DR11K

DR11K DIGITAL I/O

CZDRGE0

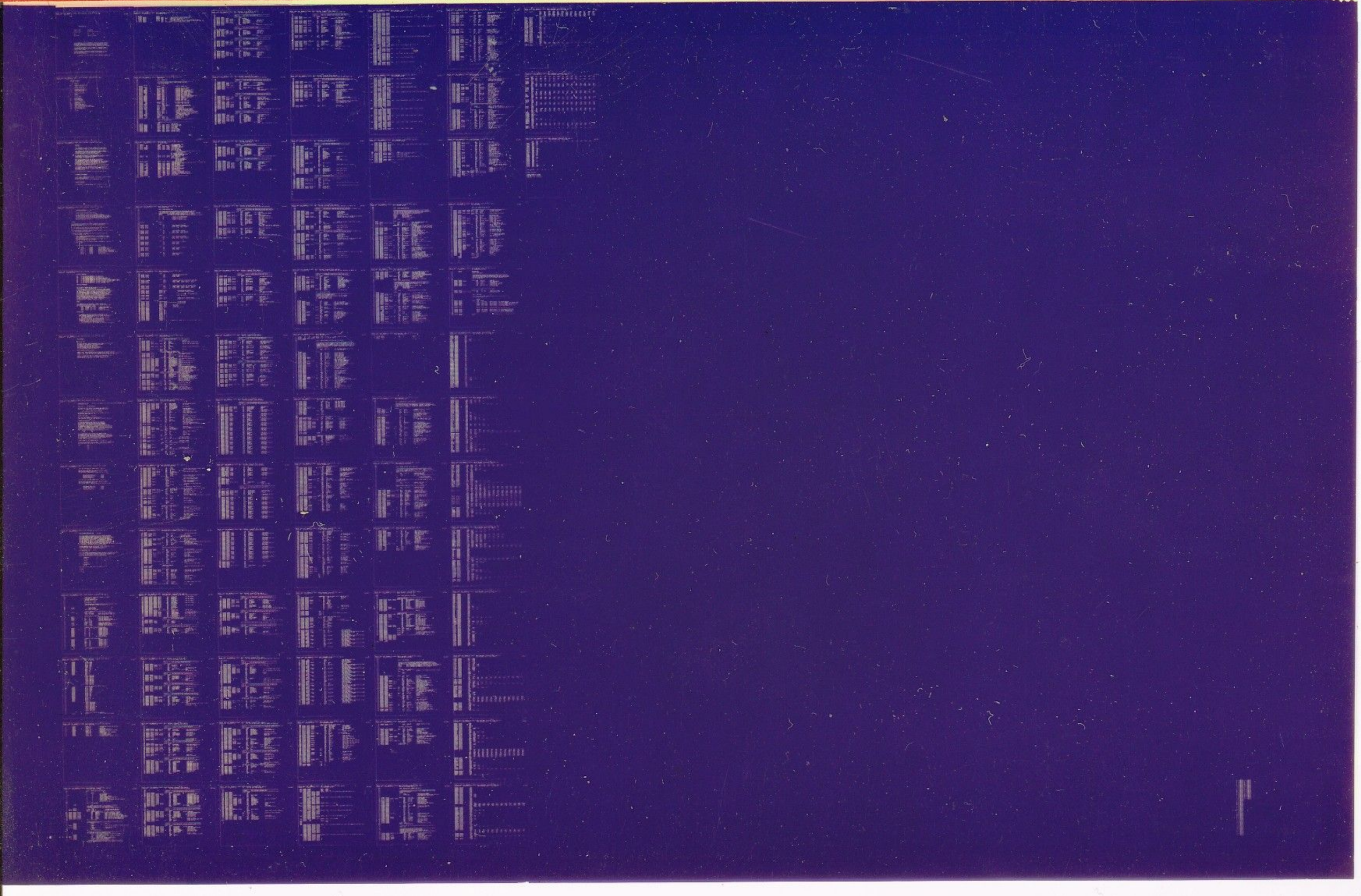
AH-8662E-MC

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA



.REM%

IDENTIFICATION

PRODUCT CODE: AC-8660E-MC

PRODUCT NAME: CZDRGE0 DR11K DIGITAL I/O

DATE: AUG. 1979

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 XL01'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 (REF 9.3).

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 "$BASE" AND "$VECT1". FOR NORMAL FACTORY CONFIG., ALL QUESTIONS SHOULD BE ANSWERED WITH A VALUE OF 0.

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

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERMINAL AND 8K 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.

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

7. RESTRICTIONS

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

8. MISCELANFOUS

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

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

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

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

9.1 LOGIC TEST <SA 200 AND 204>

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

W21A, W22A AND W23A CAN BE DIAGNOSED.

THE PROGRAM CHECKS THAT THE DR11-K 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. "$BASE" (LOC. 1244) 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. 1370) FOR THE PROPER DELAY ON A DIFFERENT CPU TYPE:

11/05	240
11/40	620

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 [illegible], 167760 AND 167750.
THE OPERATOR MAY CHANGE LOCATIONS LOC1, LOC2, LOC3
(LOC. 1402, 1404, 1406) TO RUN ON OTHER DR11K'S.

9.7 XL01 ADJUSTMENT LOOP <SA 234>

THIS LOOP ENABLES THE OPERATOR TO VERIFY OR ADJUST THE XL01/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.

```
                                .TITLE  DR11-K  LOGIC TEST  MAINDEC-11-CZDRG-E
                                ;*COPYRIGHT (C) 1979
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY R. SHOOP
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*

 167770                                 ABASE=167770
 100300                                 AVECT1=100300
                                .SBTTL  BASIC DEFINITIONS

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

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

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

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

                                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
 100000                         SW15=   100000
```

```
(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
```

```
                                                        N 1
396                                     .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>
397
398                                     .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  001454          JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
399  000204     000137  001510          JMP     @#BEGIN1        ;JUMP TO THE RESTART ADDRESS
400  000210     000137  011604          JMP     @#EXTTST        ;JUMP TO THE CONTROL LINES LOOP
401  000214     000137  011766          JMP     @#COULTR        ;JUMP TO THE COULTER FUNCTION LOOP
402  000220     000137  001464          JMP     COULTJ          ;JUMP TO SA WITH THE COULTER JUMPER CONFIG.
403  000224     000137  001476          JMP     M322J           ;JUMP TO SA WITH THE LOC-BOX JUMPER CONFIG.
404  000230     000137  012226          JMP     LOCBOX          ;JUMP TO SA OF LOC-BOX FUNCTION LOOP
405  000234     000137  012540          JMP     XL01AD          ;JUMP TO SA OF XL01 ADJUSTMENT LOOP
406                                     .SBTTL  ACT11 HOOKS
(1)
(2)                                     ;;******************************************************************
(1)                                     ;HOOKS REQUIRED BY ACT11
(1)             000240                          $SVPC=.                 ;SAVE PC
(1)             000046                          .=46
(1)  000046     011324                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)             000052                          .=52
(1)  000052     000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
(1)             000240                          .=$SVPC                 ;; RESTORE PC
407             001000                          .=1000
408                                     .SBTTL  APT PARAMETER BLOCK
(1)
(2)                                     ;;******************************************************************
(1)                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                     ;;******************************************************************
(1)             001000                          .$X=.   ;;SAVE CURRENT LOCATION
(1)             000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1)  000024     000200                          200     ;;FOR APT START UP
(1)             000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1)  000044     001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
(1)             001000                          .=.$X   ;;RESET LOCATION COUNTER
(2)                                     ;;******************************************************************
(1)                                     ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)                                     ;INTERFACE SPEC.
```

```
                                                                    B 2
   (1)
   (1)  001000                          $APTHD:
   (1)  001000  000000                  $HIBTS: .WORD   0        ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
   (1)  001002  001170                  $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
   (1)  001004  000030                  $TSTM:  .WORD   30       ;;RUN TIM OF LONGEST TEST
   (1)  001006  000010                  $PASTM: .WORD   10       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
   (1)  001010  000030                  $UNITM: .WORD   30       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
   (1)  001012  00003'                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
409
(1)
(2)
(1)
(1)
(1)
(1)          001100                     .=1100
(1)  001100                     $CMTAG:                         ;;START OF COMMON TAGS
(1)  001100  000000                     .WORD   0
(1)  001102     000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
(1)  001103     000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
(1)  001104  000000             $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
(1)  001106  000000             $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
(1)  001110  000000             $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
(1)  001112  000000             $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
(1)  001114     000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
(1)  001115     001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
(1)  001116  000000             $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
(1)  001120  000000             $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
(1)  001122  000000             $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
(1)  001124  000000             $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
(1)  001126  000000             $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
(1)  001130  000000                     .WORD   0               ;;RESERVED--NOT TO BE USED
(1)  001132  000000                     .WORD   0
(1)  001134     000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
(1)  001135     000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
(1)  001136  000000                     .WORD   0
(1)  001140  177570             SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
(1)  001142  177570             DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
(1)  001144  177560             $TKS:   177560                  ;;TTY KBD STATUS
(1)  001146  177562             $TKB:   177562                  ;;TTY KBD BUFFER
(1)  001150  177564             $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
(1)  001152  177566             $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
(1)  001154     000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
(1)  001155     002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
(1)  001156     012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
(1)  001157     000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0 YES)
(1)  001160  000000             $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
(1)  001162  000000             $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
(1)  001164     077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
(1)  001165     015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
(1)  001166  000012             $LF:    .ASCIZ  <12>            ;;LINE FEED
(2)
(2)
(2)
(3)
(2)
(2)  001170                     $MAIL:                          ;;APT MAILBOX
(2)  001170  000000             $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
(2)  001172  000000             $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
(2)  001174  000000             $TESTN: .WORD   ATESTN          ;;TEST NUMBER
(2)  001176  000000             $PASS:  .WORD   APASS           ;;PASS COUNT
(2)  001200  000000             $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
(2)  001202  000000             $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
(2)  001204  000000             $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
(2)  001206  000000             $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
```

```
(2)  001210                          $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001210      000                 $ENV:    .BYTE  AENV    ;;ENVIRONMENT BYTE
(2)  001211      000                 $ENVM:   .BYTE  AENVM   ;;ENVIRONMENT MODE BITS
(2)  001212  000000                  $SWREG:  .WORD  ASWREG  ;;APT SWITCH REGISTER
(2)  001214  000000                  $USWR:   .WORD  AUSWR   ;;USER SWITCHES
(2)  001216  000000                  $CPUOP:  .WORD  ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                  ;*                      BITS 15-11=CPU TYPE
(2)                                  ;*                              11/04=01,11/05=02,11/20=03,11/40 04,11/45-05
(2)                                  ;*                              11/70=06,PDQ=07,Q=10
(2)                                  ;*                      BIT 10=REAL TIME CLOCK
(2)                                  ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                  ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001220      000                 $MAMS1:  .BYTE  AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001221      000                 $MTYP1:  .BYTE  AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                  ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                  ;*                              900 NSEC CORE=001
(2)                                  ;*                              300 NSEC BIPOLAR=002
(2)                                  ;*                              500 NSEC MOS=003
(2)  001222  000000                  $MADR1:  .WORD  AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                  ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2)  001224      000                 $MAMS2:  .BYTE  AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001225      000                 $MTYP2:  .BYTE  AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001226  000000                  $MADR2:  .WORD  AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001230      000                 $MAMS3:  .BYTE  AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001231      000                 $MTYP3:  .BYTE  AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001232  000000                  $MADR3:  .WORD  AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001234      000                 $MAMS4:  .BYTE  AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001235      000                 $MTYP4:  .BYTE  AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001236  000000                  $MADR4:  .WORD  AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001240  100300                  $VECT1:  .WORD  AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001242  000000                  $VECT2:  .WORD  AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001244  167770                  $BASE:   .WORD  ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001246  000000                  $DEVM:   .WORD  ADEVM   ;;DEVICE MAP
(2)  001250  000000                  $CDW1:   .WORD  ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001252                          $ETEND:
(2)                                  .MEXIT
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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


(1) 001252                      $ERRTB:

                                ;ITEM   1
001252  014251                          .+1             ;STATUS REGISTER IN ERROR
001254  014671                  DH1     ;ERRPC  DRADD   STATUS EXPECTED
001256  015162                  DT1     ;$ERRPC DRADD   $BDDAT $GDDAT
001260  015224                  DF0

                                ;ITEM   2
001262  014302                  EM2     ;INPUT REGISTER IN ERROR
001264  014741                  DH2     ;ERRPC  DRADD   INPUT EXPECTED
001266  015162                  DT1     ;$ERRPC DRADD   $BDDAT $GDDAT
001270  015224                  DF0

                                ;ITEM 3
001272  014332                  EM3     ;OUTPUT REGISTER IN ERROR
001274  015007                  DH3     ;ERRPC  DRADD   OUTPUT EXPECTED
001276  015162                  DT1     ;$ERRPC DRADD   $BDDAT $GDDAT
001300  015224                  DF0

                                ;ITEM   4
001302  014363                  EM4     ;INPUT FAILED TO INTERRUPT
001304  015055                  DH4     ;ERRPC  DRADD
001306  015176                  DT4     ;$ERRPC DRADD
001310  015224                  DF0

                                ;ITEM   5
001312  014415                  EM5     ;OUTPUT FAILED TO INTERRUPT
001314  015055                  DH4     ;ERRPC  DRADD
001316  015176                  DT4     ;$ERRPC DRADD
001320  015224                  DF0

                                ;ITEM   6
001322  014450                  EM6     ;UNEXPECTED INTERRUPT
001324  015055                  DH4     ;ERRPC  DRADD
001326  015176                  DT4     ;$ERRPC DRADD
001330  015224                  DF0
```

```
                                ;ITEM   7
001332  014475                          EM7             ;OPERATOR INTERVENTION ERROR
001334  015055                          DH4             ;ERRPC  DRADD
001336  015176                          DT4             ;$ERRPC DRADD
001340  015224                          DF0

                                ;ITEM   10
001342  014531                          EM10            ;INTERRUPT INPUT BIT FAILED TO SET INPUT READY
001344  015102                          DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
001346  015206                          DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
001350  015224                          DF0

                                ;ITEM   11
001352  014614                          EM11            ;NON-INTERRUPTING INPUT BIT SET INPUT READY
001354  015102                          DH10            ;ERRPC  DRADD   STATUS  EXPECTED  INPUT BIT
001356  015206                          DT10            ;$ERRPC DRADD   $BDDAT  $GDDAT    BRLEV3
001360  015224                          DF0

001362  167770            BASEBA: 167770                  ;STARTING BASE BUS ADDRESS
001364  000300            BASEIV: 300                     ;STARTING BASE INTERRUPT ADDRESS
001366  000200            BASEBR: 200
001370  000240            CPU:    240                     ;1 MS. CPU DELAY FACTOR 240 FOR 11/05
                                                        ;                       620 FOR 11/40
001372  000000            NMBEXT: 0                       ;ADDITIONAL DR-11-K
001374  000000            NBEXT:  0

001376  167770            DRADD:  167770                  ;CURRENT DR11-K STARTING ADDRESS
001400  000300            DRIV:   300                     ;CURRENT DR11-K STARTING INTERRUPT VECTOR

001402  167770            LOC1:   167770                  ;LOC BOX #1 ADDR
001404  167760            LOC2:   167760                  ;LOC BOX #2 ADDR
001406  167750            LOC3:   167750                  ;LOC BOX #3 ADDR
001410  167770            GRSTAT: 167770          ;DR STATUS
001412  167772            GRDAI:  167772          ;INPUT REG.
001414  167774            GRDIO:  167774          ;OUTPUT REG.
001416  167775            GRBHIO: 167775          ;OUTPUT REG HIGH BYTE
001420  000000            NOTLCH: 0
001422  000000            INTBIT: 0
001424  000300            GRIVA:  300
001426  000302            GRIVSA: 302
001430  000304            GRIVB:  304
001432  000306            GRIVSB: 306
001434  000200            DIOBRL: 200
001436  000000            BRLEV1: 0
001440  000000            BRLEV2: 0
001442  000000            BRLEV3: 0
001444  000000            ODDJMP: 0
001446  000000            MINSIN: 0
001450  000000            TRANST: 0
001452  000000            JUMPER: 0       ;1 IF RUNNING H322 JUMPER CONFIG ;2 IF COULTER JUMPER CONFIG
```

```
                                        ;;*****************************************************
                                        ;       DIGITAL I-O LOGIC TEST
                                        ;*****************************************************
001454  005000                  BEGIN:  CLR     R0                      ;CLEAR R0
001456  005037  001452                  CLR     JUMPER                  ;INDICATE NORMAL FACTORY BUILD
001462  000414                          BR      RBEG
001464  012737  000002  001452  COULTJ: MOV     #2,JUMPER               ;INDICATE COULTER JUMPER CONFIG
001472  005000                          CLR     R0
001474  000407                          BR      RBEG
001476  012737  000001  001452  H322J:  MOV     #1,JUMPER               ;INDICATE LOC BOX JUMPER CONFIG
001504  005000                          CLR     R0
001506  000402                          BR      RBEG
001510  012700  177777          BEGIN1: MOV     #-1,R0                  ;LOAD R0
001514  000005                  RBEG:   RESET
                                .SBTTL  INITIALIZE THE COMMON TAGS
(1)                             ;;CLEAR THE COMMON TAGS (.$CMTAG) AREA
(1)     001516  012706  001100          MOV     #$CMTAG,R6              ;;FIRST LOCATION TO BE CLEARED
(1)     001522  005026                  CLR     (R6)+                   ;;CLEAR MEMORY LOCATION
(1)     001524  022706  001140          CMP     #SWR,R6 ;;DONE?
(1)     001530  001374                  BNE     .-6                     ;;LOOP BACK IF NO
(1)     001532  012706  001100          MOV     #STACK,SP               ;;SETUP THE STACK POINTER
(1)                             ;;INITIALIZE A FEW VECTORS
(1)     001536  012737  015244  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1)     001544  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
(1)     001552  012737  015524  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1)     001560  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
(1)     001566  012737  020254  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1)     001574  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
(1)     001602  012737  016300  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1)     001610  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
(1)     001616  013737  011272  011264          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
(1)     001624  005037  001160                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
(1)     001630  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1)     001634  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
(1)     001642  012737  001642  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1)     001650  012737  001650  001110          MOV     #.,$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)     001656  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
(2)     001662  012737  001716  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
(2)     001670  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
(2)     001676  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
(2)     001704  022777  177777  177226          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
(2)     001712  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                                                                     ;;AND  THE HARDWARE SWR IS NOT  -1
(2)     001714  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
(2)     001716  012716  001724          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
(2)     001722  000002                          RTI
(2)     001724  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
(2)     001732  012737  000174  001142          MOV     #DISPREG,DISPLAY
(2)     001740  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
(1)
(2)     001744  005037  001176                  CLR     $PASS           ;;CLEAR PASS COUNT
(2)     001750  132737  000200  001211          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
(2)     001756  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
(2)     001760  012737  001212  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
```

```
   (2) 001766                                   67$:
   523 001766 013737 001244 001362                      MOV     $BASE,BASEBA            ;GET APT DEFINED ADDRESS
   524 001774 013737 001240 001364                      MOV     $VECT1,BASEIV           ;GET VECTOR
   525 002002 042737 160000 001364                      BIC     #160000,BASEIV
   526 002010 113737 001241 001366                      MOVB    $VECT1+1,BASEBR         ;GET PRIORITY
   527 002016 042737 177437 001366                      BIC     #177437,BASEBR
   528 002024 012737 002052 000004                      MOV     #1$,@#ERRVEC            ;LOAD BUS ERROR
   529 002032 013702 001362                             MOV     BASEBA,R2               ;LOAD STARTING ADDRESS
   530 002036 005003                                    CLR     R3                      ;CLEAR COUNT
   531 002040 005712                            2$:     TST     (R2)                    ;TEST IF EXISTENT
   532 002042 162702 000010                             SUB     #10,R2                  ;EXIST, UPDATE TEST ADDRESS
   533 002046 005203                                    INC     R3                      ;UPDATE # OF DR11'S
   534 002050 000773                                    BR      2$
   535 002052 022626                            1$:     CMP     (SP)+,(SP)+             ;POP STACK
   536 002054 005703                                    TST     R3                      ;TEST IF FIRST DOES EXIST
   537 002056 001014                                    BNE     3$                      ;BR
   538 002060 032777 020000 177052                      BIT     #SW13,@SWR              ;TEST IF INHIBIT TYPEOUT
   539 002066 001002                                    BNE     4$                      ;BR IF YES
   540 002070 104401 013463                             TYPE,   BUSTRP                  ;TELL OPERATOR BASE DR11K DOESN'T EXIST
   541 002074 032777 100000 177036              4$:     BIT     #SW15,@SWR              ;TEST HALT ON ERROR
   542 002102 001401                                    BEQ     5$                      ;BR IF NO HALT
   543 002104 000000                                    HALT                            ;FIRST DR11-K DOES NOT EXIST
   544                                                                                  ;CHECK THE PROGRAM DEVICE ADDRESS
   545 002106 000754                            5$:     BR      2$                      ;TRY AGAIN
   546
   547 002110 005303                            3$:     DEC     R3                      ;ADJUST R3
   548 002112 010337 001372                             MOV     R3,NMBEXT               ;SAVE THE NUMBER OF ADDITIONAL DR11-K
   549 002116 012737 002572 000004                      MOV     #VECTRP,@#ERRVEC        ;RESET BUS ERROR
   550 002124 012737 000340 000006                      MOV     #340,@#ERRVEC+2
   551 002132 000005                            RBEG1:  RESET
   552 002134 005737 000042                             TST     @#42                    ;TEST IF UNDER A MONITOR
   553 002140 001002                                    BNE     4$                      ;BR IF
   554 002142 005700                                    TST     R0                      ;TEST R0
   555 002144 001402                                    BEQ     2$                      ;BR IF CLEARED
   556 002146 000137 003102                     4$:     JMP     IOTEST                  ;JUMP IF SET
   557 002152                                   2$:
   (1) 002152 104401 002160                             TYPE    ,65$                ;;TYPE ASCIZ STRING
   (1) 002156 000426                                    BR      64$                 ;;GET OVER THE ASCIZ
   (1)                                          ;;65$:  .ASCIZ  <15><12><15><12>/DR11-K DIGITAL INPUT OUTPUT LOGIC TEST/
   (1) 002234                                   64$:
   558 002234 104401 002242                             TYPE    ,67$                ;;TYPE ASCIZ STRING
   (1) 002240 000414                                    BR      66$                 ;;GET OVER THE ASCIZ
   (1)                                          ;;67$:  .ASCIZ  <15><12>/MAINDEC-11-CZDRG-E/<15><12>
   (1) 002272                                   66$:
   559 002272 013746 001372                             MOV     NMBEXT,-(SP)
   560 002276 104403                                    TYPOS
   561 002300 000002                                    .WORD   2
   562 002302 104401 002310                             TYPE    ,69$                ;;TYPE ASCIZ STRING
   (1) 002306 000422                                    BR      68$                 ;;GET OVER THE ASCIZ
   (1)                                          ;;69$:  .ASCIZ  /(8) ADDITIONAL DR11-K'S CONNECTED/<15><12>
   (1) 002354                                   68$:
   563 002354 022737 000001 001452                      CMP     #1,JUMPER               ;TEST IF LOC-BOX CONFIG.
   564 002362 001013                                    BNE     1$                      ;BR IF NOT
   565 002364 005037 001420                             CLR     NOTLCH                  ;LOAD LOC-BOX JUMPER CONFIG
   566 002370 012737 177777 001422                      MOV     #-1,INTBIT              ;
   567 002376 005037 001446                             CLR     MINSIN
```

```
                                                                                    1  2
   568  002402  005037  001450                CLR     TRANST
   569  002406  000137  003102                JMP     IOTEST                  ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
   570  002412  022737  000002  001452  1$:   CMP     #2,JUMPER               ;TEST IF COULTER CONFIG.
   571  002420  001015                        BNE     3$              ;BR IF NOT
   572  002422  012737  177777  001420        MOV     #-1,NOTLCH              ;LOAD COULTER JUMPER CONFIG.
   573  002430  012737  170000  001422        MOV     #170000,INTBIT
   574  002436  005037  001446                CLR     MINSIN
   575  002442  012737  170000  001450        MOV     #170000,TRANST
   576  002450  000137  003102                JMP     IOTEST                  ;RUN THE LOGIC TEST WITH COULTER JUMPERS
   577  002454  004537  003016          3$:   JSR     R5,TALK                 ;TALK TO THE ANIMALS
   578  002460  013554                        SWNLB
   579  002462  001420                        NOTLCH                          ;ABOUT NON LATCH INPUT BITS
   580  002464  004537  003016                JSR     R5,TALK                 ;TALK TO THE ANIMAL AGAIN
   581  002470  013652                        SWINTB
   582  002472  001422                        INTBIT                          ;ABOUT THE INTERRUPTING BITS
   583  002474  004537  003016                JSR     R5,TALK                 ;ISN'T THIS BOARING
   584  002500  013750                        SWPOSB
   585  002502  001446                        MINSIN                          ;AGAIN-ABOUT THE MINUS INPUT BITS
   586  002504  004537  003016                JSR     R5,TALK                 ;HO-HUM
   587  002510  014044                        SWTRAB
   588  002512  001450                        TRANST                          ;AGAIN-ABOUT THE TRANSITION BITS
   589
   590                                        .SBTTL  DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
   591  002514  012737  010000  001124        MOV     #BIT12,$GDDAT           ;LOAD TEST BIT
   592  002522  033737  001124  001446  10$:   BIT     $GDDAT,MINSIN           ;TEST IF NEG INPUT BIT
   593  002530  001407                        BEQ     11$                     ;BR IF NOT
   594  002532  033737  001124  001450        BIT     $GDDAT,TRANST           ;TEST IF TRANS. INPUT
   595  002540  001403                        BEQ     11$                     ;BR IF NOT
   596  002542  104007                        ERROR   7                       ;LOGICAL OPERATOR ERROR
   597                                                                        ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
   598  002544  000137  001454                JMP     BEGIN                   ;
   599
   600  002550  006137  001124          11$:  ROL     $GDDAT                  ;MOVE LEFT
   601  002554  103362                        BCC     10$                     ;BR UNTIL DONE
   602
   603  002556  004537  003016                JSR     R5,TALK                 ;ASK THE OPERATOR
   604  002562  014142                        SWDPOB
   605  002564  001442                        BRLEV3                          ;ABOUT PROGRAM OPTIONS
   606  002566  000137  003102                JMP     IOTEST
   607
   608                                    ;INTERRUPT TO UNEXPECTED VECTOR
   609  002572  021627  001000        VECTRP: CMP     (SP),#1000              ;TEST IF IN PROGRAM CODE
   610  002576  103402                        BLO     1$                      ;BR IF NOT
   611  002600  000000                70$:    HALT                            ;FATAL BUS TRAP IN PROGRAM AREA
   612  002602  000776                        BR      70$
   613  002604  021627  000200          1$:   CMP     (SP),#200               ;TEST IF IN SACRED VECTOR AREA
   614  002610  101002                        BHI     2$                      ;BR IF NOT
   615  002612  000000                71$:    HALT                            ;FATAL VECTOR TRAP
   616  002614  000776                        BR      71$
   617  002616  012637  003014          2$:   MOV     (SP)+,72$               ;GET ADDR .HE TRAP OCCURRED TO
   618  002622  162737  000004  003014        SUB     #4,72$                  ;MAKE REAL ADDRESS
   619  002630  032777  020000  176302        BIT     #SW13,@SWR              ;TEST IF TYPE ERROR INHIBIT
   620  002636  001050                        BNE     3$                      ;BR IF INHIBIT
   621  002640  104401  002646                TYPE    ,65$            ;;TYPE ASCIZ STRING
   (1)  002644  000417                        BR      64$             ;;GET OVER THE ASCIZ
   (1)                                ;;65$:  .ASCIZ  <15><12>/DR11K INTERRUPTED TO LOC. /
```

```
(1)  002704                                    64$:
622  002704  013746  003014                            MOV     72$,-(SP)               ;LOAD VALUE TO BE TYPED
623  002710  104402                                    TYPOC
624  002712  104401  002720                            TYPE    ,67$                ;;TYPE ASCIZ STRING
(1)  002716  000420                                    BR      66$                 ;;GET OVER THE ASCIZ
(1)                                            ;;67$:  .ASCIZ  / AND WILL NOW USE THAT VECTOR/<15><12>
(1)  002760                                    66$:
625  002760  042737  000007  003014            3$:     BIC     #7,72$                  ;MASK OFF LOWER 3 BITS
626  002766  032777  100000  176144                    BIT     #SW15,@SWR              ;TEST IF HALT ON ERROR
627  002774  001401                                    BEQ     4$                      ;BR IF NOT
628  002776  000000                                    HALT            ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
629  003000  022626                            4$:     CMP     (SP)+,(SP)+             ;CLEAN THE STACK
630  003002  013737  003014  001400                    MOV     72$,DRIV                ;LOAD NEW VECTOR ADDRESS
631  003010  000137  003126                            JMP     RBEG2                   ;TEST THE DR11K AT THE NEW VECTOR
632  003014  000000                            72$:    0
633
634                                            ;OPERATOR QUESTIONS AND ANSWER ROUTINE
635
636  003016  012537  003030                    TALK:   MOV     (R5)+,10$               ;GET ASCII POINTER
637  003022  012537  003074                            MOV     (R5)+,11$               ;GET POINTER TO DATA FROM OPERATOR
638  003026  104401                                    TYPE
639  003030  013554                            10$:    SWNLB                           ;TELL OPERATOR TO LOAD SWITCHES
640  003032  023727  001140  000176                    CMP     SWR,#SWREG              ;TEST IF SWITCH REG. EXISTS
641  003040  001006                                    BNE     1$                      ;BR IF NO
642  003042  104401  017763                            TYPE    ,$MSWR                  ;TYPE "SWR ="
643  003046  104412                                    RDOCT                           ;READ OCTAL
644  003050  012677  176064                            MOV     (SP)+,@SWR              ;SAVE THE SWITCHES
645  003054  000403                                    BR      2$
646  003056  104401                            1$:     TYPE
647  003060  014227                                    DEPCNT                          ;TELL OPERATOR TO DEPRESS CONT
648  003062  000000                                    HALT                            ;WAIT FOR OPERATOR
649  003064  017777  176050  000002            2$:     MOV     @SWR,@11$               ;LOAD SWITCH VALUE
650  003072  000205                                    RTS     R5                      ;EXIT
651  003074  001420                            11$:    NOTLCH                          ;POINTER TO DATA TO BE LOADED
652
653                                            ;UNEXPECTED INTERRUPT
654  003076  104006                            UNEXPT: ERROR   6                       ;UNEXPECTED INTERRUPT DURING A SUB-TEST
655  003100  000002                                    RTI                             ;EXIT
656
657  003102  000005                            IOTEST: RESET
658  003104  013737  001362  001376                    MOV     BASEBA,DRADD            ;LOAD INITIAL STARTING ADDRESS
659  003112  013737  001364  001400                    MOV     BASEIV,DRIV             ;LOAD INITIAL STARTING VECTOR
660  003120  013737  001372  001374                    MOV     NMBEXT,NBEXT            ;RELOAD # AVAILABLE
661  003126  000005                            RBEG2:  RESET
662  003130  012701  000240                            MOV     #240,R1                 ;LOAD R1
663  003134  012702  000242                            MOV     #242,R2
664  003140  010221                            5$:     MOV     R2,(R1)+                ;SAVE THE ADDRESS
665  003142  012721  004700                            MOV     #4700,(R1)+             ;LOAD "JSR      PC,R0"
666  003146  022222                                    CMP     (R2)+,(R2)+             ;BUMP R2
667  003150  020227  001076                            CMP     R2,#$CMTAG-2            ;TEST FOR LAST
668  003154  001371                                    BNE     5$                      ;BR UNTIL DONE
669
670  003156  000005                            IOTST1: RESET
671  003160  004737  003166                            JSR     PC,SETADD               ;SET UP BUS ADDRESS AND VECTOR
672  003164  000453                                    BR      IOTTS1
673  003166  013737  001376  001410            SETADD: MOV     DRADD,GRSTAT            ;LOAD 1ST ADDRESS
```

```
                                                                                    K 2
   674  003174  013737  001376  001412          MOV     DRADD,GRDAI             ;LOAD 2ND ADDRESS
   675  003202  062737  000002  001412          ADD     #2,GRDAI
   676  003210  013737  001376  001414          MOV     DRADD,GRDIO             ;LOAD 3RD ADDRESS
   677  003216  062737  000004  001414          ADD     #4,GRDIO
   678  003224  013737  001376  001416          MOV     DRADD,GRBHIO            ;LOAD 4TH ADDRESS
   679  003232  062737  000005  001416          ADD     #5,GRBHIO
   680  003240  013737  001400  001424          MOV     DRIV,GRIVA              ;LOAD FIRST VECTOR
   681  003246  013737  001400  001426          MOV     DRIV,GRIVSA
   682  003254  062737  000002  001426          ADD     #2,GRIVSA
   683  003262  013737  001400  001430          MOV     DRIV,GRIVB              ;LOAD 2ND VECTOR
   684  003270  062737  000004  001430          ADD     #4,GRIVB
   685  003276  013737  001400  001432          MOV     DRIV,GRIVSB
   686  003304  062737  000006  001432          ADD     #6,GRIVSB
   687  003312  000207                          RTS     PC
   688  003314  013737  001366  001434  IOTTS1: MOV     BASEBR,DIOBRL           ;LOAD BR LEVEL
   689  003322  005037  001444                  CLR     ODDJMP
   690  003326  053737  001446  001444          BIS     MINSIN,ODDJMP           ;SET MINUS INPUT BITS
   691  003334  053737  001450  001444          BIS     TRANST,ODDJMP           ;SET TRANSITION INPUT BITS
   692  003342  012777  003076  176054          MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
   (1)  003350  012777  000340  176050          MOV     #340,@GRIVSA
   (1)  003356  012777  003076  176044          MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
   (1)  003364  012777  000340  176040          MOV     #340,@GRIVSB
   693                                  ;;*****************************************************************
   (3)                                  ;*TEST 1        TEST FOR NO BUS ERRORS
   (3)                                  ;;*****************************************************************
   (2)  003372  000004                  TST1:   SCOPE
   694  003374  005077  176010                  CLR     @GRSTAT
   695  003400  005077  176006                  CLR     @GRDAI
   696  003404  005077  176004                  CLR     @GRDIO
   697
   698                                  ;;*****************************************************************
   (3)                                  ;*TEST 2        TEST THAT OUTPUT REG. CAN HOLD #-1
   (3)                                  ;;*****************************************************************
   (2)  003410  000004                  TST2:   SCOPE
   699  003412  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD EXPECTED
   700  003420  012777  177777  175766          MOV     #-1,@GRDIO              ;ALL ONES TO REGISTER
   701  003426  017737  175762  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
   702  003434  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
   703  003442  001401                          BEQ     TST3            ;;BR IF EQUAL
   704  003444  104003                          ERROR   3                       ;REG WILL NOT HOLD ONES
```

```
706                                     ;;*****************************************************************
(3)                                     ;*TEST 3        TEST THAT RESET CLEARS OUTPUT REG.
(3)                                     ;;*****************************************************************
(2)  003446  000004                     TST3:   SCOPE
(1)  003450  012737  000040  001160             MOV     #40,$TIMES      ;;DO 40 ITERATIONS
707  003456  005037  001124                     CLR     $GDDAT
708  003462  012777  177777  175724             MOV     #-1,@GRD10
709  003470  000005                             RESET                           ;SET DATA TO ALL ONES
710  003472  017737  175716  001126             MOV     @GRD10,$BDDAT           ;READ OUTPUT REG.
711  003500  001401                             BEQ     TST4            ;;BR IF EQUAL
712  003502  104003                             ERROR   3                       ;REG FAILED TO CLEAR
713
714                                     ;;*****************************************************************
(3)                                     ;*TEST 4        TEST THAT OUTPUT REG. CAN HOLD #52525
(3)                                     ;;*****************************************************************
(2)  003504  000004                     TST4:   SCOPE
715  003506  012737  052525  001124             MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
716  003514  012777  052525  175672             MOV     #52525,@GRD10
717  003522  017737  175666  001126             MOV     @GRD10,$BDDAT           ;READ OUTPUT REG.
718  003530  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
719  003536  001401                             BEQ     TST5            ;;BR IF EQUAL
720  003540  104003                             ERROR   3                       ;DATA NOT=52525
721
722                                     ;;*****************************************************************
(3)                                     ;*TEST 5        TEST THAT OUTPUT REG. CAN HOLD #125252
(3)                                     ;;*****************************************************************
(2)  003542  000004                     TST5:   SCOPE
723  003544  012737  125252  001124             MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
724  003552  012777  125252  175634             MOV     #125252,@GRD10
725  003560  017737  175630  001126             MOV     @GRD10,$BDDAT           ;READ OUTPUT REG.
726  003566  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
727  003574  001401                             BEQ     TST6            ;;BR IF EQUAL
728  003576  104003                             ERROR   3                       ;DATA NOT=125252
729
730                                     ;;*****************************************************************
(3)                                     ;*TEST 6        TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
(3)                                     ;;*****************************************************************
(2)  003600  000004                     TST6:   SCOPE
(1)  003602  012737  000004  001160             MOV     #4,$TIMES       ;;DO 4 ITERATIONS
731  003610  012737  003622  001110             MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
732  003616  005037  001124                     CLR     $GDDAT                  ;CLEAR PATTERN
733  003622  013777  001124  175564  2$:        MOV     $GDDAT,@GRD10           ;LOAD THE OUTPUT REG.
734  003630  017737  175560  001126             MOV     @GRD10,$BDDAT           ;READ THE REG.
735  003636  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
736  003644  001401                             BEQ     1$              ;;BR IF EQUAL
737  003646  104003                             ERROR   3                       ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
738  003650  005237  001124          1$:        INC     $GDDAT                  ;UPDATE THE PATTERN
739  003654  001362                             BNE     2$                      ;TRY AGAIN
```

```
                                        ;;*******************************************************
                                        ;*TEST 7         FLOAT A 1 ACROSS THE OUTPUT REGISTER
                                        ;;*******************************************************
                                TST7:   SCOPE
003656  000004                          
003660  012737  003674  001110          MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
003666  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE

003674  005077  175514                  1$:     CLR     @GRDIO          ;CLEAR OUTPUT
003700  053777  001124  175506          BIS     $GDDAT,@GRDIO           ;SET THAT BIT
003706  017737  175502  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003714  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;TEST RESULTS
003722  001401                          BEQ     2$                      ;BR IF EQUAL ?
003724  104003                          ERROR   3

003726  006337  001124                  2$:     ASL     $GDDAT          ;SHIFT EXPECTED DATA
003732  001360                          BNE     1$                      ;BR UNTIL DONE
                                        ;;*******************************************************
                                        ;*TEST 10        FLOAT A 0 ACROSS THE OUTPUT REGISTER
                                        ;;*******************************************************
                                TST10:  SCOPE
003734  000004                          
003736  012737  003752  001110          MOV     #1$,$LPERR              ;LOAD SCOPE ERROR RETURN
003744  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE

003752  012777  177777  175434  1$:     MOV     #-1,@GRDIO              ;LOAD OUTPUT TO A ONE
003760  043777  001124  175426          BIC     $GDDAT,@GRDIO           ;CLEAR A BIT
003766  017737  175422  001126          MOV     @GRDIO,$BDDAT           ;READ OUTPUT REG.
003774  005137  001126                  COM     $BDDAT                  ;COMPLEMENT IT
004000  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;EQUAL ?
004006  001401                          BEQ     2$                      ;BR IF EQUAL
004010  104003                          ERROR   3

004012  006337  001124                  2$:     ASL     $GDDAT          ;SHIFT LEFT
004016  001355                          BNE     1$                      ;BRANCH UNTIL DONE

                                        ;;*******************************************************
                                        ;*TEST 11        TEST FOR SLOW OUTPUT GATES WITH #125252
                                        ;;*******************************************************
                                TST11:  SCOPE
004020  000004                          
004022  012737  125252  001124          MOV     #125252,$GDDAT          ;LOAD EXPECTED VALUE
004030  013777  001124  175356          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
004036  005177  175352                  COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1) 004042  005177  175346              COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1) 004046  005177  175342              COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1) 004052  005177  175336              COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1) 004056  005177  175332              COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1) 004062  005177  175326              COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1) 004066  005177  175322              COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
(1) 004072  005177  175316              COM     @GRDIO                  ;COMPLEMENT OUTPUT REG.
004076  017737  175312  001126          MOV     @GRDIO,$BDDAT           ;RF D OUTPUT REG.
004104  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;' ST REGISTER
004112  001401                          BEQ     TST12           ;;BR IF CORRECT
004114  104003                          ERROR   3
```

```
                                        ;;*****************************************************************
                                        ;*TEST 12       TEST FOR SLOW OUTPUT GATES WITH #52525
                                        ;;*****************************************************************
(2) 004116 000004                       TST12:  SCOPE
781 004120 012737 052525 001124                 MOV     #52525,$GDDAT           ;LOAD EXPECTED VALUE
782 004126 013777 001124 175260                 MOV     $GDDAT,@GRD10           ;LOAD OUTPUT REG.
785 004134 005177 175254                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
(1) 004140 005177 175250                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
(1) 004144 005177 175244                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
(1) 004150 005177 175240                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
(1) 004154 005177 175234                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
(1) 004160 005177 175230                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
(1) 004164 005177 175224                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
(1) 004170 005177 175220                        COM     @GRD10                  ;COMPLEMENT OUTPUT REG.
786 004174 017737 175214 001126                 MOV     @GRD10,$BDDAT           ;READ OUTPUT REG.
787 004202 023737 001124 001126                 CMP     $GDDAT,$BDDAT           ;TEST PATTERN
788 004210 001401                               BEQ     TST13           ;;BR IF EQUAL
789 004212 104003                               ERROR   3                       ;OUTPUT REGISTER IN ERROR
                                        ;;*****************************************************************
                                        ;*TEST 13       TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN
                                        ;;*****************************************************************
(2) 004214 000004                       TST13:  SCOPE
(1) 004216 012737 000400 001160                 MOV     #400,$TIMES     ;;DO 400 ITERATIONS
792 004224 012737 004246 001110                 MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
793 004232 012737 052400 001124                 MOV     #52400,$GDDAT           ;LOAD EXPECTED DATA
794 004240 013777 001124 175146                 MOV     $GDDAT,@GRD10           ;LOAD OUTPUT REG.
795 004246 113777 001124 175140 2$:             MOVB    $GDDAT,@GRD10           ;LOAD THE OUTPUT
796 004254 017737 175134 001126                 MOV     @GRD10,$BDDAT           ;READ OUTPUT REG.
797 004262 023737 001124 001126                 CMP     $GDDAT,$BDDAT
798 004270 001401                               BEQ     1$                      ;BRANCH IF EQUAL
799 004272 104003                               ERROR   3                       ;ERROR, OUTPUT REG. IN ERROR
800 004274 105237 001124                1$:     INCB    $GDDAT                  ;UPDATE PATTERN
801 004300 001362                               BNE     2$
802
803                                     ;;*****************************************************************
                                        ;*TEST 14       TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN
                                        ;;*****************************************************************
(2) 004302 000004                       TST14:  SCOPE
(1) 004304 012737 000400 001160                 MOV     #400,$TIMES     ;;DO 400 ITERATIONS
804 004312 012737 004334 001110                 MOV     #2$,$LPERR              ;LOAD SCOPE ERROR RETURN
805 004320 012737 000252 001124                 MOV     #252,$GDDAT             ;LOAD EXPECTED
806 004326 013777 001124 175060                 MOV     $GDDAT,@GRD10           ;LOAD OUTPUT REG.
807 004334 113777 001125 175054 2$:             MOVB    $GDDAT+1,@GRBH10        ;LOAD THE HIGH BYTE
808 004342 017737 175046 001126                 MOV     @GRD10,$BDDAT           ;READ  OUTPUT REG.
809 004350 023737 001124 001126                 CMP     $GDDAT,$BDDAT
810 004356 001401                               BEQ     1$                      ;BRANCH IF EQUAL
811 004360 104003                               ERROR   3                       ;ERROR, OUTPUT REG. IN ERROR
812 004362 105237 001125                1$:     INCB    $GDDAT+1
813 004366 001362                               BNE     2$

                                        \
```

```
 815
 (3)                                    ;;*****************************************************************
 (3)                                    ;*TEST 15       TEST OUTPUT DATA ACCEPT FLAG
 (3)                                    ;;*****************************************************************
 (2) 004370  000004                     TST15:  SCOPE
 816 004372  005077  175016                     CLR     @GRDIO
 817 004376  012777  177777  175006         MOV     #-1,@GRDAI
 818 004404  012737  100000  001124         MOV     #BIT15,$GDDAT           ;LOAD EXPECTED
 819 004412  012777  100000  174770         MOV     #BIT15,@GRSTAT          ;SET BIT 15
 820 004420  017737  174764  001126         MOV     @GRSTAT,$BDDAT          ;READ STATUS
 821 004426  033737  001124  001126         BIT     $GDDAT,$BDDAT
 822 004434  001001                        BNE     TST16           ;;BR IF SET
 823 004436  104001                        ERROR   1                       ;ERROR, BIT 15 FAILED TO SET
 824
 825                                    ;;*****************************************************************
 (3)                                    ;*TEST 16       TEST OUTPUT INTERRUPT ENABLE
 (3)                                    ;;*****************************************************************
 (2) 004440  000004                     TST16:  SCOPE
 826 004442  005077  174742                     CLR     @GRSTAT                 ;CLEAR STATUS
 827 004446  012737  040000  001124         MOV     #BIT14,$GDDAT           ;LOAD EXPECTED
 828 004454  012777  040000  174726         MOV     #BIT14.@GRSTAT          ;LOAD BIT 14
 829 004462  017737  174722  001126         MOV     @GRSTAT,$BDDAT          ;READ STATUS
 830 004470  033737  001124  001126         BIT     $GDDAT,$BDDAT
 831 004476  001001                        BNE     TST17           ;;BR IF SET
 832 004500  104001                        ERROR   1                       ;ERROR BIT 14 FAILED TO SET
 833
 834                                    ;;*****************************************************************
 (3)                                    ;*TEST 17       TEST INPUT DATA READY FLAG
 (3)                                    ;;*****************************************************************
 (2) 004502  000004                     TST17:  SCOPE
 835 004504  012737  000200  001124         MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
 836 004512  012777  000200  174670         MOV     #BIT7,@GRSTAT           ;SET BIT 7
 837 004520  017737  174664  001126         MOV     @GRSTAT,$BDDAT          ;READ RESULT
 838 004526  023737  001124  001126         CMP     $GDDAT,$BDDAT           ;COMPARE
 839 004534  001401                        BEQ     TST20           ;;BR IF EQUAL
 840 004536  104001                        ERROR   1                       ;ERROR, BIT 7 FAILED TO SET
 841
 842                                    ;;*****************************************************************
 (3)                                    ;*TEST 20       TEST INPUT INTERRUPT ENABLE
 (3)                                    ;;*****************************************************************
 (2) 004540  000004                     TST20:  SCOPE
 843 004542  005077  174642                     CLR     @GRSTAT                 ;CLEAR STATUS
 844 004546  012737  000100  001124         MOV     #BIT6,$GDDAT            ;LOAD EXPECTED
 845 004554  012777  000100  174626         MOV     #BIT6,@GRSTAT           ;SET BIT 6
 846 004562  017737  174622  001126         MOV     @GRSTAT,$BDDAT          ;READ RESULT
 847 004570  023737  001124  001126         CMP     $GDDAT,$BDDAT           ;COMPARE
 848 004576  001401                        BEQ     TST21           ;;BR IF EQUAL
 849 004600  104001                        ERROR   1                       ;ERROR, BIT 6 FAILED TO SET
 850
```

```
852
(3)
(3)
(2)  004602  000004
(1)  004604  012737  000040  001160          TST21:  SCOPE
                                                      MOV     #40,$TIMES      ;;DO 40 ITERATIONS
853  004612  005077  174572                           CLR     @GRSTAT                 ;CLEAR STATUS
854  004616  005037  001124                           CLR     $GDDAT                  ;LOAD EXPECTED
855  004622  012777  140300  174560                   MOV     #BIT15!BIT14!BIT7!BIT6,@GRSTAT
856  004630  000005                                    RESET
857  004632  017737  174552  001126                   MOV     @GRSTAT,$BDDAT          ;READ RESULT
858  004640  001401                                    BEQ     TST22           ;;BR IF EQUAL
859  004642  104001                                    ERROR   1                       ;ERROR, RESET FAILED TO CLEAR DIGITAL
860                                                                                    ;STATUS REGISTER
861
865
869
(3)
(3)
(2)  004644  000004
870  004646  032777  002000  174264          TST22:  SCOPE
                                                      BIT     #BIT10,@SWR             ;TEST SWITCH BIT
871  004654  001405                                    BEQ     1$                      ;BRANCH IF DOWN
872  004656  112737  000036  001102                   MOVB    #36,$TSTNM              ;LOAD NEW TEST NUMBER
873  004664  000137  007200                           JMP     DRT21                   ;BYPASS SOME TEST USING THE EXTERNAL CABLE
874  004670                                    1$:
(1)  004670  005077  174520                           CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
875  004674  012777  177777  174510                   MOV     #-1,@GRDAI              ;CLEAR INPUT
876  004702  005037  001124                           CLR     $GDDAT                  ;CLEAR EXPECTED
877  004706  012777  000000  174500                   MOV     #0,@GRDIO               ;LOAD THE OUTPUT
878  004714  017737  174472  001126                   MOV     @GRDAI,$BDDAT           ;READ THE INPUT
879  004722  043737  001444  001126                   BIC     ODDJMP,$BDDAT           ;MASK
880  004730  023737  001124  001126                   CMP     $GDDAT,$BDDAT           ;COMPARE
881  004736  001401                                    BEQ     TST23           ;;BR IF EQUAL
882  004740  104002                                    ERROR   2                       ;ERROR, INPUT DID NOT EQUAL THE OUTPUT REG.
883
884
(3)
(3)
(2)  004742  000004
885  004744  005077  174444                  TST23:  SCOPE
                                                      CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
886  004750  012777  177777  174434                   MOV     #-1,@GRDAI              ;CLEAR INPUT
887  004756  012737  177777  001124                   MOV     #-1,$GDDAT              ;LOAD EXPECTED
888  004764  043737  001444  001124                   BIC     ODDJMP,$GDDAT           ;MASK
889  004772  013777  001124  174414                   MOV     $GDDAT,@GRDIO           ;LOAD THE OUTPUT
890  005000  017737  174406  001126                   MOV     @GRDAI,$BDDAT           ;READ INPUT
891  005006  043737  001444  001126                   BIC     ODDJMP,$BDDAT           ;MASK
892  005014  023737  001124  001126                   CMP     $GDDAT,$BDDAT           ;COMPARE
893  005022  001401                                    BEQ     TST24           ;;BR IF EQUAL
894  005024  104002                                    ERROR   2                       ;ERROR, INPUT DID NOT EQUAL THE OUTPUT
895
```

```
                                        ;;*****************************************************************
                                        ;*TEST 24        TEST INPUT WITH #52525
                                        ;;*****************************************************************
005026  000004                          TST24:  SCOPE
005030  005077  174360                          CLR     @GRDIO                          ;CLEAR OUTPUT REGISTER
005034  012777  177777  174350                  MOV     #-1,@GRDAI                      ;CLEAR INPUT
005042  012737  052525  001124                  MOV     #52525,$GDDAT                   ;LOAD EXPECTED
005050  043737  001444  001124                  BIC     ODDJMP,$GDDAT                   ;MASK
005056  013777  001124  174330                  MOV     $GDDAT,@GRDIO                   ;LOAD THE OUTPUT
005064  017737  174322  001126                  MOV     @GRDAI,$BDDAT                   ;READ INPUT
005072  043737  001444  001126                  BIC     ODDJMP,$BDDAT                   ;MASK
005100  023737  001124  001126                  CMP     $GDDAT,$BDDAT                   ;COMPARE
005106  001401                                  BEQ     TST25                   ;;BR IF EQUAL
005110  104002                                  ERROR   2                               ;ERROR, INPUT DID NOT EQUAL OUTPUT

                                        ;;*****************************************************************
                                        ;*TEST 25        TEST INPUT WITH #125252
                                        ;;*****************************************************************
005112  000004                          TST25:  SCOPE
005114  005077  174274                          CLR     @GRDIO                          ;CLEAR OUTPUT REGISTER
005120  012777  177777  174264                  MOV     #-1,@GRDAI                      ;CLEAR INPUT
005126  012737  125252  001124                  MOV     #125252,$GDDAT                  ;LOAD EXPECTED
005134  043737  001444  001124                  BIC     ODDJMP,$GDDAT                   ;MASK
005142  013777  001124  174244                  MOV     $GDDAT,@GRDIO                   ;LOAD THE OUTPUT
005150  017737  174236  001126                  MOV     @GRDAI,$BDDAT                   ;READ INPUT
005156  043737  001444  001126                  BIC     ODDJMP,$BDDAT                   ;MASK
005164  023737  001124  001126                  CMP     $GDDAT,$BDDAT                   ;COMPARE
005172  001401                                  BEQ     TST26                   ;;BR IF EQUAL
005174  104002                                  ERROR   2                               ;ERROR, INPUT DID NOT EQUAL OUTPUT
```

```
                                        ;;*****************************************************************
                                        ;*TEST 26       TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
                                        ;;*****************************************************************
005176  000004                  TST26:  SCOPE
005200  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
005206  005737  001444                  TST     ODDJMP                  ;TEST IF ANY ODD JUMPERS
005212  001461                          BEQ     TST27           ;;BR IF NONE
005214  012737  010000  001124          MOV     #BIT12,$GDDAT           ;LOAD TEST BIT
005222  033737  001444  001124  1$:     BIT     ODDJMP,$GDDAT           ;TEST IF ODD JUMPER BIT
005230  001447                          BEQ     2$                      ;BR IF NOT
005232  033737  001420  001124          BIT     NOTLCH,$GDDAT           ;TEST IF LATCHING INPUT BIT
005240  001043                          BNE     2$                      ;BR IF NOT
005242  013777  001124  174144          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
005250  012777  177777  174134          MOV     #-1,@GRDAI              ;CLEAR INPUT
005256  043777  001124  174130          BIC     $GDDAT,@GRDIO           ;CLEAR OUTPUT BIT
005264  017737  174122  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT REG
005272  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005300  001401                          BEQ     3$                      ;BR IF EQUAL
005302  104002                          ERROR   2                       ;ERROR, NEG. INPUT OR NEG. TRANSITION
                                                                        . INPUT BIT FAILED TO SET INPUT REGISTER
005304  033737  001124  001446  3$:     BIT     $GDDAT,MINSIN           ;TEST IF NEG. INPUT BIT
005312  001016                          BNE     2$                      ;BR IF
005314  012777  177777  174070          MOV     #-1,@GRDAI              ;CLEAR INPUT
005322  053777  001124  174064          BIS     $GDDAT,@GRDIO           ;LOAD OUTPUT
005330  017737  174056  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT
005336  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005344  001401                          BEQ     2$                      ;BR IF EQUAL
005346  104002                          ERROR   2                       ;ERROR, POSITIVE INPUT TRANSITION
                                                                        ;LOGIC FAILED TO SET INPUT REGISTER BIT
005350  006137  001124          2$:     ROL     $GDDAT                  ;CHANGE DATA PATTERN
005354  103322                          BCC     1$                      ;BR IF MORE DATA
```

```
                        ;;*****************************************************************
                        ;*TEST 27       FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
                        ;;*****************************************************************
(2) 005356 000004       TST27:  SCOPE
952 005360 012737 005406 001110         MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
953
954 005366 012737 000001 001440         MOV     #BIT0,BRLEV2            ;LOAD EXPECTED
955 005374 013737 001420 001442         MOV     NOTLCH,BRLEV3           ;GET NON-LATCH
956 005402 005137 001442                COM     BRLEV3                  ;COMPLEMENT
957 005406 013737 001440 001124 2$:     MOV     BRLEV2,$GDDAT           ;LOAD GOOD
958 005414 033737 001124 001444         BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
959 005422 001042                       BNE     1$                      ;BYPASS IF ODD JUMPER
960 005424 033737 001124 001420         BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCH
961 005432 001436                       BEQ     1$                      ;BR IF LATCHING
962
963 005434 013777 001124 173752         MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
964 005442 017737 173744 001126         MOV     @GRDAI,$BDDAT           ;READ INPUT
965 005450 043737 001442 001126         BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
966 005456 023737 001124 001126         CMP     $GDDAT,$BDDAT           ;COMPARE
967 005464 001401                       BEQ     3$              ;;BR IF EQUAL
968 005466 104002                       ERROR   2                       ;INPUT REGISTER IN ERROR
                                                                        ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
970
971                     ;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
972
973 005470 043777 001124 173716 3$:      BIC     $GDDAT,@GRDIO           ;CLEAR OUTPUT BIT
974 005476 017737 173710 001126         MOV     @GRDAI,$BDDAT           ;READ INPUT
975 005504 043737 001442 001126         BIC     BRLEV3,$BDDAT           ;MASK TO LATCH BITS
976 005512 005037 001124                CLR     $GDDAT                  ;CLEAR EXPECTED
977 005516 033737 001440 001126         BIT     BRLEV2,$BDDAT           ;TEST FOR BIT
978 005524 001401                       BEQ     1$              ;;BR IF CLEARED
979 005526 104002                       ERROR   2                       ;INPUT BIT LATCHED IN ERROR
                                                                        ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
981
982 005530 006337 001440        1$:     ASL     BRLEV2                  ;CHANGE PATTERN
983 005534 001324                       BNE     2$                      ;BR UNTIL DONE
984
```

```
                                        ;;*****************************************************************
                                        ;*TEST 30        FLOAT A 1 ACROSS LATCHING INPUT BITS
                                        ;;*****************************************************************
005536  000004                  TST30:  SCOPE
005540  012737  005566  001110          MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
005546  012737  000001  001124          MOV     #BIT0,$GDDAT            ;LOAD EXPECTED VALUE
005554  005077  173634                  CLR     @GRDIO                  ;CLEAR OUTPUT REG
005560  012777  177777  173624          MOV     #-1,@GRDAI              ;CLEAR INPUT

005566  033737  001124  001420  2$:     BIT     $GDDAT,NOTLCH           ;TEST FOR NON-LATCHING
005574  001021                          BNE     1$                      ;BR IF NON-LATCH
005576  033737  001124  001444          BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
005604  001015                          BNE     1$                      ;BYPASS IF ODD JUMPER

005606  013777  001124  173600          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
005614  005077  173574                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
005620  017737  173566  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT REG.
005626  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005634  001401                          BEQ     1$              ;;BR IF EQUAL
005636  104002                          ERROR   2                       ;INPUT REGISTER FAILED TO LATCH DATA

005640  053777  001124  173544  1$:     BIS     $GDDAT,@GRDAI           ;CLEAR INPUT BIT
005646  006337  001124                  ASL     $GDDAT                  ;CHANGE PATTERN
005652  001345                          BNE     2$                      ;BR UNTIL DONE

                                        ;;*****************************************************************
                                        ;*TEST 31        FLOAT A 0 ACROSS LATCHING INPUT BITS
                                        ;;*****************************************************************
005654  000004                  TST31:  SCOPE
005656  012737  005704  001110          MOV     #2$,$LPERR              ;LOAD ERROR SCOPE RETURN
005664  012737  000001  001442          MOV     #BIT0,BRLEV3            ;LOAD EXPECTED
005672  005077  173516                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
005676  012777  177777  173506          MOV     #-1,@GRDAI              ;CLEAR INPUT

005704  033737  001442  001420  2$:     BIT     BRLEV3,NOTLCH           ;TEST FOR LATCHING
005712  001032                          BNE     1$                      ;BR IF NOT
005714  033737  001124  001444          BIT     $GDDAT,ODDJMP           ;TEST IF ODD JUMPER
005722  001026                          BNE     1$                      ;BYPASS IF ODD JUMPER BIT

005724  012737  177777  001124          MOV     #-1,$GDDAT              ;LOAD
005732  043737  001420  001124          BIC     NOTLCH,$GDDAT
005740  043737  001442  001124          BIC     BRLEV3,$GDDAT           ;MAKE BRLEV3
005746  013777  001124  173440          MOV     $GDDAT,@GRDIO           ;LOAD OUTPUT
005754  005077  173434                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER

005760  017737  173426  001126          MOV     @GRDAI,$BDDAT           ;READ INPUT
005766  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
005774  001401                          BEQ     1$              ;;BR IF EQUAL
005776  104002                          ERROR   2                       ;INPUT REGISTER FAILED TO LATCH DATA

006000  053777  001442  173404  1$:     BIS     BRLEV3,@GRDAI           ;CLEAR INPUT BIT
006006  006337  001442                  ASL     BRLEV3                  ;CHANGE PATTERN
006012  001334                          BNE     2$                      ;BR UNTIL DONE
```

```
1034                                    ;;*****************************************************************
 (3)                                    ;*TEST 32       TEST FOR SLOW INPUT GATES WITH #125252
 (3)                                    ;;*****************************************************************
 (2)  006014  000004                    TST32:  SCOPE
1035
1036  006016  012737  125252  001124            MOV     #125252,$GDDAT          ;LOAD EXPECTED
1037  006024  043737  001420  001124            BIC     NOTLCH,$GDDAT           ;CONVERT
1038  006032  043737  001444  001124            BIC     ODDJMP,$GDDAT           ;MASK
1039  006040  013700  001124                    MOV     $GDDAT,R0               ;LOAD PATTERN
1040  006044  005077  173344                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
1041  006050  012777  177777  173334          MOV     #-1,@GRDAI              ;CLEAR INPUT
1042
1049  006056  010077  173332                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006062  005077  173326                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006066  017701  173320                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006072  050177  173314                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006076  005100                            COM     R0
 (1)  006100  010077  173310                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006104  005077  173304                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006110  017701  173276                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006114  050177  173272                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006120  005100                            COM     R0
 (1)  006122  010077  173266                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006126  005077  173262                    CLR     @GPDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006132  017701  173254                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006136  050177  173250                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006142  005100                            COM     R0
 (1)  006144  010077  173244                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006150  005077  173240                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006154  017701  173232                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006160  050177  173226                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006164  005100                            COM     R0
 (1)  006166  010077  173222                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006172  005077  173216                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006176  017701  173210                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006202  050177  173204                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006206  005100                            COM     R0
 (1)  006210  010077  173200                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006214  005077  173174                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006220  017701  173166                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006224  050177  173162                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006230  005100                            COM     R0
 (1)  006232  010077  173156                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006236  005077  173152                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006242  017701  173144                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006246  050177  173140                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006252  005100                            COM     R0
 (1)  006254  010077  173134                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006260  005077  173130                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006264  017701  173122                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006270  050177  173116                    BIS     R1,@GRDAI               ;CLEAR INPUT
 (1)  006274  005100                            COM     R0
 (1)  006276  010077  173112                    MOV     R0,@GRDIO               ;LOAD OUTPUT
 (2)  006302  005077  173106                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 (1)  006306  017701  173100                    MOV     @GRDAI,R1               ;READ INPUT
 (1)  006312  050177  173074                    BIS     R1,@GRDAI               ;CLEAR INPUT
```

```
   (1)  006316  005100                          COM     R0
   (1)  006320  010077  173070                 MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006324  005077  173064                 CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006330  017701  173056                 MOV     @GRDAI,R1               ;READ INPUT
   (1)  006334  050177  173052                 BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006340  005100                          COM     R0
   (1)  006342  010077  173046                 MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006346  005077  173042                 CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006352  017701  173034                 MOV     @GRDAI,R1               ;READ INPUT
   (1)  006356  050177  173030                 BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006362  005100                          COM     R0
 1050
 1051   006364  010137  001126                 MOV     R1,$BDDAT               ;LOAD READ
 1052   006370  000240                          NOP
 1053   006372  000240                          NOP
 1054   006374  000240                          NOP
 1055   006376  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMPARE
 1056   006404  001401                          BEQ     TST33           ;;BR IF EQUAL
 1057   006406  104002                          ERROR   2                       ;INPUT GATE SLOW
 1058
 1059                                  ;;*****************************************************************
   (3)                                 ;*TEST 33       TEST FOR SLOW INPUT GATES WITH #52525
   (3)                                 ;;*****************************************************************
   (2)  006410  000004                  TST33:  SCOPE
 1060
 1061   006412  012737  052525  001124          MOV     #52525,$GDDAT           ;SETUP EXPECTED
 1062   006420  043737  001444  001124          BIC     ODDJMP,$GDDAT           ;MASK ODD JUMPER BITS
 1063   006426  043737  001420  001124          BIC     NOTLCH,$GDDAT           ;CONVERT
 1064   006434  013700  001124                  MOV     $GDDAT,R0
 1065   006440  005077  172750                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 1066   006444  012777  177777  172740          MOV     #-1,@GRDAI              ;CLEAR INPUT
 1067
 1074   006452  010077  172736                  MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006456  005077  172732                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006462  017701  172724                  MOV     @GRDAI,R1               ;READ INPUT
   (1)  006466  050177  172720                  BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006472  005100                          COM     R0
   (1)  006474  010077  172714                  MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006500  005077  172710                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006504  017701  172702                  MOV     @GRDAI,R1               ;READ INPUT
   (1)  006510  050177  172676                  BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006514  005100                          COM     R0
   (1)  006516  010077  172672                  MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006522  005077  172666                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006526  017701  172660                  MOV     @GRDAI,R1               ;READ INPUT
   (1)  006532  050177  172654                  BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006536  005100                          COM     R0
   (1)  006540  010077  172650                  MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006544  005077  172644                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006550  017701  172636                  MOV     @GRDAI,R1               ;READ INPUT
   (1)  006554  050177  172632                  BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006560  005100                          COM     R0
   (1)  006562  010077  172626                  MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006566  005077  172622                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006572  017701  172614                  MOV     @GRDAI,R1               ;READ INPUT
   (1)  006576  050177  172610                  BIS     R1,@GRDAI               ;CLEAR INPUT
```

```
   (1)  006602  005100                        COM     R0
   (1)  006604  010077  172604                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006610  005077  172600                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006614  017701  172572                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006620  050177  172566                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006624  005100                        COM     R0
   (1)  006626  010077  172562                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006632  005077  172556                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006636  017701  172550                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006642  050177  172544                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006646  005100                        COM     R0
   (1)  006650  010077  172540                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006654  005077  172534                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006660  017701  172526                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006664  050177  172522                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006670  005100                        COM     R0
   (1)  006672  010077  172516                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006676  005077  172512                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006702  017701  172504                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006706  050177  172500                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006712  005100                        COM     R0
   (1)  006714  010077  172474                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006720  005077  172470                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006724  017701  172462                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006730  050177  172456                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006734  005100                        COM     R0
   (1)  006736  010077  172452                MOV     R0,@GRDIO               ;LOAD OUTPUT
   (2)  006742  005077  172446                CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
   (1)  006746  017701  172440                MOV     @GRDAI,R1               ;READ INPUT
   (1)  006752  050177  172434                BIS     R1,@GRDAI               ;CLEAR INPUT
   (1)  006756  005100                        COM     R0
1075
1076    006760  010137  001126                MOV     R1,$BDDAT               ;LOAD VALUE READ
1077    006764  000240                        NOP
1078    006766  000240                        NOP
1079    006770  000240                        NOP
1080    006772  023737  001124  001126        CMP     $GDDAT,$BDDAT           ;COMPARE
1081    007000  001401                        BEQ     TST34           ;;BR IF EQUAL
1082    007002  104002                        ERROR   2
```

```
                                        ;;*****************************************************************
                                        ;*TEST 34        TEST THAT RESET CLEARS INPUT REGISTER BITS
                                        ;;*****************************************************************
007004  000004                          TST34:  SCOPE
007006  012737  000040  001160                  MOV     #40,$TIMES          ;;DO 40 ITERATIONS
007014  005077  172374                          CLR     @GRDIO                      ;CLEAR OUTPUT REGISTER
007020  012777  177777  172364                  MOV     #-1,@GRDAI                  ;CLEAR INPUT
007026  012777  177777  172360                  MOV     #-1,@GRDIO                  ;LOAD OUTPUT
007034  005037  001124                          CLR     $GDDAT                      ;LOAD EXPECTED
007040  000005                                  RESET
007042  017737  172344  001126                  MOV     @GRDAI,$BDDAT               ;READ INPUT REG.
007050  043737  001444  001126                  BIC     ODDJMP,$BDDAT               ;MASK ODD JUMPERS
007056  001401                                  BEQ     TST35               ;;BR IF ALL BITS CLEARED
007060  104002                                  ERROR   2                           ;INPUT REG. FAILED TO CLEAR UPON RESET INST.


                                        ;;*****************************************************************
                                        ;*TEST 35        TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
                                        ;;*****************************************************************
007062  000004                          TST35:  SCOPE
007064  012737  000200  001124                  MOV     #BIT7,$GDDAT                ;LOAD EXPECTED
007072  005077  172312                          CLR     @GRSTAT
007076  012777  000000  172310                  MOV     #0,@GRDIO                   ;OUTPUT 0
007104  022727  000000  000000                  CMP     #0,#0                       ;DELAY
007112  017737  172272  001126                  MOV     @GRSTAT,$BDDAT              ;READ RESULTS
007120  023737  001124  001126                  CMP     $GDDAT,$BDDAT               ;COMPARE
007126  001401                                  BEQ     TST36               ;;BR IF EQUAL
007130  104001                                  ERROR   1                           ;INPUT DATA READY FLAG FAILED TO SET

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

                                        ;;*****************************************************************
                                        ;*TEST 36        TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
                                        ;;*****************************************************************
007132  000004                          TST36:  SCOPE
007134  012737  100000  001124                  MOV     #BIT15,$GDDAT               ;LOAD EXPECTED
007142  005077  172242                          CLR     @GRSTAT
007146  012777  000000  172240                  MOV     #0,@GRDIO
007154  022727  000000  000000                  CMP     #0,#0
007162  017700  172224                          MOV     @GRDAI,R0                   ;READ INPUT
007166  017737  172216  001126                  MOV     @GRSTAT,$BDDAT              ;READ RESULTS
007174  100401                                  BMI     TST37               ;;BR IF SET
007176  104001                                  ERROR   1                           ;INPUT DATA READY FLAG FAILED TO SET
```

```
                                                DRT21:
                                                ;;*****************************************************************
                                                ;*TEST 37        TEST FOR UNEXPECTED INTERRUPT
                                                ;;*****************************************************************
                                                TST37:  SCOPE
                                                        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
                                                        RESET
                                                        CLR     @GRSTAT
                                                        CLR     PSW
                                                        MOV     #1$,@GRIVA              ;SET UP INTERRUPT INPUT VECTOR
                                                        MOV     #340,@GRIVSA
                                                        MOV     #2$,@GRIVB              ;SET UP INTERRUPT OUTPUT VECTOR
                                                        MOV     #340,@GRIVSB
                                                        MOV     #0,@GRDIO               ;OUTPUT
                                                        MOV     @GRDAI,R0               ;INPUT
                                                        NOP
                                                        NOP
                                                        NOP
                                                        NOP
                                                        CLR     @GRSTAT                 ;CLEAR STATUS
                                                        BR      TST40           ;;
                                                1$:     ERROR   6                       ;ERROR, DIGITAL INPUT INTERRUPTED
                                                        RTI
                                                2$:     ERROR   6                       ;ERROR, DIGITAL OUTPUT INTERRUPTED
                                                        RTI
                                                ;;*****************************************************************
                                                ;*TEST 40        TEST THAT THE INPUT CAN INTR. USING THE MAINT. BIT
                                                ;;*****************************************************************
                                                TST40:  SCOPE
                                                        CLR     @GRSTAT                 ;CLEAR STATUS
                                                        MOV     #1$,@GRIVA              ;LOAD RETURN VECTOR
                                                        MOV     #340,@GRIVSA            ;
                                                        MOV     #2$,@GRIVB              ;LOAD OUTPUT VECTOR
                                                        MOV     #340,@GRIVSB
                                                        BIS     #BIT5,@GRSTAT           ;MAINT. INT C
                                                        CLR     PSW                     ;LOWER PSW
                                                        NOP
                                                        NOP
                                                        ERROR   4                       ;ERROR, INPUT FAILED TO INTERRUPT
                                                        BR      TST41           ;;BR TO NEXT TEST
                                                ;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN NO INTERRUPT WILL OCCUR
                                                ; IF INTERRUPT OCCURS 'INT DONE C' FAILED TO CLEAR INT C FLOP

                                                1$:     MOV     #2$,@GRIVA              ;LOAD INPUT VECTOR
                                                        CMP     (SP)+,(SP)+             ;POP THE STACK *4
                                                        CLR     PSW                     ;LOWER PRIOR.
                                                        NOP
                                                        NOP
                                                        BR      TST41           ;;<NEXT TEST>
                                                2$:     CMP     (SP)+,(SP)+             ;POP THE STACK
                                                        ERROR   6                       ;UNEXPECTED INTERRUPT
                                                        CLR     PSW
```

```
                                        ;;*****************************************************************
                                        ;*TEST 41        TEST THAT THE INPUT INTR. CLEARS INT. ENABLE VIA MAINT. BIT
                                        ;;*****************************************************************
                                TST41:  SCOPE
                                        CLR     @GRSTAT                 ;CLEAR STATUS
                                        MOV     #1$,@GRIVA              ;LOAD RETURN VECTOR
                                        MOV     #340,@GRIVSA            ;
                                        MOV     #2$,@GRIVB              ;LOAD OUTPUT VECTOR
                                        MOV     #340,@GRIVSB
                                        MOV     #BIT6,@GRSTAT           ;LOAD INPUT INTR. ENABLE
                                        BIS     #BIT5,@GRSTAT           ;MAINT. INT C
                                        CLR     PSW                     ;LOWER PSW
                                        NOP
                                        NOP
                                        ERROR   4                       ;ERROR, INPUT FAILED TO INTERRUPT
                                1$:     MOV     #2$,@GRIVA              ;LOAD INPUT VECTOR
                                        CMP     (SP)+,(SP)+             ;POP THE STACK *4
                                        CLR     PSW                     ;LOWER PRIOR.
                                        CLR     $GDDAT                  ;CLEAR EXPECTED
                                        MOV     @GRSTAT,$BDDAT          ;READ STATUS
                                        BIT     #BIT6,$BDDAT            ;TEST BIT 6
                                        BEQ     TST42           ;;BR IF CLEARED
                                        ERROR   1                       ;INPUT INTR. FAILED TO CLEAR
                                        BR      TST42           ;;<NEXT TEST>
                                2$:     CMP     (SP)+,(SP)+             ,POP THE STACK
                                        ERROR   6                       ;UNEXPECTED INTERRUPT
                                        CLR     PSW
                                        ;;*****************************************************************
                                        ;*TEST 42        TEST THAT THE OUTPUT CAN INTR. USING THE MAINT. BIT
                                        ;;*****************************************************************
                                TST42:  SCOPE
                                        CLR     @GRSTAT                 ;CLEAR STATUS
                                        MOV     #2$,@GRIVA              ;LOAD INPUT VECTOR
                                        MOV     #340,@GRIVSA
                                        MOV     #1$,@GRIVB              ;LOAD OUTPUT VECTOR
                                        MOV     #340,@GRIVSB
                                        BIS     #BIT13,@GRSTAT          ;MAINT. INTERRUPT
                                        CLR     PSW                     ;LOWER PSW
                                        NOP
                                        NOP
                                        ERROR   5                       ;OUTPUT FAILED TO INTERRUPT
                                        BR      TST43           ;;BR TO NEXT TEST
                                ;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN, NO INTERRUPT WILL OCCUR
                                ; IF IT DOES, 'INT DONE D HIGH' FAILED TO CLEAR 'INT D' FLOP
                                1$:     MOV     #2$,@GRIVB              ;LOAD OUTPUT VECTOR
                                        CMP     (SP)+,(SP)+             ;POP THE STACK *4
                                        CLR     PSW
                                        NOP
                                        NOP
                                        BR      TST43           ;;<NEXT TEST>
                                2$:     CMP     (SP)+,(SP)+             ;POP THE STACK
                                        ERROR   6                       ;UNEXPECTED INTERRUPT
                                        CLR     PSW
```

```
 1212                                   ;;*****************************************************************
  (3)                                   ;*TEST 43       TEST FOR INTR. FROM DRA INPUT
  (3)                                   ;;*****************************************************************
  (2)  007674  000004                   TST43:  SCOPE
 1213  007676  000240                           NOP
 1214  007700  000240                           NOP
 1215  007702  032777  002000  171230           BIT     #BIT10,@SWR             ;TEST SWITCH
 1216  007710  001401                           BEQ     1$
 1217  007712  000463                           BR      TST44           ;;
 1218  007714  005077  171470           1$:     CLR     @GRSTAT
 1219  007720  012777  010020  171476           MOV     #2$,@GRIVA              ;IN CASE OF INTERRUPT
 1220  007726  012777  000340  171472           MOV     #340,@GRIVSA
 1221  007734  012777  010016  171466           MOV     #3$,@GRIVB
 1222  007742  012777  000340  171462           MOV     #340,@GRIVSB
 1223  007750  012777  000100  171432           MOV     #BIT6,@GRSTAT           ;ENABLE INPUT INTR.
 1224  007756  012777  000000  171430           MOV     #0,@GRDIO               ;OUTPUT
 1225  007764  017700  171422                   MOV     @GRDAI,R0               ;INPUT
 1226  007770  005037  177776                   CLR     PSW                     ;LOWER PSW
 1227  007774  000240                           NOP                             ;LET INTERRUPT OCCUR
 1228  007776  000240                           NOP
 1229  010000  000240                           NOP
 1230  010002  042777  000100  171400           BIC     #BIT6,@GRSTAT
 1231  010010  000240                           NOP
 1232  010012  104004                           ERROR   4                       ;ERROR, INPUT FAILED TO INTERRUPT
 1233  010014  000404                           BR      4$              ;;
 1234  010016  104006                   3$:     ERROR   6                       ;UNEXPECTED OUTPUT INTERRUPT
 1235  010020  022626                   2$:     CMP     (SP)+,(SP)+
 1236  010022  005037  177776                   CLR     PSW
 1237  010026  005077  171356           4$:     CLR     @GRSTAT                 ;CLEAR STATUS
 1238  010032  012777  003076  171364           MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
  (1)  010040  012777  000340  171360           MOV     #340,@GRIVSA
  (1)  010046  012777  003076  171354           MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
  (1)  010054  012777  000340  171350           MOV     #340,@GRIVSB
 1239
```

```
 1241                                   ;;*****************************************************************
  (3)                                   ;*TEST 44       TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
  (3)                                   ;;*****************************************************************
  (2) 010062  000004                    TST44:  SCOPE
 1242 010064  032777  002000  171046             BIT     #BIT10,@#WR             ;TEST CABLE SWITCH
 1243 010072  001067                      BNE     TST45           ;;BYPASS IF NO I/O CABLE
 1244
 1245 010074  012737  010110  001110             MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
 1246 010102  012737  000001  001442             MOV     #BIT0,BRLEV3            ;LOAD INTERRUPT BIT
 1247 010110  005037  001126          1$:        CLR     $BDDAT                  ;CLEAR BAD DATA
 1248 010114  012737  000200  001124             MOV     #BIT7,$GDDAT            ;LOAD GOOD DATA
 1249
 1250 010122  033737  001442  001422             BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
 1251 010130  001445                      BEQ     3$              ;;NO TRY NEXT BIT
 1252 010132  005077  171256                      CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 1253 010136  012777  177777  171246             MOV     #-1,@GRDAI              ;CLEAR INPUT
 1254 010144  005077  171240                      CLR     @GRSTAT                 ;CLEAR STATUS
 1255 010150  053777  001442  171236             BIS     BRLEV3,@GRDIO           ;LOAD OUTPUT
 1256 010156  043777  001442  171230             BIC     BRLEV3,@GRDIO           ;CLEAR OUTPUT
 1257 010164  053777  001442  171222             BIS     BRLEV3,@GRDIO           ;LOAD OUTPUT
 1258 010172  005077  171212                      CLR     @GRSTAT                 ;CLEAR FLAG FROM DATA READY
 1259                                                                             ;SHOULD REMAIN SET VIA DIRECT SET SIDE
 1260                                                                             ;IF INTERRUPT INPUT SWITCH IS ON
 1261 010176  105777  171206                      TSTB    @GRSTAT                 ;TEST READY BIT
 1262 010202  100401                      BMI     2$              ;;BR IF SET
 1263 010204  104010                      ERROR   10                              ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
 1264                                                                             ;?? DID OPERATOR GIVE CORRECT
 1265                                                                             ;INPUT INTERRUPT BITS ??
 1266
 1267 010206  043777  001442  171200  2$:        BIC     BRLEV3,@GRDIO           ;CLEAR OUTPUT
 1268 010214  053777  001442  171170             BIS     BRLEV3,@GRDAI           ;CLEAR INPUT
 1269 010222  005037  001124                      CLR     $GDDAT                  ;CLEAR EXPECTED
 1270 010226  005077  171156                      CLR     @GRSTAT                 ;CLEAR STATUS
 1271 010232  117737  171152  001126             MOVB    @GRSTAT,$BDDAT          ;READ STATUS
 1272 010240  100001                      BPL     3$              ;;BR IF CLEARED
 1273 010242  104001                      ERROR   1                               ;INPUT READY FAILED TO CLEAR
 1274
 1275 010244  006337  001442          3$:        ASL     BRLEV3                  ;CHANGE BIT
 1276 010250  001317                      BNE     1$                              ;BR IF NOT DONE
```

```
                                        ;;*****************************************************************
                                        ;*TEST 45       TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
                                        ;;*****************************************************************
010252  000004                  TST45:  SCOPE
010254  032777  002000  170656          BIT     #BIT10,@SWR             ;TEST CABLE SWITCH
010262  001050                          BNE     TST46           ;;BYPASS IF NO I/O CABLE
010264  012737  010300  001110          MOV     #1$,$LPERR              ;LOAD ERROR SCOPE RETURN
010272  012737  000001  001442          MOV     #BIT0,BRLEV3            ;LOAD NON-INTERRUPT BIT
010300  012737  000200  001126  1$:     MOV     #200,$BDDAT             ;LOAD BAD DATA
010306  005037  001124                  CLR     $GDDAT                  ;CLEAR GOOD DATA
010312  033737  001442  001422          BIT     BRLEV3,INTBIT           ;TEST IF THIS BIT WILL INTERRUPT
010320  001026                          BNE     3$              ;;YES SKIP AND TRY NEXT BIT
010322  005077  171066                  CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
010326  012777  177777  171056          MOV     #-1,@GRDAI              ;CLEAR INPUT
010334  005077  171050                  CLR     @GRSTAT                 ;CLEAR STATUS
010340  053777  001442  171046          BIS     BRLEV3,@GRDIO           ;LOAD OUTPUT
010346  043777  001442  171040          BIC     BRLEV3,@GRDIO           ;CLEAR OUTPUT
010354  053777  001442  171032          BIS     BRLEV3,@GRDIO           ;SET OUTPUT
010362  005077  171022                  CLR     @GRSTAT                 ;CLEAR FLAG FROM DATA READY
                                                                        ;SHOULD REMAIN SET VIA DIRECT SET SIDE
010366  105777  171016                  TSTB    @GRSTAT                 ;TEST READY BIT
010372  100001                          BPL     3$              ;;BR IF CLEAR
010374  104011                          ERROR   11                      ;INPUT NON-INTERRUPT BIT SET INPUT READY
                                                                        ;?? DID OPERATOR GIVE CORRECT
                                                                        ;INPUT INTERRUPT BITS ??
010376  006337  001442          3$:     ASL     BRLEV3                  ;CHANGE BIT
010402  001336                          BNE     1$                      ;BR IF NOT DONE
```

```
                                        ;;*****************************************************************
                                        ;*TEST 46        TEST FOR INTR. FROM DRA OUTPUT
                                        ;;*****************************************************************
                                TST46:  SCOPE
                                        NOP
                                        NOP
010412  032777  002000  170520          BIT     #BIT10,@SWR             ;TEST SWITCH
010420  001401                          BEQ     1$
010422  000461                          BR      TST47           ;;
010424  005077  170760          1$:     CLR     @GRSTAT
010430  012777  010524  170772          MOV     #2$,@GRIVB      ;IN CASE OF INTERRUPT
010436  012777  000340  170766          MOV     #340,@GRIVSB
010444  013777  001426  170752          MOV     GRIVSA,@GRIVA
010452  005077  170750                  CLR     @GRIVSA
010456  012777  040000  170724          MOV     #BIT14,@GRSTAT          ;ENABLE OUTPUT INTR.
010464  012777  000000  170722          MOV     #0,@GRDIO               ;OUTPUT
010472  017700  170714                  MOV     @GRDAI,R0               ;INPUT
010476  005037  177776                  CLR     PSW                     ;LOWER PSW
010502  000240                          NOP                             ;LET INTERRUPT OCCUR
010504  000240                          NOP
010506  000240                          NOP
010510  042777  040000  170672          BIC     #BIT14,@GRSTAT
010516  000240                          NOP
010520  104005                          ERROR   5                       ;ERROR, OUTPUT FAILED TO INTERRUPT
010522  000401                          BR      3$              ;;
010524  022626                  2$:     CMP     (SP)+,(SP)+
010526                          3$:
010526  012777  003076  170670          MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
010534  012777  000340  170664          MOV     #340,@GRIVSA
010542  012777  003076  170660          MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
010550  012777  000340  170654          MOV     #340,@GRIVSB
010556  005037  177776                  CLR     PSW
010562  005077  170622                  CLR     @GRSTAT                 ;CLEAR STATUS

                                        ;;*****************************************************************
                                        ;*TEST 47        PRE-INTERRUPT SETUP
                                        ;;*****************************************************************
010566  000004                  TST47:  SCOPE
010570  012737  000001  001160          MOV     #1,$TIMES       ;;DO 1 ITERATION
010576  000005                          RESET
010600  042737  177437  001434          BIC     #177437,DIOBRL
010606  001001                          BNE     1$
010610  104007                          ERROR   7                       ;BR LEVEL INDICATED FOR DIGITAL I/O WAS=0
010612  022737  000340  001434  1$:     CMP     #340,DIOBRL
010620  001001                          BNE     2$
010622  104007                          ERROR   7                       ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 7
010624  013737  001434  001436  2$:     MOV     DIOBRL,BRLEV1
010632  162737  000040  001436          SUB     #40,BRLEV1
010640  013737  001434  001440          MOV     DIOBRL,BRLEV2
```

```
                                        ;;*******************************************************************
                                        ;*TEST 50        TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT
                                        ;;*******************************************************************
1346                                    TST50:  SCOPE
 (3)
 (3)
 (2)  010646  000004
1347  010650  005077  170534                   CLR     @GRSTAT                 ;CLEAR ENABLES
1348  010654  012777  010726  170542           MOV     #1$,@GRIVA              ;SET UP VECTORS
1349  010662  013737  001436  177776           MOV     BRLEV1,PSW              ;CHANGE PSW
1350  010670  005737  001452                   TST     JUMPER                  ;TEST IF FACTORY MODE
1351  010674  001403                           BEQ     2$                      ;BR IF YES
1352  010676  012737  000200  177776           MOV     #200,PSW                ;LOAD BR LEVEL #5-1 IF LOC-BOX OR COULTER MODE
1353  010704  052777  000040  170476  2$:      BIS     #BIT5,@GRSTAT           ;GENERATE INPUT MAINT. INTERRUPT
1354  010712  000240                           NOP
1355  010714  000240                           NOP
1356  010716  000240                           NOP
1357  010720  000240                           NOP
1358  010722  104004                           ERROR   4                       ;ERROR, NO INTERRUPT FROM DIGITAL I/O INPUT
1359  010724  000401                           BR      3$              ;;BR TO NEXT TEST
1360  010726  022626                   1$:     CMP     (SP)+,(SP)+
1361  010730                           3$:
 (1)  010730  012777  003076  170465           MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
 (1)  010736  012777  000340  170462           MOV     #340,@GRIVSA
 (1)  010744  012777  003076  170456           MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
 (1)  010752  012777  000340  170452           MOV     #340,@GRIVSB
1362  010760  005037  177776                   CLR     PSW                     ;LOWER PSW
1363                                    ;;*******************************************************************
                                        ;*TEST 51        TEST FOR NO INTR. FROM DRA INPUT ON LEVEL INDICATED  VIA MAINT. INT
                                        ;;*******************************************************************
 (3)
 (3)
 (2)  010764  000004
1364  010766  005077  170416           TST51:  CLR     @GRSTAT                 ;CLEAR ENABLES
1365  010772  012777  011060  170424           MOV     #1$,@GRIVA              ;SET UP VECTORS
1366  011000  013737  001440  177776           MOV     BRLEV2,PSW              ;CHANGE PSW
1367  011006  005737  001452                   TST     JUMPER                  ;TEST IF FACTORY MODE
1368  011012  001403                           BEQ     4$                      ;BR IF YES
1369  011014  012737  000240  177776           MOV     #240,PSW                ;LOAD BR LEVEL #5 IF LOC-BOX OR COULTER MODE
1370  011022  052777  000040  170360  4$:      BIS     #BIT5,@GRSTAT           ;GENERATE INPUT MAINT. INTERRUPT
1371  011030  000240                           NOP
1372  011032  000240                           NOP
1373  011034  000240                           NOP
1374                                    ;SUB-TEST, NOW LOWER THE PSW AND ALLOW THE INTERRUPT
1375  011036  012777  011072  170360           MOV     #2$,@GRIVA              ;RESET VECTOR
1376  011044  005037  177776                   CLR     PSW                     ;LOWER PSW
1377  011050  000240                           NOP
1378  011052  000240                           NOP
1379  011054  000240                           NOP
1380  011056  000403                           BR      3$                      ;ERROR
1381  011060  022626                   1$:     CMP     (SP)+,(SP)+
1382  011062  005037  177776                   CLR     PSW
1383  011066  104006                   3$:     ERROR   6                       ;UNEXPECTED INTERRUPT <IS BR LEVEL PLUG CORRECT>
1384  011070  000401                           BR      5$              ;;
1385  011072  022626                   2$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1386  011074                           5$:
 (1)  011074  012777  003076  170322           MOV     #UNEXPT,@GRIVA          ;RESET INPUT VECTOR
 (1)  011102  012777  000340  170316           MOV     #340,@GRIVSA
 (1)  011110  012777  003076  170312           MOV     #UNEXPT,@GRIVB          ;RESET OUTPUT VECTOR
 (1)  011116  012777  000340  170306           MOV     #340,@GRIVSB
1387  011124  005037  177776                   CLR     PSW
```

```
                                        ;;*****************************************************************
                                        ;*TEST 52        DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
                                        ;;*****************************************************************
                                TST52:  SCOPE
                                        MOV     #1,$TIMES       ;;DO 1 ITERATION
                                BYPASS: TST     NBEXT                   ;TEST IF ANY
                                        BEQ     1$                      ;BR IF NONE
                                        BIT     #SW12,@SWR              ;TEST BIT12 OF SWR
                                        BNE     BYPAS1                  ;INHIBIT TESTING NEXT DR11-K
                                        SUB     #10,DRADD               ;UPDATE DEVICE ADDRESS
                                        ADD     #10,DRIV                ;UPDATE DEVICE VECTOR
                                        DEC     NBEXT                   ;ANOTHER ONE ?
                                        BR      BYPAS1                  ;BR IF ANOTHER
                                1$:     MOV     BASEBA,DRADD            ;RELOAD ADDRESS
                                        MOV     BASEIV,DRIV             ;RELOAD VECTOR
                                        MOV     NMBEXT,NBEXT            ;RELOAD NUMBER
                                        JMP     $EOP                    ;DONE
                                BYPAS1: MOV     #-1,R0
                                        JMP     RBEG2                   ;TEST ANOTHER UNIT

                                .SBTTL  END OF PASS ROUTINE

                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO BYPAS1

011236                          $EOP:
011236  000004                          SCOPE
011240  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
011244  005037  001160                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
011250  005237  001176                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
011254  042737  100000  001176          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
011262  005327                          DEC     (PC)+           ;;LOOP?
011264  000001                  $EOPCT: .WORD   1
011266  003022                          BGT     $DOAGN          ;;YES
011270  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
011272  000001                  $ENDCT: .WORD   1
011274  011264                          $EOPCT
011276  104401  011343                  TYPE    ,$ENDMG         ;;TYPE "END PASS #"
011302  013746  001176                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
011306  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
011310  104401  011340                  TYPE    ,$NULL          ;;TYPE A NULL CHARACTER
011314  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
011320  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
011322  000005                          RESET                   ;;CLEAR THE WORLD
011324  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
011326  000240                          NOP                     ;;SAVE ROOM
011330  000240                          NOP                     ;;FOR
011332  000240                          NOP                     ;;ACT11
011334                          $DOAGN:
011334  000137                          JMP     @(PC)+          ;;RETURN
011336  011226                  $RTNAD: .WORD   BYPAS1
```

```
(1)  011340  377     377     000   $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
(1)  011343  015  042412  042116   $ENDMG: .ASCIZ  <15><12>/END PASS #/
(1)  011350 050040  051501  020123
(1)  011356 000043
1406                               .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)                                ;;**********************************************************************
(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)
(1)  011360                        $TYPDS:
(3)  011360  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  011362  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  011364  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  011366  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  011370  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  011372  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  011376  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  011402  100004                        BPL     1$              ;;BR IF INPUT IS POS.
(1)  011404  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  011406  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  011414  005000                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  011416  012703  011574                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  011422  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  011426  005002                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  011430  016001  011564                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  011434  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
(1)  011436  002402                        BLT     4$              ;;BR IF DONE
(1)  011440  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
(1)  011442  000774                        BR      3$
(1)  011444  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  011446  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  011450  001002                        BNE     5$              ;;FALL THROUGH IF 0
(1)  011452  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  011454  100407                        BMI     7$              ;;BR IF YES
(1)  011456  106316                5$:     ASLB    (SP)            ;;MSD?
(1)  011460  103003                        BCC     6$              ;;BR IF NO
(1)  011462  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  011470  052702  000060        6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  011474  052702  000040        7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  011500  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  011502  005720                        TST     (R0)+           ;;JUST INCREMENTING
(1)  011504  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
(1)  011510  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
(1)  011512  003002                        BGT     8$              ;;GO TO EXIT
(1)  011514  010502                        MOV     R5,R2           ;;GET THE LSD
(1)  011516  000764                        BR      6$              ;;GO CHANGE TO ASCII
(1)  011520  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  011522  100003                        BPL     9$              ;;BR IF NO
(1)  011524  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
```

```
  (1)  011532  105013                    9$:     CLRB    (R3)            ;;SET THE TERMINATOR
  (3)  011534  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
  (3)  011536  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
  (3)  011540  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
  (3)  011542  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
  (3)  011544  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
  (1)  011546  1044C1  011574                    TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
  (1)  011552  016666  000002  000004            MOV     2(SP),4(SP)     ;;ADJUST THE STACK
  (1)  011560  012616                            MOV     (SP)+,(SP)
  (1)  011562  000002                            RTI                     ;;RETURN TO USER
  (1)  011564  023420                    $DTBL:  10000.
  (1)  011566  001750                            1000.
  (1)  011570  000144                            100.
  (1)  011572  000012                            10.
  (1)  011574  000004                    $DBLK:  .BLKW   4
 1407
 1408                                            .SBTTL  MISC. EXTERNAL LOGIC TEST
 1409  011604  012706  001100            EXTTST: MOV     #STACK,SP
 1410  011610  004737  003166                    JSR     PC,SETADD               ;SET UP ADDRESS AND VECTOR
 1411  011614  012737  040000  011710    2$:     MOV     #BIT14,EXTCNT           ;LOAD COUNT
 1412  011622                            1$:
  (1)  011522  005077  167566                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 1413  011626  012777  125252  167560            MOV     #125252,@GRDIO          ;LOAD OUTPUT
 1414  011634  017737  167552  011706            MOV     @GRDAI,EXTTMP           ;READ INPUT
 1415  011642  013777  011706  167542            MOV     EXTTMP,@GRDAI           ;CLEAR INPUT
 1416  011650  005077  167540                    CLR     @GRDIO                  ;CLEAR OUTPUT REGISTER
 1417  011654  012777  052525  167532            MOV     #52525,@GRDIO           ;LOAD OUTPUT
 1418  011662  017737  167524  011706            MOV     @GRDAI,EXTTMP           ;READ INPUT
 1419  011670  013777  011706  167514            MOV     EXTTMP,@GRDAI           ;CLEAR INPUT
 1420  011676  005337  011710                    DEC     EXTCNT                  ;FINISHED COUNT
 1421  011702  001347                            BNE     1$
 1422  011704  000743                            BR      2$                      ;LOOP
 1423
 1424  011706  000000                    EXTTMP: 0
 1425  011710  000000                    EXTCNT: 0
 1426
 1427                                            ;SUBROUTINE TO LOAD TRAP AND POWER FAIL VECTORS
 1428  011712  012737  020254  000034    SETTRP: MOV     #$TRAP,@#TRAPVEC        ;LOAD TRAP VECTOR
 1429  011720  012737  000340  000036            MOV     #340,@#TRAPVEC+2
 1430  011726  012737  016300  000024            MOV     #$PWRDN,@#PWRVEC        ;LOAD POWER FAIL VECTOR
 1431  011734  012737  000340  000026            MOV     #340,@#PWRVEC+2
 1432  011742  012737  011752  000004            MOV     #1$,@#4                 ;SET UP FOR UNEXPECTED TRAP
 1433  011750  000207                            RTS     PC                      ;EXIT
 1434  011752  022626                    1$:     CMP     (SP)+,(SP)+
 1435  011754  104401  013463                    TYPE    ,BUSTRP
 1436  011760  000000                            HALT
 1437  011762  000137  001454                    JMP     BEGIN
```

```
1439
1440  011766  012706  001100          COULTR: MOV     #STACK,SP
1441  011772  004737  011712                  JSR     PC,SETTRP               ;LOAD TRAP AND POWER FAIL VECTORS
1442  011776  004737  003166                  JSR     PC,SETADD               ;SETUP BUS ADDRESS AND VECTOR
1443  012002  005077  167402                  CLR     @GRSTAT                 ;CLEAR INPUT STATUR REG
1444  012006  012777  012050  167410          MOV     #20$,@GRIVA             ;LOAD INPUT VECTOR
1445  012014  012777  000340  167404          MOV     #340,@GRIVSA
1446  012022  104401  013524          1$:     TYPE,RUNMSG                     ;TELL OPERATOR TO "RUN SAMPLE"
1447  012026  012701  020336                  MOV     #BUFFER,R'              ;LOAD POINTER FOR RESULTS
1448  012032  012777  000100  167350  4$:     MOV     #BIT6,@GRSTAT           ;ENABLE INTERRUPT
1449  012040  005037  177776                  CLR     PS                      ;LOWER PSW
1450  012044  000001                          WAIT                            ;WAIT FOR OPERATOR
1451  012046  000771                          BR      4$
1452  012050  022626          20$:    CMP     (SP)+,(SP)+             ;POP STACK
1453  012052  012737  000106  011706          MOV     #70.,EXTTMP             ;LOAD # OF 1 MSEC DELAYS
1454  012060  013700  001370          6$:     MOV     CPU,R0  ,LOAD 1 MSEC. DELAY WEIGHT ;240 FOR 11/05  ;620 FOR 11/40
1455  012064  005300          3$:     DEC     R0                      ;DELAY
1456  012066  001376                  BNE     3$
1457  012070  005337  011706                  DEC     EXTTMP                  ;FINISHED ALL MSEC. DELAY ?
1458  012074  001371                  BNE     6$                      ;BR IF NOT
1459  012076  017700  167310                  MOV     @GRDAI,R0               ;SAVE RESULTS
1460  012102  010021                          MOV     R0,(R1)+                ;SAVE IN BUFFER
1461  012104  012777  177777  167300          MOV     #-1,@GRDAI              ;CLEAR INPUT
1462  012112  042700  007777                  BIC     #7777,R0                ;MASK
1463  012116  001345                          BNE     4$                      ;BR IF NOT LAST SAMPLE
1464  012120  010103                          MOV     R1,R3                   ;COPY R1
1465  012122  014300          5$:     MOV     -(R3),R0                ;GET RESULTS
1466  012124  004737  012140                  JSR     PC,40$                  ;CONVERT AND TYPE
1467  012130  022703  020336                  CMP     #BUFFER,R3              ;FINISHED ?
1468  012134  001732                          BEQ     1$                      ;BR IF YES
1469  012136  000771                          BR      5$                      ;CONT. TYPING UNTIL DONE
1470                          ;SUBROUTINE FOR THE COULTER TEST
1471  012140  012737  000055  013546  40$:     MOV     #55,MSGRUS+2            ;FIX ASCII MESSAGE
1472  012146  012702  013553                  MOV     #MSPNT1,R2              ;LOAD DEST. POINTER
1473  012152  012737  000004  011706          MOV     #4,EXTTMP               ;LOAD COUNT
1474  012160  000404                          BR      12$
1475  012162  006000          10$:    ROR     R0
1476  012164  006000                  ROR     R0
1477  012166  006000                  ROR     R0
1478  012170  006000                  ROR     R0
1479  012172  010001          12$:    MOV     R0,R1                   ;LOAD R1
1480  012174  042701  177760                  BIC     #177760,R1              ;MASK
1481  012200  052701  000060                  BIS     #60,R1                  ;MAKE DIGIT
1482  012204  110142                          MOVB    R1,-(R2)                ;SAVE DIGIT
1483  012206  005337  011706                  DEC     EXITMP                  ;FINISHED ?
1484  012212  001363                          BNE     10$                     ;BR IF NOT
1485  012214  000337  013546                  SWAB    MSGRUS+2                ;ADJUST MESSAGE
1486  012220  104401  013544                  TYPE,   MSGRUS
1487  012224  000207                          RTS     PC                      ;EXIT
```

```
1489                                            .SBTTL  LOC-BOX LAMP AND SWITCH LOOP
1490  012226  012706  001100          LOCBOX: MOV     #STACK,SP
1491  012232  004737  011712                  JSR     PC,SETTRP
1492  012236  104401  013304                  TYPE    ,LOCHDR                 ;TELL OPERATOR WHAT THIS IS
1493  012242  004537  012346                  JSR     R5,INADR                ;GET THE ADDRESSES
1494  012246  001402  012532                  LOC1,LOC1Y
1495  012252  000412                          BR      LOCA                    ;BR IF DONE
1496  012254  004537  012346                  JSR     R5,INADR                ;GET NEXT ADDRESS
1497  012260  001404  012534                  LOC2,LOC2Y
1498  012264  000405                          BR      LOCA                    ;BR IF DONE
1499  012266  004537  012346                  JSR     R5,INADR                ;GET NEXT ADDRESS
1500  012272  001406  012536                  LOC3,LOC3Y
1501  012276  000400                          BR      LOCA
1502
1503  012300  005737  012532          LOCA:   TST     LOC1Y                   ;TEST IF SELECTED
1504  012304  001403                          BEQ     1$                      ;BR IF NOT
1505  012306  004537  012460                  JSR     R5,LOUP1                ;TEST FOR SWITCHES AND LOAD LAMPS
1506  012312  001402                          LOC1
1507  012314  005737  012534          1$:     TST     LOC2Y                   ;TEST IF SELECTED
1508  012320  001403                          BEQ     2$                      ;BR IF NOT
1509  012322  004537  012460                  JSR     R5,LOUP1                ;LOAD NEXT SET
1510  012326  001404                          LOC2
1511  012330  005737  012536          2$:     TST     LOC3Y                   ;TEST IF SELECTED
1512  012334  001403                          BEQ     3$
1513  012336  004537  012460                  JSR     R5,LOUP1                ;LOAD NEXT SET
1514  012342  001406                          LOC3
1515  012344  000755                  3$:     BR      LOCA                    ;LOOP BACK
1516
1517  012346  013537  012452          INADR:  MOV     @(R5)+,10$              ;GET BUS ADDRESS
1518  012352  012537  012456                  MOV     (R5)+,12$               ;GET INDICATOR
1519  012356  005077  000074                  CLR     @12$                    ;CLEAR FLAG
1520  012362  104401  013345          1$:     TYPE,   INADRH                  ;ASK FOR INPUT
1521  012366  013746  012452                  MOV     10$,-(SP)               ;TYPE CURRENT ADDRESS
1522  012372  104402                          TYPOC
1523  012374  104401  013410                  TYPE,   INDADR                  ;ADD YES NO
1524  012400  104411                          RDLIN                           ;READ CHAR. AND ECHO
1525  012402  000240                          NOP
1526  012404  000240                          NOP
1527  012406  013637  012454                  MOV     @(SP)+,11$              ;GET THE ANSWER
1528  012412  042737  177640  012454          BIC     #177640,11$             ;MASK OFF BITS
1529  012420  001413                          BEQ     3$                      ;BR IF CR
1530  012422  022737  000116  012454          CMP     #'N,11$                 ;TEST IF NO
1531  012430  001406                          BEQ     2$                      ;BR IF NO
1532  012432  022737  000131  012454          CMP     #'Y,11$                 ;TEST IF YES
1533  012440  001350                          BNE     1$                      ;BR IF OTHER
1534  012442  005277  000010                  INC     @12$                    ;SET YES FLAG
1535  012446  005725                  2$:     TST     (R5)+
1536  012450  000205                  3$:     RTS     R5                      ;EXIT
1537  012452  000000                  10$:    0
1538  012454  000000                  11$:    0
1539  012456  000000                  12$:    0
```

```
1541
1542                                    ;SUBROUTINE TO LOAD THE LAMPS FROM THE LOC BOX SWITCHES
1543  012460  013537  012526    LOUP1:  MOV     @(R5)+,40$              ;GET ARG.
1544  012464  001001                    BNE     1$                      ;BR IF YES
1545  012466  000205                    RTS     R5                      ;EXIT
1546  012470  013702  012526    1$:     MOV     40$,R2
1547  012474  062702  000002            ADD     #2,R2
1548  012500  010203                    MOV     R2,R3
1549  012502  062703  000002            ADD     #2,R3
1550  012506  011200                    MOV     (R2),R0
1551  012510  011301                    MOV     (R3),R1
1552  012512  040100                    BIC     R1,R0                   ;MASK OFF BITS
1553  012514  041213                    BIC     (R2),(R3)               ;LOAD OUTPUT BITS
1554  012516  060013                    ADD     R0,(R3)         ;LOAD
1555  012520  012712  177777            MOV     #-1,(R2)                ;CLEAR INPUT
1556  012524  000205                    RTS     R5                      ;EXIT
1557
1558  012526  000000            40$:    0
1559  012530  000000            41$:    0
1560  012532  000000            LOC1Y:  0               ;YES/NO ANSWER TO LOC BOX #1
1561  012534  000000            LOC2Y:  0
1562  012536  000000            LOC3Y:  0
1563
1564                                    .SBTTL  XL01 ADJUSTMENT ROUTINE
1565  012540  012706  001100    XL01AD: MOV     #STACK,SP
1566  012544  004737  011712            JSR     PC,SETTRP
1567  012550  104401  013426            TYPE    ,XL0HDR         ;TELL OPERATOR
1568  012554  004537  012346            JSR     R5,INADR                ;GET BUS ADDRESS
1569  012560  001402  012532            LOC1,LOC1Y
1570  012564  000412                    BR      XL010
1571  012566  004537  012346            JSR     R5,INADR                ;
1572  012572  001404  012534            LOC2,LOC2Y
1573  012576  000405                    BR      XL010
1574  012600  004537  012346            JSR     R5,INADR
1575  012604  001406  012536            LOC3,LOC3Y
1576  012610  000400                    BR      XL010
1577
1584  012612  005737  012532    XL010:  TST     LOC1Y                   ;TEST IF SELECTED
1585  012616  001431                    BEQ     1$                      ;BR IF NOT SELECTED
1586  012620  004537  013066            JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012624     000     000            .BYTE   0,0                     ;LOAD CH# AND SHIFT COUNT
 (1)  012626  000007                    7                               ;LOAD DATA MASK
 (1)  012630  001402                    LOC1                            ;DR11K BUS ADDRESS
1587  012632  004537  013066            JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012636     001     003            .BYTE   1,3                     ;LOAD CH# AND SHIFT COUNT
 (1)  012640  000070                    70                              ;LOAD DATA MASK
 (1)  012642  001402                    LOC1                            ;DR11K BUS ADDRESS
1588  012644  004537  013066            JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012650     002     006            .BYTE   2,6                     ;LOAD CH# AND SHIFT COUNT
 (1)  012652  000700                    700                             ;LOAD DATA MASK
 (1)  012654  001402                    LOC1                            ;DR11K BUS ADDRESS
1589  012656  004537  013066            JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
 (1)  012662     003     011            .BYTE   3,11                    ;LOAD CH# AND SHIFT COUNT
 (1)  012664  007000                    7000                            ;LOAD DATA MASK
 (1)  012666  001402                    LOC1                            ;DR11K BUS ADDRESS
1590  012670  004537  013066            JSR     R5,XLOADJ               ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
```

```
   (1)  012674     004     014                .BYTE   4,14              ;LOAD CH# AND SHIFT COUNT
   (1)  012676  070000                        70000                     ;LOAD DATA MASK
   (1)  012700  001402                        LOC1                      ;DR11K BUS ADDRESS
 1591   012702  005737  012534        1$:     TST     LOC2Y             ;TEST IF SELECTED
 1592   012706  001431                        BEQ     2$                ;BR IF NOT
 1593   012710  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  012714     005     000                .BYTE   5,0               ;LOAD CH# AND SHIFT COUNT
   (1)  012716  000007                        7                         ;LOAD DATA MASK
   (1)  012720  001404                        LOC2                      ;DR11K BUS ADDRESS
 1594   012722  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  012726     006     003                .BYTE   6,3               ;LOAD CH# AND SHIFT COUNT
   (1)  012730  000070                        70                        ;LOAD DATA MASK
   (1)  012732  001404                        LOC2                      ;DR11K BUS ADDRESS
 1595   012734  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  012740     007     006                .BYTE   7,6               ;LOAD CH# AND SHIFT COUNT
   (1)  012742  000700                        700                       ;LOAD DATA MASK
   (1)  012744  001404                        LOC2                      ;DR11K BUS ADDRESS
 1596   012746  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  012752     010     011                .BYTE   10,11             ;LOAD CH# AND SHIFT COUNT
   (1)  012754  007000                        7000                      ;LOAD DATA MASK
   (1)  012756  001404                        LOC2                      ;DR11K BUS ADDRESS
 1597   012760  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  012764     011     014                .BYTE   11,14             ;LOAD CH# AND SHIFT COUNT
   (1)  012766  070000                        70000                     ;LOAD DATA MASK
   (1)  012770  001404                        LOC2                      ;DR11K BUS ADDRESS
 1598   012772  005737  012536        2$:     TST     LOC3Y             ;TEST IF SELECTED
 1599   012776  001431                        BEQ     3$                ;BR IF NOT
 1600   013000  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  013004     012     000                .BYTE   12,0              ;LOAD CH# AND SHIFT COUNT
   (1)  013006  000007                        7                         ;LOAD DATA MASK
   (1)  013010  001406                        LOC3                      ;DR11K BUS ADDRESS
 1601   013012  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  013016     013     003                .BYTE   13,3              ;LOAD CH# AND SHIFT COUNT
   (1)  013020  000070                        70                        ;LOAD DATA MASK
   (1)  013022  001406                        LOC3                      ;DR11K BUS ADDRESS
 1602   013024  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  013030     014     006                .BYTE   14,6              ;LOAD CH# AND SHIFT COUNT
   (1)  013032  000700                        700                       ;LOAD DATA MASK
   (1)  013034  001406                        LOC3                      ;DR11K BUS ADDRESS
 1603   013036  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  013042     015     011                .BYTE   15,11             ;LOAD CH# AND SHIFT COUNT
   (1)  013044  007000                        7000                      ;LOAD DATA MASK
   (1)  013046  001406                        LOC3                      ;DR11K BUS ADDRESS
 1604   013050  004537  013066                JSR     R5,XLOADJ         ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
   (1)  013054     016     014                .BYTE   16,14             ;LOAD CH# AND SHIFT COUNT
   (1)  013056  070000                        70000                     ;LOAD DATA MASK
   (1)  013060  001406                        LOC3                      ;DR11K BUS ADDRESS
 1605   013062  000137  012612        3$:     JMP     XL010
```

```
1607
1608  013066  112577  000206          ;ANALOG CONVERSION AND LOC BOX DRIVER
1609  013072  052777  020000  000176  XLOADJ: MOVB    (R5)+,@ADCS1            ;LOAD CHANNEL #
1610  013100  112537  013270                  BIS     #BIT13,@ADCS            ;SET UNIPOLAR
1611  013104  012537  013274                  MOVB    (R5)+,40$               ;GET SHIFT COUNT
1612  013110  013537  013272                  MOV     (R5)+,42$               ;GET DATA MASK
1613  013114  001464                          MOV     @(R5)+,41$              ;GET DR11K BUS ADDRESS
1614  013116  105277  000154                  BEQ     70$             ;BR IF NONE
1615  013122  005001                          INCB    @ADCS                   ;CONVERT THE DATA
1616  013124  062737  000004  013272          CLR     R1
1617  013132  105777  000140          1$:     ADD     #4,41$          ;GET LAMP ADDRESS
1618  013136  100375                          TSTB    @ADCS                   ;WAIT FOR DONE
1619  013140  017700  000136                  BPL     1$
1620  013144  162700  000004                  MOV     @ADBR,R0                ;READ THE DATA
1621  013150  100432                          SUB     #4,R0                   ;TEST IF AT UPPER END
1622  013152  162700  000014                  BMI     7$                      ;BR IF YES
1623  013156  100425                          SUB     #14,R0                  ;TEST IF NEAR UPPER END
1624  013160  162700  000060                  BMI     6$                      ;BR IF YES
1625  013164  100420                          SUB     #60,R0                  ;TEST IF GETTING NEAR UPPER END
1626  013166  162700  001530                  BMI     5$                      ;BR IF YES
1627  013172  100407                          SUB     #1530,R0                ;TEST IF GETTING NEAR LOWER END
1628  013174  162700  000060                  BMI     2$                      ;BR IF YES
1629  013200  100406                          SUB     #60,R0                  ;TEST IF NEAR LOWER END
1630  013202  162700  000014                  BMI     3$                      ;BR IF YES
1631  013206  100405                          SUB     #14,R0                  ;TEST IF AT LOWER END
1632  013210  000412                          BMI     4$                      ;BR IF YES
1633  013212  012701  000004          2$:     BR      7$                      ;BR IF AT STOP
1634  013216  062701  000002          3$:     MOV     #4,R1                   ;LOAD #
1635  013222  005201                  4$:     ADD     #2,R1
1636  013224  000404                          INC     R1
1637  013226  012701  000002          5$:     BR      7$                      ;BR AND LOAD LAMP'S
1638  013232  062701  000004          6$:     MOV     #2,R1                   ;LOAD #
1639  013236  013700  013270          7$:     ADD     #4,R1
1640  013242  006301                  10$:    MOV     40$,R0                  ;LOAD SHIFT COUNT
1641  013244  005300                          ASL     R1                      ;SHUFFEL THE DATA
1642  013246  100375                          DEC     R0
1643  013250  006201                          BPL     10$                     ;BR IF NOT DONE
1644  013252  043777  013274  000012          ASR     R1                      ;RE ADJUST
1645  013260  050177  000006                  BIC     42$,@41$                ;CLEAR OLD DATA IN LAMPS
1646  013264  005001                          BIS     R1,@41$                 ;LOAD THE LAMPS
1647  013266  000205                  70$:    CLR     R1
1648                                          RTS     R5                      ;EXIT
1649  013270  000000                  40$:    0
1650  013272  000000                  41$:    0
1651  013274  000000                  42$:    0
```

```
                                                                    N 4
 1653  013276  170400                    ADCS:   170400
 1654  013300  170401                    ADCS1:  170401
 1655  013302  170402                    ADBR:   170402
 1656  013304  005015  047514  026503    LOCHDR: .ASCIZ  <15><12>/LOC-BOX SWITCH AND LAMP LOOP/<15><12>
       013312  047502  020130  053523
       013320  052111  044103  040440
       013326  042116  046040  046501
       013334  020120  047514  050117
       013342  005015     000
 1657  013345     015  052412  042523    INADRH: .ASCIZ  <15><12>/USE LOC-BOX <DR11K AT BUS ADDR. /
       013352  046040  041517  041055
       013360  054117  036040  051104
       013366  030461  020113  052101
       013374  041040  051525  040440
       013402  042104  027122  000040
 1658  013410  037040  020040  020131    INDADR: .ASCIZ  / >  Y OR N ? /
       013416  051117  047040  037440
       013424  000040
 1659  013426  005015  046130  030460    XLOHDR: .ASCIZ  <15><12>/XL01 POT ADJUSTMENT LOOP/<15><12>
       013434  050040  052117  040440
       013442  045104  051525  046524
       013450  047105  020124  047514
       013456  050117  005015     000
 1660  013463     015  041012  051525    BUSTRP: .ASCIZ  <15><12>/BUS TIME-OUT ON SELECTED DR11K/
       013470  052040  046511  026505
       013476  052517  020124  047117
       013504  051440  046105  041505
       013512  042524  020104  051104
       013520  030461  000113
 1661  013524  005015  051012  047125    RUNMSG: .ASCIZ  <15><12><12>/RUN SAMPLE/<15><12>
       013532  051440  046501  046120
       013540  006505  000012
 1662  013544  005015  047055  047116    MSGRUS: .ASCII  <15><12>/-NNNN/
       013552     116
 1663  013553     000                    MSPNT1: .BYTE   0
 1664                                            .EVEN
 1665
 1666
 1667  013554  042523  020124  053523    SWNLB:  .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING INPUT BITS/
       013562  052111  044103  051040
       013570  043505  051511  042524
       013576  020122  044502  051524
       013604  042440  052521  046101
       013612  052040  020117  044124
       013620  020105  047516  026516
       013626  040514  041524  044510
       013634  043516  044440  050116
       013642  052125  041040  052111
       013650  000123
 1668  013652  042523  020124  053523    SWINTB: .ASCIZ  /SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING INPUT BITS/
       013660  052111  044103  051040
       013666  043505  051511  042524
       013674  020122  044502  051524
       013702  042440  052521  046101
       013710  052040  020117  044124
       013716  020105  047111  042524
```

```
                013724  051122  050125  044524
                013732  043516  044440  050116
                013740  052125  041040  052111
                013746  000123
1669            013750  042523  020124  053523  SWPOSB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO POSITIVE INPUT BITS/
                013756  052111  044103  051040
                013764  043505  051511  042524
                013772  020122  044502  051524
                014000  030440  026465  031061
                014006  042440  052521  046101
                014014  052040  020117  047520
                014022  044523  044524  042526
                014030  044440  050116  052125
                014036  041040  052111  000123
1670            014044  042523  020124  053523  SWTRAB: .ASCIZ  /SET SWITCH REGISTER BITS 15-12 EQUAL TO TRANSITION INPUT BITS/
                014052  052111  044103  051040
                014060  043505  051511  042524
                014066  020122  044502  051524
                014074  030440  026465  031061
                014102  042440  052521  046101
                014110  052040  020117  051124
                014116  047101  044523  044524
                014124  047117  044440  050116
                014132  052125  040040  052111
                014140  000123
1671            014142  042523  020124  053523  SWDPOB: .ASCIZ  /SET SWITCH REGISTER WITH THE DESIRED PROGRAM OPTIONS/
                014150  052111  044103  051040
                014156  043505  051511  042524
                014164  020122  044527  044124
                014172  052040  042510  042040
                014200  051505  051111  042105
                014206  050040  047522  051107
                014214  046501  047440  052120
                014222  047511  051516     000
1672            014227     015  042012  050105  DEPCNT: .ASCIZ  <15><12>/DEPRESS CONT./<15><12>
                014234  042522  051523  041440
                014242  047117  027124  005015
                014250     000
1673            014251     123  040524  052524  EM1:    .ASCIZ  /STATUS REGISTER IN ERROR/
                014256  020123  042522  044507
                014264  052123  051105  044440
                014272  020116  051105  047522
                014300  000122
1674            014302  047111  052520  020124  EM2:    .ASCIZ  /INPUT REGISTER IN ERROR/
                014310  042522  044507  052123
                014316  051105  044440  020116
                014324  051105  047522  000122
1675            014332  052517  050124  052125  EM3:    .ASCIZ  /OUTPUT REGISTER IN ERROR/
                014340  051040  043505  051511
                014346  042524  020122  047111
                014354  042440  051122  051117
                014362     000
1676            014363     111  050116  052125  EM4:    .ASCIZ  /INPUT FAILED TO INTERRUPT/
                014370  043040  044501  042514
                014376  020104  047524  044440
                014404  052116  051105  052522
```

```
                014412  052120     000
1677            014415     117  052125  052520  EM5:    .ASCIZ  /OUTPUT FAILED TO INTERRUPT/
                014422  020124  040506  046111
                014430  042105  052040  020117
                014436  047111  042524  051122
                014444  050125  000124
1678            014450  047125  054105  042520  EM6:    .ASCIZ  /UNEXPECTED INTERRUPT/
                014456  052103  042105  044440
                014464  052116  051105  052522
                014472  052120     000
1679            014475     117  042520  040522  EM7:    .ASCIZ  /OPERATOR INTERVENTION ERROR/
                014502  047524  020122  047111
                014510  042524  053122  047105
                014516  044524  047117  042440
                014524  051122  051117     000
1680            014531     111  052116  051105  EM10:   .ASCIZ  /INTERRUPT INPUT BIT FAILED TO SET INPUT READY FLAG/
                014536  052522  052120  044440
                014544  050116  052125  041040
                014552  052111  043040  044501
                014560  042514  020104  047524
                014566  051440  052105  044440
                014574  050116  052125  051040
                014602  040505  054504  043040
                014610  040514  000107
1681            014614  047516  026516  047111  EM11:   .ASCIZ  /NON-INTERRUPT INPUT BIT SET INPUT READY FLAG/
                014622  042524  051122  050125
                014630  020124  047111  052520
                014636  020124  044502  020124
                014644  042523  020124  047111
                014652  052520  020124  042522
                014660  042101  020131  046106
                014666  043501     000
1682
1683            014671     105  051122  041520  DH1:    .ASCIZ  /ERRPC   DRADD  TSTNUM   STATUS  EXPECTED/
                014676  020040  042040  040522
                014704  042104  052011  052123
                014712  052516  020115  020040
                014720  052123  052101  051525
                014726  020040  054105  042520
                014734  052103  042105     000
1684            014741     105  051122  041520  DH2:    .ASCIZ  /ERRPC   DRADD  TSTNUM  INPUT   EXPECTED/
                014746  020040  042040  040522
                014754  042104  052011  052123
                014762  052516  004515  047111
                014770  052520  020124  020040
                014776  054105  042520  052103
                015004  042105     000
1685            015007     105  051122  041520  DH3:    .ASCIZ  /ERRPC   DRADD  TSTNUM  OUTPUT  EXPECTED/
                015014  020040  042040  040522
                015022  042104  052011  052123
                015030  052516  004515  052517
                015036  050124  052125  020040
                015044  054105  042520  052103
                015052  042105     000
1686            015055     105  051122  041520  DH4:    .ASCIZ  /ERRPC   DRADD  TSTNUM/
                015062  020040  042040  040522
```

```
        015070  042104  052011  052123
        015076  052516  000115
1687    015102  051105  050122  020103  DH10:   .ASCIZ  /ERRPC   DRADD  TSTNUM  STATUS  EXPECT  INPUT BIT/
        015110  020040  051104  042101
        015116  004504  051524  047124
        015124  046525  051411  040524
        015132  052524  020123  042440
        015140  050130  041505  020124
        015146  044440  050116  052125
        015154  041040  052111     000
1688            015162                  .EVEN
1689    015162  001116  001376  015242  DT1:    $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,0
        015170  001126  001124  000000
1690    015176  001116  001376  015242  DT4:    $ERRPC,DRADD,TSTNUM,0
        015204  000000
1691    015206  001116  001376  015242  DT10:   $ERRPC,DRADD,TSTNUM,$BDDAT,$GDDAT,BRLEV3,0
        015214  001126  001124  001442
        015222  000000
1692    015224  000000  000000  000000  DF0:    0,0,0,0,0,0,0
        015232  000000  000000  000000
        015240  000000
1693    015242  000000                  TSTNUM: 0
```

```
                                               .SBTTL  SCOPE HANDLER ROUTINE
 (1)
 (2)                                           ;;*****************************************************************
 (1)                                           ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 (1)                                           ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 (1)                                           ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
 (1)                                           ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 (1)                                           ;*SW14=1        LOOP ON TEST
 (1)                                           ;*SW11=1        INHIBIT ITERATIONS
 (1)                                           ;*SW09=1        LOOP ON ERROR
 (1)                                           ;*SW08=1        LOOP ON TEST IN SWR<7:0>
 (1)                                           ;*CALL
 (1)                                           ;*       SCOPE             ;;SCOPE=IOT
 (1)
 (1)
 (1) 015244                                    $SCOPE:
 (1) 015244 104407                                      CKSWR                      ;;TEST  FOR CHANGE IN SOFT-SWR
 (1) 015246 032777 040000 163664               1$:      BIT     #BIT14,@SWR        ;;LOOP ON PRESENT TEST?
 (1) 015254 001114                                      BNE     $OVER              ;;YES IF SW14=1
 (1)                                           ;#####START OF CODE FOR THE XOR TESTER#####
 (1) 015256 000416                             $XTSTR:  BR      6$                 ;;IF RUNNING ON THE "XOR" TESTER CHANGE
 (1)                                                                               ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 (1) 015260 013746 000004                               MOV     @#ERRVEC,-(SP)     ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 (1) 015264 012737 015304 000004                        MOV     #5$,@#ERRVEC       ;;SET FOR TIMEOUT
 (1) 015272 005737 177060                               TST     @#177060           ;;TIME OUT ON XOR?
 (1) 015276 012637 000004                               MOV     (SP)+,@#ERRVEC     ;;RESTORE THE ERROR VECTOR
 (1) 015302 000463                                      BR      $SVLAD             ;;GO TO THE NEXT TEST
 (1) 015304 022626                             5$:      CMP     (SP)+,(SP)+        ;;CLEAR THE STACK AFTER A TIME OUT
 (1) 015306 012637 000004                               MOV     (SP)+,@#ERRVEC     ;;RESTORE THE ERROR VECTOR
 (1) 015312 000423                                      BR      7$                 ;;LOOP ON THE PRESENT TEST
 (1) 015314                                    6$:;#####END OF CODE FOR THE XOR TESTER#####
 (1) 015314 032777 000400 163616                        BIT     #BIT08,@SWR        ;;LOOP ON SPEC. TEST?
 (1) 015322 001404                                      BEQ     2$                 ;;BR IF NO
 (1) 015324 127737 163610 001102                        CMPB    @SWR,$TSTNM        ;;ON THE RIGHT TEST?    SWR<7:0>
 (1) 015332 001465                                      BEQ     $OVER              ;;BR IF YES
 (1) 015334 105737 001103                      2$:      TSTB    $ERFLG             ;;HAS AN ERROR OCCURRED?
 (1) 015340 001421                                      BEQ     3$                 ;;BR IF NO
 (1) 015342 123737 001115 001103                        CMPB    $ERMAX,$ERFLG      ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 (1) 015350 101015                                      BHI     3$                 ;;BR IF NO
 (1) 015352 032777 001000 163560                        BIT     #BIT09,@SWR        ;;LOOP ON ERROR?
 (1) 015360 001404                                      BEQ     4$                 ;;BR IF NO
 (1) 015362 013737 001110 001106               7$:      MOV     $LPERR,$LPADR      ;;SET LOOP ADDRESS TO LAST SCOPE
 (1) 015370 000446                                      BR      $OVER
 (1) 015372 105037 001103                      4$:      CLRB    $ERFLG             ;;ZERO THE ERROR FLAG
 (1) 015376 005037 001160                               CLR     $TIMES             ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 (1) 015402 000415                                      BR      1$                 ;;ESCAPE TO THE NEXT TEST
 (1) 015404 032777 004000 163526               3$:      BIT     #BIT11,@SWR        ;;INHIBIT ITERATIONS?
 (1) 015412 001011                                      BNE     1$                 ;;BR IF YES
 (1) 015414 005737 001176                               TST     $PASS              ;;IF FIRST PASS OF PROGRAM
 (1) 015420 001406                                      BEQ     1$                 ;;      INHIBIT ITERATIONS
 (1) 015422 005237 001104                               INC     $ICNT              ;;INCREMENT ITERATION COUNT
 (1) 015426 023737 001160 001104                        CMP     $TIMES,$ICNT       ;;CHECK THE NUMBER OF ITERATIONS MADE
 (1) 015434 002024                                      BGE     $OVER              ;;BR IF MORE ITERATION REQUIRED
 (1) 015436 012737 000001 001104               1$:      MOV     #1,$ICNT           ;;REINITIALIZE THE ITERATION COUNTER
 (1) 015444 013737 015522 001160                        MOV     $MXCNT,$TIMES      ;;SET NUMBER OF ITERATIONS TO DO
 (1) 015452 105237 001102                      $SVLAD:  INCB    $TSTNM             ;;COUNT TEST NUMBERS
```

```
(1)  015456  113737  001102  001174          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1)  015464  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1)  015470  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  015474  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  015500  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  015506  013777  001102  163426  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  015514  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(1)  015520  000002                          RTI                     ;;FIXES PS
(1)  015522  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
1697                                 .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)  015524                          $ERROR:
(1)  015524  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(2)  015526  113737  001102  015242          MOVB    $TSTNM,TSTNUM
(1)  015534  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
(1)  015540  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
(1)  015542  013777  001102  163372          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  015550  005237  001112                  INC     $ERTTL          ;;INC THE ERROR COUNT
(1)  015554  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  015560  162737  000002  001116          SUB     #2,$ERRPC
(1)  015566  117737  163324  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  015574  032777  020000  163336          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
(1)  015602  001004                          BNE     20$             ;;SKIP TYPEOUTS
(1)  015604  004737  015716                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(1)  015610  104401  001165                  TYPE    ,$CRLF
(1)  015614                          20$:
(1)  015614  122737  000001  001210          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  015622  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
(1)  015624  113737  001114  015636          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1)  015632  004737  020024                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
(1)  015636     000                  21$:    .BYTE   0
(1)  015637     000                          .BYTE   0
(1)  015640  000777                  22$:    BR      22$             ;;APT ERROR LOOP
(1)  015642  005777  163272          2$:     TST     @SWR            ;;HALT ON ERROR
(1)  015646  100002                          BPL     3$              ;;SKIP IF CONTINUE
(1)  015650  000000                          HALT                    ;;HALT ON ERROR.
(1)  015652  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
(1)  015654  032777  001000  163256  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
(1)  015662  001402                          BEQ     4$              ;;BR IF NO
(1)  015664  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
(1)  015670  005737  001162          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
(1)  015674  001402                          BEQ     5$              ;;BR IF NONE
(1)  015676  013716  001162                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  015702                          5$:
(1)  015702  022737  011324  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
```

```
(1)  015710  001001                BNE     6$              ;;BRANCH IF NO
(1)  015712  000000                HALT                    ;;YES
(1)  015714                 6$
(1)  015714  000002                RTI                     ;;RETURN
```

```
                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

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

                                $ERRTYP:
016716                                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
016716  104401  001165                  MOV     R0,-(SP)        ;;SAVE R0
016722  010046                          CLR     R0              ;;PICKUP THE ITEM INDEX
016724  005000                          BISB    @#$ITEMB,R0
016726  153700  001114                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
016732  001004                                                  ;;TYPE THE PC OF THE ERROR
                                        MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
016734  013746  001116                                          ;;ERROR ADDRESS
                                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
016740  104402                          BR      6$              ;;GET OUT
016742  000426                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
016744  005300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
016746  006300                          ASL     R0
016750  006300                          ASL     R0
016752  006300                          ADD     #$ERRTP,R0      ;;FORM TABLE POINTER
016754  062700  001252                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
016760  012037  015770                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
016764  001404                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
016766  104401                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
016770  000000                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
016772  104401  001165          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
016776  012037  016006                  BEQ     5$              ;;SKIP TYPEOUT IF 0
017002  001404                          TYPE                    ;;TYPE THE "DATA HEADER"
017004  104401                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
017006  000000                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
017010  104401  001165          5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
017014  011000                          BNE     7$              ;;GO TYPE THE DATA
017016  001004                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
017020  012600                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
017022  104401  001165                  RTS     PC              ;;RETURN
017026  000207                  7$:
017030                                  MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
017030  013046                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
017032  104402                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
017034  005710                          BEQ     6$              ;;BR IF NO
017036  001770                          TYPE    ,8$             ;;TYPE TWO(2) SPACES
017040  104401  016046                  BR      7$              ;;LOOP
017044  000771                  8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
017046  020040      000                 .EVEN
017052
```

```
                                .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

016052  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
016056  116637  000001  016275          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
016064  112637  016277                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
016070  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
016074  000406                          BR      $TYPON
016076  112737  000001  016275  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
016104  112737  000006  016277          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
016112  112737  000005  016274  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
016120  010346                          MOV     R3,-(SP)        ;;SAVE R3
016122  010446                          MOV     R4,-(SP)        ;;SAVE R4
016124  010546                          MOV     R5,-(SP)        ;;SAVE R5
016126  113704  016277                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
016132  005404                          NEG     R4
016134  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
016140  110437  016276                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
016144  113704  016275                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
016150  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
016154  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
016156  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
016160  000404                          BR      3$              ;;GO DO MSB
016162  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
016164  006105                          ROL     R5
016166  006105                          ROL     R5
016170  010503                          MOV     R5,R3
016172  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
016174  105337  016276                  DECB    $OMODE          ;;TYPE THIS DIGIT?
016200  100016                          BPL     7$              ;;BR '  NO
016202  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
016206  001002                          BNE     4$              ;;TEST FOR 0
016210  005704                          TST     R4              ;;SUPPRESS THIS 0?
```

```
(1)  016212  001403                        BEQ     5$              ;;BR IF YES
(1)  016214  005204                 4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  016216  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  016222  052703  000040         5$:    BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  016226  110337  016272                MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  016232  104401  016272                TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  016236  105337  016274         7$:    DECB    $OCNT           ;;COUNT BY 1
(1)  016242  003347                        BGT     2$              ;;BR IF MORE TO DO
(1)  016244  002402                        BLT     6$              ;;BR IF DONE
(1)  016246  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  016250  000744                        BR      2$              ;;GO DO THE LAST DIGIT
(1)  016252  012605                 6$:    MOV     (SP)+,R5        ;;RESTORE R5
(1)  016254  012604                        MOV     (SP)+,R4        ;;RESTORE R4
(1)  016256  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  016260  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  016266  012616                        MOV     (SP)+,(SP)
(1)  016270  000002                        RTI                     ;;RETURN
(1)  016272     000                 8$:    .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  016273     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  016274     000                 $OCNT: .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  016275     000                 $OFILL:.BYTE   0               ;  'ERO FILL SWITCH
(1)  016276  000000                 $OMODE:.WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;POWER DOWN ROUTINE
 (1) 016300 012737 016444 000024        $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
 (1) 016306 012737 000340 000026                MOV     #340,@#PWRVEC+2 ;;PRIO:7
 (3) 016314 010046                              MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 (3) 016316 010146                              MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 (3) 016320 010246                              MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 (3) 016322 010346                              MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 (3) 016324 010446                              MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 (3) 016326 010546                              MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 (3) 016330 017746 162604                       MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
 (1) 016334 010637 016450                       MOV     SP,$SAVR6       ;;SAVE SP
 (1) 016340 012737 016352 000024                MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
 (1) 016346 000000                              HALT
 (1) 016350 000776                              BR      .-2             ;;HANG UP
 (1)
 (2)                                    ;;*****************************************************************
 (1)                                    ;POWER UP ROUTINE
 (1) 016352 012737 016444 000024        $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
 (1) 016360 013706 016450                       MOV     $SAVR6,SP       ;;GET SP
 (1) 016364 005037 016450                       CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
 (1) 016370 005237 016450               1$:     INC     $SAVR6          ;;WAIT FOR THE INC
 (1) 016374 001375                              BNE     1$              ;;OF  WORD
 (3) 016376 012677 162536                       MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
 (3) 016402 012605                              MOV     (SP)+,R5        ;;POP STACK INTO R5
 (3) 016404 012604                              MOV     (SP)+,R4        ;;POP STACK INTO R4
 (3) 016406 012603                              MOV     (SP)+,R3        ;;POP STACK INTO R3
 (3) 016410 012602                              MOV     (SP)+,R2        ;;POP STACK INTO R2
 (3) 016412 012601                              MOV     (SP)+,R1        ;;POP STACK INTO R1
 (3) 016414 012600                              MOV     (SP)+,R0        ;;POP STACK INTO R0
 (1) 016416 012737 016300 000024                MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
 (1) 016424 012737 000340 000026                MOV     #340,@#PWRVEC+2 ;;PRIO:7
 (1) 016432 104401                              TYPE                    ;;REPORT THE POWER FAILURE
 (1) 016434 016452                      $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
 (1) 016436 012716                              MOV     (PC)+,(SP)      ;;RESTART AT IOTEST
 (1) 016440 003102                      $PWRAD: .WORD   IOTEST          ;;RESTART ADDRESS
 (1) 016442 000002                              RTI
 (1) 016444 000000                      $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
 (1) 016446 000776                              BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
 (1) 016450 000000                      $SAVR6: 0                       ;;PUT THE SP HERE
     016452 005015 042522 052123        PWRMSG: .ASCIZ  <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
     016460 051101 044524 043516
     016466 040440 052106 051105
     016474 040440 050040 053517
     016502 051105 043040 044501
     016510 052514 042522 005015
     016516    000
            016520                                      .EVEN
```

```
 1710
 1711                                   .SBTTL  TYPE ROUTINE
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
  (1)                                   ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
  (1)                                   ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
  (1)                                   ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIPED.
  (1)                                   ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
  (1)                                   ;*
  (1)                                   ;*CALL:
  (1)                                   ;*1) USING A TRAP INSTRUCTION
  (1)                                   ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
  (1)                                   ;*OR
  (1)                                   ;*      TYPE
  (1)                                   ;*      MESADR
  (1)                                   ;*
  (1)
  (1)  016520  105737  001157           $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
  (1)  016524  100002                           BPL     1$              ;;BR IF YES
  (1)  016526  000000                           HALT                    ;;HALT HERE IF NO TERMINAL
  (1)  016530  000430                           BR      3$              ;;LEAVE
  (1)  016532  010046                   1$:     MOV     R0,-(SP)        ;;SAVE R0
  (1)  016534  017600  000002                   MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
  (1)  016540  122737  000001  001210           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)  016546  001011                           BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
  (1)  016550  132737  000100  001211           BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
  (1)  016556  001405                           BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
  (1)  016560  010037  016570                   MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
  (1)  016564  004737  020014                   JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
  (1)  016570  000000                   61$:    .WORD   0               ;;MESSAGE ADDRESS
  (1)  016572  132737  000040  001211   62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
  (1)  016600  001003                           BNE     60$             ;;YES,SKIP TYPE OUT
  (1)  016602  112046                   2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
  (1)  016604  001005                           BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
  (1)  016606  005726                           TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
  (1)  016610  012600                   60$:    MOV     (SP)+,R0        ;;RESTORE R0
  (1)  016612  062716  000002           3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
  (1)  016616  000002                           RTI                     ;;RETURN
  (1)  016620  122716  000011           4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
  (1)  016624  001430                           BEQ     8$
  (1)  016626  122716  000200                   CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
  (1)  016632  001006                           BNE     5$
  (1)  016634  005726                           TST     (SP)+           ;;POP  <CR><LF> EQUIV
  (1)  016636  104401                           TYPE                    ;;TYPE A CR AND LF
  (1)  016640  001165                           $CRLF
  (1)  016642  105037  016776                   CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
  (1)  016646  000755                           BR      2$              ;;GET NEXT CHARACTER
  (1)  016650  004737  016732           5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
  (1)  016654  123726  001156           6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
  (1)  016660  001350                           BNE     2$              ;;IF NO GO GET NEXT CHAR.
  (1)  016662  013746  001154                   MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
  (1)                                           ;;AND THE NULL CHAR.
  (1)  016666  105366  000001           7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
  (1)  016672  002770                           BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
  (1)  016674  004737  016732                   JSR     PC,$TYPEC       ;;GO TYPE A NULL
```

```
   (1)  016700  105337  016776                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
   (1)  016704  000770                        BR      7$              ;;LOOP
   (1)
   (1)                                      ;HORIZONTAL TAB PROCESSOR
   (1)
   (1)  016706  112716  000040          8$:   MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
   (1)  016712  004737  016732          9$:   JSR     PC,$TYPEC       ;;TYPE A SPACE
   (1)  016716  132737  000007  016776        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
   (1)  016724  001372                        BNE     9$              ;;TAB STOP
   (1)  016726  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
   (1)  016730  000724                        BR      2$              ;;GET NEXT CHARACTER
   (1)  016732  105777  162212          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
   (1)  016736  100375                        BPL     $TYPEC
   (1)  016740  116677  000002  162204        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
   (1)  016746  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
   (1)  016754  001003                        BNE     1$              ;;BRANCH IF NO
   (1)  016756  105037  016776                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
   (1)  016762  000406                        BR      $TYPEX          ;;EXIT
   (1)  016764  122766  000012  000002  1$.   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
   (1)  016772  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
   (1)  016774  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
   (1)  016776  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
   (1)  017000  000207          $TYPEX: RTS     PC
   (1)
```

```
1713                                    .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 (1)                                    ;*CHANGE IT TO BINARY.
 (1)                                    ;*CALL:
 (1)                                    ;*      RDOCT                   ;;READ AN OCTAL NUMBER
 (1)                                    ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
 (1)                                    ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
 (1)
 (1)  017002  011646                    $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
 (1)  017004  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
 (3)  017012  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 (3)  017014  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 (3)  017016  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 (1)  017020  104411                    1$:     RDLIN                   ;;READ AN ASCIZ LINE
 (1)  017022  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
 (1)  017024  005001                            CLR     R1              ;;CLEAR DATA WORD
 (1)  017026  005002                            CLR     R2
 (1)  017030  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
 (1)  017032  001412                            BEQ     3$              ;;IF ZERO GET OUT
 (1)  017034  006301                            ASL     R1              ;;*2
 (1)  017036  006102                            ROL     R2
 (1)  017040  006301                            ASL     R1              ;;*4
 (1)  017042  006102                            ROL     R2
 (1)  017044  006301                            ASL     R1              ;;*8
 (1)  017046  006102                            ROL     R2
 (1)  017050  042716  177770                    BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
 (1)  017054  062601                            ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
 (1)  017056  000764                            BR      2$              ;;LOOP
 (1)  017060  005726                    3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
 (1)  017062  010166  000012                    MOV     R1,12(SP)       ;;SAVE THE RESULT
 (1)  017066  010237  017102                    MOV     R2,$HIOCT
 (3)  017072  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
 (3)  017074  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
 (3)  017076  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
 (1)  017100  000002                            RTI                     ;;RETURN
 (1)  017102  000000                    $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
1714                                    .SBTTL  TTY INPUT ROUTINE
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    .ENABL  LSB
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
 (1)                                    ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
 (1)                                    ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
 (1)                                    ;*WHEN OPERATING IN TTY FLAG MODE.
 (1)  017104  022737  000176  001140    $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
 (1)  017112  001074                            BNE     15$             ;;BRANCH IF NO
 (1)  017114  105777  162024                    TSTB    @$TKS           ;;CHAR THERE?
 (1)  017120  100071                            BPL     15$             ;;IF NO, DON'T WAIT AROUND
 (1)  017122  117746  162020                    MOVB    @$TKB,-(SP      ;;SAVE THE CHAR
 (1)  017126  042716  177600                    BIC     #^C177,(SP,     ;;STRIP-OFF THE ASCII
 (1)  017132  022726  000007                    CMP     #7,(SP)+        ;;IS IT A CONTROL G?
 (1)  017136  001062                            BNE     15$             ;;NO, RETURN TO USER
```

```
(1)  017140  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  017146  001456                          BEQ     15$             ;;BRANCH IF YES
(1)
(1)  017150  104401  017756                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  017154  104401  017763          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  017160  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  017164  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  017166  104401  017774                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  017172  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
(1)  017174  005046                          CLR     -(SP)           ;;THE NEW SWR
(1)  017176  105777  161742          7$:     TSTB    @$TKS           ;;CHAR THERE?
(1)  017202  100375                          BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  017204  117746  161736                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  017210  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1)  017214  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  017220  001005                          BNE     10$             ;;BRANCH IF NOT
(1)  017222  104401  017751                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  017226  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  017232  000757                          BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  017234  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
(1)  017240  001022                          BNE     16$             ;;BRANCH IF NO
(1)  017242  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  017246  001403                          BEQ     11$             ;;BRANCH IF YES
(1)  017250  016677  000002  161662          MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  017256  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
(1)  017262  104401  001165          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  017266  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  017274  001003                          BNE     15$             ;;BRANCH IF NOT
(1)  017276  012777  000100  161640          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  017304  000002                  15$:    RTI                     ;;RETURN
(1)  017306  004737  016732          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  017312  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
(1)  017316  002420                          BLT     18$             ;;BRANCH IF YES
(1)  017320  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
(1)  017324  003015                          BGT     18$             ;;BRANCH IF YES
(1)  017326  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  017332  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  017336  001403                          BEQ     17$             ;;BRANCH IF YES
(1)  017340  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  017342  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
(1)  017344  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
(1)  017346  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  017352  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  017356  000707                          BR      7$              ;;GET THE NEXT ONE
(1)  017360  104401  001164          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  017364  000720                          BR      20$             ;;SIMULATE CONTROL-U
(1)                                  .DSABL  LSB
(1)
(1)
(2)                                  ;;**************************************************************
```

```
(1)                                     ;*THIS ROUTINE  ILL 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)  017366  011646                     $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  017370  016666  000004  000002             MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  017376  105777  161542             1$:     TSTB    @$TKS           ;;WAIT FOR
(1)  017402  100375                             BPL     1$              ;;A CHARACTER
(1)  017404  117766  161536  000004             MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  017412  042766  177600  000004             BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  017420  026627  000004  000023             CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  017426  001013                             BNE     3$              ;;BRANCH IF NO
(1)  017430  105777  161510             2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  017434  100375                             BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  017436  117746  161504                     MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  017442  042716  177600                     BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  017446  022627  000021                     CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  017452  001366                             BNE     2$              ;;IF NOT DISCARD IT
(1)  017454  000750                             BR      1$              ;;YES, RESUME
(1)  017456  026627  000004  000140     3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  017464  002407                             BLT     4$              ;;BRANCH IF YES
(1)  017466  026627  000004  000175             CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  017474  003003                             BGT     4$              ;;BRANCH IF YES
(1)  017476  042766  000040  000004             BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
(1)  017504  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)  017506  010346                     $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  017510  005046                             CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  017512  012703  017742             1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  017516  022703  017757             2$:     CMP     #$TTYIN+7,R3    ;;BUFFER FULL?
(1)  017522  101456                             BLOS    4$              ;;BR IF YES
(1)  017524  104410                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  017526  112613                             MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  017530  122713  000177             10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  017534  001022                             BNE     5$              ;;BR IF NO
(1)  017536  005716                             TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  017540  001007                             BNE     6$              ;;BR IF NO
(1)  017542  112737  000134  017740             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  017550  104401  017740                     TYPE    ,9$
(1)  017554  012716  177777                     MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  017560  005303                     6$:     DEC     R3              ;;BACKUP BY ONE
(1)  017562  020327  017742                     CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  017566  103434                             BLO     4$              ;;BR IF YES
(1)  017570  111337  017740                     MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  017574  104401  017740                     TYPE    ,9$             ;;GO TYPE
(1)  017600  000746                             BR      2$              ;;GO READ ANOTHER CHAR.
(1)  017602  005716                     5$:     TST     (SP)            ;;RUBOUT KEY SET?
```

```
(1)  017604  001406                        BEQ     7$              ;;BR IF NO
(1)  017606  112737  000134  017740        MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  017614  104401  017740                TYPE    ,9$
(1)  017620  005016                        CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  017622  122713  000025        7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  017626  001003                        BNE     8$              ;;BR IF NO
(1)  017630  104401  017751                TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
(1)  017634  000726                        BR      1$              ;;GO START OVER
(1)  017636  122713  000022        8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "^R"?
(1)  017642  001011                        BNE     3$              ;;BRANCH IF NO
(1)  017644  105013                        CLRB    (R3)            ;;CLEAR THE CHARACTER
(1)  017646  104401  001165                TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
(1)  017652  104401  017742                TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  017656  000717                        BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  017660  104401  001164        4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  017664  000712                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  017666  111337  017740        3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  017672  104401  017740                TYPE    ,9$
(1)  017676  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  017702  001305                        BNE     2$              ;;LOOP IF NOT RETURN
(1)  017704  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  017710  104401  001166                TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  017714  005726                        TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  017716  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  017720  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  017722  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  017730  012766  017742  000004        MOV     #$TTYIN,4(SP)
(1)  017736  000002                        RTI                     ;;RETURN
(1)  017740     000                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  017741     000                        .BYTE   0               ;;TERMINATOR
(1)  017742  000007                $TTYIN: .BLKB   7               ;;RESERVE 7 BYTES FOR TTY INPUT
(1)  017751     136  005525  000012 $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
(1)  017756  043536  005015     000 $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
(1)  017763     015  051412  051127 $MSWR:  .ASCIZ  <15><12>/SWR - /
(1)  017770  036440  000040
(1)  017774  020040  042516  020127 $MNEW:  .ASCIZ  /  NEW   /
(1)  020002  020075     000
(1)          020006                .EVEN
1715
```

```
1717
1718                                    .SBTTL  APT COMMUNICATIONS ROUTINE
 (1)
 (2)                                    ;;*****************************************************************
 (1)  020006  112737  000001  020252  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
 (1)  020014  112737  000001  020250  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
 (1)  020022  000403                   BR      $ATYC
 (1)  020024  112737  000001  020252  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
 (1)  020032                           $ATYC:
 (3)  020032  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 (3)  020034  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 (1)  020036  105737  020250                   TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
 (1)  020042  001450                           BEQ     5$              ;;IF NOT:  BR
 (1)  020044  122737  000001  001210           CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
 (1)  020052  001031                           BNE     3$              ;;IF NOT:  BR
 (1)  020054  132737  000100  001211           BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
 (1)  020062  001425                           BEQ     3$              ;;IF NOT:  BR
 (1)  020064  017600  000004                   MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
 (1)  020070  062766  000002  000004           ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 (1)  020076  005737  001170           1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
 (1)  020102  001375                           BNE     1$              ;;IF NOT:  WAIT
 (1)  020104  010037  001204                   MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
 (1)  020110  105720                   2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
 (1)  020112  001376                           BNE     2$
 (1)  020114  163700  001204                   SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
 (1)  020120  006200                           ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
 (1)  020122  010037  001206                   MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
 (1)  020126  012737  000004  001170           MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
 (1)  020134  000413                           BR      5$
 (1)  020136  017637  000004  020162   3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
 (1)  020144  062766  000002  000004           ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
 (3)  020152  013746  177776                   MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
 (1)  020156  004737  016520                   JSR     PC,$TYPE        ;;CALL TYPE MACRO
 (1)  020162  000000                   4$:     .WORD   0
 (1)  020164                           5$:
 (1)  020164  105737  020252           10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
 (1)  020170  001416                           BEQ     12$             ;;IF NOT:  PR
 (1)  020172  005737  001210                   TST     $ENV            ;;RUNNING UNDER APT?
 (1)  020176  001413                           BEQ     12$             ;;IF NOT:  BR
 (1)  020200  005737  001170           11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
 (1)  020204  001375                           BNE     11$             ;;IF NOT:  WAIT
 (1)  020206  017637  000004  001172           MOV     @4(SP),$FATAL   ;;GET ERROR #
 (1)  020214  062766  000002  000004           ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 (1)  020222  005237  001170                   INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
 (1)  020226  105037  020252           12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
 (1)  020232  105037  020251                   CLRB    $LFLG           ;;CLEAR LOG FLAG
 (1)  020236  105037  020250                   CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
 (3)  020242  012601                           MOV     (SP)+,R1        ;;POP STACK INTO R1
 (3)  020244  012600                           MOV     (SP)+,R0        ;;POP STACK INTO R0
 (1)  020246  000207                           RTS     PC              ;;RETURN
 (1)  020250     000                   $MFLG:  .BYTE   0               ;;MESSG. FLAG
 (1)  020251     000                   $LFLG:  .BYTE   0               ;;LOG FLAG
 (1)  020252     000                   $FFLG:  .BYTE   0               ;;FATAL FLAG
 (1)          020254                           .EVEN
 (1)          000200                   APTSIZE=200
 (1)          000001                   APTENV=001
```

```
  (1)           000000
  (1)           000040
 1719
 (1)
  (2)
  (1)                                   ;;**********************************************************
  (1)                                   ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
  (1)                                   ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
  (1)                                   ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
  (1)                                   ;*GO TO THAT ROUTINE.
  (1)
  (1)  020254  010046                   $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
  (1)  020256  016600  000002                   MOV     2(SP),R0        ;;GET TRAP ADDRESS
  (1)  020262  005740                           TST     -(R0)           ;;BACKUP BY 2
  (1)  020264  111000                           MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
  (1)  020266  006300                           ASL     R0              ;;POSITION FOR INDEXING
  (1)  020270  016000  020310                   MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
  (1)  020274  000200                           RTS     R0              ;;GO TO ROUTINE
  (1)
  (1)
  (1)                                   ;;THIS IS USE  O HANDLE THE "GETPRI" MACRO
  (1)
  (1)  020276  011646                   $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
  (1)  020300  016666  000004  000002           MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
  (1)  020306  000002                           RTI                     ;;RESTORE THE PSW
  (1)
  (3)                                   .SBTTL  TRAP TABLE
  (3)
  (3)                                   ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
  (3)                                   ;*BY THE "TRAP" INSTRUCTION.
  (3)
  (5)                                   ;       ROUTINE
  (3)                                   ;       -------
  (3)  020310  020276                   $TRPAD: .WORD   $TRAP2
  (3)  020312  016520                           $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
  (3)  020314  016076                           $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
  (3)  020316  016052                           $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
  (3)  020320  016112                           $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
  (3)  020322  011360                           $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
  (1)
  (3)  020324  017154                           $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
  (1)
  (3)  020326  017104                           $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
  (3)  020330  017366                           $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
  (3)  020332  017506                           $RDLIN  ;;CALL=RDLIN    TRAP+11('04411) TTY TYPEIN STRING ROUTINE
  (3)  020334  017002                           $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
 1720
 1721  020336  000000                   BUFFER: 0                       ;10 WORD BUFFER FOR THE COULTER INTERFACE TEST
 1722          000001                   .END
```

```
ABASE = 167770            392#    409
ACDW1 = 000000            409
ACDW2 = 000000            409
ACPUOP= 000000            409
ADBR    013302           1619    1655#
ADCS    013276           1609*   1614*   16'7    1653#
ADCS1   013300           1608*   1654#
ADDW0 = 000000            409
ADDW1 = 000000            409
ADDW10= 000000            409
ADDW11= 000000            409
ADDW12= 000000            409
ADDW13= 000000            409
ADDW14= 000000            409
ADDW15= 000000            409
ADDW2 = 000000            409
ADDW3 = 000000            409
ADDW4 = 000000            409
ADDW5 = 000000            409
ADDW6 = 000000            409
ADDW7 = 000000            409
ADDW8 = 000000            409
ADDW9 = 000000            409
ADEVCT= 000000            409
ADEVM = 000000            409
AENV  = 000000            409
AENVM = 000000            409
AFATAL= 000000            409
AMADR1= 000000            409
AMADR2= 000000            409
AMADR3= 000000            409
AMADR4= 000000            409
AMAMS1= 000000            409
AMAMS2= 000000            409
AMAMS3= 000000            409
AMAMS4= 000000            409
AMSGAD= 000000            409
AMSGLG= 000000            409
AMSGTY= 000000            409
AMTYP1= 000000            409
AMTYP2= 000000            409
AMTYP3= 000000            409
AMTYP4= 000000            409
APASS = 000000            409
APRIOR= 000000            409
APTCSU= 000040           171     1718#
APTENV= 000001           1697    1711    1718#
APTSIZ= 000200            522    1718#
APTSPO= 000100           1711    1718#
ASWREG= 000000            409
ATESTN= 000000            409
AUNIT = 000000            409
AUSWR = 000000            409
AVECT1= 100300            393#    409
AVECT2= 000000            409
BASEBA  001362            467#    523*    529     658     1398
```

```
BASEBR  001366            469#    526*    527*    688
BASEIV  001364            468#    524*    525*    659    1399
BEGIN   001454            398     511#    598     1437
BEGIN1  001510            399     520#
BIT0  = 000001            394#    743     756     954     988     1010    1246    1283
BIT00 = 000001            394#
BIT01 = 000002            394#
BIT02 = 000004            394#
BIT03 = 000010            394#
BIT04 = 000020            394#
BIT05 = 000040            394#
BIT06 = 000100            394#
BIT07 = 000200            394#
BIT08 = 000400            394#    1696
BIT09 = 001000            394#    1696    1697
BIT1  = 000002            394#
BIT10 = 002000            394#    870     1215    1242    1279    1310
BIT11 = 004000            394#    1696
BIT12 = 010000            394#    591     924
BIT13 = 020000            394#    1193    1609    1697
BIT14 = 040000            394#    827     828     855     1318    1325    1411    1696
BIT15 = 100000            394#    818     819     855     1108
BIT2  = 000004            394#
BIT3  = 000010            394#
BIT4  = 000020            394#
BIT5  = 000040            394#    1144    1170    1353    1370
BIT6  = 000100            394#    844     845     855     1169    1180    1223    1230    1448
BIT7  = 000200            394#    835     836     855     1096    1248
BIT8  = 000400            394#
BIT9  = 001000            394#
BPTVEC= 000014            394#
BRLEV1  001436            493#    1342*   1343*   1349
BRLEV2  001440            494#    954*    957     977     982*    1344*   1366
BRLEV3  001442            495#    605     955*    956*    965     975     1010*   1014    1021    1030    1031*   1246*   1250
                          1255    1256    1257    1267    1268    1275*   1283*   1287    1292    1293    1294    1304*   1691
BUFFER  020336            1447    1467    1721#
BUSTRP  013463            540     1435    1660#
BYPASS  011140            1390#
BYPAS1  011226            1393    1397    1402#   1405
CKSWR = 104407            1696    1697    1719#
COULTJ  001464            402     514#
COULTR  011766            401     1440#
CPU     001370            470#    1454
CR    = 000015            394#    171*
CRLF  = 000200            394#    1711
DDISP = 177570            394#    409     522
DEPCNT  014227            647     1672#
DF0     015224            415     421     427     433     439     445     453     459     465     1692#
DH1     014671            413     1683#
DH10    015102            457     463     1687#
DH2     014741            419     1684#
DH3     015007            425     1685#
DH4     015055            431     437     443     451     1686#
DIOBRL  001434            492#    688*    1336*   1339    1342    1344
DISPLA  001142            409#    522*    1696*   1697*
DISPRE  000174            398#    522
```

```
DRADD   001376          475#    658*    673     674     676     678     1394*   1398*   1689    1690    1691
DRIV    001400          476#    630*    659*    680     681     683     685     1395*   1399*
DRT21   007200          873     1118#
DSWR  = 177570          394#    409     522
DT1     015162          414     420     426     1689#
DT10    015206          458     464     1691#
DT4     015176          432     438     444     452     1690#
EPTVEC= 000030          394#    522*
EM1     014251          412     1673#
EM10    014531          456     1680#
EM11    014614          462     1681#
EM2     014302          418     1674#
EM3     014332          424     1675#
EM4     014363          430     1676#
EM5     014415          436     1677#
EM6     014450          442     1678#
EM7     014475          450     1679#
ERRVEC= 000004          394#    522*    528*    549*    550*    1696*
EXTCNT  011710          1411*   1420*   1425#
EXTTMP  011706          1414*   1415    1418*   1419    1424#   1453*   1457*   1473*   1483*
EXTTST  011604          400     1409#
GNS   = ****** U        398     557     558     562     621     624     1719
GRBHIO  001416          485#    678*    679*    807*
GRDAI   001412          483#    674*    675*    695*    817*    875*    878     886*    890     899*    903     911*    915
                        930*    932     940*    942     964     974     990*    999     1004*   1012*   1025    1030*   1041*
                        1049*   1066*   1074*   1086*   1090    1112    1127    1225    1253*   1268*   1290*   1320    1414
                        1415*   1418    1419*   1459    1461*
GRDIO   001414          484#    676*    677*    696*    700*    701     708*    710     716*    717     724*    725     733*
                        734     745*    746*    747     758*    759*    760     771*    774*    775     782*    785*    786
                        794*    795*    796     806*    808     816*    874*    877*    885*    889*    898*    902*    910*
                        914*    929*    931*    941*    963*    973*    989*    997*    998*    1011*   1022*   1023*   1040*
                        1049*   1065*   1074*   1085*   1087*   1[      1110*   1126*   1224*   1252*   1255*   1256*   1257*
                        1267*   1289*   1292*   1293*   1294*   1_      1412*   1413*   1416*   1417*
GRIVA   001424          488#    680*    692*    1122*   1140*   11.     1165*   1175*   1189*   1219*   1238*   1316*   1330*
                        1348*   1361*   1365*   1375*   1386*   1444*
GRIVB   001430          490#    683*    684*    692*    1124*   1142*   1167*   1191*   1201*   1221*   1238*   1314*   1330*
                        1361*   1386*
GRIVSA  001426          489#    681*    682*    692*    1123*   1141*   1166*   1190*   1220*   1238*   1316    1317*   1330*
                        1361*   1386*   1445*
GRIVSB  001432          491#    685*    686*    692*    1125*   1143*   1168*   1192*   1222*   1238*   1315*   1330*   1361*
                        1386*
GRSTAT  001410          482#    673*    694*    819*    820     826*    828*    829     836*    837     843*    845*    846
                        853*    855*    857     1097*   1100    1109*   1113    1120*   1132*   1139*   1144*   1164*   1169*
                        1170*   1179    1188*   1193*   1218*   1223*   1230*   1237*   1254*   1258*   1261    1270*   1271
                        1291*   1295*   1297    1313*   1318*   1325*   1332*   1347*   1353*   1364*   1370*   1443*   1448*
GTSWR = 104406          1719#
HT    = 000011          394#    1711
H322J   001476          403     517#
INADR   012346          1493    1496    1499    1517#   1568    1571    1574
INADRH  013345          1520    1657#
INDADR  013410          1523    1658#
INTBIT  001422          487#    566*    573*    582     1250    1287
IOTEST  003102          556     569     576     606     657#    1706
IOTST1  003156          670#
IOTTS1  003314          672     688#
IOTVEC= 000020          394#    522*
```

```
JUMPER  001452          499#    512*    514*    517*    563     570     1350    1367
LF    = 000012          394#    1711
LOCA    012300         1495    1498    1501    1503#    1515
LOCBOX  012226          404    1490#
LOCHDR  013304         1492    1656#
LOC1    001402          479#   1494    1506    1569    1586    1587    1588    1589    1590
LOC1Y   012532         1494    1503    1560#   1569    1584
LOC2    001404          480#   1497    1510    1572    1593    1594    1595    1596    1597
LOC2Y   012534         1497    1507    1561#   1572    1591
LOC3    001406          481#   1500    1514    1575    1600    1601    1602    1603    1604
LOC3Y   012536         1500    1511    1562*   1575    1598
LOUP1   012460         1505    1509    1513    1543#
MINSIN  001446          497#    567*    574*    585     592     690     938
MSGRUS  013544         1471*   1485*   1486    1662#
MSPNT1  013553         1472    1663#
NBEXT   001374          473#    660*   1390    1396*   1400*
NMBEXT  001372          472#    548*    559     660    1400
NOTLCH  001420          486#    565*    572*    579     651     927     955     960     992    1014    1020    1037    1063
ODDJMP  001444          496#    689*    690*    691*    879     888     891     901     904     913     916     922     925
                        958     994    1016    1038    1062    1091

PIRQ  = 177772          394#
PIRQVE= 000240          394#
PR0   = 000000          394#
PR1   = 000040          394#
PR2   = 000100          394#
PR3   = 000140          394#
PR4   = 000200          394#
PR5   = 000240          394#
PR6   = 000300          394#
PR7   = 000340          394#
PS    = 177776          394#   1449*
PSW   = 177776          394#   1121*   1145*   1155*   1161*   1171*   1177*   1186*   1194*   1203*   1209*   1226*   1236*
                       1321*   1331*   1349*   1352*   1362*   1366*   1369*   1376*   1382*   1387*
PWRMSG  016452         1706    1707#
PWRVEC= 000024          394#    522*   1430*   1431*   1706*
RBEG    001514          513     516     519     521#
RBEG1   002132          551#
RBEG2   003126          631     661#   1403
RDCHR = 104410         1714    1719#
RDLIN = 104411         1524    1713    1719#
RDOCT = 104412          643    1719#
RESVEC= 000010          394#
RUNMSG  013524         1446    1661#
SETADD  003166          671     673#   1410    1442
SETTRP  011712         1428#   1441    1491    1566
STACK = 001100          394#    522    1409    1440    1490    1565
STKLMT= 177774          394#
SWDPOB  014142          604    1671#
SWINTB  013652          581    1668#
SWNLB   013554          578     639    1667#
SWPOSB  013750          584    1669#
SWR     001140          409#    522*    538     541     619     626     640     644*    649     870    1215    1242    1279
                       1310    1392    1696    1697    1706*   1714*
SWREG   000176          398#    522     640    1714
SWTRAB  014044          587    1670#
SW0   = 000001          394#
```

```
SW00  = 000001            394#
SW01  = 000002            394#
SW02  = 000004            394#
SW03  = 000010            394#
SW04  = 000020            394#
SW05  = 000040            394#
SW06  = 000100            394#
SW07  = 000200            394#
SW08  = 000400            394#
SW09  = 001000            394#
SW1   = 000002            394#
SW10  = 002000            394#
SW11  = 004000            394#
SW12  = 010000            394#   1392
SW13  = 020000            394#    538     619
SW14  = 040000            394#
SW15  = 100000            394#    541     626
SW2   = 000004            394#
SW3   = 000010            394#
SW4   = 000020            394#
SW5   = 000040            394#
SW6   = 000100            394#
SW7   = 000200            394#
SW8   = 000400            394#
SW9   = 001000            394#
TALK    003016            577     580     583     586     603     636#
TBITVE= 000014            394#
TKVEC = 000060            394#
TPVEC = 000064            394#
TRANST  001450            498#    568*    575*    588     594     691
TRAPVE= 000034            394#    522*   1428*   1429*
TRTVEC= 000014            394#
TSTNUM  015242           1689    1690    1691    1693#   1697*
TST1    003372            693#
TST10   003734            754#
TST11   004020            769#
TST12   004116            777     780#
TST13   004214            788     791#
TST14   004302            803#
TST15   004370            815#
TST16   004440            822     825#
TST17   004502            831     834#
TST2    003410            698#
TST20   004540            839     842#
TST21   004602            848     852#
TST22   004644            858     869#
TST23   004742            881     884#
TST24   005026            893     897#
TST25   005112            906     909#
TST26   005176            918     921#
TST27   005356            923     951#
TST3    003446            703     706#
TST30   005536            986#
TST31   005654           1008#
TST32   006014           1034#
TST33   006410           1056    1059#
```

```
TST34   007004          1081    1084#
TST35   007062          1092    1095#
TST36   007132          1102    1107#
TST37   007200          1114    1118#
TST4    003504           711     714#
TST40   007312          1133    1138#
TST41   007424          1149    1158    1163#
TST42   007562          1181    1183    1187#
TST43   007674          1198    1206    1212#
TST44   010062          1217    1241#
TST45   010252          1243    1278#
TST46   010404          1280    1307#
TST47   010566          1312    1334#
TST5    003542           719     722#
TST50   010646          1346#
TST51   010764          1363#
TST52   011130          1389#
TST6    003600           727     730#
TST7    003656           741#
TYPDS = 104405          1405    1719#
TYPE  = 104401           540     557     558     562     621     624     638     642     646    1405    1406    1435    1446
                        1486    1492    1520    1523    1567    1697    1700    1703    1706    1711    1714    1719#
TYPOC = 104402           623    1522    1700    1714    1719#
TYPON = 104404          1719#
TYPOS = 104403           560    1719#
UNEXPT  003076           654#    692    1238    1330    1361    1386
VECTRP  002572           549     609#
XLOADJ  013066          1586    1587    1588    1589    1590    1593    1594    1595    1596    1597    1600    1601    1602
                        1603    1604    1608#
XLOHDR  013426          1567    1659#
XL01AD  012540           405    1565#
XL010   012612          1570    1573    1576    1584#   1605
$APTHD  001000           408#
$ASTAT= ****** U        1718
$ATYC   020032          1718#
$ATY1   020006          1718#
$ATY3   020014          1711    1718#
$ATY4   020024          1697    1718#
$AUTOB  001134           409#    1714
$BASE   001244           409#     523
$BDADR  001122           409#
$BDDAT  001126           409#     701*    702     710*    717*    718     725*    726     734*    735     747*    748     760*
                         761*    762     775*    776     786*    787     796*    797     808*    809     820*    821     829*
                         830     837*    838     846*    847     857*    878*    879*    880     890*    891*    892     903*
                         904*    905     915*    916*    917     932*    933     942*    943     964*    965*    966     974*
                         975*    977     999*   1000    1025*   1026    1051*   1055    1076*   1080    1090*   1091*   1100*
                        1101    1113*   1179*   1180    1247*   1271*   1284*   1689    1691
$CDW1   001250           409#
$CHARC  016776          1711#*
$CKSWR  017104          1714#    1719
$CMTAG  001100           409#     522     667
$CM3  = 000000           409#
$CNTLG  017756          1714#
$CNTLU  017751          1714#
$CPUOP  001216           409#
$CRLF   001165           409#    1697    1700    1711    1714
```

```
$DBLK   011574          1406#
$DEVCT  001200           409#
$DEVM   001246           409#
$DOAGN  011334          1405#
$DTBL   011564          1406#
$ENDAD  011324           406    1405#   1697
$ENDCT  011272           522    1405#
$ENDMG  011343          1405#
$ENULL  011340          1405#
$ENV    001210           409#   1697    1711    1718
$ENVM   001211           409#    522    1711    1718
$EOP    011236          1401    1405#
$EOPCT  011264           522*   1405#
$ERFLG  001103           409#   1696*   1697*
$ERMAX  001115           409#    522*   1696*
$ERROR  015524           522    1697#
$ERRPC  001116           409#   1689    1690    1691    1697*   1700
$ERRTB  001252           409#   1700
$ERRTY  015716          1697    1700#
$ERTTL  001112           409#   1697*
$ESCAP  001162           409#    522*   1696*   1697
$ETABL  001210           409#
$ETEND  001252           408     409#
$FATAL  001172           409#   1718*
$FFLG   020252          1718#*
$FILLC  001156           409#   1711
$FILLS  001155           409#   1711
$GDADR  001120           409#
$GDDAT  001124           409#    591*    592     594     600*    699*    702     707*    715*    718     723*    726     732*
                         733     735     738*    743*    746     748     752*    756*    759     762     766*    770*    771
                         776     781*    782     787     793*    794     795     797     800*    805*    806     807     809
                         812*    818*    821     827*    830     835*    838     844*    847     854*    876*    880     887*
                         888*    889     892     900*    901*    902     905     912*    913*    914     917     924*    925
                         927     929     931     933     938     941     943     948*    957*    958     960     963     966
                         973     976*    988*    992     994     997     1000    1004    1005*   1016    1019*   1020*   1021*
                        1022    1026    1036*   1037*   1038*   1039    1055    1061*   1062*   1063*   1064    1080    1088*
                        1096*   1101    1108*   1178*   1248*   1269*   1285*   1689    1691
$GET42  011314          1405#
$GTSWR  017154          1714#   1719
$HD   = 000000           390
$HIBTS  001000           408#
$HIOCT  017102          1713#*
$ICNT   001104           409#   1696*
$ILLUP  016444          1706#
$INTAG  001135           409#   1714
$ITEMB  001114           409#   1697*   1700
$LF     001166           409#   1697    1711    1714
$LFLG   020251          1718#*
$LPADR  001106           409#    522*   1696*
$LPERR  001110           409#    522*    731*    742*    755*    792*    804*    952*    987*   1009*   1245*   1282*   1696*
                        1697
$MADR1  001222           409#
$MADR2  001226           409#
$MADR3  001232           409#
$MADR4  001236           409#
$MAIL   001170           408     409#    522     1696    1697    1711
```

```
$MAMS1  001220            409#
$MAMS2  001224            409#
$MAMS3  001230            409#
$MAMS4  001234            409#
$MBADR  001002            408#
$MFLG   020250           1718#*
$MNEW   017774           1714#
$MSGAD  001204            409#   1718*
$MSGLG  001206            409#   1718*
$MSGTY  001170            409#   1718*
$MSWR   017753            642    1714#
$MTYP1  001221            409#
$MTYP2  001225            409#
$MTYP3  001231            409#
$MTYP4  001235            409#
$MXCNT  015522           1696#
$NULL   001154            409#   1711
$NWTST= 000001            693#    698#    706#    714#    722#    730#    741#    754#    769#    780#    791#    803#    815#
                          825#    834#    842#    852#    869#    884#    897#    909#    921#    951#    986#   1008#   1034#
                         1059#   1084#   1095#   1107#   1118#   1138#   1163#   1187#   1212#   1241#   1278#   1307#   1334#
                         1346#   1363#   1389#
$OCNT   016274           1703#*
$OMODE  016276           1703#*
$OVER   015506           1696#
$PASS   001176            409#    522*   1405*   1696
$PASTM  001006            408#
$PWRAD  016440           1706#
$PWRDN  016300             522    1430    1706#
$PWRMG  016434           1706#
$PWRUP  016352           1706#
$QUES   001164            409#   1697    1711    1714
$RDCHR  017366           1714#   1719
$RDDEC= ****** U         1719
$RDLIN  017506           1714#   1719
$RDOCT  017002           1713#   1719
$RDSZ = 000007           1714#
$RTNAD  011336           1405#
$R2A  = ****** U         1719
$SAVRE= ****** U         1719
$SAVR6  016450           1706#*
$SCOPE  015244             522   1696#
$SETUP= 000137             447#    522    1405    1696    1697    1714
$STUP = 177777             447#
$SVLAD  015452           1696#
$SVPC = 000240             406#
$SWR  = 165400             383#    390     396     409     522     693     698     706     714     722     730     741     754
                           769     780     791     803     815     825     834     842     852     869     884     897     909
                           921     951     986    1008    1034    1059    1084    1095    1107    1118    1138    1163    1187
                          1212    1241    1278    1307    1334    1346    1363    1389    1405    1696    1697    1706
$SWREG  001212            409#    522
$SWRMK= 000000             396    1696
$TESTN  001174            409#   1696*
$TIMES  001160            409#    522*    706*    730*    791*    803*    852*    921*   1084*   1118*   1334*   1389*   1405*
                         1696*
$TKB    001146            409#   1714
$TKS    001144            409#   1714*
```

```
$TN    = 000053          384#    390     693#    698#    703     706#    711     714#    719     722#    727     730#    741#
                         754#    769#    777     780#    788     791#    803#    815#    822     825#    831     834#    839
                         842#    848     852#    858     869#    881     884#    893     897#    906     909#    918     921#
                         923     951#    986#    1008#   1034#   1056    1059#   1081    1084#   1092    1095#   1102    1107#
                        1114    1118#   1133    1138#   1149    1158    1163#   1181    1183    1187#   1198    1206    1212#
                        1217    1241#   1243    1278#   1280    1307#   1312    1334#   1346#   1363#   1389#
$TPB     001152          409#   1711*
$TPFLG   001157          409#   1711
.TPS     001150          409#   1711
$TRAP    020254          522    1428    1719#
$TRAP2   020276         1719#
$TRP   = 000013         1719#
$TRPAD   020310         1719#
$TSTM    001004          408#
$TSTNM   001102          409#    872*    1405*   1696*   1697
$TTYIN   017742         1714#
$TYPBN=  ****** U       1719
$TYPDS   011360         1406#   1719
$TYPE    016520         1711#   1718    1719
$TYPEC   016732         1711#   1714
$TYPEX   017000         1711#
$TYPOC   016076         1703#   1719
$TYPON   016112         1703#   1719
$TYPOS   016052         1703#   1719
$UNIT    001202          409#
$UNITM   001010          408#
$USWR    001214          409#
$VECT1   001240          409#    524     526
$VECT2   001242          409#
$XTSTR   015256         1696#
$$GET4=  000000         1405#
$OFILL   016275         1703#*
$40CAT=  ****** U       1696    1697
.      = 020340          398#    406#    407#    408#    409#    522     557#    558#    621#    1405    1406#   1688#   1696
                        1697    1700#   1706    1708#   1711    1714#   1718#
.$ASTA=  ****** U       1718
.$X    = 001000          408#
```

ADJXL	1578#	1586	1587	1588	1589	1590	1593	1594	1595	1596	1597	1600	1601	1602	1603
	1604														
CLERIN	866#	875	886	899	911	1012	1041	1066	1086	1253	1290				
CLEROT	862#	874	885	898	910	998	1011	1023	1040	1049	1065	1074	1085	1252	1289
	1412	1416													
CLRVCT	501#	692	1238	1330	1361	1386									
COMMEN	394#														
ENDCOM	394#														
ERROR	394#	596	654	704	712	720	728	737	750	764	778	789	799	811	823
	832	840	849	859	882	894	907	919	935	945	968	979	1002	1028	1057
	1082	1093	1103	1115	1134	1136	1148	1160	1174	1182	1185	1197	1208	1232	1234
	1263	1273	1299	1327	1338	1341	1358	1383							
ESCAPE	394#														
GETPRI	394#														
GETSWR	387#	394#													
MULT	394#														
NEWTST	394#	693	698	706	714	722	730	741	754	769	780	791	803	815	825
	834	842	852	869	884	897	909	921	951	986	1008	1034	1059	1084	1095
	1107	1118	1138	1163	1187	1212	1241	1278	1307	1334	1346	1363	1389		
POP	394#	1406	1706	1713	1718										
PUSH	394#	1406	1706	1713	1718										
REPORT	394#														
SCOPE	394#	693	698	706	714	722	730	741	754	769	780	791	803	815	825
	834	842	852	869	884	897	909	921	951	986	1008	1034	1059	1084	1095
	1107	1118	1138	1163	1187	1212	1241	1278	1307	1334	1346	1363	1389	1405	
SETPRI	394#														
SETTRA	1719#														
SETUP	394#	522													
SKIP	394#	703	711	719	727	736	777	788	822	831	839	848	858	881	893
	906	918	923	967	978	1001	1027	1056	1081	1092	1102	1114	1133	1149	1158
	1181	1183	1198	1206	1217	1233	1243	1251	1262	1272	1280	1288	1298	1312	1328
	1359	1384													
SLASH	394#														
SPACE	394#														
STARS	394#	406	408	409	693	698	706	714	722	730	741	754	769	780	791
	803	815	825	834	842	852	869	884	897	909	921	951	986	1008	1034
	1059	1084	1095	1107	1118	1138	1163	1187	1212	1241	1278	1307	1334	1346	1363
	1389	1405	1406	1696	1697	1700	1703	1706	1711	1713	1714	1718	1719		
SWRSU	394#	522#													
TRMTRP	1719#														
TYPBIN	394#														
TYPDEC	394#	1405													
TYPNAM	394#														
TYPNUM	394#														
TYPOCS	394#														
TYPOCT	394#	1700	1714												
TYPTXT	394#	557	558	562	621	624									
$$CMRE	409#														
$$CMTM	409#														
$$ESCA	394#														
$$NEWT	394#	693	698	706	714	722	730	741	754	769	780	791	803	815	825
	834	842	852	869	884	897	909	921	951	986	1008	1034	1059	1084	1095
	1107	1118	1138	1163	1187	1212	1241	1278	1307	1334	1346	1363	1389		
$$SET	1719#														
$$SETM	522#														
$$SKIP	394#	703	711	719	727	777	788	822	831	839	848	858	881	893	906

```
          918     923    1056    1081    1092    1102    1114    1133    1149    1158    1181    1183    1198    1206    1217
         1243    1280    1312
.EQUAT    385#    394
.HEADE    385#    390
.SETUP    386#    447
.SWRHI    387#    396
.SWRLO    396#
.$ACTI    388#    406
.$APTB    388#    409#
.$APTH    388#    408
.$APTY    388#   1718
.$CATC    385#    398
.$CATA    385#    409
.$EOP     385#   1405
.$ERRO    385#   1697
.$ERRT    387#   1700
.$PARM    386#
.$POWE    386#   1706
.$RDOC    387#   1713
.$READ    386#   1714
.$SAVE    386#
.$SCOP    386#   1696
.$SPAC    386#
.$SWDO    386#
.$TRAP    386#   1719
.$TYPD    387#   1406
.$TYPE    385#    386#   1711
.$TYPO    385#   1703

. ABS.  020340     000     CON   RW    REL   GBL   D

 ERRORS DETECTED:  0

 CZDRGE,CZDRGE/CR=CZDRGE
 RUN-TIME. 26 13 1 SECONDS
 RUN-TIME RATIO: 76/42=1.7
 CORE USED:  25K  (49 PAGES)
```