    539*    550     651    1395
NOTLCH  001344           479#    556*    563*    570     642     918     946     951     983    1005    1011    1028    1054
ODDJMP  001370           489#    680*    681*    682*    870     879     882     892     895     904     907     913     916
                         949     985    1007    1029    1053    1082
PC    =%000007           390#    662*    678*   1400*   1405*   1427*   1431*   1432*   1456*   1477*   1481*   1550*   1675*
                        1678*   1664    1689*   1692*
PIRQ  = 177772           390#
PIRQVE= 000240           390#
PR0   = 000000           390#
PR1   = 000040           390#
PR2   = 000100           390#
PR3   = 000140           390#
PR4   = 000200           390#
PR5   = 000240           390#
PR6   = 000300           390#
PR7   = 000340           390#
PS    = 177776           390#   1439*
PSW   = 177776           390#   1112*   1138*   1149*   1156*   1166*   1172*   1181*   1189*   1198*   1204*   1221*   1231*
                        1316*   1326*   1344*   1347*   1357*   1361*   1364*   1371*   1377*   1382*
PWRMSG  016174          1684    1685#
PWRVEC= 000024           390#    515*   1425*   1426*   1684*
RBEG    001440           506     509     512     514#
RBEG1   002004           542#
RBEG2   002776           622     652#   1398
RDCHR = 104410          1692    1696#
RDLIN = 104411          1508    1691    1696#
RDOCT = 104412           634    1696#
RESVEC= 000010           390#
RUNMSG  013302          1436    1639#
R0    =%000000           390#    504*    508*    511*    513*    545    1030*   1040*   1055*   1065*   1103*   1118*   1220*
```

```
                    1315*  1397*  1400*  1401*  1444*  1445*  1449*  1450   1452*  1455*  1465*  1466*  1467*
                    1468*  1469   1534*  1536*  1538   1597*  1598*  1600*  1602*  1604*  1606*  1608*  1617*
                    1619*  1678*  1684*  1689*  1691*  1696*
R1      =%000001     390#   653*   655*   656*  1040*  1042   1065*  1067   1401*  1437*  1450*  1454   1469*
                    1470*  1471*  1472   1535*  1536   1593*  1611*  1612*  1613*  1615*  1616*  1618*  1621*
                    1623   1624*  1684*  1691*
R2      =%000002     390#   520*   522    523*   654*   655    657    658    1401*  1462*  1472*  1530*  1531*
                    1532   1534   1537   1539*  1684*  1691*
R3      =%000003     390#   521*   524*   527    538*   539    1401*  1454*  1455   1457   1532*  1533*  1535
                    1537*  1538*  1681*  1684*  1692*
R4      =%000004     390#  1681*  1684*
R5      =%000005     390#   568*   571*   574*   577*   594*   627    628    641*   1401*  1483*  1486*  1489*
                    1493*  1495*  1497*  1502   1503   1519   1520*  1527   1529*  1540*  1552*  1555*  1558*
                    1568*  1569*  1570*  1571*  1572*  1573*  1574*  1575*  1576*  1577*  1578*  1579*  1580*
                    1581*  1582*  1586   1588   1589   1590   1625*  1681*  1684*
R6      =%000006     390#   515*
R7      =%000007     390#
SETADD   003036      662    664#  1405   1432
SETTRP   011562     1423#  1431   1481   1550
SP      =%000006     390#   515*   526    550*   600    604    608    613*   620    635    1148   1154   1171
                    1179   1197   1202   1230   1324   1355   1376   1380   1400*  1401*  1404*  1430*  1442
                    1480*  1505*  1511   1549*  1674*  1675*  1678*  1681*  1684*  1689*  1691*  1692*  1696*
STACK = 001100       390#   515   1404   1430   1480   1549
STKLMT= 177774       390#
SWDPOB   013720      595   1649#
SWINTB   013430      572   1646#
SWNLB    013332      569    630   1645#
SWPOSB   013526      575   1647#
SWR      001140      402#   515*   529    532    610    617    631    635*   640    861    1210   1237   1274
                    1305   1367   1674   1675   1684*  1692*
SWREG    000176      394#   515    631    1692
SWTRAB   013622      578   1648#
SW0     = 000001     390#
SW00    = 000001     390#
SW01    = 000002     390#
SW02    = 000004     390#
SW03    = 000010     390#
SW04    = 000020     390#
SW05    = 000040     390#
SW06    = 000100     390#
SW07    = 000200     390#
SW08    = 000400     390#
SW09    = 001000     390#
SW1     = 000002     390#
SW10    = 002000     390#
SW11    = 004000     390#
SW12    = 010000     390#  1387
SW13    = 020000     390#   529    610
SW14    = 040000     390#
SW15    = 100000     390#   532    617
SW2     = 000004     390#
SW3     = 000010     390#
SW4     = 000020     390#
SW5     = 000040     390#
```

```
SW6    = 000100            390#
SW7    = 000200            390#
SW8    = 000400            390#
SW9    = 001000            390#
TALK     002666            568     571     574     577     594     627#
TBITVE= 000014             390#
TKVEC  = 000060            390#
TPVEC  = 000064            390#
TRANST   001374            491#    559#    566#    579     585     682
TRAPVE= 000034             390#    515#   1423#   1424#
TRTVEC= 000014             390#
TSTNUM   015020           1667    1668    1669    1671#   1675#
TST1     003242            684#
TST10    003604            745#
TST11    003670            760#
TST12    003766            768     771#
TST13    004064            779     782#
TST14    004152            794#
TST15    004240            806#
TST16    004310            813     816#
TST17    004352            822     825#
TST2     003260            689#
TST20    004410            830     833#
TST21    004452            939     843#
TST22    004514            849     860#
TST23    004612            872     875#
TST24    004676            884     888#
TST25    004762            897     900#
TST26    005046            909     912#
TST27    005226            914     942#
TST3     003316            694     697#
TST30    005406            977#
TST31    005524            999#
TST32    005664           1025#
TST33    006260           1047    1050#
TST34    006654           1072    1075#
TST35    006732           1083    1086#
TST36    007002           1093    1098#
TST37    007050           1105    1109#
TST4     003354            702     705#
TST40    007162           1124    1131#
TST41    007274           1142    1152    1158#
TST42    007432           1176    1178    1182#
TST43    007544           1193    1201    1207#
TST44    007732           1212    1236#
TST45    010122           1238    1273#
TST46    010254           1275    1302#
TST47    010436           1307    1329#
TST5     003412            710     713#
TST50    010516           1341#
TST51    010634           1358#
TST52    011000           1384#
TST6     003450            718     721#
TST7     003526            732#
```

```
TYPDS = 104405           1400    1695*
TYPE  = 104401            531     548     549     553     612     615     629     633     637    1400    1401    1436    1476
                         1482    1504    1507    1551    1675    1678    1681    1684    1689    1692    1696*
TYPOC = 104402            614    1506    1678    1692    1696*
TYPON = 104404           1696*
TYPOS = 104403            551    1696*
UNEXPT 002746             645*    683    1233    1325    1356    1381
VEC~?P 002442             540     600*
XLOADJ 012644            1568    1569    1570    1571    1572    1573    1574    1575    1576    1577    1578    1579    1580
                         1581    1582    1586*
XLOHDR 013204            1551    1637*
XLO1AD 012340             401    1549*
XLO10  012412            1554    1557    1560    1568*   1583
$AUTOB 001134             402*   1692
$BDADR 001122             402*
$BDDAT 001126             402*    692*    693     701*    708*    709     716*    717     725*    726     738*    739     751*
                          752*    753     766*    767     777*    778     787*    788     799*    800     811*    812     820*
                          821     828*    829     837*    838     848*    869*    870*    871     881*    882*    883     894*
                          895*    896     906*    907*    908     923*    924     933*    934     955*    956*    957     965*
                          966*    968     990*    991    1016*   1017    1042*   1046    1067*   1071    1081*   1082*   1091*
                         1092    1104*   1174*   1175    1242*   1266*   1279*   1667    1669
$CHARC 016456            1689**
$CKSWR 016564            1692*   1696
$CMTAG 001100             402*    515     658
$CM1  = 000002            402*
$CM2  = 000004            402*
$CM3  = 000002            402*
$CNTLG 017436            1692*
$CNTLU 017431            1692*
$CRLF  001173             402*   1675    1678    1689    1692
$DBLK  011444            1401*
$DCAGN 011204            1400*
$DTBL  011434            1401*
$ENDAD 011174            1400*   1675
$ENDCT 011142             515    1400*
$ENDMG 011213            1400*
$ENULL 011210            1400*
$EOP   011106            1396    1400*
$EOPCT 011134             515*   1400*
$ERFLG 001103             402*   1674*   1675*
$ERMAX 001115             402*    515*   1674*
$ERROR 015274             515    1675*
$ERRPC 001116             402*   1667    1668    1669    1675*   1678
$ERRTB 001176             402*   1678
$ERRTY 015440            1675    1678*
$ERTTL 001112             402*   1675*
$ESCAP 001170             402*    515*   1674*   1675
$FILLC 001156             402*   1689
$FILLS 001155             402*   1689
$GDADR 001120             402*
$GDDAT 001124             402*    582*    583     585     591*    690*    693     698*    706*    709     714*    717     723*
                          724     726     729*    734*    737     739     743*    747*    750     753     757*    761*    762
                          767     772*    773     778     784*    785     786     788     791*    796*    797     798     800
                          803*    809*    812     818*    821     826*    829     835*    838     845*    867*    871     878*
```

```
                        879*   880    883    891*   892*   893    896    903*   904*   905    908    915*   916
                        918    920    922    924    929    932    934    939*   948*   949    951    954    957
                        964    967*   979*   983    985    988    991    995    996*  1007   1010*  1011*  1012*
                       1013   1017   1027*  1028*  1029*  1030   1046   1052*  1053*  1054*  1055   1071   1079*
                       1087*  1092   1099*  1173*  1243*  1264*  1280*  1667   1669
$GET42  011164         1400#
$GTSWR  016634         1692#  1696
$HD   = 000000          388
$HIOCT  016562         1691#*
$ICNT   001104          402#  1674*
$ILLUP  016166         1684#
$INTAG  001135          402#  1692
$ITEMB  001114          402#  1675*  1678
$LF     001174          402#  1675   1689   1692
$LPADR  001106          402#   515*  1674*
$LPERR  001110          402#   515*   722*   733*   746*   783*   795*   943*   978*  1000*  1240*  1277*  1674*
                       1675
$MAIL = ****** U         515   1674   1675   1689
$MNEW   017454         1692#
$MSWR   017443          633   1692#
$MXCNT  015272         1674#
$NULL   001154          402#  1689
$NWTST= 000001          684#   689#   697#   705#   713#   721#   732#   745#   760#   771#   782#   794#   806#
                        816#   825#   833#   843#   860#   875#   888#   900#   912#   942#   977#   999#  1025#
                       1050#  1075#  1086#  1098#  1109#  1131#  1158#  1182#  1207#  1236#  1273#  1302#  1329#
                       1341#  1358#  1384#
$OCNT   016016         1681#*
$OMODE  016020         1681#*
$OVER   015256         1674#
$PASS   001100          402#  1400*  1674
$PWRAD  016162         1684#
$PWRDN  016022          515   1425   1684#
$PWRMG  016156         1684#
$PWRUP  016074         1684#
$QUES   001172          402#  1675   1689   1692
$RDCHR  017046         1692#  1696
$RDDEC= ****** U       1696
$RDLIN  017166         1692#  1696
$RDOCT  016462         1691#  1696
$RDSZ = 000007         1692#
$REGAD  001160          402#
$REG0   001162          402#
$REG1   001164          402#
$RTNAD  011206         1400#
$R2A  = ****** U       1696
$SAVRE= ****** U       1696
$SAVR6  016172         1684#*
$SCOPE  015022          515   1674#
$SETUP= 000137          440#   515   1400   1674   1675   1692
$STUP = 177777          440#
$SVLAD  015230         1674#
$SWR  = 165400          382#   388    392    402    515    684    689    697    705    713    721    732    745
                        760    771    782    794    806    816    825    833    843    860    875    888    900
                        912    942    977    999   1025   1050   1075   1086   1098   1109   1131   1158   1182
```

```
                    1207    1236    1273    1302    1329    1341    1358    1384    1400    1674    1675    1684
$SWRMK= 000000       392    1674
$TIMES  001166       402*    515*    697*    721*    782*    794*    843*    912*   1075*   1109*   1329*   1384*   1400*
                    1674*
$TKB    001146       402*   1692
$TKS    001144       402*   1692*
$TN    = 000053      383*    388     684*    689*    694     697*    702     705*    710     713*    718     721*    732*
                     745*    760*    768     771*    779     782*    794*    806*    813     816*    822     825*    830
                     833*    839     843*    849     860*    872     875*    884     888*    897     900*    909     912*
                     914     942*    977*    999*   1025*   1047    1050*   1072    1075*   1083    1086*   1093    1098*
                    1105    1109*   1124    1131*   1142    1152    1158*   1176    1178    1182*   1193    1201    1207*
                    1212    1236*   1238    1273*   1275    1302*   1307    1329*   1341*   1358*   1384*
$TPB    001152       402*   1689*
$TPFLG  001157       402*   1689
$TPS    001150       402*   1689
$TRAP   017466       515    1423    1696#
$TRAP2  017510      1696#
$TRP   = 000013     1696#
$TRPAD  017522      1696#
$TSTNM  001102       402*    863*   1400*   1674*   1675
$TTYIN  017422      1692#
$TYPBN= ****** U    1696
$TYPDS  011230      1401#   1696
$TYPE   016242      1689#   1696
$TYPEC  016412      1689#   1692
$TYPEX  016460      1689#
$TYPOC  015620      1681#   1696
$TYPON  015634      1681#   1696
$TYPOS  015574      1681#   1696
$XTSTR  015034      1674#
$$GET4= 000000      1400#
$OFILL  016017      1681#*
$40CAT= ****** U    1674    1675
.      = 017552      394#    402#    515     548#    612#    1400    1401#   1666#   1674    1675    1678#   1684    1686#
                    1689    1692#
```

```
ADJXL    1562*  1568  1569  1570  1571  1572  1573  1574  1575  1576  1577  1578  1579  1580  1581
         1582
CLERIN    857*   866   877   890   902  1003  1032  1057  1077  1248  1285
CLEROT    853*   865   876   889   901   989  1002  1014  1031  1040  1056  1065  1076  1247  1284
         1407  1411
CLRVCT    494*   683  1233  1325  1356  1381
COMMEN    390*
ENDCOM    390*
ERROR     390*   587   645   695   703   711   719   728   741   755   769   780   790   802   814
          823   831   840   850   873   885   898   910   926   936   959   970   993  1019  1048
         1073  1084  1094  1106  1126  1129  1141  1155  1169  1177  1180  1192  1203  1227  1229
         1258  1268  1294  1322  1333  1336  1353  1378
ESCAPE    390*
GETPRI    390*
GETSWR    386*   390*
MULT      390*
NEWTST    390*   684   689   697   705   713   721   732   745   760   771   782   794   806   816
          825   833   843   860   875   888   900   912   942   977   999  1025  1050  1075  1086
         1098  1109  1131  1158  1182  1207  1236  1273  1302  1329  1341  1358  1384
POP       390*  1401  1684  1691
PUSH      390*  1401  1684  1691
REPORT    390*
SCOPE     390*   684   689   697   705   713   721   732   745   760   771   782   794   806   816
          825   833   843   860   875   888   900   912   942   977   999  1025  1050  1075  1086
         1098  1109  1131  1158  1182  1207  1236  1273  1302  1329  1341  1358  1384  1400
SETPRI    390*
SETTRA   1696*
SETUP     390*   515
SKIP      390*   694   702   710   718   727   768   779   813   822   830   839   849   872   884
          897   909   914   958   969   992  1018  1047  1072  1083  1093  1105  1124  1142  1152
         1176  1178  1193  1201  1212  1228  1238  1246  1257  1267  1275  1283  1293  1307  1323
         1354  1379
SLASH     390*
SPACE     390*
STARS     390*   402   684   689   697   705   713   721   732   745   760   771   782   794   806
          816   825   833   843   860   875   888   900   912   942   977   999  1025  1050  1075
         1086  1098  1109  1131  1158  1182  1207  1236  1273  1302  1329  1341  1358  1384  1400
         1401  1674  1675  1678  1681  1684  1689  1691  1692  1696
SWRSU     390*   515*
TRMTRP   1696*
TYPBIN    390*
TYPDEC    390*  1400
TYPNAM    390*
TYPNUM    390*
TYPOCS    390*
TYPOCT    390*  1678  1692
TYPTXT    390*   548   549   553   612   615
$$CMRE    402*
$$CMTM    402*
$$ESCA    390*
$$NEWT    390*   684   689   697   705   713   721   732   745   760   771   782   794   806   816
          825   833   843   860   875   888   900   912   942   977   999  1025  1050  1075  1086
         1098  1109  1131  1158  1182  1207  1236  1273  1302  1329  1341  1358  1384
$$SET    1696*
```

```
$$SKIP    390*    694    702    710    718    768    779    813    822    830    839    849    872    884    897
           909     914   1047   1072   1083   1093   1105   1124   1142   1152   1176   1178   1193   1201   1212
          1238    1275   1307
.EQUAT    384*    390
.HEADE    384*    388
.SETUP    385*    440
.SWRHI    386*    392
.SWRLO    392*
.$CATC    384*    394
.$CMTA    384*    402
.$EOP     384*   1400
.$ERRO    384*   1675
.$ERRT    386*   1678
.$PARM    385*
.$POWE    385*   1684
.$RDOC    386*   1691
.$READ    385*   1692
.$SAVE    385*
.$SCOP    385*   1674
.$SPAC    385*
.$SWDO    385*
.$TRAP    385*   1696
.$TYPD    386*   1401
.$TYPE    384*    385*   1689
.$TYPO    384*   1681
```

```
ADD      666   668   670   673   675   677  1390  1401  1531  1533  1538  1594  1612  1616  1678
        1681  1689  1691  1692
ASL      743   757   973   996  1022  1270  1299  1618  1678  1691  1692  1696
ASLB    1401
ASR     1621
BCC      592   940  1401
BEQ      533   546   580   586   618   694   702   710   718   727   740   754   768   779   789
         801   830   839   849   862   872   884   897   909   914   917   925   935   952   958
         969   992  1018  1047  1072  1083  1093  1176  1211  1246  1306  1346  1363  1386  1400
        1458  1513  1515  1591  1674  1675  1678  1681  1689  1691  1692
BGE     1674
BGT     1400  1401  1681  1692
BHI      605  1674
BIC      616   750   870   879   882   892   895   904   907   922   956   964   966  1011  1012
        1028  1029  1053  1054  1082  1225  1251  1262  1288  1320  1331  1400  1452  1470  1512
        1536  1537  1622  1681  1691  1692
BIS      681   682   737   932   995  1021  1040  1065  1137  1165  1188  1250  1252  1263  1287
        1289  1348  1365  1401  1471  1587  1623  1681  1692
BISB    1678
BIT      529   532   583   585   610   617   812   821   861   916   918   929   949   951   968
         983   985  1005  1007  1175  1210  1237  1245  1274  1292  1305  1387  1674  1675
BITB    1689
BLO      601  1692
BLOS    1692
BLT     1401  1681  1689  1692
BMI     1105  1257  1401  1599  1601  1603  1605  1607  1609
BNE      515   528   530   544   555   562   611   632   659   730   744   758   792   804   813
         822   919   930   950   974   984   986   997  1006  1008  1023  1238  1271  1275  1283
        1300  1332  1335  1388  1401  1416  1446  1448  1453  1474  1517  1528  1674  1675  1678
        1681  1684  1689  1692
BPL     1267  1293  1401  1596  1620  1675  1681  1689  1692
BR       506   509   512   515   525   536   548   549   553   603   607   612   615   636   663
        1124  1142  1152  1178  1193  1201  1212  1228  1307  1323  1354  1375  1379  1392  1401
        1417  1441  1459  1464  1485  1488  1491  1500  1554  1557  1560  1610  1614  1674  1678
        1681  1684  1689  1691  1692
CLR      504   505   508   511   515   521   556   558   559   565   680   685   686   687   698
         723   736   807   817   834   844   845   865   867   876   889   901   967   980   989
        1002  1014  1031  1040  1056  1065  1076  1073  1088  1100  1111  1112  1123  1132  1138
        1149  1156  1159  1166  1172  1173  1181  1183  1189  1198  1204  1213  1221  1231  1232
        1242  1247  1249  1253  1264  1265  1280  1284  1286  1290  1308  1312  1316  1326  1327
        1342  1357  1359  1371  1377  1382  1400  1401  1407  1411  1433  1439  1593  1624  1674
        1678  1681  1684  1691  1692
CLRB    1401  1674  1689  1692
CMP      515   526   554   561   600   604   620   631   657   658   693   709   717   726   739
         753   767   778   788   800   829   838   871   883   896   908   924   934   957   991
        1017  1046  1071  1090  1092  1102  1148  1154  1171  1179  1197  1202  1230  1324  1334
        1355  1376  1380  1401  1442  1457  1514  1516  1674  1675  1692
CMPB    1674  1689  1692
COM      752   765   776   947  1040  1065
DEC      538  1391  1400  1415  1445  1447  1473  1619  1678  1692
DECB    1681  1689
EMT      390
HALT     394   534   602   606   619   639  1675  1684  1689
INC      524   729  1400  1401  1518  1613  1674  1675  1681  1684  1692
```

```
.TITLE    388
.WORD     394     402     552     1400    1678    1681    1684    1689    1691    1696


 ERRORS DETECTED:  0

 *DSKZ:DZDRGD,DZDRGD/CRF=DZDRGD
 RUN-TIME:  63 36 5 SECONDS
 CORE USED:  23K
```