

DR11-K

DIGITAL I/O TEST
MD-11-DZDRG-D

EP-DZDRG-D-DL-B
COPYRIGHT 1977
FICHE 1 OF 1

MAR 1977
digital
MADE IN USA

This microfiche card contains a grid of frames, each displaying digital test data. The data is organized into columns and rows, with some frames showing waveforms and others showing binary sequences. The frames are arranged in a grid that is approximately 12 frames wide and 15 frames high. The data is printed in a small, high-contrast font, typical of microfiche technology. The frames contain various types of information, including what appears to be test results, timing diagrams, and possibly control signals. The overall layout is dense and structured, typical of a technical test document.

801

EOF1DZDRBASE0411
DZDRGD.F11

08018000LOGIC 729224MAINDEC-11-DZ88880 411 MACY112870880ZDR69588C-76 08:22 P00810000

770224
SEQ 0003

.REM%

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZDRG-D-D
PRODUCT NAME: DR11-K DIGITAL I/O TEST
DATE: JANUARY 1977
MAINTAINER: DIAGNOSTIC GROUP

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1975, 1977 BY DIGITAL EQUIPMENT CORPORATION.

TABLE OF CONTENTS

1. ABSTRACT
2. EQUIPMENT REQUIREMENTS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESSES
5. OPERATING PROCEDURE
6. ERRORS
7. RESTRICTIONS
8. MISCELANEOUS
 - 8.1 EXECUTION TIMES
 - 8.2 DEVICE ADDRESSES
9. PROGRAM DESCRIPTION
 - 9.1 LOGIC TEST
 - 9.2 CONTROL LINE LOOP
 - 9.3 COULTER FUNCTION LOOP
 - 9.4 LOGIC TEST WITH COULTER JUMPER'S
 - 9.5 LOGIC TEST WITH H321 LOC-BOX JUMPER'S
 - 9.6 H321 FUNCTION LOOP
 - 9.7 XLO1 ADJUSTMENT LOOP
10. LISTING

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 XLD1'S. THE ROUTINES SERVE THE SAME FUNCTION AS THE "PDL #3 AND PDL #4" ONLINE TEST PROGRAMS.

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

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

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

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERMINAL AND 4K OF MEMORY
DR11-K OPTION INSTALLED
BCOBR-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 TYPINGS
SW 12 = 1	010000	LOOP ON CURRENTLY SELECTED DR11-K
SW 11 = 1	004000	INHIBIT SUB-TEST INTERACTIONS
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 THE STARTING ADDRESS OF THE H321 "LOC-BOX" FUNCTION LOOP
 234 IS THE STARTING ADDRESS OF THE XLO1 ADJUSTMENT LOOP

5. OPERATING PROCEDURE

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

*** NOTE: OPERATOR MUST INSERT INFORMATION WHEN REQUESTED BY PROGRAM. THE MACHINE WILL TYPE: NEW = AFTER NEW = THE INFORMATION IS INSERTED. REFER TO SECTION 4.1 2) ***

6. ERRORS

THIS PROGRAM USES THE DIAGNOSTIC 'SYSMAC' PACKAGE FOR ERROR REPORTING AND TYPEOUT. REFER TO THE "ERROR POINTER TABLE" FOR TYPE AND DESCRIPTION OF ERRORS.

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

7. RESTRICTIONS

1. IF SEQUENTIAL DR11-K'S, ALL DR11-K'S MUST BE IN THE SAME INTERRUPT SWITCHES AND DATA PATH JUMPER CONFIGURATION.
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 INITIALIZATION QUESTIONS OR AN ERROR WILL OCCUR.
4. FOR MULTIPLE GROUPS OF CONSECUTIVE DR11-K'S:
THIS DIAGNOSTIC MUST BE RUN FOR EACH GROUP.

8. MISCELANEOUS

8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 60 SECONDS FOR COMPLETION:
ON A PDP11/05 TYPE AND WILL TYPE 'END PASS NNNN.'
THE CONTROL LINE LOOP WILL NEVER EXIT.
THE COULTER INTERFACE LOOP WILL NEVER EXIT.
THE H321 LOC-BOX FUNCTION LOOP WILL NEVER EXIT.
THE XLO1 ADJUSTMENT LOOP WILL NEVER EXIT.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS (AT APPROX. 1300)

"BASEBA" (LOC. 1306) CONTAINS THE DR11-K BASE DEVICE ADDRESS <767770>
"BASEIV" (LOC. 1310) CONTAINS THE DR11-K BASE INTERRUPT VECTOR <300>
"BASEBR" (LOC. 1312) CONTAINS THE DR11-K BR LEVEL #4 <200>

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

9. PROGRAM DESCRIPTION

9.1 LOGIC TEST <SA 200 AND 204>

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

W21A, W22A AND W23A CAN BE DIAGNOSED.

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

9.2 CONTROL LINE LOOP <SA 210>

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

9.3 COULTER FUNCTION LOOP <SA 214>

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

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

THE COULTER INPUT DATA IS STORED IN A MEMORY BUFFER. THE HIGHER FOUR BITS ARE THE TEST NUMBERS WITH ZERO BEING THE RUN TERMINATOR. UPON COMPLETION OF A RUN (SAMPLE), THE PROGRAM WILL TYPE THE RESULTS IN THE FOLLOWING FORMAT AND THEN RESTART THE RUN SEQUENCE. THE TYPED RESULTS CAN BE COMPARED TO THE COULTER TYPED OUTPUT 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 -- PDL03" THE OPERATOR SELECTS THE LOC-BOX'S, VIA DR11K BUS ADDRESS, AND THEN DEPRESSES EACH SWITCH. THE PROGRAM WILL COMPLEMENT THE LAMP ABOVE THE SWITCH. THE OPERATOR SHOULD DEPRESS THE SWITCHES SLOWLY TO VERIFY THAT ONLY DEPRESSING A SWITCH WILL CAUSE THE LAMP TO CHANGE AND THAT RELEASING THE SWITCH DOES NOT. IF THE LATER IS DETECTED THE DR11-K JUMPER CONFIGURATION SHOULD BE QUESTIONED? THE PROGRAM MAY OCCASIONLY MISS A SWITCH CLOSURE.

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

9.7 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 -- PDL04" THE H321 LAMP'S OPERATE AS SHOWN BELOW. REFER TO THE PDL-11 INSTALLATION MANUAL FOR ADJUSTMENT DETAILS.

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

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

10. LISTING

ATTACHED.

%

```

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

```



```
(1)          000004      .*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)          000010      ERRVEC= 4      ;; TIME OUT AND OTHER ERRORS
(1)          000014      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
```



```

442                                     ;ITEM 7
443 001256 014253                       EM7           ;OPERATOR INTERVENTION ERROR
444 001260 014633                       DH4           ;ERRPC DRADD
445 001262 014754                       DT4           ;$ERRPC DRADD
446 001264 015002                       DFO
447
448                                     ;ITEM 10
449 001266 014307                       EM10          ; INTERRUPT INPUT BIT FAILED TO SET INPUT READY
450 001270 014660                       DH10          ;ERRPC DRADD STATUS EXPECTED INPUT BIT
451 001272 014764                       DT10          ;$ERRPC DRADD $BDDAT $GDDAT BRLEV3
452 001274 015002                       DFO
453
454                                     ;ITEM 11
455 001276 014372                       EM11          ;NON-INTERRUPTING INPUT BIT SET INPUT READY
456 001300 014660                       DH10          ;ERRPC DRADD STATUS EXPECTED INPUT BIT
457 001302 014764                       DT10          ;$ERRPC DRADD $BDDAT $GDDAT BRLEV3
458 001304 015002                       DFO
459
460 001306 167770                       BASEBA: 167770 ; STARTING BASE BUS ADDRESS
461 001310 000300                       BASEIV: 300    ; STARTING BASE INTERRUPT ADDRESS
462 001312 000200                       BASEBR: 200
463 001314 000240                       CPU: 240      ; 1 MS. CPU DELAY FACTOR 240 FOR 11/05
464                                     ; ADDITIONAL DR-11-K 620 FOR 11/40
465 001316 000000                       NMBEXT: 0
466 001320 000000                       NBEXT: 0
467
468 001322 167770                       DRADD: 167770 ; CURRENT DR11-K STARTING ADDRESS
469 001324 000300                       DRIV: 300     ; CURRENT DR11-K STARTING INTERRUPT VECTOR
470
471
472 001326 167770                       LOC1: 167770   ; LOC BOX #1 ADDR
473 001330 167760                       LOC2: 167760   ; LOC BOX #2 ADDR
474 001332 167750                       LOC3: 167750   ; LOC BOX #3 ADDR
475 001334 167770                       GRSTAT: 167770 ; DR STATUS
476 001336 167772                       GRDAI: 167772  ; INPUT REG.
477 001340 167774                       GRDIO: 167774  ; OUTPUT REG.
478 001342 167775                       GRBHIO: 167775 ; OUTPUT REG HIGH BYTE
479 001344 000000                       NOTLCH: 0
480 001346 000000                       INTBIT: 0
481 001350 000300                       GRIVA: 300
482 001352 000302                       GRIVSA: 302
483 001354 000304                       GRIVB: 304
484 001356 000306                       GRIVSB: 306
485 001360 000200                       DIOBRL: 200
486 001362 000000                       BRLEV1: 0
487 001364 000000                       BRLEV2: 0
488 001366 000000                       BRLEV3: 0
489 001370 000000                       ODDJMP: 0
490 001372 000000                       MINSIN: 0
491 001374 000000                       TRANST: 0
492 001376 000000                       JUMPER: 0
493

```

:1 IF RUNNING H322 JUMPER CONFIG ;2 IF COULTER JUMPER CONFIG

```

501      ;:*****
502      ;:      DIGITAL I-O LOGIC TEST
503      ;:*****
504      001400 005000      BEGIN: CLR      R0      ;CLEAR R0
505      001402 005037 001376      CLR      JUMPER      ;INDICATE NORMAL FACTORY BUILD
506      001406 000414      BR      RBEG
507      001410 012737 000002 001376      COULTJ: MOV      #2,JUMPER      ;INDICATE COULTER JUMPER CONFIG
508      001416 005000      CLR      R0
509      001420 000407      BR      RBEG
510      001422 012737 000001 001376      H322J: MOV      #1,JUMPER      ;INDICATE LOC BOX JUMPER CONFIG
511      001430 005000      CLR      R0
512      001432 000402      BR      RBEG
513      001434 012700 177777      BEGIN1: MOV     #-1,R0      ;LOAD R0
514      001440 000005      RBEG:  RESET
515      .SBTTL INITIALIZE THE COMMON TAGS
(1)      ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
(1)      001442 012706 001100      MOV      #CMTAG,R6      ;:FIRST LOCATION TO BE CLEARED
(1)      001446 005026      CLR      (R6)+      ;:CLEAR MEMORY LOCATION
(1)      001450 022706 001140      CMP      #SWR,R6 ;:DONE?
(1)      001454 001374      BNE      -6      ;:LOOP BACK IF NO
(1)      001456 012706 001100      MOV      #STACK,SP      ;:SETUP THE STACK POINTER
(1)      ;:INITIALIZE A FEW VECTORS
(1)      001462 012737 015022 000020      MOV      #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
(1)      001470 012737 000340 000022      MOV      #340,@IOTVEC+2 ;:LEVEL 7
(1)      001476 012737 015274 000030      MOV      #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
(1)      001504 012737 000340 000032      MOV      #340,@EMTVEC+2 ;:LEVEL 7
(1)      001512 012737 017466 000034      MOV      #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
(1)      001520 012737 000340 000036      MOV      #340,@TRAPVEC+2 ;:LEVEL 7
(1)      001526 012737 016022 000024      MOV      #SPWRDN,@PWRVEC ;:POWER FAILURE VECTOR
(1)      001534 012737 000340 000026      MOV      #340,@PWRVEC+2 ;:LEVEL 7
(1)      001542 013737 011142 011134      MOV      $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
(1)      001550 005037 001166      CLR      $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
(1)      001554 005037 001170      CLR      $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
(1)      001560 112737 000001 001115      MOV      #1,$SERMAX ;:ALLOW ONE ERROR PER TEST
(1)      001566 012737 001566 001106      MOV      #,$LPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1)      001574 012737 001574 001110      MOV      #,$LPERR ;:SETUP THE ERROR LOOP ADDRESS
(2)      ;:SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
(2)      ;:EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
(2)      001602 013746 000004      MOV      @ERRVEC,-(SP) ;:SAVE ERROR VECTOR
(2)      001606 012737 001642 000004      MOV      #64,$ERRVEC ;:SET UP ERROR VECTOR
(2)      001614 012737 177570 001140      MOV      #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
(2)      001622 012737 177570 001142      MOV      #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
(2)      001630 022777 177777 177302      CMP      #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
(2)      001636 001012      BNE      66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)      ;:AND THE HARDWARE SWR IS NOT = -1
(2)      001640 000403      BR      65$ ;:BRANCH IF NO TIMEOUT
(2)      001642 012716 001650      64$: MOV      #65$,(SP) ;:SET UP FOR TRAP RETURN
(2)      001646 000002      RTI
(2)      001650 012737 000176 001140      65$: MOV      #SWREG,SWR ;:POINT TO SOFTWARE SWR
(2)      001656 012737 000174 001142      MOV      #DISPREG,DISPLAY
(2)      001664 012637 000004      66$: MOV      (SP)+,@ERRVEC ;:RESTORE ERROR VECTOR
(1)
516      001670 000240      NOP
517      001672 000240      NOP

```

```

518 001674 000240      NOP
519 001676 012737 001724 000004  MOV #1$, @#ERRVEC ;LOAD BUS ERROR
520 001704 013702 001306      MOV BASEBA, R2 ;LOAD STARTING ADDRESS
521 001710 005003      CLR R3 ;CLEAR COUNT
522 001712 005712      2$: TST (R2) ;TEST IF EXISTENT
523 001714 162702 000010  SUB #10, R2 ;EXIST, UPDATE TEST ADDRESS
524 001720 005203      INC R3 ;UPDATE # OF DR11'S
525 001722 000773      BR 2$
526 001724 022626      1$: CMP (SP)+, (SP)+ ;POP STACK
527 001726 005703      TST R3 ;TEST IF FIRST DOES EXIST
528 001730 001014      BNE 3$ ;BR
529 001732 032777 020000 177200  BIT #SW13, @SWR ;TEST IF INHIBIT TYPEOUT
530 001740 001002      BNE 4$ ;BR IF YES
531 001742 104401 013241  TYPE, BUSTRP ;TELL OPERATOR BASE DR11K DOESN'T EXIST
532 001746 032777 100000 177164  4$: BIT #SW15, @SWR ;TEST HALT ON ERROR
533 001754 001401      BEQ 5$ ;BR IF NO HALT
534 001756 000000      HALT ;FIRST DR11-K DOES NOT EXIST
535 ;CHECK THE PROGRAM DEVICE ADDRESS
536 001760 000754      5$: BR 2$ ;TRY AGAIN
537
538 001762 005303      3$: DEC R3 ;ADJUST R3
539 001764 010337 001316  MOV R3, NMBEXT ;SAVE THE NUMBER OF ADDITIONAL DR11-K
540 001770 012737 002442 000004  MOV #VECTRP, @#ERRVEC ;RESET BUS ERROR
541 001776 012737 000340 000006  MOV #340, @#ERRVEC+2
542 002004 000005      RBEG1: RESET
543 002006 005737 000042  TST @#42 ;TEST IF UNDER A MONITOR
544 002012 001002      BNE 4$ ;BR IF
545 002014 005700      TST R0 ;TEST R0
546 002016 001402      BEQ 2$ ;BR IF CLEARED
547 002020 000137 002752  4$: JMP IOTEST ;JUMP IF SET
548 002024  2$:
(1) 002024 104401 002032  TYPE ,65$ ;:TYPE ASCIZ STRING
(1) 002030 000426      BR ,64$ ;:GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <15><12><15><12>/DR11-K DIGITAL INPUT OUTPUT LOGIC TEST/
(1) ;:64$:
549 002106 104401 002114  TYPE ,67$ ;:TYPE ASCIZ STRING
(1) 002112 000413      BR ,66$ ;:GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <15><12>/MAINDEC-11-DZDRGD/<15><12>
(1) ;:66$:
550 002142 013746 001316  MOV NMBEXT, -(SP)
551 002146 104403      TYPOS
552 002150 000002      .WORD 2
553 002152 104401 002160  TYPE ,69$ ;:TYPE ASCIZ STRING
(1) 002156 000422      BR ,68$ ;:GET OVER THE ASCIZ
(1) ;:69$: .ASCIZ /(8) ADDITIONAL DR11-K'S CONNECTED/<15><12>
(1) ;:68$:
554 002224 022737 000001 001376  CMP #1, JUMPER ;TEST IF LOC-BOX CONFIG.
555 002232 001013      BNE 1$ ;BR IF NOT
556 002234 005037 001344      CLR NOTLCH ;LOAD LOC-BOX JUMPER CONFIG
557 002240 012737 177777 001346  MOV #-1, INTBIT
558 002246 005037 001372      CLR MINSIN
559 002252 005037 001374      CLR TRANST
560 002256 000137 002752      JMP IOTEST ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
561 002262 022737 000002 001376  1$: CMP #2, JUMPER ;TEST IF COULTER CONFIG.
    
```

```

562 002270 001015          BNE      3$          ;BR IF NOT
563 002272 C12737 177777 001344  MOV      #-1,NOTLCH ;LOAD COULTER JUMPER CONFIG.
564 002300 012737 170000 001346  MOV      #170000,INTBIT
565 002306 005037 001372          CLR      MINSIN
566 002312 012737 170000 001374  MOV      #170000,TRANST
567 002320 000137 002752          JMP      IOTEST      ;RUN THE LOGIC TEST WITH COULTER JUMPERS
568 002324 004537 002666          JSR      R5,TALK     ;TALK TO THE ANIMALS
569 002330 013332          SWNLB
570 002332 001344          NOTLCH      ;ABOUT NON LATCH INPUT BITS
571 002334 004537 002666          JSR      R5,TALK     ;TALK TO THE ANIMAL AGAIN
572 002340 013430          SWINTB
573 002342 001346          INTBIT      ;ABOUT THE INTERRUPTING BITS
574 002344 004537 002666          JSR      R5,TALK     ;ISN'T THIS BOARING
575 002350 013526          SWPOSB
576 002352 001372          MINSIN      ;AGAIN-ABOUT THE MINUS INPUT BITS
577 002354 004537 002666          JSR      R5,TALK     ;HO-HUM
578 002360 013622          SWTRAB
579 002362 001374          TRANST      ;AGAIN-ABOUT THE TRANSITION BITS
580
581          .SBTTL  DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
582 002364 012737 010000 001124  MOV      #BIT12,$GDDAT ;LOAD TEST BIT
583 002372 033737 001124 001372 10$:  BIT      $GDDAT,MINSIN ;TEST IF NEG INPUT BIT
584 002400 001407          BEQ      11$        ;BR IF NOT
585 002402 033737 001124 001374  BIT      $GDDAT,TRANST ;TEST IF TRANS. INPUT
586 002410 001403          BEQ      11$        ;BR IF NOT
587 002412 104007          ERROR      7        ;LOGICAL OPERATOR ERROR
588          ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
589 002414 000137 001400          JMP      BEGIN
590
591 002420 006137 001124          11$:  ROL      $GDDAT      ;MOVE LEFT
592 002424 103362          BCC      10$        ;BR UNTIL DONE
593
594 002426 004537 002666          JSR      R5,TALK     ;ASK THE OPERATOR
595 002432 013720          SWDPOB
596 002434 001366          BRLEV3
597 002436 000137 002752          JMP      IOTEST      ;ABOUT PROGRAM OPTIONS
598
599          ;INTERRUPT TO UNEXPECTED VECTOR
600 002442 021627 001000  VECTR:  CMP      (SP),#1000 ;TEST IF IN PROGRAM CODE
601 002446 103402          BLO      1$        ;BR IF NOT
602 002450 000000          70$:  HALT
603 002452 000776          BR      70$        ;FATAL BUS TRAP IN PROGRAM AREA
604 002454 021627 000200  1$:  CMP      (SP),#200 ;TEST IF IN SACRED VECTOR AREA
605 002460 101002          BHI      2$        ;BR IF NOT
606 002462 000000          71$:  HALT
607 002464 000776          BR      71$        ;FATAL VECTOR TRAP
608 002466 012637 002664          2$:  MOV      (SP)+,72$ ;GET ADDR THE TRAP OCCURRED TO
609 002472 162737 000004 002664  SUB      #4,72$      ;MAKE REAL ADDRESS
610 002500 032777 020000 176432  BIT      #SW13,ASWR ;TEST IF TYPE ERROR INHIBIT
611 002506 001050          BNE      3$        ;BR IF INHIBIT
612 002510 104401 002516          TYPE      65$      ;:TYPE ASCIZ STRING
(1) 002514 000417          BR      64$        ;:GET OVER THE ASCIZ
(1)          ;:65$: .ASCIZ <15><12>/DR11K INTERRUPTED TO LOC. /
(1)          64$:

```

```

613 002554 013746 002664      MOV      72$,-(SP)          ;LOAD VALUE TO BE TYPED
614 002560 104402                TYP0C
615 002562 104401 002570      TYPE      ,67$           ;;TYPE ASCIZ STRING
(1) 002566 000420                BR        66$           ;;GET OVER THE ASCIZ
(1)                                ;;67$: .ASCIZ / AND WILL NOW USE THAT VECTOR<15><12>
(1) 002630                66$:
616 002630 042737 000007 002664 3$:      BIC      #7,72$         ;MASK OFF LOWER 3 BITS
617 002636 032777 100000 176274    BIT      #SW15,2SWR     ;TEST IF HALT ON ERROR
618 002644 001401                BEQ      4$           ;BR IF NOT
619 002646 000000                HALT
520 002650 022626                4$:      CMP      (SP)+,(SP)+    ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
621 002652 013737 002664 001324    MOV      72$,DRIV      ;CLEAN THE STACK
622 002660 000137 002776                JMP      RBEG2         ;LOAD NEW VECTOR ADDRESS
623 002664 000000                72$:      0                   ;TEST THE DR11K AT THE NEW VECTOR
624
625                                ;OPERATOR QUESTIONS AND ANSWER ROUTINE
626
627 002667 012537 002700    TALK:    MOV      (R5)+,10$    ;GET ASCII POINTER
628 002672 012537 002744                MOV      (R5)+,11$    ;GET POINTER TO DATA FROM OPERATOR
629 002676 104401                TYPE
630 002700 013332                10$:     SWNLB
631 002702 023727 001140 000176    CMP      SWR,#SWREG    ;TELL OPERATOR TO LOAD SWITCHES
632 002710 001006                BNE      1$           ;TEST IF SWITCH REG. EXISTS
633 002712 104401 017443    TYPE      ,SMSWR      ;BR IF NO
634 002716 104412                RDOCT
635 002720 012677 176214    MOV      (SP)+,2SWR    ;TYPE "SWR ="
636 002724 000403                BR        2$           ;READ OCTAL
637 002726 104401                1$:     TYPE
638 002730 014005                DEPCNT
639 002732 000000                HALT
640 002734 017777 176200 000002 2$:      MOV      2SWR,211$    ;TELL OPERATOR TO DEPRESS CONT
641 002742 000205                RTS      R5           ;WAIT FOR OPERATOR
642 002744 001344                11$:    NOTLCH           ;LOAD SWITCH VALUE
643
644                                ;UNEXPECTED INTERRUPT
645 002746 104006    UNEXPT:  ERROR 6       ;UNEXPECTED INTERRUPT DURING A SUB-TEST
646 002750 000002                RTI
647
648 002752 000005                IOTEST: RESET
649 002754 013737 001306 001322    MOV      BASEBA,DRADD ;LOAD INITIAL STARTING ADDRESS
650 002762 013737 001310 001324    MOV      BASEIV,DRIV  ;LOAD INITIAL STARTING VECTOR
651 002770 013737 001316 001320    MOV      NMBEXT,NBEXT ;RELOAD # AVAILABLE
652 002776 000005                RBEG2:  RESET
653 003000 012701 000240                MOV      #240,R1      ;LOAD R1
654 003004 012702 000242                MOV      #242,R2
655 003010 010221                5$:     MOV      R2,(R1)+
656 003012 012721 004700                MOV      #4700,(R1)+ ;SAVE THE ADDRESS
657 003016 022222                CMP      (R2)+,(R2)+ ;LOAD "JSR PC,RC"
658 003020 020227 001076                CMP      R2,#$CMTAG-2 ;BUMP R2
659 003024 001371                BNE      5$           ;TEST FOR LAST
                                ;BR UNTIL DONE
    
```

```

661 003026 000005          IOTST1: RESET
662 003030 004737 003036      JSR      PC,SETADD          ;SET UP BUS ADDRESS AND VECTOR
663 003034 000453          BR       IOTTS1
664 003036 013737 001322 001334 SETADD: MOV    DRADD,GRSTAT      ;LOAD 1ST ADDRESS
665 003044 013737 001322 001336      MOV    DRADD,GRDAI        ;LOAD 2ND ADDRESS
666 003052 062737 000002 001336      ADD    #2,GRDAI
667 003060 013737 001322 001340      MOV    DRADD,GRDIO        ;LOAD 3RD ADDRESS
668 003066 062737 000004 001340      ADD    #4,GRDIO
669 003074 013737 001322 001342      MOV    DRADD,GRBHIO       ;LOAD 4TH ADDRESS
670 003102 062737 000005 001342      ADD    #5,GRBHIO
671 003110 013737 001324 001350      MOV    DRIV,GRIVA         ;LOAD FIRST VECTOR
672 003116 013737 001324 001352      MOV    DRIV,GRIVSA
673 003124 062737 000002 001352      ADD    #2,GRIVSA
674 003132 013737 001324 001354      MOV    DRIV,GRIVB         ;LOAD 2ND VECTOR
675 003140 062737 000004 001354      ADD    #4,GRIVB
676 003146 013737 001324 001356      MOV    DRIV,GRIVSB
677 003154 062737 000006 001356      ADD    #6,GRIVSB
678 003162 000207          RTS      PC
679 003164 013737 001312 001360 IOTTS1: MOV    BASEBR,DIOBRL      ;LOAD BR LEVEL
680 003172 005037 001370          CLR     ODDJMP
681 003176 053737 001372 001370      BIS    MINSIN,ODDJMP      ;SET MINUS INPUT BITS
682 003204 053737 001374 001370      BIS    TRANST,ODDJMP     ;SET TRANSITION INPUT BITS
683 003212 012777 002746 176130      MOV    #UNEXPT,@GRIVA    ;RESET INPUT VECTOR
(1) 003220 012777 000340 176124      MOV    #340,@GRIVSA
(1) 003226 012777 002746 176120      MOV    #UNEXPT,@GRIVB    ;RESET OUTPUT VECTOR
(1) 003234 012777 000340 176114      MOV    #340,@GRIVSB
684 (3) ;*****
685 (3) ;*TEST 1 TEST FOR NO BUS ERRORS
686 (3) ;*****
(2) 003242 000004          TST1:  SCOPE
685 003244 005077 176064      CLR     @GRSTAT
686 003250 005077 176062      CLR     @GRDAI
687 003254 005077 176060      CLR     @GRDIO
688
689 ;*****
(3) ;*TEST 2 TEST THAT OUTPUT REG. CAN HOLD #-1
(3) ;*****
(2) 003260 000004          TST2:  SCOPE
690 003262 012737 177777 001124      MOV    #-1,$GDDAT        ;LOAD EXPECTED
691 003270 012777 177777 176042      MOV    #-1,@GRDIO        ;ALL ONES TO REGISTER
692 003276 017737 176036 001126      MOV    @GRDIO,$BDDAT     ;READ OUTPUT REG.
693 003304 023737 001124 001126      CMP    $GDDAT,$BDDAT     ;COMPARE
694 003312 001401          BEQ     TST3              ;;BR IF EQUAL
695 003314 104003          ERROR 3                  ;REG WILL NOT HOLD ONES

```

```

697
(3)
(3)
(2) 003316 000004
(1) 003320 012737 000040 001166
698 003326 005037 001124
699 003332 012777 177777 176000
700 003340 000005
701 003342 017737 175772 001126
702 003350 001401
703 003352 104003
704
705
(3)
(3)
(2) 003354 000004
706 003356 012737 052525 001124
707 003364 012777 052525 175746
708 003372 017737 175742 001126
709 003400 023737 001124 001126
710 003406 001401
711 003410 104003
712
713
(3)
(3)
(2) 003412 000004
714 003414 012737 125252 001124
715 003422 012777 125252 175710
716 003430 017737 175704 001126
717 003436 023737 001124 001126
718 003444 001401
719 003446 104003
720
721
(3)
(3)
(2) 003450 000004
(1) 003452 012737 000004 001166
722 003460 012737 003472 001110
723 003466 005037 001124
724 003472 013777 001124 175640 2$:
725 003500 017737 175634 001126
726 003506 023737 001124 001126
727 003514 001401
728 003516 104003
729 003520 005237 001124 1$:
730 003524 001362

```

```

;*****
;*TEST 3 TEST THAT RESET CLEARS OUTPUT REG.
;*****
TST3: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
CLR $GDDAT
MOV #-1,$GRDIO
RESET ;SET DATA TO ALL ONES
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
BEQ TST4 ;;BR IF EQUAL
ERROR 3 ;REG FAILED TO CLEAR

;*****
;*TEST 4 TEST THAT OUTPUT REG. CAN HOLD #52525
;*****
TST4: SCOPE
MOV #52525,$GDDAT ;LOAD EXPECTED VALUE
MOV #52525,$GRDIO
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST5 ;;BR IF EQUAL
ERROR 3 ;DATA NOT=52525

;*****
;*TEST 5 TEST THAT OUTPUT REG. CAN HOLD #125252
;*****
TST5: SCOPE
MOV #125252,$GDDAT ;LOAD EXPECTED VALUE
MOV #125252,$GRDIO
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST6 ;;BR IF EQUAL
ERROR 3 ;DATA NOT=125252

;*****
;*TEST 6 TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
;*****
TST6: SCOPE
MOV #4,$TIMES ;;DO 4 ITERATIONS
MOV #2,$SLPERR ;LOAD SCOPE ERROR RETURN
CLR $GDDAT ;CLEAR PATTERN
MOV $GDDAT,$GRDIO ;LOAD THE OUTPUT REG.
MOV $GRDIO,$BDDAT ;READ THE REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;;BR IF EQUAL
ERROR 3 ;OUTPUT REG. FAILED TO HOLD COUNT PATTERN
INC $GDDAT ;UPDATE THE PATTERN
BNE 2$ ;TRY AGAIN

```

```

732 (3) ::*****
733 (3) ::*TEST 7 FLOAT A 1 ACROSS THE OUTPUT REGISTER
734 (2) ::*****
003526 000004 †ST7: SCOPE
735 003530 012737 003544 001110 MOV #1$,$LPERR ;LOAD SCOPE ERROR RETURN
736 003536 012737 000001 001124 MOV #BIT0,$GDDAT ;LOAD EXPECTED VALUE
737 003544 005077 175570 1$: CLR @GRDIO ;CLEAR OUTPUT
738 003550 053777 001124 175562 BIS $GDDAT,@GRDIO ;SET THAT BIT
739 003556 017737 175556 001126 MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
740 003564 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST RESULTS
741 003572 001401 BEQ 2$ ;BR IF EQUAL ?
742 003574 104003 ERROR 3
743 003576 006337 001124 2$: ASL $GDDAT ;SHIFT EXPECTED DATA
744 003602 001360 BNE 1$ ;BR UNTIL DONE
745 (3) ::*****
746 (3) ::*TEST 10 FLOAT A 0 ACROSS THE OUTPUT REGISTER
747 (2) ::*****
003604 000004 †ST10: SCOPE
748 003606 012737 003622 001110 MOV #1$,$LPERR ;LOAD SCOPE ERROR RETURN
749 003614 012737 000001 001124 MOV #BIT0,$GDDAT ;LOAD EXPECTED VALUE
750 003622 012777 177777 175510 1$: MOV #-1,@GRDIO ;LOAD OUTPUT TO A ONE
751 003630 043777 001124 175502 BIC $GDDAT,@GRDIO ;CLEAR A BIT
752 003636 017737 175476 001126 MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
753 003644 005137 001126 001126 COM $BDDAT ;COMPLEMENT IT
754 003650 023737 001124 001126 CMP $GDDAT,$BDDAT ;EQUAL ?
755 003656 001401 BEQ 2$ ;BR IF EQUAL
756 003660 104003 ERROR 3
757 003662 006337 001124 2$: ASL $GDDAT ;SHIFT LEFT
758 003666 001355 BNE 1$ ;BRANCH UNTIL DONE
759 (3) ::*****
760 (3) ::*TEST 11 TEST FOR SLOW OUTPUT GATES WITH #125252
761 (2) ::*****
003670 000004 †ST11: SCOPE
762 003672 012737 125252 001124 MOV #125252,$GDDAT ;LOAD EXPECTED VALUE
763 003700 013777 001124 175432 MOV $GDDAT,@GRDIO ;LOAD OUTPUT
764 003706 005177 175426 COM @GRDIO ;COMPLEMENT OUTPUT REG.
765 (1) 003712 005177 175422 COM @GRDIO ;COMPLEMENT OUTPUT REG.
766 (1) 003716 005177 175416 COM @GRDIO ;COMPLEMENT OUTPUT REG.
767 (1) 003722 005177 175412 COM @GRDIO ;COMPLEMENT OUTPUT REG.
768 (1) 003726 005177 175406 COM @GRDIO ;COMPLEMENT OUTPUT REG.
769 (1) 003732 005177 175402 COM @GRDIO ;COMPLEMENT OUTPUT REG.
770 (1) 003736 005177 175376 COM @GRDIO ;COMPLEMENT OUTPUT REG.
771 (1) 003742 005177 175372 COM @GRDIO ;COMPLEMENT OUTPUT REG.
772 003746 017737 175366 001126 MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
773 003754 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST REGISTER
774 003762 001401 BEQ TST12 ;;BR IF CORRECT
775 003764 104003 ERROR 3

```

```

771
(3)
(3)
(2) 003766 000004
772 003770 012737 052525 001124
773 003776 013777 001124 175334
776 004004 005177 175330
(1) 004010 005177 175324
(1) 004014 005177 175320
(1) 004020 005177 175314
(1) 004024 005177 175310
(1) 004030 005177 175304
(1) 004034 005177 175300
(1) 004040 005177 175274
777 004044 017737 175270 001126
778 004052 023737 001124 001126
779 004060 001401
780 004062 104003
781
782
(3)
(3)
(2) 004064 000004
(1) 004066 012737 000400 001166
783 004074 012737 004116 001110
784 004102 012737 052400 001124
785 004110 013777 001124 175222
786 004116 113777 001124 175214
787 004124 017737 175210 001126
788 004132 023737 001124 001126
789 004140 001401
790 004142 104003
791 004144 105237 001124
792 004150 001362
793
794
(3)
(3)
(2) 004152 000004
(1) 004154 012737 000400 001166
795 004162 012737 004204 001110
796 004170 012737 000252 001124
797 004176 013777 001124 175134
798 004204 113777 001125 175130
799 004212 017737 175122 001126
800 004220 023737 001124 001126
801 004226 001401
802 004230 104003
803 004232 105237 001125
804 004236 001362

```

```

*****
*TEST 12 TEST FOR SLOW OUTPUT GATES WITH #52525
*****
TST12: SCOPE
MOV #52525,$GDDAT ;LOAD EXPECTED VALUE
MOV $GDDAT,$GRDIO ;LOAD OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
COM $GRDIO ;COMPLEMENT OUTPUT REG.
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;TEST PATTERN
BEQ TST13 ;;BR IF EQUAL
ERROR 3 ;OUTPUT REGISTER IN ERROR

*****
*TEST 13 TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN
*****
TST13: SCOPE
MOV #400,$TIMES ;;DO 400 ITERATIONS
MOV #2,$LPERR ;LOAD SCOPE ERROR RETURN
MOV #52400,$GDDAT ;LOAD EXPECTED DATA
MOV $GDDAT,$GRDIO ;LOAD OUTPUT REG.
2$: MOVB $GDDAT,$GRDIO ;LOAD THE OUTPUT
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT
BEQ 1$ ;BRANCH IF EQUAL
ERROR 3 ;ERROR, OUTPUT REG. IN ERROR
1$: INCB $GDDAT ;UPDATE PATTERN
BNE 2$

*****
*TEST 14 TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN
*****
TST14: SCOPE
MOV #400,$TIMES ;;DO 400 ITERATIONS
MOV #2,$LPERR ;LOAD SCOPE ERROR RETURN
MOV #252,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,$GRDIO ;LOAD OUTPUT REG.
2$: MOVB $GDDAT+1,$GREHIO ;LOAD THE HIGH BYTE
MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT
BEQ 1$ ;BRANCH IF EQUAL
ERROR 3 ;ERROR, OUTPUT REG. IN ERROR
1$: INCB $GDDAT+1
BNE 2$

```

```

806
(3)
(3)
(2) 004240 000004
807 004242 005077 175072
808 004246 012777 177777 175062
809 004254 012737 100000 001124
810 004262 012777 100000 175044
811 004270 017737 175040 001126
812 004276 033737 001124 001126
813 004304 001001
814 004306 104001
815
816
(3)
(3)
(2) 004310 000004
817 004312 005077 175016
818 004316 012737 040000 001124
819 004324 012777 040000 175002
820 004332 017737 174776 001126
821 004340 033737 001124 001126
822 004346 001001
823 004350 104001
824
825
(3)
(3)
(2) 004352 000004
826 004354 012737 000200 001124
827 004362 012777 000200 174744
828 004370 017737 174740 001126
829 004376 023737 001124 001126
830 004404 001401
831 004406 104001
832
833
(3)
(3)
(2) 004410 000004
834 004412 005077 174716
835 004416 012737 000100 001124
836 004424 012777 000100 174702
837 004432 017737 174676 001126
838 004440 023737 001124 001126
839 004446 001401
840 004450 104001
841

```

```

*****
*TEST 15 TEST OUTPUT DATA ACCEPT FLAG
*****
TST15: SCOPE
CLR @GRDIO
MOV #-1,@GRDAI
MOV #BIT15,$GDDAT ;LOAD EXPECTED
MOV #BIT15,@GRSTAT ;SET BIT 15
MOV @GRSTAT,$BDDAT ;READ STATUS
BIT $GDDAT,$BDDAT
BNE TST16 ;;BR IF SET
ERROR 1 ;ERROR, BIT 15 FAILED TO SET

```

```

*****
*TEST 16 TEST OUTPUT INTERRUPT ENABLE
*****
TST16: SCOPE
CLR @GRSTAT ;CLEAR STATUS
MOV #BIT14,$GDDAT ;LOAD EXPECTED
MOV #BIT14,@GRSTAT ;LOAD BIT 14
MOV @GRSTAT,$BDDAT ;READ STATUS
BIT $GDDAT,$BDDAT
BNE TST17 ;;BR IF SET
ERROR 1 ;ERROR BIT 14 FAILED TO SET

```

```

*****
*TEST 17 TEST INPUT DATA READY FLAG
*****
TST17: SCOPE
MOV #BIT7,$GDDAT ;LOAD EXPECTED
MOV #BIT7,@GRSTAT ;SET BIT 7
MOV @GRSTAT,$BDDAT ;READ RESULT
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST20 ;;BR IF EQUAL
ERROR 1 ;ERROR, BIT 7 FAILED TO SET

```

```

*****
*TEST 20 TEST INPUT INTERRUPT ENABLE
*****
TST20: SCOPE
CLR @GRSTAT ;CLEAR STATUS
MOV #BIT6,$GDDAT ;LOAD EXPECTED
MOV #BIT6,@GRSTAT ;SET BIT 6
MOV @GRSTAT,$BDDAT ;READ RESULT
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST21 ;;BR IF EQUAL
ERROR 1 ;ERROR, BIT 6 FAILED TO SET

```

```

843
(3)
(3)
(2) 004452 000004
(1) 004454 012737 000040 001166
844 004462 005077 174646
845 004466 005037 001124
846 004472 012777 140300 174634
847 004500 000005
848 004502 017737 174626 001126
849 004510 001401
850 004512 104001
851
852
856
860
(3)
(3)
(2) 004514 000004
861 004516 032777 002000 174414
862 004524 001405
863 004526 112737 000036 001102
864 004534 000137 307050
865 004540
(1) 004540 005077 174574
866 004544 012777 177777 174564
867 004552 005037 001124
868 004556 012777 000000 174554
869 004564 017737 174546 001126
870 004572 043737 001370 001126
871 004600 023737 001124 001126
872 004606 001401
873 004610 104002
874
875
(3)
(3)
(2) 004612 000004
876 004614 005077 174520
877 004620 012777 177777 174510
878 004626 012737 177777 001124
879 004634 043737 001370 001124
880 004642 013777 001124 174470
881 004650 017737 174462 001126
882 004656 043737 001370 001126
883 004664 023737 001124 001126
884 004672 001401
885 004674 104002
886

```

```

*****
*TEST 21 TEST THAT RESET CLEARS THE DIGITAL STATUS REGISTER
*****
TST21: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
CLR @GRSTAT ;CLEAR STATUS
CLR $GDDAT ;LOAD EXPECTED
MOV #BIT15!BIT14!BIT7!BITS,@GRSTAT
RESET
MOV @GRSTAT,$BDDAT ;READ RESULT
BEQ TST22 ;;BR IF EQUAL
ERROR 1 ;ERROR, RESET FAILED TO CLEAR DIGITAL
;STATUS REGISTER

*****
*TEST 22 TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
*****
TST22: SCOPE
BIT #BIT10,@SWR ;TEST SWITCH BIT
BEQ 1$ ;BRANCH IF DOWN
MOVB #36,$STNM ;LOAD NEW TEST NUMBER
JMP DRT21 ;BYPASS SOME TEST USING THE EXTERNAL CABLE

1$:
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
CLR $GDDAT ;CLEAR EXPECTED
MOV #0,@GRDIO ;LOAD THE OUTPUT
MOV @GRDAI,$BDDAT ;READ THE INPUT
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST23 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL THE OUTPUT REG.

*****
*TEST 23 TEST INPUT WITH #-1
*****
TST23: SCOPE
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
MOV #-1,$GDDAT ;LOAD EXPECTED
BIC ODDJMP,$GDDAT ;MASK
MOV $GDDAT,@GRDIO ;LOAD THE OUTPUT
MOV @GRDAI,$BDDAT ;READ INPUT
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST24 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL THE OUTPUT

```

```

888
(3)
(3)
(2) 004676 000004
889 004700 005077 174434
890 004704 012777 177777 174424
891 004712 012737 052525 001124
892 004720 043737 001370 001124
893 004726 013777 001124 174404
894 004734 017737 174376 001126
895 004742 043737 001370 001126
896 004750 023737 001124 001126
897 004756 001401
898 004760 104002
899
900
(3)
(3)
(2) 004762 000004
901 004764 005077 174350
902 004770 012777 177777 174340
903 004776 012737 125252 001124
904 005004 043737 001370 001124
905 005012 013777 001124 174320
906 005020 017737 174312 001126
907 005026 043737 001370 001126
908 005034 023737 001124 001126
909 005042 001401
910 005044 104002

```

```

*****
*TEST 24 TEST INPUT WITH #52525
*****
TST24: SCOPE
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
MOV #52525,$GDDAT ;LOAD EXPECTED
BIC ODDJMP,$GDDAT ;MASK
MOV $GDDAT,@GRDIO ;LOAD THE OUTPUT
MOV @GRDAI,$BDDAT ;READ INPUT
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST25 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL OUTPUT

```

```

*****
*TEST 25 TEST INPUT WITH #125252
*****
TST25: SCOPE
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
MOV #125252,$GDDAT ;LOAD EXPECTED
BIC ODDJMP,$GDDAT ;MASK
MOV $GDDAT,@GRDIO ;LOAD THE OUTPUT
MOV @GRDAI,$BDDAT ;READ INPUT
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST26 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL OUTPUT

```

```

912          ;:*****
(3)          ;:TEST 26      TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
(3)          ;:*****
(2) 005046 000004          TST26: SCOPE
(1) 005050 012737 000001 001166      MOV      #1,$TIMES      ;;DO 1 ITERATION
913 005056 005737 001370          TST      ODDJMP          ;:TEST IF ANY ODD JUMPERS
914 005062 001461          BEQ      TST27          ;:BR IF NONE
915 005064 012737 010000 001124      MOV      #BIT12,$GDDAT ;:LOAD TEST BIT
916 005072 033737 001370 001124 1$: BIT      ODDJMP,$GDDAT ;:TEST IF ODD JUMPER BIT
917 005100 001447          BEQ      2$           ;:BR IF NOT
918 005102 033737 001344 001124      BIT      NOTLCH,$GDDAT ;:TEST IF LATCHING INPUT BIT
919 005110 001043          BNE      2$           ;:BR IF NOT
920 005112 013777 001124 174220      MOV      $GDDAT,$GRDIO ;:LOAD OUTPUT
921 005120 012777 177777 174210      MOV      #-1,$GRDAI    ;:CLEAR INPUT
922 005126 043777 001124 174204      BIC      $GDDAT,$GRDIO ;:CLEAR OUTPUT BIT
923 005134 017737 174176 001126      MOV      $GRDAI,$BDDAT ;:READ INPUT REG
924 005142 023737 001124 001126      CMP      $GDDAT,$BDDAT ;:COMPARE
925 005150 001401          BEQ      3$           ;:BR IF EQUAL
926 005152 104002          ERROR    2           ;:ERROR, NEG. INPUT OR NEG. TRANSITION
927                                     ;: INPUT BIT FAILED TO SET INPUT REGISTER
928
929 005154 033737 001124 001372 3$: BIT      $GDDAT,MINSIN ;:TEST IF NEG. INPUT BIT
930 005162 001016          BNE      2$           ;:BR IF
931 005164 012777 177777 174144      MOV      #-1,$GRDAI    ;:CLEAR INPUT
932 005172 053777 001124 174140      BIS      $GDDAT,$GRDIO ;:LOAD OUTPUT
933 005200 017737 174132 001126      MOV      $GRDAI,$BDDAT ;:READ INPUT
934 005206 023737 001124 001126      CMP      $GDDAT,$BDDAT ;:COMPARE
935 005214 001401          BEQ      2$           ;:BR IF EQUAL
936 005216 104002          ERROR    2           ;:ERROR, POSITIVE INPUT TRANSITION
937                                     ;:LOGIC FAILED TO SET INPUT REGISTER BIT
938
939 005220 006137 001124          2$: ROL      $GDDAT      ;:CHANGE DATA PATTERN
940 005224 103322          1$: BCC      1$           ;:BR IF MORE DATA

```

```

942      ::*****
(3)      : *TEST 27      FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
(3)      : *****
(2)      TST27:  SCOPE
943      005226 000004          MOV      #2$, $LPERR          ;LOAD ERROR SCOPE RETURN
944      005230 012737 005256 001110
945      005236 012737 000001 001364          MOV      #BIT0, BRLEV2          ;LOAD EXPECTED
946      005244 013737 001344 001366          MOV      NOTLCH, BRLEV3        ;GET NON-LATCH
947      005252 005137 001366          COM          ;COMPLEMENT
948      005256 013737 001364 001124 2$:      MOV      BRLEV2, $GDDAT        ;LOAD GOOD
949      005264 033737 001124 001370          BIT      $GDDAT, ODDJMP        ;TEST IF ODD JUMPER
950      005272 001042          BNE      1$                    ;BYPASS IF ODD JUMPER
951      005274 033737 001124 001344          BIT      $GDDAT, NOTLCH        ;TEST FOR NON-LATCH
952      005302 001436          BEQ      1$                    ;BR IF LATCHING
953
954      005304 013777 001124 174026          MOV      $GDDAT, @GRDIO        ;LOAD OUTPUT
955      005312 017737 174020 001126          MOV      @GRDAI, $BDDAT        ;READ INPUT
956      005320 043737 001366 001126          BIC      BRLEV3, $BDDAT        ;MASK TO LATCH BITS
957      005326 023737 001124 001126          CMP      $GDDAT, $BDDAT        ;COMPARE
958      005334 001401          BEQ      3$                    ;;BR IF EQUAL
959      005336 104002          ERROR    2                    ;INPUT REGISTER IN ERROR
960                                     ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
961
962      ;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
963
964      005340 043777 001124 173772 3$:      BIC      $GDDAT, @GRDIO        ;CLEAR OUTPUT BIT
965      005346 017737 173764 001126          MOV      @GRDAI, $BDDAT        ;READ INPUT
966      005354 043737 001366 001126          BIC      BRLEV3, $BDDAT        ;MASK TO LATCH BITS
967      005362 005037 001124          CLR      $GDDAT                ;CLEAR EXPECTED
968      005366 033737 001364 001126          BIT      BRLEV2, $BDDAT        ;TEST FOR BIT
969      005374 001401          BEQ      1$                    ;;BR IF CLEARED
970      005376 104002          ERROR    2                    ;INPUT BIT LATCHED IN ERROR
971                                     ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
972
973      005400 006337 001364 1$:      ASL      BRLEV2                ;CHANGE PATTERN
974      005404 001324          BNE      2$                    ;BR UNTIL DONE
975

```

```

977      ::*****
(3)      ::*TEST 30      FLOAT A 1 ACROSS LATCHING INPUT BITS
(3)      ::*****
(2)      †ST30:  SCOPE
978      005406 000004      MOV      #2$, $LPERR      ;LOAD ERROR SCOPE RETURN
979      005410 012737 005436 001110      MOV      #BIT0, $GDDAT    ;LOAD EXPECTED VALUE
979      005416 012737 000001 001124      CLR      @GRDIO          ;CLEAR OUTPUT REG
980      005424 005077 173710      MOV      #-1, @GRDAI     ;CLEAR INPUT
981      005430 012777 177777 173700
982
983      005436 033737 001124 001344 2$:   BIT      $GDDAT, NOTLCH   ;TEST FOR NON-LATCHING
984      005444 001021      BNE      1$             ;BR IF NON-LATCH
985      005446 033737 001124 001370      BIT      $GDDAT, ODDJMP  ;TEST IF ODD JUMPER
986      005454 001015      BNE      1$             ;BYPASS IF ODD JUMPER
987
988      005456 013777 001124 173654      MOV      $GDDAT, @GRDIO  ;LOAD OUTPUT
989      005464 005077 173650      CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
990      005470 017737 173642 001126      MOV      @GRDAI, $BDDAT  ;READ INPUT REG.
991      005476 023737 001124 001126      CMP      $GDDAT, $BDDAT  ;COMPARE
992      005504 001401      BEQ      1$             ;;BR IF EQUAL
993      005506 104002      ERROR    2             ;INPUT REGISTER FAILED TO LATCH DATA
994
995      005510 053777 001124 173620 1$:   BIS      $GDDAT, @GRDAI  ;CLEAR INPUT BIT
996      005516 006337 001124      ASL      $GDDAT          ;CHANGE PATTERN
997      005522 001345      BNE      2$             ;BR UNTIL DONE
998
999      ::*****
(3)      ::*TEST 31      FLOAT A 0 ACROSS LATCHING INPUT BITS
(3)      ::*****
(2)      †ST31:  SCOPE
1000     005524 000004      MOV      #2$, $LPERR     ;LOAD ERROR SCOPE RETURN
1001     005526 012737 005554 001110      MOV      #BIT0, BRLEV3   ;LOAD EXPECTED
1002     005534 012737 000001 001366      CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1003     005542 005077 173572      MOV      #-1, @GRDAI     ;CLEAR INPUT
1004     005546 012777 177777 173562
1005     005554 033737 001366 001344 2$:   BIT      BRLEV3, NOTLCH  ;TEST FOR LATCHING
1006     005562 001032      BNE      1$             ;BR IF NOT
1007     005564 033737 001124 001370      BIT      $GDDAT, ODDJMP  ;TEST IF ODD JUMPER
1008     005572 001026      BNE      1$             ;BYPASS IF ODD JUMPER BIT
1009
1010     005574 012737 177777 001124      MOV      #-1, $GDDAT     ;LOAD
1011     005602 043737 001344 001124      BIC      NOTLCH, $GDDAT   ;MAKE BRLEV3
1012     005610 043737 001366 001124      BIC      BRLEV3, $GDDAT  ;LOAD OUTPUT
1013     005616 013777 001124 173514      MOV      $GDDAT, @GRDIO  ;CLEAR OUTPUT REGISTER
1014     005624 005077 173510      CLR      @GRDIO
1015
1016     005630 017737 173502 001126      MOV      @GRDAI, $BDDAT  ;READ INPUT
1017     005636 023737 001124 001126      CMP      $GDDAT, $BDDAT  ;COMPARE
1018     005644 001401      BEQ      1$             ;;BR IF EQUAL
1019     005646 104002      ERROR    2             ;INPUT REGISTER FAILED TO LATCH DATA
1020
1021     005650 053777 001366 173460 1$:   BIS      BRLEV3, @GRDAI  ;CLEAR INPUT BIT
1022     005656 006337 001366      ASL      BRLEV3          ;CHANGE PATTERN
1023     005662 001334      BNE      2$             ;BR UNTIL DONE

```

```

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

```

```

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

```

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

```

1075
(3)
(3)
(2) 006654 000004
(1) 006656 012737 000040 001166
1076 006664 005077 172450
1077 006670 012777 177777 172440
1078 006676 012777 177777 172434
1079 006704 005037 001124
1080 006710 000005
1081 006712 017737 172420 001126
1082 006720 043737 001370 001126
1083 006726 001401
1084 006730 104002
1085
1086
(3)
(3)
(2) 006732 000004
1087 006734 012737 000200 001124
1088 006742 005077 172366
1089 006746 012777 000000 172364
1090 006754 022727 000000 000000
1091 006762 017737 172346 001126
1092 006770 023737 001124 001126
1093 006776 001401
1094 007000 104001
1095
1096
1097
1098
(3)
(3)
(2) 007002 000004
1099 007004 012737 100000 001124
1100 007012 005077 172316
1101 007016 012777 000000 172314
1102 007024 022727 000000 000000
1103 007032 017700 172300
1104 007036 017737 172272 001126
1105 007044 100401
1106 007046 104001
1107

```

```

*****
*TEST 34 TEST THAT RESET CLEARS INPUT REGISTER BITS
*****
TST34: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
MOV #-1,@GRDIO ;LOAD OUTPUT
CLR $GDDAT ;LOAD EXPECTED
RESET
MOV @GRDAI,$BDDAT ;READ INPUT REG.
BIC ODDJMP,$BDDAT ;MASK ODD JUMPERS
BEQ TST35 ;;BR IF ALL BITS CLEARED
ERROR 2 ;INPUT REG. FAILED TO CLEAR UPON RESET INST.

*****
*TEST 35 TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
*****
TST35: SCOPE
MOV #BIT7,$GDDAT ;LOAD EXPECTED
CLR @GRSTAT
MOV #0,@GRDIO ;OUTPUT 0
CMP #0,#0 ;DELAY
MOV @GRSTAT,$BDDAT ;READ RESULTS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST36 ;;BR IF EQUAL
ERROR 1 ;INPUT DATA READY FLAG FAILED TO SET

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

*****
*TEST 36 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
*****
TST36: SCOPE
MOV #BIT15,$GDDAT ;LOAD EXPECTED
CLR @GRSTAT
MOV #0,@GRDIO
CMP #0,#0
MOV @GRDAI,R0 ;READ INPUT
MOV @GRSTAT,$BDDAT ;READ RESULTS
BMI TST37 ;;BR IF SET
ERROR 1 ;INPUT DATA READY FLAG FAILED TO SET

```

```

1109 007050
(4)
(3)
(3)
(2) 007050 000004
(1) 007052 012737 000040 001166
1110 007060 000005
1111 007062 005077 172246
1112 007066 005037 177776
1113 007072 012777 007152 172250
1114 007100 012777 000340 172244
1115 007106 012777 007156 172240
1116 007114 012777 000340 172234
1117 007122 012777 000000 172210
1118 007130 017700 172202
1119 007134 000240
1120 007136 000240
1121 007140 000240
1122 007142 000240
1123 007144 005077 172164
1124 007150 000404
1125
1126 007152 104006
1127 007154 000002
1128
1129 007156 104006
1130 007160 000002
1131
(3)
(3)
(2) 007162 000004
1132 007164 005077 172144
1133 007170 012777 007242 172152
1134 007176 012777 000340 172146
1135 007204 012777 007264 172142
1136 007212 012777 000340 172136
1137 007220 052777 000040 172106
1138 007226 005037 177776
1139 007232 000240
1140 007234 000240
1141 007236 104004
1142 007240 000415
1143
1144
1145
1146
1147 007242 012777 007264 172100
1148 007250 022626
1149 007252 005037 177776
1150 007256 000240
1151 007260 000240
1152 007262 000404
1153
1154 007264 022626

```

```

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 ;SET UP INTERRUPT OUTPUT VECTOR
MOV #2$,@GRIVB
MOV #340,@GRIVSB
MOV #0,@GRDIO ;OUTPUT
MOV @GRDAI,RO ;INPUT
NOP
NOP
NOP
NOP
CLR @GRSTAT ;CLEAR STATUS
BR TST40 ;;
1$: 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$,@GRIVB ;LOAD OUTPUT VECTOR
MOV #340,@GRIVSB
BIS #BITS,@GRSTAT ;MAINT. INT C
CLR PSW ;LOWER PSW
NOP
NOP
ERROR 4 ;ERROR, INPUT FAILED TO INTERRUPT
BR TST41 ;;BR TO NEXT TEST
;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN NO INTERRUPT WILL OCCUR
; IF INTERRUPT OCCURS 'INT DONE C' FAILED TO CLEAR INT C FLOP
1$: 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

```

K03

DR11-K LOGIC TEST MAINDEC-11-DZDRGD MACY11 27(663) 19-DEC-76 08:22 PAGE 25-1
DZDRGD.P11 T40 TEST THAT THE INPUT CAN INTR. USING THE MAINT. BIT

SEQ 0038

1155 007266 104006
1156 007270 005037 177776

ERROR 6
CLR PSW

:UNEXPECTED INTERRUPT

```

1158 (3) *****
1159 (3) *TEST 41 TEST THAT THE INPUT INTR. CLEARS INT. ENABLE VIA MAINT. BIT
1160 (2) *****
1161 007274 000004 TST41: SCOPE
1162 007276 005077 172032 CLR @GRSTAT ;CLEAR STATUS
1163 007302 012777 007360 172040 MOV #1$, @GRIVA ;LOAD RETURN VECTOR
1164 007310 012777 000340 172034 MOV #340, @GRIVSA ;
1165 007316 012777 007422 172030 MOV #2$, @GRIVB ;LOAD OUTPUT VECTOR
1166 007324 012777 000340 172024 MOV #340, @GRIVSB ;
1167 007332 012777 000100 171774 MOV #BIT6, @GRSTAT ;LOAD INPUT INTR. ENABLE
1168 007340 052777 000040 171766 BIS #BITS, @GRSTAT ;MAINT. INT C
1169 007346 005037 177776 CLR PSW ;LOWER PSW
1170 007352 000240 NOP
1171 007354 000240 NOP
1172 007356 104004 ERROR 4 ;ERROR, INPUT FAILED TO INTERRUPT
1173 007360 012777 007422 171762 1$: MOV #2$, @GRIVA ;LOAD INPUT VECTOR
1174 007366 022626 CMP (SP)+, (SP)+ ;POP THE STACK #4
1175 007370 005037 177776 CLR PSW ;LOWER PRIOR.
1176 007374 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
1177 007400 017737 171730 001126 MOV @GRSTAT, $BDDAT ;READ STATUS
1178 007406 032737 000100 001126 BIT #BIT6, $BDDAT ;TEST BIT 6
1179 007414 001406 BEQ TST42 ;;BR IF CLEARED
1180 007416 104001 ERROR 1 ;INPUT INTR. FAILED TO CLEAR
1181 007420 000404 BR TST42 ;;<NEXT TEST>
1182 007422 022626 2$: CMP (SP)+, (SP)+ ;POP THE STACK
1183 007424 104006 ERROR 6 ;UNEXPECTED INTERRUPT
1184 007426 005037 177776 CLR PSW
1185 (3) *****
1186 (3) *TEST 42 TEST THAT THE OUTPUT CAN INTR. USING THE MAINT. BIT
1187 (2) *****
1188 007432 000004 TST42: SCOPE
1189 007434 005077 171674 CLR @GRSTAT ;CLEAR STATUS
1190 007440 012777 007534 171702 MOV #2$, @GRIVA ;LOAD INPUT VECTOR
1191 007446 012777 000340 171676 MOV #340, @GRIVSA ;
1192 007454 012777 007512 171672 MOV #1$, @GRIVB ;LOAD OUTPUT VECTOR
1193 007462 012777 000340 171666 MOV #340, @GRIVSB ;
1194 007470 052777 020000 171636 BIS #BIT13, @GRSTAT ;MAINT. INTERRUPT
1195 007476 005037 177776 CLR PSW ;LOWER PSW
1196 007502 000240 NOP
1197 007504 000240 NOP
1198 007506 104005 ERROR 5 ;OUTPUT FAILED TO INTERRUPT
1199 007510 000415 BR TST43 ;;BR TO NEXT TEST
1200 ;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN, NO INTERRUPT WILL OCCUR
1201 ; IF IT DOES, 'INT DONE D HIGH' FAILED TO CLEAR 'INT D' FLOP
1202 007512 012777 007534 171634 1$: MOV #2$, @GRIVB ;LOAD OUTPUT VECTOR
1203 007520 022626 CMP (SP)+, (SP)+ ;POP THE STACK #4
1204 007522 005037 177776 CLR PSW
1205 007526 000240 NOP
1206 007530 000240 NOP
1207 007532 000404 BR TST43 ;;<NEXT TEST>
1208 007534 022626 2$: CMP (SP)+, (SP)+ ;POP THE STACK
1209 007536 104006 ERROR 6 ;UNEXPECTED INTERRUPT
1210 007540 005037 177776 CLR PSW

```

```

1207      ;:*****
(3)      ;*TEST 43      TEST FOR INTR. FROM DRA INPUT
(3)      ;:*****
(2) 007544 000004      †ST43: SCOPE
1209 007546 000240      NOP
1209 007550 000240      NOP
1210 007552 032777 002000 171360      BIT      #BIT10, @SWR      ;TEST SWITCH
1211 007560 001401      BEQ      1$
1212 007562 000463      BR      TST44      ;;
1213 007564 005077 171544      CLR      @GRSTAT      ;;
1214 007570 012777 007670 171552      1$: MOV      #2$, @GRIVA      ;IN CASE OF INTERRUPT
1215 007576 012777 000340 171546      MOV      #340, @GRIVSA
1216 007604 012777 007666 171542      MOV      #3$, @GRIVB
1217 007612 012777 000340 171536      MOV      #340, @GRIVSB
1218 007620 012777 000100 171506      MOV      #BIT6, @GRSTAT      ;ENABLE INPUT INTR.
1219 007626 012777 000000 171504      MOV      #0, @GRDIO      ;OUTPUT
1220 007634 017700 171476      MOV      @GRDAI, R0      ;INPUT
1221 007640 005037 177776      CLR      PSW      ;LOWER PSW
1222 007644 000240      NOP      ;LET INTERRUPT OCCUR
1223 007646 000240      NOP
1224 007650 000240      NOP
1225 007652 042777 000100 171454      BIC      #BIT6, @GRSTAT
1226 007660 000240      NOP
1227 007662 104004      ERROR 4      ;ERROR, INPUT FAILED TO INTERRUPT
1228 007664 000404      BR      4$      ;;
1229 007666 104006      3$: ERROR 6      ;UNEXPECTED OUTPUT INTERRUPT
1230 007670 022626      2$: CMP      (SP)+, (SP)+
1231 007672 005037 177776      CLR      PSW
1232 007676 005077 171432      4$: CLR      @GRSTAT      ;CLEAR STATUS
1233 007702 012777 002746 171440      MOV      #UNEXPT, @GRIVA      ;RESET INPUT VECTOR
(1) 007710 012777 000340 171434      MOV      #340, @GRIVSA
(1) 007716 012777 002746 171430      MOV      #UNEXPT, @GRIVB      ;RESET OUTPUT VECTOR
(1) 007724 012777 000340 171424      MOV      #340, @GRIVSB
1234

```

```

1236 (3) *****
1237 (3) :*TEST 44 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
1238 (2) *****
1239 (2) 007732 000004 TST44: SCOPE
1240 007734 032777 002000 171176 BIT #BIT10,ASWR ;TEST CABLE SWITCH
1241 007742 001067 BNE TST45 ;;BYPASS IF NO I/O CABLE
1242 007744 012737 007760 001110 MOV #1$,SLPERR ;LOAD ERROR SCOPE RETURN
1243 007752 012737 000001 001366 MOV #BIT0,BRLEV3 ;LOAD INTERRUPT BIT
1244 007760 005037 001126 1$: CLR $BDDAT ;CLEAR BAD DATA
1245 007764 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD GOOD DATA
1246 007772 033737 001366 001346 BIT BRLEV3,INTBIT ;TEST IF THIS BIT WILL INTERRUPT
1247 010000 001445 BEQ 3$ ;;NO TRY NEXT BIT
1248 010002 005077 171332 CLR @GRDIO ;CLEAR OUTPUT REGISTER
1249 010006 012777 177777 171322 MOV #-1,@GRDAI ;CLEAR INPUT
1250 010014 005077 171314 CLR @GRSTAT ;CLEAR STATUS
1251 010020 053777 001366 171312 BIS BRLEV3,@GRDIO ;LOAD OUTPUT
1252 010026 043777 001366 171304 BIC BRLEV3,@GRDIO ;CLEAR OUTPUT
1253 010034 053777 001366 171276 BIS BRLEV3,@GRDIO ;LOAD OUTPUT
1254 010042 005077 171266 CLR @GRSTAT ;CLEAR FLAG FROM DATA READY
1255 ;SHOULD REMAIN SET VIA DIRECT SET SIDE
1256 010046 105777 171262 TSTB @GRSTAT ;IF INTERRUPT INPUT SWITCH IS ON
1257 010052 100401 BMI 2$ ;TEST READY BIT
1258 010054 104010 ERROR 10 ;;BR IF SET
1259 ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
1260 ;?? DID OPERATOR GIVE CORRECT
1261 ;INPUT INTERRUPT BITS ??
1262 010056 043777 001366 171254 2$: BIC BRLEV3,@GRDIO ;CLEAR OUTPUT
1263 010064 053777 001366 171244 BIS BRLEV3,@GRDAI ;CLEAR INPUT
1264 010072 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
1265 010076 005077 171232 CLR @GRSTAT ;CLEAR STATUS
1266 010102 117737 171226 001126 MOVB @GRSTAT,$BDDAT ;READ STATUS
1267 010110 100001 BPL 3$ ;;BR IF CLEARED
1268 010112 104001 ERROR 1 ;INPUT READY FAILED TO CLEAR
1269
1270 010114 006337 001366 3$: ASL BRLEV3 ;CHANGE BIT
1271 010120 001317 BNE 1$ ;BR IF NOT DONE

```

```

1273      ;:*****
(3)      ;:TEST 45      TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
(3)      ;:*****
(2)      TST45:  SCOPE
1274 010122 000004      BIT      #BIT10,@SWR      ;TEST CABLE SWITCH
1275 010132 001050      BNE      TST46      ;;BYPASS IF NO I/O CABLE
1276
1277 010134 012737 010150 001110      MOV      #1$,SLPERR      ;LOAD ERROR SCOPE RETURN
1278 010142 012737 000001 001366      MOV      #BIT0,BRLEV3      ;LOAD NON-INTERRUPT BIT
1279 010150 012737 000200 001126 1$:  MOV      #200,$BDDAT      ;LOAD BAD DATA
1280 010156 005037 001124      CLR      $GDDAT      ;CLEAR GOOD DATA
1281
1282 010162 033737 001366 001346      BIT      BRLEV3,INTBIT      ;TEST IF THIS BIT WILL INTERRUPT
1283 010170 001026      BNE      3$      ;;YES SKIP AND TRY NEXT BIT
1284 010172 005077 171142      CLR      @GRDIO      ;CLEAR OUTPUT REGISTER
1285 010176 012777 177777 171132      MOV      #-1,@GRDAI      ;CLEAR INPUT
1286 010204 005077 171124      CLR      @GRSTAT      ;CLEAR STATUS
1287 010210 053777 001366 171122      BIS      BRLEV3,@GRDIO      ;LOAD OUTPUT
1288 010216 043777 001366 171114      BIC      BRLEV3,@GRDIO      ;CLEAR OUTPUT
1289 010224 053777 001366 171106      BIS      BRLEV3,@GRDIO      ;SET OUTPUT
1290 010232 005077 171076      CLR      @GRSTAT      ;CLEAR FLAG FROM DATA READY
1291      ;SHOULD REMAIN SET VIA DIRECT SET SIDE
1292 010236 105777 171072      TSTB    @GRSTAT      ;TEST READY BIT
1293 010242 100001      BPL     3$      ;;BR IF CLEAR
1294 010244 104011      ERROR   11      ;INPUT NON-INTERRUPT BIT SET INPUT READY
1295      ;?? DID OPERATOR GIVE CORRECT
1296      ;INPUT INTERRUPT BITS ??
1297
1298
1299 010246 006337 001366      3$:  ASL     BRLEV3      ;CHANGE BIT
1300 010252 001336      BNE     1$      ;BR IF NOT DONE

```

```

1302 (3) *****
1303 (3) *TEST 46 TEST FOR INTR. FROM DRA OUTPUT
1304 (2) *****
1305 010254 000004 TST46: SCOPE
1306 010256 000240 NOP
1307 010260 000240 NOP
1308 010262 032777 002000 170650 BIT #BIT10, @SWR ;TEST SWITCH
1309 010270 001401 BEQ 1$
1310 010272 000461 BR TST47 ;;
1311 010274 005077 171034 1$: CLR @GRSTAT
1312 010300 012777 010374 171046 MOV #2$, @GRIVB ;IN CASE OF INTERRUPT
1313 010306 012777 000340 171042 MOV #340, @GRIVSB
1314 010314 013777 001352 171026 MOV GRIVSA, @GRIVA
1315 010322 005077 171024 CLR @GRIVSA
1316 010326 012777 040000 171000 MOV #BIT14, @GRSTAT ;ENABLE OUTPUT INTR.
1317 010334 012777 000000 170776 MOV #0, @GRDIO ;OUTPUT
1318 010342 017700 170770 MOV @GRDAI, @R ;INPUT
1319 010346 005037 177776 CLR PSW ;LOWER PSW
1320 010352 000240 NOP ;LET INTERRUPT OCCUR
1321 010354 000240 NOP
1322 010356 000240 NOP
1323 010360 042777 040000 170746 BIC #BIT14, @GRSTAT
1324 010366 000240 NOP
1325 010370 104005 ERROR 5 ;ERROR, OUTPUT FAILED TO INTERRUPT
1326 010372 000401 BR 3$
1327 010374 022626 2$: CMP (SP)+, (SP)+ ;;
1328 010376 012777 002746 170744 3$: MOV #UNEXPT, @GRIVA ;RESET INPUT VECTOR
1329 010404 012777 000340 170740 MOV #340, @GRIVSA
1330 010412 012777 002746 170734 MOV #UNEXPT, @GRIVB ;RESET OUTPUT VECTOR
1331 010420 012777 000340 170730 MOV #340, @GRIVSB
1332 010426 005037 177776 CLR PSW
1333 010432 005077 170676 CLR @GRSTAT ;CLEAR STATUS
1334 *****
1335 *TEST 47 PRE-INTERRUPT SETUP
1336 *****
1337 (2) TST47: SCOPE
1338 (1) 010436 000004 MOV #1, $TIMES ;;DO 1 ITERATION
1339 010440 012737 000001 001166 RESET
1340 010446 000005 BIC #177437, @DIOBRL
1341 010450 042737 177437 001360 BNE 1$
1342 010456 001001 ERROR 7 ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 0
1343 010460 104007 ERROR 7
1344 010462 022737 000340 001360 1$: CMP #340, @DIOBRL
1345 010470 001001 BNE 2$
1346 010472 104007 ERROR 7 ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 7
1347 010474 013737 001360 001362 2$: MOV @DIOBRL, @BRLEV1
1348 010502 162737 000040 001362 SUB #40, @BRLEV1
1349 010510 013737 001360 001364 MOV @DIOBRL, @BRLEV2

```

```

1341 (3) ;*****
1342 (3) ;*TEST 50 TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT
1343 (2) ;*****
1344 (2) †T50: SCOPE
1345 010516 000004 CLR @GRSTAT ;CLEAR ENABLES
1346 010520 005077 170610 MOV #1$,@GRIVA ;SET UP VECTORS
1347 010524 012777 010576 170616 MOV BRLEV1,PSW ;CHANGE PSW
1348 010532 013737 001362 177776 TST JUMPER ;TEST IF FACTORY MODE
1349 010540 005737 001376 BEQ 2$ ;BR IF YES
1350 010544 001403 MOV #200,PSW ;LOAD BR LEVEL #5-1 IF LOC-BOX OR COULTER MODE
1351 010546 012737 000200 177776 2$: BIS #BITS,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
1352 010554 052777 000040 170552 NOP
1353 010562 000240 NOP
1354 010564 000240 NOP
1355 010566 000240 NOP
1356 010570 000240 ERROR 4 ;ERROR, NO INTERRUPT FROM DIGITAL I/O INPUT
1357 010572 104004 BR 3$ ;;BR TO NEXT TEST
1358 010574 000401 1$: CMP (SP)+,(SP)+
1359 010576 022626 3$: MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
1360 (1) 010600 012777 002746 170542 MOV #340,@GRIVSA ;RESET OUTPUT VECTOR
1361 (1) 010606 012777 000340 170536 MOV #UNEXPT,@GRIVB
1362 (1) 010614 012777 002746 170532 MOV #340,@GRIVSB
1363 (1) 010622 012777 000340 170526 CLR PSW ;LOWER PSW
1357 010630 005037 177776
1358 (3) ;*****
1359 (3) ;*TEST 51 TEST FOR NO INTR. FROM DRA INPUT ON LEVEL INDICATED VIA MAINT. INT
1360 (2) ;*****
1361 (2) †T51: SCOPE
1362 010634 000004 CLR @GRSTAT ;CLEAR ENABLES
1363 010636 005077 170472 MOV #1$,@GRIVA ;SET UP VECTORS
1364 010642 012777 010730 170500 MOV BRLEV2,PSW ;CHANGE PSW
1365 010650 013737 001364 177776 TST JUMPER ;TEST IF FACTORY MODE
1366 010656 005737 001376 BEQ 4$ ;BR IF YES
1367 010662 001403 MOV #240,PSW ;LOAD BR LEVEL #5 IF LOC-BOX OR COULTER MODE
1368 010664 012737 000240 177776 4$: BIS #BITS,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
1369 010672 052777 000040 170434 NOP
1370 010700 000240 NOP
1371 010702 000240 NOP
1372 010704 000240 ;SUB-TEST, NOW LOWER THE PSW AND ALLOW THE INTERRUPT
1373 010706 012777 010742 170434 MOV #2$,@GRIVA ;RESET VECTOR
1374 010714 005037 177776 CLR PSW ;LOWER PSW
1375 010720 000240 NOP
1376 010722 000240 NOP
1377 010724 000240 BR 3$ ;ERROR
1378 010726 000403 1$: CMP (SP)+,(SP)+
1379 010730 022626 3$: CLR PSW
1380 010732 005037 177776 ERROR 6 ;UNEXPECTED INTERRUPT <IS BR LEVEL PLUG CORRECT>
1381 010736 104006 BR 5$ ;;
1382 010740 000401 2$: CMP (SP)+,(SP)+ ;POP THE STACK
1383 010742 022626 5$: MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
1384 (1) 010744 012777 002746 170376 MOV #340,@GRIVSA ;RESET OUTPUT VECTOR
1385 (1) 010752 012777 000340 170372 MOV #UNEXPT,@GRIVB
1386 (1) 010760 012777 002746 170366

```

```

(1) 010766 012777 000340 170362      MOV  #340, @GRIVSB
1382 010774 005037 177776      CLR  PSW
1383
1384      ;:*****
(3)      ;*TEST 52      DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
(3)      ;:*****
(2) 011000 000004      †ST52: SCOPE
(1) 011002 012737 000001 001166      MOV  #1, $TIMES      ;;DO 1 ITERATION
1385 011010 005737 001320      BYPASS: TST  NBEXT      ;TEST IF ANY
1386 011014 001415      BEQ  1$      ;BR IF NONE
1387 011016 032777 010000 170114      BIT  #SW12, @SWR      ;TEST BIT12 OF SWR
1388 011024 001024      BNE  BYPAS1      ;INHIBIT TESTING NEXT DR11-K
1389 011026 162737 000010 001322      SUB  #10, DRADD      ;UPDATE DEVICE ADDRESS
1390 011034 062737 000010 001324      ADD  #10, DRIV      ;UPDATE DEVICE VECTOR
1391 011042 005337 001320      DEC  NBEXT      ;ANOTHER ONE ?
1392 011046 000413      BR  BYPAS1      ;BR IF ANOTHER
1393 011050 013737 001306 001322 1$: MOV  BASEBA, DRADD      ;RELOAD ADDRESS
1394 011056 013737 001310 001324      MOV  BASEIV, DRIV      ;RELOAD VECTOR
1395 011064 013737 001316 001320      MOV  NMBEXT, NBEXT      ;RELOAD NUMBER
1396 011072 000137 011106      JMP  $EOP      ;DONE
1397 011076 012700 177777      BYPAS1: MOV  #-1, R0
1398 011102 000137 002776      JMP  RBEG2      ;TEST ANOTHER UNIT
1399
1400      .SBTTL  END OF PASS ROUTINE.
(1)
(2)      ;:*****
(1)      ;*INCREMENT THE PASS NUMBER ($PASS)
(1)      ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
(1)      ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
(1)      ;*IF THERES A MONITOR GO TO IT
(1)      ;*IF THERE ISN'T JUMP TO BYPAS1
(1)
(1) 011106      $EOP:
(1) 011106 000004      SCOPE
(1) 011110 005037 001102      CLR  $STNM      ;;ZERO THE TEST NUMBER
(1) 011114 005037 001166      CLR  $TIMES      ;;ZERO THE NUMBER OF ITERATIONS
(1) 011120 005237 001100      INC  $PASS      ;;INCREMENT THE PASS NUMBER
(1) 011124 042737 100000 001100      BIC  #100000, $PASS      ;;DON'T ALLOW A NEG. NUMBER
(1) 011132 005327      DEC  (PC)+      ;;LOOP?
(1) 011134 000001      $EOPCT: .WORD 1
(1) 011136 003022      BGT  $DOAGN      ;;YES
(1) 011140 012737      MOV  (PC)+, @ (PC)+      ;;RESTORE COUNTER
(1) 011142 000001      $ENDCT: .WORD 1
(1) 011144 011134      $EOPCT
(1) 011146 104401 011213      TYPE  $ENDMG      ;;TYPE "END PASS #"
(2) 011152 013746 001100      MOV  $PASS, -(SP)      ;;SAVE $PASS FOR TYPEOUT
(2) 011156 104405      TYPDS      ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 011160 104401 011210      TYPE  $ENULL      ;;TYPE A NULL CHARACTER
(1) 011164 013700 000042      $GET42: MOV  @#42, R0      ;;GET MONITOR ADDRESS
(1) 011170 001405      BEQ  $DOAGN      ;;BRANCH IF NO MONITOR
(1) 011172 000005      RESET      ;;CLEAR THE WORLD
(1) 011174 004710      $ENDAD: JSR  PC, (R0)      ;;GO TO MONITOR
(1) 011176 000240      NOP      ;;SAVE ROOM
(1) 011200 000240      NOP      ;;FOR

```



```

(1) 011362 003002          BGT      8$          ;; GO TO EXIT
(1) 011364 010502          MOV      R5,R2      ;; GET THE LSD
(1) 011366 000764          BR       6$          ;; GO CHANGE TO ASCII
(1) 011370 105726          3$: TSTB   (SP)+      ;; WAS THE LSD THE FIRST NON-ZERO?
(1) 011372 100003          BPL     9$          ;; BR IF NO
(1) 011374 116663 177777 177776 9$: MOVB   -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
(1) 011402 105013          CLRB   (R3)        ;; SET THE TERMINATOR
(3) 011404 012605          MOV    (SP)+,R5    ;; POP STACK INTO R5
(3) 011406 012603          MOV    (SP)+,R3    ;; POP STACK INTO R3
(3) 011410 012602          MOV    (SP)+,R2    ;; POP STACK INTO R2
(3) 011412 012601          MOV    (SP)+,R1    ;; POP STACK INTO R1
(3) 011414 012600          MOV    (SP)+,R0    ;; POP STACK INTO R0
(1) 011416 104401 011444   TYPE   $DBLK        ;; NOW TYPE THE NUMBER
(1) 011422 016666 000002 000004 MOV    2(SP),4(SP) ;; ADJUST THE STACK
(1) 011430 012616          MOV    (SP)+,(SP)
(1) 011432 000002          RTI                    ;; RETURN TO USER
(1) 011434 023420          $DTBL: 10000.
(1) 011436 001750          1000.
(1) 011440 000144          100.
(1) 011442 000012          10.
(1) 011444 000004          $DBLK: .BLKW 4

1402
1403          .SBTTL MISC. EXTERNAL LOGIC TEST
1404 011454 012706 001100   EXTTST: MOV    #STACK,SP
1405 011460 004737 003036   JSR    PC,SETADD   ;; SET UP ADDRESS AND VECTOR
1406 011464 012737 040000 011560 2$: MOV    #BIT14,EXTCNT ;; LOAD COUNT
1407 011472          1$:
(1) 011472 005077 167642          CLR    @GRDIO      ;; CLEAR OUTPUT REGISTER
1408 011476 012777 125252 167634 MOV    #125252,@GRDIO ;; LOAD OUTPUT
1409 011504 017737 167626 011556 MOV    @GRDAI,EXTTMP ;; READ INPUT
1410 011512 013777 011556 167 16 MOV    EXTTMP,@GRDAI ;; CLEAR INPUT
1411 011520 005077 167614          CLR    @GRDIO      ;; CLEAR OUTPUT REGISTER
1412 011524 012777 052525 167606 MOV    #52525,@GRDIO ;; LOAD OUTPUT
1413 011532 017737 167600 011556 MOV    @GRDAI,EXTTMP ;; READ INPUT
1414 011540 013777 011556 167570 MOV    EXTTMP,@GRDAI ;; CLEAR INPUT
1415 011546 005337 011560   DEC    EXTCNT      ;; FINISHED COUNT
1416 011552 001347          BNE   1$
1417 011554 000743          BR    2$          ;; LOOP
1418
1419 011556 000000          EXTTMP: 0
1420 011560 000000          EXTCNT: 0
1421
1422          ;SUBROUTINE TO LOAD TRAP AND POWER FAIL VECTORS
1423 011562 012737 017466 000034 SETTRP: MOV    #STRAP,@TRAPVEC ;; LOAD TRAP VECTOR
1424 011570 012737 000340 000036 MOV    #340,@TRAPVEC+2
1425 011576 012737 016022 000024 MOV    #SPWRDN,@PWRVEC ;; LOAD POWER FAIL VECTOR
1426 011604 012737 000340 000026 MOV    #340,@PWRVEC+2
1427 011612 000207          R      PC          ;; EXIT
    
```

```

1429          .SBTTL COULTER INTERFACE TEST
1430 011614 012706 001100 COULTR: MOV #STACK,SP
1431 011620 004737 011562 JSR PC,SET+ ;LOAD TRAP AND POWER FAIL VECTORS
1432 011624 004737 003036 JSR PC,SET+ ;SETUP BUS ADDRESS AND VECTOR
1433 011630 005077 167500 CLR @GRSTAT ;CLEAR INPUT STATUS REG
1434 011634 012777 011676 167506 MOV #20$,@GRIVA ;LOAD INPUT VECTOR
1435 011642 012777 000340 167502 MOV #340,@GRIVSA
1436 011650 104401 013302 1$: TYPE,RUNMSG ;TELL OPERATOR TO "RUN SAMPLE"
1437 011654 012701 017550 MOV #BUFFER,R1 ;LOAD POINTER FOR RESULTS
1438 011660 012777 000100 167446 4$: MOV #BIT6,@GRSTAT ;ENABLE INTERRUPT
1439 011666 005037 177776 CLR PS ;LOWER PSW
1440 011672 000001 WAIT ;WAIT FOR OPERATOR
1441 011674 000771 BR 4$
1442 011676 022626 20$: CMP (SP)+,(SP)+ ;POP STACK
1443 011700 012737 000106 011556 MOV #70,EXTTMP ;LOAD # OF 1 MSEC DELAYS
1444 011706 013700 001314 6$: MOV CPU,R0 ;LOAD 1 MSEC. DELAY WEIGHT ;240 FOR 11/05 ;620 FOR 11/40
1445 011712 005300 3$: DEC R0 ;DELAY
1446 011714 001376 BNE 3$
1447 011716 005337 011556 DEC EXTTMP ;FINISHED ALL MSEC. DELAY ?
1448 011722 001371 BNE 6$ ;BR IF NOT
1449 011724 017700 167406 MOV @GRDAI,R0 ;SAVE RESULTS
1450 011730 010021 MOV RC,(R1)+ ;SAVE IN BUFFER
1451 011732 012777 177777 167376 MOV #-1,@GRDAI ;CLEAR INPUT
1452 011740 042700 007777 BIC #7777,R0 ;MASK
1453 011744 001345 BNE 4$ ;BR IF NOT LAST SAMPLE
1454 011746 010103 MOV R1,R3 ;COPY R1
1455 011750 014300 5$: MOV -(R3),R0 ;GET RESULTS
1456 011752 004737 011766 JSR PC,40$ ;CONVERT AND TYPE
1457 011756 022703 017550 CMP #BUFFER,R3 ;FINISHED ?
1458 011762 001732 BEQ 1$ ;BR IF YES
1459 011764 000771 BR 5$ ;CONT. TYPING UNTIL DONE
1460          ;SUBROUTINE FOR THE COULTER TEST
1461 011766 012737 000055 013324 40$: MOV #55,MSGRUS+2 ;FIX ASCII MESSAGE
1462 011774 012702 013331 MOV #MSPNT1,R2 ;LOAD DEST. POINTER
1463 012000 012737 000004 011556 MOV #4,EXTTMP ;LOAD COUNT
1464 012006 000404 BR 12$
1465 012010 006000 10$: ROR R0
1466 012012 006000 ROR R0
1467 012014 006000 ROR R0
1468 012016 006000 ROR R0
1469 012020 010001 12$: MOV R0,R1 ;LOAD R1
1470 012022 042701 177760 BIC #177760,R1 ;MASK
1471 012026 052701 000060 BIS #60,R1 ;MAKE DIGIT
1472 012032 110142 MOVB R1,-(R2) ;SAVE DIGIT
1473 012034 005337 011556 DEC EXTTMP ;FINISHED ?
1474 012040 001363 BNE 10$ ;BR IF NOT
1475 012042 000337 013324 SWAB MSGRUS+2 ;ADJUST MESSAGE
1476 012046 104401 013322 TYPE,MSGRUS
1477 012052 000207 RTS PC ;EXIT
    
```

```

1479          .SBTTL LOC-BOX LAMP AND SWITCH LOOP
1480 012054 012706 001100 LOCBOX: MOV #STACK,SP
1481 012060 004737 011562 JSR PC,SETTRP
1482 012064 104401 013062 TYPE ,LOCHDR ; TELL OPERATOR WHAT THIS IS
1483 012070 004537 012152 JSR R5,INADR ; GET THE ADDRESSES
1484 012074 001326 012332 LOC1,LOC1Y
1485 012100 000412 BR LOCA ; BR IF DONE
1486 012102 004537 012152 JSR R5,INADR ; GET NEXT ADDRESS
1487 012106 001330 012334 LOC2,LOC2Y
1488 012112 000405 BR LOCA ; BR IF DONE
1489 012114 004537 012152 JSR R5,INADR ; GET NEXT ADDRESS
1490 012120 001332 012336 LOC3,LOC3Y
1491 012124 000400 BR LOCA
1492
1493 012126 004537 012260 LOCA: JSR R5,LOUP1 ; TEST FOR SWITCHES AND LOAD LAMPS
1494 012132 001326 LOC1
1495 012134 004537 012260 JSR R5,LOUP1 ; LOAD NEXT SET
1496 012140 001330 LOC2
1497 012142 004537 012260 JSR R5,LOUP1 ; LOAD NEXT SET
1498 012146 001332 LOC3
1499
1500 012150 000766 BR LOCA ; LOOP BACK
1501
1502 012152 013537 012252 INADR: MOV @ (R5)+,10$ ; GET BUS ADDRESS
1503 012156 013537 012256 MOV @ (R5)+,12$ ; GET INDICATOR
1504 012162 104401 013123 1$: TYPE, INADRH ; ASK FOR INPUT
1505 012166 013746 012252 MOV 10$,-(SP) ; TYPE CURRENT ADDRESS
1506 012172 104402 TYPOC
1507 012174 104401 013166 TYPE, INADR ; ADD YES NO
1508 012200 104411 RDLIN ; READ CHAR. AND ECHO
1509 012202 000240 NOP
1510 012204 000240 NOP
1511 012206 013637 012254 MOV @ (SP)+,11$ ; GET THE ANSWER
1512 012212 042737 177640 012254 BIC #177640,11$ ; MASK OFF BITS
1513 012220 001413 BEQ 3$ ; BR IF CR
1514 012222 022737 000116 012254 CMP #'N,11$ ; TEST IF NO
1515 012230 001406 BEQ 2$ ; BR IF NO
1516 012232 022737 000131 012254 CMP #'Y,11$ ; TEST IF YES
1517 012240 001350 BNE 1$ ; BR IF OTHER
1518 012242 005277 000010 INC @12$ ; SET YES FLAG
1519 012246 005725 2$: TST (R5)+
1520 012250 000205 3$: RTS R5 ; EXIT
1521 012252 000000 10$: 0
1522 012254 000000 11$: 0
1523 012256 000000 12$: 0
    
```

```

1525
1526
1527 012260 013537 012326 ;SUBROUTINE TO LOAD THE LAMPS FROM THE LOC BOX SWITCHES
LOUP1: MOV 3(R5)+,40$ ;GET ARG.
1528 012264 001001 BNE 1$ ;BR IF YES
1529 012266 000205 RTS R5 ;EXIT
1530 012270 013702 012326 1$: MOV 40$,R2
1531 012274 062702 000002 ADD #2,R2
1532 012300 010203 MOV R2,R3
1533 012302 062703 000002 ADD #2,R3
1534 012306 011200 MOV (R2),R0
1535 012310 011301 MOV (R3),R1
1536 012312 040100 BIC R1,R0 ;MASK OFF BITS
1537 012314 041213 BIC (R2),(R3) ;LOAD OUTPUT BITS
1538 012316 060013 ADD R0,(R3) ;LOAD
1539 012320 012712 177777 MOV #-1,(R2) ;CLEAR INPUT
1540 012324 000205 RTS R5 ;EXIT
1541
1542 012326 000000 40$: 0
1543 012330 000000 41$: 0
1544 012332 000000 LOC1Y: 0 ;YES/NO ANSWER TO LOC BOX #1
1545 012334 000000 LOC2Y: 0
1546 012336 000000 LOC3Y: 0
1547
1548
1549 012340 012706 001100 XLO1AD: .SBTTL XLO1 ADJUSTMENT ROUTINE
1550 012344 004737 011562 MOV #STACK,SP
1551 012350 104401 013204 JSR PC,SETTRP
1552 012354 004537 012152 TYPE XLOHDR ;TELL OPERATOR
1553 012360 001326 012332 JSR R5,INADR ;GET BUS ADDRESS
1554 012364 000412 LOC1,LOC1Y
1555 012366 004537 012152 BR XLO10
1556 012372 001330 012334 JSR R5,INADR ;
1557 012376 000405 LOC2,LOC2Y
1558 012400 004537 012152 BR XLO10
1559 012404 001332 012336 JSR R5,INADR
1560 012410 000400 LOC3,LOC3Y
1561 BR XLO10
1562
1563 012412 XLO10:
(1) 012412 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1) 012416 000 000 .BYTE 0,0 ;LOAD CH# AND SHIFT COUNT
(1) 012420 000007 7 ;LOAD DATA MASK
(1) 012422 001326 LOC1 ;DR11K BUS ADDRESS
1569 012424 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1) 012430 001 003 .BYTE 1,3 ;LOAD CH# AND SHIFT COUNT
(1) 012432 000070 70 ;LOAD DATA MASK
(1) 012434 001326 LOC1 ;DR11K BUS ADDRESS
1570 012436 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1) 012442 002 006 .BYTE 2,6 ;LOAD CH# AND SHIFT COUNT
(1) 012444 000700 700 ;LOAD DATA MASK
(1) 012446 001326 LOC1 ;DR11K BUS ADDRESS
1571 012450 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1) 012454 003 011 .BYTE 3,11 ;LOAD CH# AND SHIFT COUNT
(1) 012456 007000 7000 ;LOAD DATA MASK
(1) 012460 001326 LOC1 ;DR11K BUS ADDRESS
    
```

1572	012462	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012466	004	014	.BYTE	4,14	:LOAD CH# AND SHIFT COUNT
(1)	012470	070000		70000		:LOAD DATA MASK
(1)	012472	001326		LOC1		:DR11K BUS ADDRESS
1573	012474	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012500	005	000	.BYTE	5,0	:LOAD CH# AND SHIFT COUNT
(1)	012502	000007		7		:LOAD DATA MASK
(1)	012504	001330		LOC2		:DR11K BUS ADDRESS
1574	012506	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012512	006	003	.BYTE	6,3	:LOAD CH# AND SHIFT COUNT
(1)	012514	000070		70		:LOAD DATA MASK
(1)	012516	001330		LOC2		:DR11K BUS ADDRESS
1575	012520	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012524	007	006	.BYTE	7,6	:LOAD CH# AND SHIFT COUNT
(1)	012526	000700		700		:LOAD DATA MASK
(1)	012530	001330		LOC2		:DR11K BUS ADDRESS
1576	012532	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012536	010	011	.BYTE	10,11	:LOAD CH# AND SHIFT COUNT
(1)	012540	007000		7000		:LOAD DATA MASK
(1)	012542	001330		LOC2		:DR11K BUS ADDRESS
1577	012544	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012550	011	014	.BYTE	11,14	:LOAD CH# AND SHIFT COUNT
(1)	012552	070000		70000		:LOAD DATA MASK
(1)	012554	001330		LOC2		:DR11K BUS ADDRESS
1578	012556	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012562	012	000	.BYTE	12,0	:LOAD CH# AND SHIFT COUNT
(1)	012564	000007		7		:LOAD DATA MASK
(1)	012566	001332		LOC3		:DR11K BUS ADDRESS
1579	012570	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012574	013	003	.BYTE	13,3	:LOAD CH# AND SHIFT COUNT
(1)	012576	000070		70		:LOAD DATA MASK
(1)	012600	001332		LOC3		:DR11K BUS ADDRESS
1580	012602	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012606	014	006	.BYTE	14,6	:LOAD CH# AND SHIFT COUNT
(1)	012610	000700		700		:LOAD DATA MASK
(1)	012612	001332		LOC3		:DR11K BUS ADDRESS
1581	012614	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012620	015	011	.BYTE	15,11	:LOAD CH# AND SHIFT COUNT
(1)	012622	007000		7000		:LOAD DATA MASK
(1)	012624	001332		LOC3		:DR11K BUS ADDRESS
1582	012626	004537	012644	JSR	R5,XLOADJ	:NOW CONVERT THE DATA AND DISPLAY IN LAMPS
(1)	012632	016	014	.BYTE	16,14	:LOAD CH# AND SHIFT COUNT
(1)	012634	070000		70000		:LOAD DATA MASK
(1)	012636	001332		LOC3		:DR11K BUS ADDRESS
1583	012640	000137	012412	JMP	XLO10	
1584						
1585						
1586	012644	112577	000206	:ANALOG CONVERSION AND LOC BOX DRIVER		
1587	012650	052777	020000	XLOADJ: MOVB	(R5)+,ADCS1	:LOAD CHANNEL #
1588	012656	112537	013046	BIS	#BIT13,ADCS	:SET UNIPOLAR
1589	012662	012537	013052	MOVB	(R5)+,40\$:GET SHIFT COUNT
1590	012666	013537	013050	MOV	(R5)+,42\$:GET DATA MASK
1591	012672	001464		MOV	2(R5)+,41\$:GET DR11K BUS ADDRESS
1592	012674	105277	000154	BEQ	70\$:BR IF NONE
				INCB	ADCS	:CONVERT THE DATA

1593	012700	005001			CLR	R1	
1594	012702	062737	000004	013050	ADD	#4,41\$;GET LAMP ADDRESS
1595	012710	105777	000140		TSTB	ADCS	;WAIT FOR DONE
1596	012714	100375			BPL	1\$	
1597	012716	017700	000136		MOV	ADBR,RO	;READ THE DATA
1598	012722	162700	000004		SUB	#4,RO	;TEST IF AT UPPER END
1599	012726	100432			BMI	7\$;BR IF YES
1600	012730	162700	000014		SUB	#14,RO	;TEST IF NEAR UPPER END
1601	012734	100425			BMI	6\$;BR IF YES
1602	012736	162700	000060		SUB	#60,RO	;TEST IF GETTING NEAR UPPER END
1603	012742	100420			BMI	5\$;BR IF YES
1604	012744	162700	001530		SUB	#1530,RO	;TEST IF GETTING NEAR LOWER END
1605	012750	100407			BMI	2\$;BR IF YES
1606	012752	162700	000060		SUB	#60,RO	;TEST IF NEAR LOWER END
1607	012756	100406			BMI	3\$;BR IF YES
1608	012760	162700	000014		SUB	#14,RO	;TEST IF AT LOWER END
1609	012764	100405			BMI	4\$;BR IF YES
1610	012766	000412			BR	7\$;BR IF AT STOP
1611	012770	012701	000004		MOV	#4,R1	;LOAD #
1612	012774	062701	000002		ADD	#2,R1	
1613	013000	005201			INC	R1	
1614	013002	000404			BR	7\$;BR AND LOAD LAMP'S
1615	013004	012701	000002		MOV	#2,R1	;LOAD #
1616	013010	062701	000004		ADD	#4,R1	
1617	013014	013700	013046		MOV	40\$,RO	;LOAD SHIFT COUNT
1618	013020	006301			ASL	R1	;SHUFFEL THE DATA
1619	013022	005300			DEC	RO	
1620	013024	100375			BPL	10\$;BR IF NOT DONE
1621	013026	006201			ASR	R1	;RE ADJUST
1622	013030	043777	013052	000012	BIC	42\$,AD41\$;CLEAR OLD DATA IN LAMPS
1623	013036	050177	000006		BIS	R1,AD41\$;LOAD THE LAMPS
1624	013042	005001			CLR	R1	
1625	013044	000205			RTS	R5	;EXIT
1626							
1627	013046	000000			0		
1628	013050	000000			0		
1629	013052	000000			0		

1631	013054	170400			ADCS:	170400	
1632	013056	170401			ADCS1:	170401	
1633	013060	170402			ADBR:	170402	
1634	013062	005015	047514	026503	LOCHDR:	.ASCIZ	<15><12>/LOC-BOX SWITCH AND LAMP LOOP/<15><12>
	013070	047502	020130	053523			
	013076	052111	044103	040440			
	013104	042116	046040	046501			
	013112	020120	047514	050117			
	013120	005015	000				
1635	013123	015	052412	042523	INADRH:	.ASCIZ	<15><12>/USE LOC-BOX <DR11K AT BUS ADDR. /
	013130	046040	041517	041055			
	013136	054117	036040	051104			
	013144	030461	020113	052101			
	013152	041040	051525	040440			
1636	013160	042104	027122	000040	INDADR:	.ASCIZ	/ > Y OR N ? /
	013166	037040	020040	020131			
	013174	051117	047040	037440			
	013202	000040					
1637	013204	005015	046130	030460	XLOWDR:	.ASCIZ	<15><12>/XLO1 POT ADJUSTMENT LOOP/<15><12>
	013212	050040	052117	040440			
	013220	045104	051525	046524			
	013226	047105	020124	047514			
	013234	050117	005015	000			
1638	013241	015	041012	051525	BUSTRP:	.ASCIZ	<15><12>/BUS TIME-OUT ON SELECTED DR11K/
	013246	052040	046511	026505			
	013254	052517	020124	047117			
	013262	051440	046105	041505			
	013270	042524	020104	051104			
	013276	030461	000113				
1639	013302	005015	051012	047125	RUNMSG:	.ASCIZ	<15><12><12>/RUN SAMPLE/<15><12>
	013310	051440	046501	046120			
	013316	006505	000012				
1640	013322	005015	047055	047116	MSGRUS:	.ASCII	<15><12>/-NNNN/
	013330	116					
1641	013331	000			MSPNT1:	.BYTE	0
1642						.EVEN	
1643							
1644							
1645	013332	042523	020124	053523	SWNLB:	.ASCIZ	/SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING INPUT BITS/
	013340	052111	044103	051040			
	013346	043505	051511	042524			
	013354	020122	044502	051524			
	013362	042440	052521	046101			
	013370	052040	020117	044124			
	013376	020105	047516	026516			
	013404	040514	041524	044510			
	013412	043516	044440	050116			
	013420	052125	041040	052111			
	013426	000123					
1646	013430	042523	020124	053523	SWINTB:	.ASCIZ	/SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING INPUT BITS/
	013436	052111	044103	051040			
	013444	043505	051511	042524			
	013452	020122	044502	051524			
	013460	042440	052521	046101			

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

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

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

(1)	015730	001002		BNE	4\$:: TEST FOR 0
(1)	015732	005704		TST	R4	:: SUPPRESS THIS 0?
(1)	015734	001403		BEQ	5\$:: BR IF YES
(1)	015736	005204		INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
(1)	015740	052703	000060	BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
(1)	015744	052703	000040	BIS	#' R3	:: MAKE ASCII IF NOT ALREADY
(1)	015750	110337	016014	MOVB	R3, 8\$:: SAVE FOR TYPING
(1)	015754	104401	016014	TYPE	8\$:: GO TYPE THIS DIGIT
(1)	015760	105337	016016	7\$: DECB	\$OCNT	:: COUNT BY 1
(1)	015764	003347		BGT	2\$:: BR IF MORE TO DO
(1)	015766	002402		BLT	6\$:: BR IF DONE
(1)	015770	005204		INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
(1)	015772	000744		BR	2\$:: GO DO THE LAST DIGIT
(1)	015774	012605		6\$: MOV	(SP)+,R5	:: RESTORE R5
(1)	015776	012604		MOV	(SP)+,R4	:: RESTORE R4
(1)	016000	012603		MOV	(SP)+,R3	:: RESTORE R3
(1)	016002	016666	000002 000004	MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
(1)	016010	012616		MOV	(SP)+,(SP)	
(1)	016012	000002		RTI		:: RETURN
(1)	016014	000		8\$: .BYTE	0	:: STORAGE FOR ASCII DIGIT
(1)	016015	000		.BYTE	00	:: TERMINATOR FOR TYPE ROUTINE
(1)	016016	000		\$OCNT: .BYTE	00	:: OCTAL DIGIT COUNTER
(1)	016017	000		\$OFI: .BYTE	00	:: ZERO FILL SWITCH
(1)	016020	000000		\$OMODE: .WORD	0	:: NUMBER OF DIGITS TO TYPE

1683
1684

.SBTTL POWER DOWN AND UP ROUTINES

(1)
(2)
(1)
(1) 016022 012737 016166 000024
(1) 016030 012737 000340 000026
(3) 016036 010046
(3) 016040 010146
(3) 016042 010246
(3) 016044 010346
(3) 016046 010446
(3) 016050 010546
(3) 016052 017746 163062
(1) 016056 010637 016172
(1) 016062 012737 016074 000024
(1) 016070 000000
(1) 016072 000776

::*****

:POWER DOWN ROUTINE

\$PWRDN: MOV \$SILLUP, @PWRVEC ;; SET FOR FAST UP
MOV #340, @PWRVEC+2 ;; PRIO:7
RO, -(SP) ;; PUSH R0 ON STACK
R1, -(SP) ;; PUSH R1 ON STACK
R2, -(SP) ;; PUSH R2 ON STACK
R3, -(SP) ;; PUSH R3 ON STACK
R4, -(SP) ;; PUSH R4 ON STACK
R5, -(SP) ;; PUSH R5 ON STACK
@SWR, -(SP) ;; PUSH @SWR ON STACK
SP, \$SAVR6 ;; SAVE SP
MOV \$PWRUP, @PWRVEC ;; SET UP VECTOR
HALT
BR -2 ;; HANG UP

::*****

:POWER UP ROUTINE

(1) 016074 012737 016166 000024
(1) 016102 013706 016172
(1) 016106 005037 016172
(1) 016112 005237 016172
(1) 016116 001375
(3) 016120 012677 163014
(3) 016124 012605
(3) 016126 012604
(3) 016130 012603
(3) 016132 012602
(3) 016134 012601
(3) 016136 012600
(1) 016140 012737 016022 000024
(1) 016146 012737 000340 000026
(1) 016154 104401
(1) 016156 016174
(1) 016160 012716
(1) 016162 002752
(1) 016164 000002
(1) 016166 000000
(1) 016170 000776
(1) 016172 000000

\$PWRUP: MOV \$SILLUP, @PWRVEC ;; SET FOR FAST DOWN
MOV \$SAVR6, SP ;; GET SP
CLR \$SAVR6 ;; WAIT LOOP FOR THE TTY
1\$: INC \$SAVR6 ;; WAIT FOR THE INC
BNE 1\$;; OF WORD
MOV (SP)+, @SWR ;; POP STACK INTO @SWR
MOV (SP)+, R5 ;; POP STACK INTO R5
MOV (SP)+, R4 ;; POP STACK INTO R4
MOV (SP)+, R3 ;; POP STACK INTO R3
MOV (SP)+, R2 ;; POP STACK INTO R2
MOV (SP)+, R1 ;; POP STACK INTO R1
MOV (SP)+, R0 ;; POP STACK INTO R0
MOV \$PWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
MOV #340, @PWRVEC+2 ;; PRIO:7
TYPE PWRMSG ;; REPORT THE POWER FAILURE
\$PWRMG: .WORD PWRMSG ;; POWER FAIL MESSAGE POINTER
MOV (PC)+, (SP) ;; RESTART AT IOTEST
\$PWRAD: .WORD IOTEST ;; RESTART ADDRESS
RTI
\$SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
\$SAVR6: 0 ;; PUT THE SP HERE

1685 016174 005015 042522 052123
016202 051101 044524 043516
016210 040440 052106 051105
016216 040440 050040 053517
016224 051105 043040 044501
016232 052514 042522 005015
016240 000

1686

016242

.EVEN

1688
1689

.SBTTL TYPE ROUTINE

```

(1) ;*****
(2) ;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 REQUIRED.
(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) $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
(1) 016242 105737 001157 BPL 1$ ;; BR IF YES
(1) 016246 100002 HALT ;; HALT HERE IF NO TERMINAL
(1) 016250 000000 BR 3$ ;; LEAVE
(1) 016252 000407 1$: MOV RO,-(SP) ;; SAVE RO
(1) 016254 010046 MOV 22(SP),RO ;; GET ADDRESS OF ASCIZ STRING
(1) 016256 017600 000002 2$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 016262 112046 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
(1) 016264 001005 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
(1) 016266 005726 60$: MOV (SP)+,RO ;; RESTORE RO
(1) 016270 012600 3$: ADD #2,(SP) ;; ADJUST RETURN PC
(1) 016272 062716 000002 RTI ;; RETURN
(1) 016276 000002 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
(1) 016300 122716 000011 BEQ 8$
(1) 016304 001430 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
(1) 016306 122716 000200 BNE 5$
(1) 016312 001006 TST (SP)+ ;; POP <CR><LF> EQUIV
(1) 016314 005726 TYPE ;; TYPE A CR AND LF
(1) 016316 104401 $CRLF
(1) 016320 001173 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
(1) 016322 105037 016456 BR 2$ ;; GET NEXT CHARACTER
(1) 016326 000755 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
(1) 016330 004737 016412 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
(1) 016334 123726 001156 BNE 2$ ;; IF NO GO GET NEXT CHAR.
(1) 016340 001350 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
(1) 016342 013746 001154 AND THE NULL CHAR.
(1) 016346 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
(1) 016352 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 016354 004737 016412 JSR PC,$TYPEC ;; GO TYPE A NULL
(1) 016360 105337 016456 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
(1) 016364 000770 BR 7$ ;; LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 8$: MOVB #' ,(SP) ;; REPLACE TAB WITH SPACE
(1) 016366 112716 000040 9$: JSR PC,$TYPEC ;; TYPE A SPACE
(1) 016372 004737 016412

```

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

```

1691          .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 $HIOCT
(1)          016462 011646          SRDOCT: MOV      (SP), -(SP)      ;; PROVIDE SPACE FOR THE
(1)          016464 016666 000004 000002  MOV      4(SP), 2(SP)      ;; INPUT NUMBER
(3)          016472 010046          MOV      R0, -(SP)        ;; PUSH R0 ON STACK
(3)          016474 010146          MOV      R1, -(SP)        ;; PUSH R1 ON STACK
(3)          016476 010246          MOV      R2, -(SP)        ;; PUSH R2 ON STACK
(1)          016500 104411          1$:      RDLIN          ;; READ AN ASCII LINE
(1)          016502 012600          MOV      (SP)+, R0        ;; GET ADDRESS OF 1ST CHARACTER
(1)          016504 005001          CLR      R1              ;; CLEAR DATA WORD
(1)          016506 005002          CLR      R2
(1)          016510 112046          2$:      MOVB      (R0)+, -(SP)      ;; PICKUP THIS CHARACTER
(1)          016512 001412          BEQ      3$              ;; IF ZERO GET OUT
(1)          016514 006301          ASL      R1              ;; *2
(1)          016516 006102          ROL      R2
(1)          016520 006301          ASL      R1              ;; *4
(1)          016522 006102          ROL      R2
(1)          016524 006301          ASL      R1              ;; *8
(1)          016526 006102          ROL      R2
(1)          016530 042716 177770          BIC      #1C7, (SP)      ;; STRIP THE ASCII JUNK
(1)          016534 062601          ADD      (SP)+, R1        ;; ADD IN THIS DIGIT
(1)          016536 000764          BR       2$              ;; LOOP
(1)          016540 005726          3$:      TST      (SP)+          ;; CLEAN TERMINATOR FROM STACK
(1)          016542 010166 000012          MOV      R1, 12(SP)      ;; SAVE THE RESULT
(1)          016546 010237 016562          MOV      R2, $HIOCT
(3)          016552 012602          MOV      (SP)+, R2
(3)          016554 012601          MOV      (SP)+, R1
(3)          016556 012600          MOV      (SP)+, R0
(1)          016560 000002          RTI
(1)          016562 000000          $HIOCT: .WORD 0          ;; HIGH ORDER BITS GO HERE

```

```

1692          .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)          016564 022737 000176 001140 $CKSWR: CMP      #SWREG, SWR      ;; IS THE SOFT-SWR SELECTED?
(1)          016572 001074          BNE      1$              ;; BRANCH IF NO
(1)          016574 105777 162344          TSTB     @STKS          ;; CHAR THERE?
(1)          016600 100071          BPL      1$              ;; IF NO, DON'T WAIT AROUND
(1)          016602 117746 162340          MOVB     @STKB, -(SP)    ;; SAVE THE CHAR
(1)          016606 042716 177600          BIC      #1C177, (SP)   ;; STRIP-OFF THE ASCII

```

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


```

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

```


SW6	=	000100	390#					
SW7	=	000200	390#					
SW8	=	000400	390#					
SW9	=	001000	390#					
TALK		002666	568	571	574	577	594	627#
TBITVE	=	000014	390#					
TKVEC	=	000060	390#					
TPVEC	=	000064	390#					
TRANST		001374	491#	559*	566*	579	585	682
TRAPVE	=	000034	390#	515*	1423*	1424*		
TRTVEC	=	000014	390#					
TSTNUM		015020	1667	1668	1669	1671#	1675*	
TST1		003242	684#					
TST10		003604	745#					
TST11		003670	760#					
TST12		003766	768	771#				
TST13		004064	779	782#				
TST14		004152	794#					
TST15		004240	806#					
TST16		004310	813	816#				
TST17		004352	822	825#				
TST2		003260	689#					
TST20		004410	830	833#				
TST21		004452	839	843#				
TST22		004514	849	860#				
TST23		004612	872	875#				
TST24		004676	884	888#				
TST25		004762	897	900#				
TST26		005046	909	912#				
TST27		005226	914	942#				
TST3		003316	694	697#				
TST30		005406	977#					
TST31		005524	999#					
TST32		005664	1025#					
TST33		006260	1047	1050#				
TST34		006654	1072	1075#				
TST35		006732	1083	1086#				
TST36		007002	1093	1098#				
TST37		007050	1105	1109#				
TST4		003354	702	705#				
TST40		007162	1124	1131#				
TST41		007274	1142	1152	1158#			
TST42		007432	1176	1178	1182#			
TST43		007544	1193	1201	1207#			
TST44		007732	1212	1235#				
TST45		010122	1238	1273#				
TST46		010254	1275	1302#				
TST47		010436	1307	1329#				
TST5		003412	710	713#				
TST50		010516	1341#					
TST51		010634	1358#					
TST52		011000	1384#					
TST6		003450	718	721#				
TST7		003526	732#					

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

DR11-K LOGIC
DZDRGD.P11

TEST MAINDEC-11-DZDRGD
CROSS REFERENCE TABLE

MACY11 27(663) 19-DEC-76 09:22 PAGE 42-11

SEQ 0080

ADD	666 1681	668 1689	670 1691	673 1692	675	677	1390	1401	1531	1533	1538	1594	1612	1616	1678
ASL	743	757	973	936	1022	1270	1299	1618	1678	1691	1692	1696			
ASLB	1401														
ASR	1621														
BCC	592	940	1401												
BEG	533	546	501	586	618	694	702	710	718	727	740	754	768	779	789
	801	830	839	849	862	872	884	897	909	914	917	925	935	952	958
	969	992	1018	1047	1072	1083	1093	1176	1211	1246	1306	1346	1363	1386	1400
	1458	1513	1515	1591	1674	1675	1678	1681	1689	1691	1692				
BGE	1674														
BGT	1400	1401	1681	1692											
BHI	605	1674													
BIC	616	750	870	879	882	892	895	904	907	922	956	964	966	1011	1012
	1028	1029	1053	1054	1082	1225	1251	1262	1288	1320	1331	1400	1452	1470	1512
	1536	1537	1622	1681	1691	1692									
BIS	681	682	737	932	995	1021	1040	1065	1137	1165	1188	1250	1252	1263	1287
	1289	1348	1365	1401	1471	1587	1623	1681	1692						
BISB	1678														
BIT	529	532	583	585	610	617	812	821	861	916	918	929	949	951	968
	983	985	1005	1007	1175	1210	1237	1245	1274	1282	1305	1387	1674	1675	
BITB	1689														
BLO	601	1692													
BLOS	1692														
BLT	1401	1681	1689	1692											
BMI	1105	1257	1401	1599	1601	1603	1605	1607	1609						
BNE	515	528	530	544	555	562	611	632	659	730	744	758	792	804	813
	822	919	930	950	974	984	986	997	1006	1008	1023	1238	1271	1275	1283
	1300	1332	1335	1388	1401	1416	1446	1448	1453	1474	1517	1528	1674	1675	1678
	1681	1684	1689	1692											
BPL	1267	1293	1401	1596	1620	1675	1681	1689	1692						
BR	506	509	512	515	525	536	548	549	553	603	607	612	615	636	663
	1124	1142	1152	1178	1193	1201	1212	1228	1307	1323	1354	1375	1379	1392	1401
	1417	1441	1459	1464	1485	1488	1491	1500	1554	1557	1560	1610	1614	1674	1678
	1681	1684	1689	1691	1692										
CLR	504	505	508	511	515	521	556	558	559	565	680	685	686	687	698
	723	736	807	817	834	844	845	865	867	876	889	901	967	980	989
	1002	1014	1031	1040	1056	1065	1076	1073	1088	1100	1111	1112	1123	1132	1138
	1149	1156	1159	1166	1172	1173	1181	1183	1189	1198	1204	1213	1221	1231	1232
	1242	1247	1249	1253	1264	1265	1280	1284	1286	1290	1308	1312	1316	1325	1327
	1342	1357	1359	1371	1377	1382	1400	1401	1407	1411	1433	1439	1593	1624	1674
	1678	1681	1684	1691	1692										
CLRB	1401	1674	1689	1692											
CMP	515	526	554	561	600	604	620	631	657	658	693	709	717	726	739
	753	767	778	788	800	829	838	871	883	896	908	924	934	957	991
	1017	1046	1071	1090	1092	1102	1148	1154	1171	1179	1197	1202	1230	1324	1334
	1355	1376	1380	1401	1442	1457	1514	1516	1674	1675	1692				
	1674	1689	1692												
CMPB	752	765	776	947	1040	1065									
COM	538	1391	1400	1415	1445	1447	1473	1619	1678	1692					
DECB	1681	1689													
EMT	390														
HALT	394	534	602	606	619	639	1675	1684	1689						
INC	524	729	1400	1401	1518	1613	1674	1675	1681	1684	1692				

007

DR11-K LOGIC TEST MAINDEC-11-DZDRGD
DZDRGD.P11 CROSS REFERENCE TABLE

MACY11 27(663) 19-DEC-76 08:22 PAGE 42-14

SEQ 0083

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

ERRORS DETECTED: 0

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