

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 48 frames of digital test data, arranged in 8 rows and 6 columns. Each frame displays a different set of test results, including binary patterns, timing diagrams, and data tables. The frames are separated by a grid of small holes. The data is printed in a light color on a dark background, typical of microfiche technology.

801

00010ZDRG080411
020900.F11

00010000LOGIC 708224MAINDEC-11-DZ0000 411 MACY1128AC000ZDRG0804-76 09:22 P008:0000

770224
SEP 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 LCC-BOX JUMPER'S
 - 9.6 H321 FUNCTION LOOP
 - 9.7 XLO1 ADJJSTMENT 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 D #3 AND PDL D #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 SJB-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 NMMN.'
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 (700)

*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 16 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 SHCULD BE QUESTIONED? THE PROGRAM MAY OCCASIONLY MISS A SWITCH CLOSURE.

THE BUS ADDRESSES OF THE DR11K'S ARE 167770, 167760 AND 157750. 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 -- FDL04" THE H321 LAMP'S OPERATE AS SHOWN BELOW. REFER TO THE PDL-11 INSTALLATION MANUAL FOR ADJUSTMENT DETAILS.

APPROACHING	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
;*COPYRIGHT (C) 1976
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY RAYMOND SHOOP
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
;*

```

389

.SBTTL BASIC DEFINITIONS

390

```

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

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

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

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

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

```

```

(1) 040000
(1) 020000
(1) 010000
(1) 004000
(1) 002000
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001

```

```

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

```

```

(1) 100000
(1) 040000
(1) 020000
(1) 010000
(1) 004000
(1) 002000
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001

```

```

:*DATA BIT DEFINITIONS (BIT00 TO BIT15)
BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8
.EQUIV BIT07,BIT7
.EQUIV BIT06,BIT6
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3
.EQUIV BIT02,BIT2
.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

```


442			:ITEM	7		
443	001256	014253		EM7	:OPERATOR INTERVENTION ERROR	
444	001260	014633		DH4	:ERRPC DRADD	
445	001262	014754		DT4	:SERRPC DRADD	
446	001264	015002		DF0		
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	:SERRPC DRADD \$BDDAT \$GDDAT BRLEV3	
452	001274	015002		DF0		
453						
454			:ITEM	11		
455	001276	014372		EM11	:NON-INTERRUPTING INPUT BIT SET INPUT READY	
456	001300	014660		DH11	:ERRPC DRADD STATUS EXPECTED INPUT BIT	
457	001302	014764		DT10	:SERRPC DRADD \$BDDAT \$GDDAT BRLEV3	
458	001304	015002		DF0		
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					: 620 FOR 11/40	
465	001316	000000	NMBEXT:	0	:ADDITIONAL DR-11-K	
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	GSTAT:	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	DIOBR:	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	:1 IF RUNNING H322 JUMPER CONFIG ;2 IF COLLIER JUMPER CONFIG	
493						

```

501 ::*****
502 : DIGITAL I-O LOGIC TEST
503 :*****
504 BEGIN: CLR RO ;CLEAR RO
505 001400 005000 001376 CLR JUMPER ;INDICATE NORMAL FACTORY BUILD
506 001402 005037 001376 BR RBEG
507 001406 000414 COULTJ: MOV #2,JUMPER ;INDICATE COULTER JUMPER CONFIG
508 001410 012737 000002 001376 CLR RO
509 001416 005000 BR RBEG
510 001420 000407 H322J: MOV #1,JUMPER ;INDICATE LOC BOX JUMPER CONFIG
511 001422 012737 000001 001376 CLR RO
512 001430 005000 BR RBEG
513 001432 000402 BEGIN1: MOV #-1,RO ;LOAD RO
514 001434 012700 RBEG: RESET
515 001440 000005 .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 00502E 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 #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001574 012737 001574 001110 MOV #,$SLPERR ;: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 55$ ;:BRANCH IF NO TIMEOUT
(2) 001642 012716 001650 64$: MOV #65$,(SP) ;:SET UP FOR TRAP RETJRN
(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

```



```

562 002270 001015          BNE      3$          ;BR IF NOT
563 002272 C12737 177777 001344  MOV     #-1,NOTLCH ;LOAD COULTER J MPER 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     RS,TALK     ;TALK TO THE ANIMALS
569 002330 013332          SWNLB
570 002332 001344          NOTLCH ;ABOUT NOM LATCH INPUT BITS
571 002334 004537 002666  JSR     RS,TALK     ;TALK TO THE ANIMAL AGAIN
572 002340 013430          SWINTB
573 002342 001346          INTBIT ;ABOUT THE INTERRUPTING BITS
574 002344 004537 002666  JSR     RS,TALK     ;ISN'T THIS BOARRING
575 002350 013526          SWPOSB
576 002352 001372          MINSIN ;AGAIN-ABOUT THE MINUS INPUT BITS
577 002354 004537 002666  JSR     RS,TALK     ;HC-HUM
578 002360 013622          SJTRAB
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     $GDDAT,MINSIN ;TEST IF NEG INPUT BIT
584 002400 001407          BEQ     -1$        ;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     RS,TALK     ;ASK THE OPERATOR
595 002432 013720          SWDPOB
596 002434 001366          BRLEV3 ;ABOUT PROGRAM OPTIONS
597 002436 000137 002752          JMP     IOTEST
598
599          ;INTERRUPT TO UNEXPECTED VECTOR
600 002442 021627 001000  VECTRP: CMP     (SP),#1000 ;TEST IF IN PROGRAM CODE
601 002446 103402          BLO     1$        ;BR IF NOT
602 002450 000000          70$:  HALT ;FATAL BUS TRAP IN PROGRAM AREA
603 002452 000776          BR     70$
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$:  HALT ;FATAL VECTOR TRAP
607 002464 000776          BR     71$
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,DSWR ;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
(2)
(3)          ;:65$: .ASCIZ <15><12>/DR11K INTERRUPTED TO LOC.
(4)          64$:

```

```

613 002554 013746 002664      MOV      725, -(SP)          ;LOAD VALUE TO BE TYPED
614 002560 104402              TYP0C
615 002562 104401 002570      TYPE      675              ;;TYPE ASCIZ STRING
        002566 000420          BR        665              ;;GET OVER THE ASCIZ
        (1) (1)                ;;675: .ASCIZ / AND WILL NOW USE THAT VECTOR<<15>><12>
        (1) (1)                665:
616 002630 042737 000007 002664 35:      BIC      #7, 725          ;MASK OFF LOWER 3 BITS
617 002636 032777 100000 176274      BIT      #SW15, 2SWR      ;TEST IF HALT ON ERROR
618 002644 001401              BEQ      45              ;BR IF NOT
619 002646 000000          HALT      ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
520 002650 022626          45:      CMP      (SP)+, (SP)+      ;CLEAN THE STACK
621 002652 013737 002664 001324      MOV      725, DR1V      ;LOAD NEW VECTOR ADDRESS
622 002660 000137 002776          JMP      RBEG2          ;TEST THE DR11K AT THE NEW VECTOR
623 002664 000000          725:      0
624
625                ;OPERATOR QUESTIONS AND ANSWER ROUTINE
626
627 002667 012537 002700      TALK:    MOV      (R5)+, 105      ;GET ASCII POINTER
628 002672 012537 002744          MOV      (R5)+, 115      ;GET POINTER TO DATA FROM OPERATOR
629 002676 104401              TYPE
630 002700 013332          105:      SWNLB          ;TELL OPERATOR TO LOAD SWITCHES
631 002702 023727 001140 000176      CMP      SWR, #SWREG      ;TEST IF SWITCH REG. EXISTS
632 002710 001006          BNE      15              ;BR IF NO
633 002712 104401 017443          TYPE      .SMSWR        ;TYPE "SWR ="
634 002716 104412          RDOCT          ;READ OCTAL
635 002720 012677 176214          MOV      (SP)+, 2SWR      ;SAVE THE SWITCHES
636 002724 000403          BR        25
637 002726 104401          15:      TYPE
638 002730 014005          DEPCNT        ;TELL OPERATOR TO DEPRESS CONT
639 002732 000000          HALT          ;WAIT FOR OPERATOR
640 002734 017777 176200 000002 25:      MOV      2SWR, 2115      ;LOAD SWITCH VALUE
641 002742 000205          RTS      R5              ;EXIT
642 002744 001344          115:     NOTLCH          ;POINTER TO DATA TO BE LOADED
643
644                ;UNEXPECTED INTERRUPT
645 002746 104006          UNEXPT: ERROR 6          ;UNEXPECTED INTERRUPT DURING A SUB-TEST
646 002750 000002          RTI              ;EXIT
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, DR1V     ;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          55:      MOV      R2, (R1)+        ;SAVE THE ADDRESS
656 003012 012721 004700          MOV      #4700, (R1)+     ;LOAD "JSR PC, RC"
657 003016 022222          CMP      (R2)+, (R2)+     ;BUMP R2
658 003020 020227 001076          CMP      R2, #SCMTAG-2   ;TEST FOR LAST
659 003027 001371          BNE      55              ;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 0C1334 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 0C1340    MOV    DRADD,GRDIO        ;LOAD 3RD ADDRESS
668 003066 062737 000004 001340    ADD    #4,GRDIO
669 003074 013737 001322 001342    MOV    DRADD,GRBPIO       ;LOAD 4TH ADDRESS
670 003102 062737 000005 001342    ADD    #5,GRBPIO
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,DI0BR      ;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)          ;*TEST 1          TEST FOR NO BUS ERRORS
(3)          ;*****
(2) 003242 000004          IOTST1: 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          IOTST2: 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     IOT3             ;;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 RETJRN
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)      ;*TEST 7      FLOAT A 1 ACROSS THE OUTPUT REGISTER
(3)      ;:*****
(2)      003526 000004
733      003530 012737 003544 001110      †ST7:  SCOPE
734      003536 012737 000001 001124      MOV      #15,$LPERR      ;LOAD SCOPE ERROR RETURN
735      ;:
736      003544 005077 175570      1$:    CLR      @GRDIO      ;CLEAR OUTPUT
737      003550 053777 001124 175562      BIS      $GDDAT,@GRDIO      ;SET THAT BIT
738      003556 017737 175556 001126      MOV      @GRDIO,$BDDAT      ;READ OUTPUT REG.
739      003564 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;TEST RESULTS
740      003572 001401      BEQ      2$      ;BR IF EQUAL ?
741      003574 104003      ERROR    3
742      ;:
743      003576 006337 001124      2$:    ASL      $GDDAT      ;SHIFT EXPECTED DATA
744      003602 001360      BNE      1$      ;BR UNTIL DONE
745      ;:*****
(3)      ;*TEST 10     FLOAT A 0 ACROSS THE OUTPUT REGISTER
(3)      ;:*****
(2)      003604 000004
746      003606 012737 003622 001110      †ST10: SCOPE
747      003614 012737 000001 001124      MOV      #15,$LPERR      ;LOAD SCOPE ERROR RETURN
748      ;:
749      003622 012777 177777 175510      1$:    MOV      #-1,@GRDIO      ;LOAD OUTPUT TO A ONE
750      003630 043777 001124 175502      BIC      $GDDAT,@GRDIO      ;CLEAR A BIT
751      003636 017737 175476 001126      MOV      @GRDIO,$BDDAT      ;READ OUTPUT REG.
752      003644 005137 001126      COM      $BDDAT      ;COMPLEMENT IT
753      003650 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;EQUAL ?
754      003656 001401      BEQ      2$      ;BR IF EQUAL
755      003660 104003      ERROR    3
756      ;:
757      003662 006337 001124      2$:    ASL      $GDDAT      ;SHIFT LEFT
758      003666 001355      BNE      1$      ;BRANCH UNTIL DONE
759      ;:*****
(3)      ;*TEST 11     TEST FOR SLOW OUTPUT GATES WITH #125252
(3)      ;:*****
(2)      003670 000004
761      003672 012737 125252 001124      †ST11: SCOPE
762      003700 013777 001124 175432      MOV      #125252,$GDDAT      ;LOAD EXPECTED VALUE
763      003706 005177 175426      MOV      $GDDAT,@GRDIO      ;LOAD OUTPUT
764      (1) 003712 005177 175422      COM      @GRDIO      ;COMPLEMENT OUTPUT REG.
765      (1) 003716 005177 175416      COM      @GRDIO      ;COMPLEMENT OUTPUT REG.
766      (1) 003722 005177 175412      COM      @GRDIO      ;COMPLEMENT OUTPUT REG.
767      (1) 003726 005177 175406      COM      @GRDIO      ;COMPLEMENT OUTPUT REG.
768      (1) 003732 005177 175402      COM      @GRDIO      ;COMPLEMENT OUTPUT REG.
769      (1) 003736 005177 175376      COM      @GRDIO      ;COMPLEMENT OUTPUT REG.
770      (1) 003742 005177 175372      COM      @GRDIO      ;COMPLEMENT OUTPUT REG.
771      003746 017737 175366 001126      MOV      @GRDIO,$BDDAT      ;READ OUTPUT REG.
772      003754 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;TEST REGISTER
773      003762 001401      BEQ      TST12      ;:BR IF CORRECT
774      003764 104003      ERROR    3

```

```

771 (3) *****
772 (3) *TEST 12 TEST FOR SLOW OUTPUT GATES WITH #52525
773 (3) *****
774 (2) ST12: SCOPE
775 (1) 003766 000004 MOV #52525,$GDDAT ;LOAD EXPECTED VALUE
776 (1) 003770 012737 052525 001124 MOV $GDDAT,$GRDIO ;LOAD OUTPUT REG.
777 (1) 003776 013777 001124 175334 COM $GRDIO ;COMPLEMENT OUTPUT REG.
778 (1) 004004 005177 175330 COM $GRDIO ;COMPLEMENT OUTPUT REG.
779 (1) 004010 005177 175324 COM $GRDIO ;COMPLEMENT OUTPUT REG.
780 (1) 004014 005177 175320 COM $GRDIO ;COMPLEMENT OUTPUT REG.
781 (1) 004020 005177 175314 COM $GRDIO ;COMPLEMENT OUTPUT REG.
782 (1) 004024 005177 175310 COM $GRDIO ;COMPLEMENT OUTPUT REG.
783 (1) 004030 005177 175304 COM $GRDIO ;COMPLEMENT OUTPUT REG.
784 (1) 004034 005177 175300 COM $GRDIO ;COMPLEMENT OUTPUT REG.
785 (1) 004040 005177 175274 COM $GRDIO ;COMPLEMENT OUTPUT REG.
786 (1) 004044 017737 175270 001126 MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
787 (1) 004052 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST PATTERN
788 (1) 004060 001401 BEQ TST13 ;;BR IF EQUAL
789 (1) 004062 104003 ERROR 3 ;OUTPUT REGISTER IN ERROR
790 *****
791 (3) *TEST 13 TEST THAT OUTPUT CAN HOLD LOW BYTE COUNT PATTERN
792 (3) *****
793 (2) TST13: SCOPE
794 (1) 004064 000004 MOV #400,$TIMES ;;DO 400 ITERATIONS
795 (1) 004066 012737 000400 001166 MOV #2,$LPERR ;LOAD SCOPE ERROR RETURN
796 (1) 004074 012737 004116 001110 MOV #52400,$GDDAT ;LOAD EXPECTED DATA
797 (1) 004102 012737 052400 001124 MOV $GDDAT,$GRDIO ;LOAD OUTPUT REG.
798 (1) 004110 013777 001124 175222 2$: MOVB $GDDAT,$GRDIO ;LOAD THE OUTPUT
799 (1) 004116 113777 001124 175214 MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
800 (1) 004124 017737 175210 001126 CMP $GDDAT,$BDDAT
801 (1) 004132 023737 001124 001126 BEQ 1$ ;BRANCH IF EQUAL
802 (1) 004140 001401 ERROR 3 ;ERROR, OUTPUT REG. IN ERROR
803 (1) 004142 104003 1$: INCB $GDDAT ;UPDATE PATTERN
804 (1) 004144 105237 001124 BNE 2$
805 (1) 004150 001362 *****
806 (3) *TEST 14 TEST THAT OUTPUT CAN HOLD HIGH BYTE COUNT PATTERN
807 (3) *****
808 (2) TST14: SCOPE
809 (1) 004152 000004 MOV #400,$TIMES ;;DO 400 ITERATIONS
810 (1) 004154 012737 000400 001166 MOV #2,$LPERR ;LOAD SCOPE ERROR RETURN
811 (1) 004162 012737 004204 001110 MOV #252,$GDDAT ;LOAD EXPECTED
812 (1) 004170 012737 000252 001124 MOV $GDDAT,$GRDIO ;LOAD OUTPUT REG.
813 (1) 004176 013777 001124 175134 2$: MOVB $GDDAT+1,$GREHIO ;LOAD THE HIGH BYTE
814 (1) 004204 113777 001125 175130 MOV $GRDIO,$BDDAT ;READ OUTPUT REG.
815 (1) 004212 017737 175122 001126 CMP $GDDAT,$BDDAT
816 (1) 004220 023737 001124 001126 BEQ 1$ ;BRANCH IF EQUAL
817 (1) 004226 001401 ERROR 3 ;ERROR, OUTPUT REG. IN ERROR
818 (1) 004230 104003 1$: INCB $GDDAT+1
819 (1) 004232 105237 001125 BNE 2$
820 (1) 004236 001362

```

```

806      ;:*****
(3)      ;*TEST 15      TEST OUTPUT DATA ACCEPT FLAG
(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)      ;*TEST 16      TEST OUTPUT INTERRUPT ENABLE
(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)      ;*TEST 17      TEST INPUT DATA READY FLAG
(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)      ;*TEST 20      TEST INPUT INTERRUPT ENABLE
(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

```

```

;:*****
;*TEST 15      TEST OUTPUT DATA ACCEPT FLAG
;:*****
TST15:  SCOPE
        CLR      @GRDIO
        MOV      #-1,@GRDAI
        MOV      #BIT15,$GDDAT      ;LOAD EXPECTED
        MOV      @GRSTAT,$BDDAT    ;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,$GDDAT  ;LOAD EXPECTED
        MOV      @GRSTAT,$BDDAT ;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,$GDDAT   ;LOAD EXPECTED
        MOV      @GRSTAT,$BDDAT ;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,$GDDAT  ;LOAD EXPECTED
        MOV      @GRSTAT,$BDDAT ;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 174645
845 004466 005037 001124
846 004472 012777 140300 174634
847 004500 00000E
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 007050
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 IS ;BRANCH IF DOWN
MOVB #36, $STNM ;LOAD NEW TEST NUMBER
JMP DRT21 ;BYPASS SOME TEST USING THE EXTERNAL CABLE

IS:
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 @DJMP, $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 @DJMP, $GDDAT ;MASK
MOV $GDDAT, @GRDIO ;LOAD THE OUTPUT
MOV @GRDAI, $BDDAT ;READ INPUT
BIC @DJMP, $BDDAT ;MASK
CMP $GDDAT, $BDDAT ;COMPARE
BEQ TST24 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL THE OUTPUT

```

```

888      ::*****
(3)      :*TEST 24      TEST INPUT WITH #52525
(3)      :*****
(2)      †ST24:  SCOPE
889      004676  000004      CLR      @GRDIO      ;CLEAR OUTPUT REGISTER
890      004700  005077  174434  MOV      #-1,@GRDAI  ;CLEAR INPUT
891      004704  012777  177777  174424  MOV      #52525,$GDDAT ;LOAD EXPECTED
892      004712  012737  052525  001124  BIC      ODDJMP,$GDDAT ;MASK
893      004720  043737  001370  001124  MOV      $GDDAT,@GRDIO ;LOAD THE OUTPUT
894      004726  013777  001124  174404  MOV      @GRDAI,$BDDAT ;READ INPUT
895      004734  017737  174376  001126  BIC      ODDJMP,$BDDAT ;MASK
896      004742  043737  001370  001126  CMP      $GDDAT,$BDDAT ;COMPARE
897      004750  023737  001124  001126  BEQ      TST25      ;;BR IF EQUAL
898      004756  001401      ERROR      2      ;ERROR, INPUT DID NOT EQUAL OUTPUT
899      004760  104002
900      ::*****
(3)      :*TEST 25      TEST INPUT WITH #125252
(3)      :*****
(2)      †ST25:  SCOPE
901      004762  000004      CLR      @GRDIO      ;CLEAR OUTPUT REGISTER
902      004764  005077  174350  MOV      #-1,@GRDAI  ;CLEAR INPUT
903      004770  012777  177777  174340  MOV      #125252,$GDDAT ;LOAD EXPECTED
904      004776  012737  125252  001124  BIC      ODDJMP,$GDDAT ;MASK
905      005004  043737  001370  001124  MOV      $GDDAT,@GRDIO ;LOAD THE OUTPUT
906      005012  013777  001124  174320  MOV      @GRDAI,$BDDAT ;READ INPUT
907      005020  017737  174312  001126  BIC      ODDJMP,$BDDAT ;MASK
908      005026  043737  001370  001126  CMP      $GDDAT,$BDDAT ;COMPARE
909      005034  023737  001124  001126  BEQ      TST26      ;;BR IF EQUAL
910      005042  001401      ERROR      2      ;ERROR, INPUT DID NOT EQUAL OUTPUT
910      005044  104002

```

```

912      ;:*****
(3)      ;*TEST 26      TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
(3)      ;:*****
(2)      005046 00C004      †ST26: 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      $GDDAT      ;CHANGE DATA PATTERN
940      005224 103322      BCC     1$      ;BR IF MORE DATA

```

```

942          ::*****
943          :*TEST 27      FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
944          :*****
945          †ST27:  SCOPE
946          005226 000004          MOV      #25,$LPERR          ;LOAD ERROR SCOPE RETURN
947          005230 012737 005256 001110          ;
948          005236 012737 000001 001364          MOV      #BIT0,BRLEV2          ;LOAD EXPECTED
949          005244 013737 001344 001366          MOV      NOTLCH,BRLEV3          ;GET NON-LATCH
950          005252 005137 001366          COM          ;COMPLEMENT
951          005256 013737 001364 001124 23:      MOV      BRLEV2,$GDDAT          ;LOAD GOOD
952          005264 033737 001124 001370          BIT      $GDDAT,ODDJMP          ;TEST IF ODC JUMPER
953          005272 001042          BNE      1$          ;BYPASS IF ODC JUMPER
954          005274 033737 001124 001344          BIT      $GDDAT,NOTLCH          ;TEST FOR NON-LATCH
955          005302 001436          BEQ      1$          ;BR IF LATCHING
956          005304 013777 001124 174026          MOV      $GDDAT,$GRDIO          ;LOAD OUTPUT
957          005312 017737 174020 001126          MOV      $GRDAI,$BDDAT          ;READ INPUT
958          005320 043737 001366 001126          BIC      BRLEV3,$BDDAT          ;MASK TO LATCH BITS
959          005326 023737 001124 001126          CMP      $GDDAT,$BDDAT          ;COMPARE
960          005334 001401          BEQ      3$          ::BR IF EQUAL
961          005336 104002          ERROR    2          ;INPUT REGISTER IN ERROR
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          ;
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
979      005406 000004      MOV      #25,$LPERR      ;LOAD ERROR SCOPE RETURN
979      005410 012737 005436 001110      MOV      #BIT0,$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      #25,$LPERR      ;LOAD ERROR SCOPE RETURN
1001     005526 012737 005554 001110      MOV      #BIT0,BRLEV3    ;LOAD EXPECTED
1002     005534 012737 000001 001366      CLR      @GRDIO          ;CLEAR OUTPUT REGISTER
1003     005542 005077 173572      MOV      #-1,@GRDAI     ;CLEAR INPUT
1004
1005     005554 033737 001366 001344 2$:  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) :*TEST 32 TEST FOR SLOW INPUT GATES WITH #125252
(3) :*****
(2) 005664 000004 †ST32: SCOPE
1026
1027 005666 012737 125252 001124 MOV #125252,$GDDAT ;LOAD EXPECTED
1028 005674 043737 001344 001124 BIC NOTLCH,$GDDAT ;CONVERT
1029 005702 043737 001370 001124 BIC ODDJMP,$GDDAT ;MASK
1030 005710 013700 001124 MOV $GDDAT,R0 ;LOAD PATTERN
1031 005714 005077 173420 CLR @GRDIO ;CLEAR OUTPUT REGISTER
1032 005720 012777 177777 173410 MOV #-1,@GRDAI ;CLEAR INPUT
1033
1040 005726 010077 173406 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 005732 005077 173402 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 005736 017701 173374 MOV @GRDAI,R1 ;READ INPUT
(1) 005742 050177 173370 BIS R1,@GRDAI ;CLEAR INPUT
(1) 005746 005100 COM R0
(1) 005750 010077 173364 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 005754 005077 173360 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 005760 017701 173352 MOV @GRDAI,R1 ;READ INPUT
(1) 005764 050177 173346 BIS R1,@GRDAI ;CLEAR INPUT
(1) 005770 005100 COM R0
(1) 005772 010077 173342 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 005776 005077 173336 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006002 017701 173330 MOV @GRDAI,R1 ;READ INPUT
(1) 006006 050177 173324 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006012 005100 COM R0
(1) 006014 010077 173320 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 006020 005077 173314 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006024 017701 173306 MOV @GRDAI,R1 ;READ INPUT
(1) 006030 050177 173302 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006034 005100 COM R0
(1) 006036 010077 173276 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 006042 005077 173272 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006046 017701 173264 MOV @GRDAI,R1 ;READ INPUT
(1) 006052 050177 173260 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006056 005100 COM R0
(1) 006060 010077 173254 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 006064 005077 173250 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006070 017701 173242 MOV @GRDAI,R1 ;READ INPUT
(1) 006074 050177 173236 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006100 005100 COM R0
(1) 006102 010077 173232 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 006106 005077 173226 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006112 017701 173220 MOV @GRDAI,R1 ;READ INPUT
(1) 006116 050177 173214 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006122 005100 COM R0
(1) 006124 010077 173210 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 006130 005077 173204 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006134 017701 173176 MOV @GRDAI,R1 ;READ INPUT
(1) 006140 050177 173172 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006144 005100 COM R0
(1) 006146 010077 173166 MOV R0,@GRDIO ;LOAD OUTPUT
(2) 006152 005077 173162 CLR @GRDIO ;CLEAR OUTPUT REGISTER

```

```

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

```

(1)	006432	010077	172702	MOV	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	PO, @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 005670 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)
(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,$GDDAT ;LOAD EXPECTED
CLR @GRSTAT
MOV #0,@GRDIC ;OUTPUT 0
CMP #0,#0 ;DELAY
MOV @GRSTAT,$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

```

J03

DR11-K LOGIC TEST MAINDEC-11-DZDRGD MACY11 27(663) 19-DEC-76 08:22 PAGE 25
 DZCRGC.P11 T36 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET

SEQ 0037

```

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
:*****
†ST37: SCOPE
MOV #40, $TIMES ;;DC 40 ITERATIONS
RESET
CLR @GRSTAT
CLR PSW
MOV #15, @GRIVA ;SET UP INTERRUPT INPUT VECTOR
MOV #340, @GRIVSA
MOV #25, @GRIVB ;SET UP INTERRUPT OUTPUT VECTOR
MOV #340, @GRIVSB
MOV #0, @GRDIO ;OUTPUT
MOV @GRDAI, RO ;INPUT
NOP
NOP
NOP
NOP
CLR @GRSTAT ;CLEAR STATUS
BR TST40 ;;
1$: 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
:*****
†ST40: SCOPE
CLR @GRSTAT ;CLEAR STATUS
MOV #15, @GRIVA ;LOAD RETURN VECTOR
MOV #340, @GRIVSA
MOV #25, @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 FLAG
1$: MOV #25, @GRIVA ;LOAD INPUT VECTOR
CMP (SP)+, (SP)+ ;POP THE STACK *4
CLR PSW ;LOWER PRIOR.
NOP
NOP
BR TST41 ;;<NEXT TEST>
2$: 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 0039

1155 007266 104306
1156 007270 005037 177776

ERROR 6
CLR FGW

:UNEXPECTED INTERRUPT

```

1158      ;*****
1159      ;*TEST 41      TEST THAT THE INPUT INTR. CLEARS INT. ENABLE VIA MAINT. BIT
1160      ;*****
1161      TST41: SCOPE
1162      CLR      @GRSTAT      ;CLEAR STATUS
1163      MOV      #15,@GRIVA    ;LOAD RETURN VECTOR
1164      MOV      #340,@GRIVSA  ;
1165      MOV      #25,@GRIVB    ;LOAD OUTPUT VECTOR
1166      MOV      #340,@GRIVSB  ;
1167      MOV      #BIT6,@GRSTAT ;LOAD INPUT INTR. ENABLE
1168      BIS      #BITS,@GRSTAT ;MAINT. INT C
1169      CLR      PSW          ;LOWER PSW
1170      NOP
1171      NOP
1172      ERROR   4            ;ERROR, INPUT FAILED TO INTERRUPT
1173      MOV      #25,@GRIVA    ;LOAD INPUT VECTOR
1174      CMP      (SP)+,(SP)+   ;POP THE STACK #4
1175      CLR      PSW          ;LOWER PRIOR.
1176      CLR      $GDDAT        ;CLEAR EXPECTED
1177      MOV      @GRSTAT,$BDDAT ;READ STATUS
1178      BIT      #BIT6,$BDDAT  ;TEST BIT 6
1179      BEQ      TST42        ;;BR IF CLEARED
1180      ERROR   1            ;INPUT INTR. FAILED TO CLEAR
1181      BR       TST42        ;;<NEXT TEST>
1182      2$:     CMP      (SP)+,(SP)+ ;POP THE STACK
1183      ERROR   6            ;UNEXPECTED INTERRUPT
1184      CLR      PSW
1185      ;*****
1186      ;*TEST 42      TEST THAT THE OUTPUT CAN INTR. USING THE MAINT. BIT
1187      ;*****
1188      TST42: SCOPE
1189      CLR      @GRSTAT      ;CLEAR STATUS
1190      MOV      #25,@GRIVA    ;LOAD INPUT VECTOR
1191      MOV      #340,@GRIVSA  ;
1192      MOV      #15,@GRIVB    ;LOAD OUTPUT VECTOR
1193      MOV      #340,@GRIVSB  ;
1194      BIS      #BIT13,@GRSTAT ;MAINT. INTERRUPT
1195      CLR      PSW          ;LOWER PSW
1196      NOP
1197      NOP
1198      ERROR   5            ;OUTPUT FAILED TO INTERRUPT
1199      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      1$:     MOV      #25,@GRIVB ;LOAD OUTPUT VECTOR
1203      CMP      (SP)+,(SP)+   ;POP THE STACK #4
1204      CLR      PSW
1205      NOP
1206      NOP
1207      BR       TST43        ;;<NEXT TEST>
1208      2$:     CMP      (SP)+,(SP)+ ;POP THE STACK
1209      ERROR   6            ;UNEXPECTED INTERRUPT
1210      CLR      PSW

```

```

1207 (3) ;*****
      (3) ;*TEST 43 TEST FOR INTR. FROM DRA INPUT
      (2) ;*****
1208 007544 000004 ↑TST43: SCOPE
1209 007546 000240 NOP
1210 007552 032777 002000 171360 BIT #BIT10,DSWR ;TEST SWITCH
1211 007560 001401 BEQ 1$
1212 007562 000463 BR TST44 ;;
1213 007564 005077 171544 1$: CLR @GRSTAT ;;
1214 007570 012777 007670 171552 MOV #25,@GRIVA ;IN CASE OF INTERRUPT
1215 007576 012777 000340 171546 MOV #340,@GRIVSA
1216 007604 012777 007666 171542 MOV #35,@GRIVB
1217 007612 012777 000340 171536 MOV #340,@GRIVSB
1218 007620 012777 000100 171506 MOV #BIT6,@GRSTAT ;ENABLE INPUT INTR.
1219 007626 012777 000000 171504 MOV #0,@GRDIO ;OUTPLT
1220 007634 017700 171476 MOV @GRDAI,R0 ;INPUT
1221 007640 005037 177776 CLR PSW ;LOWER PSW
1222 007644 000240 NOP ;LET INTERRUPT OCCUR
1223 007646 000240 NOP
1224 007650 000240 NOP
1225 007652 042777 000100 171454 BIC #BIT6,@GRSTAT
1226 007660 000240 NOP
1227 007662 104004 ERROR 4 ;ERROR, INPUT FAILED TO INTERRUPT
1228 007664 000404 BR 4$ ;;
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 INPLT VECTOR
      007710 012777 000340 171434 MOV #340,@GRIVSA
      007716 012777 002746 171430 MOV #UNEXPT,@GRIVB ;RESET OUTPUT VECTOR
1234 007724 012777 000340 171424 MOV #340,@GRIVSB

```

```

1236 (3)
1237 (3)
1238 (2) 007732 006004
1239 007734 032777 002000 171175
1240 007742 001067
1241 007744 012737 007760 001110
1242 007752 012737 000001 001366
1243 007760 005037 001126
1244 007764 012737 000200 001124
1245 007772 033737 001366 001346
1246 010000 001445
1247 010002 005077 171332
1248 010006 012777 177777 171322
1249 010014 005077 171314
1250 010020 053777 001366 171312
1251 010026 043777 001366 171304
1252 010034 053777 001366 171276
1253 010042 005077 171266
1254
1255
1256 010046 105777 171262
1257 010052 100401
1258 010054 104010
1259
1260
1261
1262 010056 043777 001366 171254 2S:
1263 010064 053777 001366 171244
1264 010072 005037 001124
1265 010076 005077 171232
1266 010102 117737 171226 001126
1267 010110 100001
1268 010112 104001
1269
1270 010114 006337 001366 3S:
1271 010120 001317

```

```

*****
: TEST 44 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
*****
↑ST44: SCOPE
BIT #BIT10, 2SWR ;TEST CABLE SWITCH
BNE TST45 ;;BYPASS IF NO I/O CABLE
MOV #1S, $LPERR ;LOAD ERROR SCOPE RETURN
MOV #BIT0, BRLEV3 ;LOAD INTERRUPT BIT
1S: CLR $BDDAT ;CLEAR BAD DATA
MOV #BIT7, $GDDAT ;LOAD GOOD DATA
BIT BRLEV3, INTBIT ;TEST IF THIS BIT WILL INTERRUPT
BEQ 3S ;;NO TRY NEXT BIT
CLR 2GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1, 2GRDAI ;CLEAR INPUT
CLR 2GRSTAT ;CLEAR STATUS
BIS BRLEV3, 2GRDIO ;LOAD OUTPUT
BIC BRLEV3, 2GRDIO ;CLEAR OUTPUT
BIS BRLEV3, 2GRDIO ;LOAD OUTPUT
CLR 2GRSTAT ;CLEAR FLAG FROM DATA READY
;SHOULD REMAIN SET VIA DIRECT SET SIDE
;IF INTERRUPT INPUT SWITCH IS ON
;TEST READY BIT
2S: TSTB 2GRSTAT
BMI 2S ;;BR IF SET
ERROR 10 ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
;?? DID OPERATOR GIVE CORRECT
;INPUT INTERRUPT BITS ??
BIC BRLEV3, 2GRDIO ;CLEAR OUTPUT
BIS BRLEV3, 2GRDAI ;CLEAR INPUT
CLR $GDDAT ;CLEAR EXPECTED
CLR 2GRSTAT ;CLEAR STATUS
MOV 2GRSTAT, $BDDAT ;READ STATUS
3S: BPL 3S ;;BR IF CLEARED
ERROR 1 ;INPUT READY FAILED TO CLEAR
ASL BRLEV3 ;CHANGE BIT
BNE 1S ;BR IF NOT DONE

```

```

1273          ;:*****
(3)          ;*TEST 45      TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
(3)          ;:*****
(2) 010122 000004
1274 010124 032777 002000 171006 TST45: SCOPE
1275 010132 001050          BIT      #BIT10,@SWR          ;TEST CABLE SWITCH
          BNE      TST46          ;;BYPASS IF NO I/O CABLE
1276
1277 010134 012737 010150 001110          MOV      #15,$LPERR          ;LOAD ERROR SCOPE RETURN
1278 010142 012737 000001 001366          MOV      #BIT0,BRLEV3          ;LOAD NON-INTERRUPT BIT
1279 010150 012737 000200 001126 1$: 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)
1304      (2) 010254 000004
1305      010256 000240
1306      010260 000240
1307      010262 032777 002000 170650
1308      010270 001401
1309      010272 000461
1310      010274 005077 171034
1311      010300 012777 010374 171046
1312      010306 012777 000340 171042
1313      010314 013777 001352 171026
1314      010322 005077 171024
1315      010326 012777 040000 171000
1316      010334 012777 000000 170776
1317      010342 017700 170770
1318      010346 005037 177776
1319      010352 000240
1320      010354 000240
1321      010356 000240
1322      010360 042777 040000 170746
1323      010366 000240
1324      010370 104005
1325      010372 000401
1326      010374 022626
1327      010376
1328      (1) 010376 012777 002746 170744
1329      (1) 010404 012777 000340 170740
1330      (1) 010412 012777 002746 170734
1331      (1) 010420 012777 000340 170730
1332      010426 005037 177776
1333      010432 005077 170676
1334      (3)
1335      (3)
1336      (2) 010436 000004
1337      (1) 010440 012737 000001 001166
1338      010446 000005
1339      010450 042737 177437 001360
1340      010456 001001
1341      010460 104007
1342      010462 022737 000340 001360
1343      010470 001001
1344      010472 104007
1345      010474 013737 001360 001362
1346      010502 162737 000040 001362
1347      010510 013737 001360 001364

*****
*TEST 46 TEST FOR INTR. FROM DRA OUTPUT
*****
TST46: SCOPE
NOP
NOP
BIT #BIT10,DSWR ;TEST SWITCH
BEQ 1$
BR TST47 ;;
CLR @GRSTAT
MOV #25,@GRIVB ;IN CASE OF INTERRUPT
MOV #340,@GRIVSB
MOV GRIVSA,@GRIVA
CLR @GRIVSA
MOV #BIT14,@GRSTAT ;ENABLE OUTPUT INTR.
MOV #0,@GRDIO ;OUTPUT
MOV @GRDAI,R0 ;INPUT
CLR PSW ;LOWER PSW
;LET INTERRUPT OCCUR
NOP
NOP
NOP
BIC #BIT14,@GRSTAT
NOP
ERROR 5 ;ERROR, OUTPUT FAILED TO INTERRUPT
BR 3$ ;;
CMP (SP)+,(SP)+
2$:
3$:
MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
MOV #340,@GRIVSA
MOV #UNEXPT,@GRIVB ;RESET OUTPUT VECTOR
MOV #340,@GRIVSB
CLR PSW
CLR @GRSTAT ;CLEAR STATUS

*****
*TEST 47 PRE-INTERRUPT SETUP
*****
TST47: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
RESET
BIC #177437,DIOBRL
BNE 1$
ERROR 7 ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 0
CMP #340,DIOBRL
BNE 2$
ERROR 7 ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 7
2$:
MOV DIOBRL,BRLEV1
SUB #40,BRLEV1
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) †ST50: 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$: ;BIS #BITS,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
1352 010554 052777 000040 170552 ;NOP
1353 010562 000240 ;NOP
1354 010564 000240 ;NOP
1355 010566 000240 ;NOP
1356 010570 000240 ;NOP
1357 010572 104004 ;ERROR 4 ;ERROR, NO INTERRUPT FROM DIGITAL I/O INPUT
1358 010574 000401 ;BR 3$ ;:BR TO NEXT TEST
1359 010576 022626 1$: ;CMP (SP)+,(SP)+
1360 010600 012777 002746 170542 3$: ;MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
1361 010606 012777 000340 170536 ;MOV #340,@GRIVSA ;RESET OUTPUT VECTOR
1362 010614 012777 002746 170532 ;MOV #UNEXPT,@GRIVB
1363 010622 012777 000340 170526 ;MOV #340,@GRIVSB
1364 010630 005037 177776 ;CLR PSW ;LOWER PSW
1358 (3) ;*****
1359 (3) ;*TEST 51 TEST FOR NO INTR. FROM DRA INPUT ON LEVEL INDICATED VIA MAINT. INT
1360 (2) ;*****
1361 (2) †ST51: 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$: ;BIS #BITS,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
1369 010672 052777 000040 170434 ;NOP
1370 010700 000240 ;NOP
1371 010702 000240 ;NOP
1372 010704 000240 ;NOP
1373 010706 012777 010742 170434 ;SUB-TEST, NOW LOWER THE PSW AND ALLOW THE INTERRUPT
1374 010714 005037 177776 ;MOV #2$,@GRIVA ;RESET VECTOR
1375 010720 000240 ;CLR PSW ;LOWER PSW
1376 010722 000240 ;NOP
1377 010724 000240 ;NOP
1378 010726 000403 ;BR 3$ ;ERROR
1379 010730 022626 1$: ;CMP (SP)+,(SP)+
1380 010732 005037 177776 ;CLR PSW
1381 010736 104006 3$: ;ERROR 6 ;UNEXPECTED INTERRUPT (IS BR LEVEL PLUG CORRECT)
1382 010740 000401 ;BR 5$ ;
1383 010742 022626 2$: ;CMP (SP)+,(SP)+ ;POP THE STACK
1384 010744 000240 5$: ;MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
1385 010752 012777 000340 170372 ;MOV #340,@GRIVSA ;RESET OUTPUT VECTOR
1386 010760 012777 002746 170366 ;MOV #UNEXPT,@GRIVB

```

```

(1) 010766 012777 000340 170362
1382 C10774 005037 177776
1383
1384
(2)
(3)
(3)
(2) 011000 000004
(1) 011002 012737 000001 001166
1385 011010 005737 001320
1386 011014 001415
1387 011016 032777 010000 170114
1388 011024 001024
1389 011026 162737 000010 001322
1390 011034 062737 000010 001324
1391 011042 005337 001320
1392 011046 000413
1393 011050 013737 001306 001322
1394 011056 013737 001310 001324
1395 011064 013737 001316 001320
1396 011072 000137 011106
1397 011076 012700 177777
1398 011102 000137 002776
1399
1400
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 011106
(1) 011106 000004
(1) 011110 005037 001102
(1) 011114 005037 001166
(1) 011120 005237 001100
(1) 011124 042737 100000 001100
(1) 011132 005327
(1) 011134 000001
(1) 011136 003022
(1) 011140 012737
(1) 011142 000001
(1) 011144 011134
(1) 011146 104401 011213
(2) 011152 013746 001100
(2) 011156 104405
(1) 011160 104401 011210
(1) 011164 013700 000042
(1) 011170 001405
(1) 011172 000005
(1) 011174 004710
(1) 011176 000240
(1) 011200 000240

```

```

MOV #340, @GRIVSB
CLR PSW
;*****
;*TEST 52 DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
;*****
↑ST52: SCOPE
MOV #1, $TIMES ;;DO 1 ITERATION
BYPASS: TST NBEXT ;TEST IF ANY
BEQ 1$ ;BR IF NONE
BIT #SW12, @SWR ;TEST BIT12 OF SWR
BNE BYPAS1 ;INHIBIT TESTING NEXT DR11-K
SUB #10, DRADD ;UPDATE DEVICE ADDRESS
ADD #10, DRIV ;UPDATE DEVICE VECTOR
DEC NBEXT ;ANOTHER ONE ?
BR BYPAS1 ;BR IF ANOTHER
1$: MOV BASEBA, DRADD ;RELOAD ADDRESS
MOV BASEIV, DRIV ;RELOAD VECTOR
MOV NMBEXT, NBEXT ;RELOAD NUMBER
JMP $EOP ;DONE
BYPAS1: MOV #-1, RO
JMP RBEG2 ;TEST ANOTHER UNIT

.SBTTL END OF PASS ROUTINE

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

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

```

```

(1) 011202 000240
(1) 011204
(1) 011204 000137
(1) 011206 011076
(1) 011210 377 377 000
(1) 011213 015 042412 042116
(1) 011220 050040 051501 020123
(1) 011226 000043
1401
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2) 011230
(3) 011230 010046
(3) 011232 010146
(3) 011234 010246
(3) 011236 010346
(3) 011240 010546
(1) 011242 012746 020200
(1) 011246 016605 000020
(1) 011252 100004
(1) 011254 005405
(1) 011256 112766 000055 000001
(1) 011264 005000
(1) 011266 012703 011444
(1) 011272 112723 000040
(1) 011276 005002
(1) 011300 016001 011434
(1) 011304 160105
(1) 011306 002402
(1) 011310 005202
(1) 011312 000774
(1) 011314 060105
(1) 011316 005702
(1) 011320 001002
(1) 011322 105716
(1) 011324 100407
(1) 011326 106316
(1) 011330 103003
(1) 011332 116663 000001 177777
(1) 011340 052702 000060
(1) 011344 052702 000040
(1) 011350 110223
(1) 011352 005720
(1) 011354 020027 000010
(1) 011360 002746

```

```

NOP ::ACT11
SDOAGN:
JMP 2(PC)+ ::RETURN
SRTNAD: .WORD BYPASS
SENULL: .BYTE -1,-1,0 ::NULL CHARACTER STRING
SENDMG: .ASCIZ <15><12>/END PASS */

.SBT'L CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

:*****
:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
:SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
:NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
:BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
:REPLACED WITH SPACES.
:CALL:
:* MOV NUM,-(SP) ::PLT THE BINARY NUMBER ON THE STACK
:* TYPDS ::GO TO THE ROUTINE

$TYPDS:
MOV R0,-(SP) ::PUSH R0 ON STACK
MOV R1,-(SP) ::PLSH R1 ON STACK
MOV R2,-(SP) ::PUSH R2 ON STACK
MOV R3,-(SP) ::PUSH R3 ON STACK
MOV R5,-(SP) ::PUSH R5 ON STACK
MOV #20200,-(SP) ::SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 ::GET THE INPUT NUMBER
BPL '$ ::BR IF INPUT IS POS.
NEG '$ ::MAKE THE BINARY NUMBER POS.
MOVB #'-,1(SP) ::MAKE THE ASCII NUMBER NEG.
1$: CLR R0 ::ZERO THE CONSTANTS INDEX
MOV #SDBLK,R3 ::SETUP THE OUTPUT POINTER
MOVB #'',(R3)+ ::SET THE FIRST CHARACTER TO A BLANK
2$: CLR R2 ::CLEAR THE BCD NUMBER
MOV $D*BL(R0),R1 ::GET THE CONSTANT
3$: SLB R1,R5 ::FORM THIS BCD DIGIT
BLT 4$ ::BR IF DONE
INC R2 ::INCREASE THE BCD DIGIT BY 1
BR 3$
4$: ADD R1,R5 ::ADD BACK THE CONSTANT
TST R2 ::CHECK IF BCD DIGIT=C
BNE 5$ ::FALL THROUGH IF 0
*STB (SP) ::STILL DOING LEADING 0'S?
BMI 7$ ::BR IF YES
5$: ASLB (SP) ::MSD?
BCC 6$ ::BR IF NO
MOVB 1(SP),-1(R3) ::YES--SET THE SIGN
6$: BIS #'0,R2 ::MAKE THE BCD DIGIT ASCII
7$: BIS #'',(R2) ::MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOV R2,(R3)+ ::PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST (R0)+ ::JUST INCREMENTING
CMP R0,#10 ::CHECK THE TABLE INDEX
BLT 2$ ::GO DO THE NEXT DIGIT

```

```

(1) 011362 003002          BGT      8$          ;; GO TO EXIT
(1) 011364 010502          MOV      R5,R2      ;; GET THE LSD
(1) 011366 000764          BR       6$          ;; GO CHANGE TO ASCII
(1) 011370 105726          3$: TSTB      (SP)+   ;; WAS THE LSD THE FIRST NON-ZERO?
(1) 011372 100003          BPL      9$          ;; BR IF NO
(1) 011374 116663 177777 17777E  MOVB     -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
(1) 011402 105013          9$: CLRB      (R3)    ;; SET THE TERMINATOR
(3) 011404 012605          MOV      (SP)+,R5   ;; POP STACK INTO R5
(3) 011406 012603          MOV      (SP)+,R3   ;; POP STACK INTO R3
(3) 011410 012602          MOV      (SP)+,R2   ;; POP STACK INTO R2
(3) 011412 012601          MOV      (SP)+,R1   ;; POP STACK INTO R1
(3) 011414 012600          MOV      (SP)+,R0   ;; POP STACK INTO R0
(1) 011416 104401 011444    TYPE     $DBLK      ;; NOW TYPE THE NUMBER
(1) 011422 016666 000002 000004  MOV      2(SP),4(SP) ;; ADJUST THE STACK
(1) 011430 012616          MOV      (SP)+,(SP)
(1) 011432 000002          RTI                    ;; RETURN TO USER
(1) 011434 023420          $DTBL: 10000.
(1) 011436 001750          1000.
(1) 011440 000144          100.
(1) 011442 000012          10.
(1) 011444 000004          $DBLK: .BLKW      4

1402
1403          .SBTTL  MISC. EXTERNAL LOGIC TEST
1404 011454 012706 001100    EXTTST: MOV      #STACK,SP
1405 011460 004737 003036    JSR      PC,SETADD  ;; SET UP ADDRESS AND VECTOR
1406 011464 012737 040000 011560  2$: MOV      #BIT14,EXTCNT ;; LOAD COUNT
1407 011472          1$:
(1) 011472 005077 167642          CLR      @GRDIO     ;; CLEAR OUTPUT REGISTER
1408 011476 012777 125252 167634  MOV      #125252,@GRDIO ;; LOAD OUTPUT
1409 011504 017737 167626 011556  MOV      @GRDAI,EXTTMP ;; READ INPUT
1410 011512 013777 011556 167 16  MOV      EXTTMP,@GRDAI ;; CLEAR INPUT
1411 011520 005077 167614          CLR      @GRDIO     ;; CLEAR OUTPUT REGISTER
1412 011524 012777 052525 167606  MOV      #52525,@GRDIO ;; LOAD OUTPUT
1413 011532 017737 167600 011556  MOV      @GRDAI,EXTTMP ;; READ INPUT
1414 011540 013777 011556 167570  MOV      EXTTMP,@GRDAI ;; CLEAR INPUT
1415 011546 005337 011560    DEC      EXTCNT     ;; FINISHED COUNT
1416 011552 001347          BNE     1$
1417 011554 000743          BR      2$          ;; LOCP
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
1426 011604 012737 000340 000026  MOV      #340,@PWRVEC+2 ;; LOAD POWER FAIL VECTOR
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 PC
1434 011634 012777 011676 167506 MOV #20$,@GRIVA ;LOAD INPUT VECTOR
1435 011642 012777 000340 167502 MOV #347,@GRIVSA
1436 011650 104401 013302 1$: 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,RO ;LOAD 1 MSEC. DELAY WEIGHT ;240 FOR 11/05 ;620 FOR 11/40
1445 011712 005300 3$: DEC RO ;DELAY
1446 011714 001376 BNE 3$
1447 011716 005337 011556 DEC EXTTMP ;FINISHED ALL MSEC. DELAY ?
1448 011722 001371 BNE 6$ ;BR IF NOT
1449 011724 017700 167406 MOV @GRDAI,RO ;SAVE RESULTS
1450 011730 010021 MOV RC,(R1)+ ;SAVE IN BUFFER
1451 011732 012777 177777 167376 MOV #-1,@GRDAI ;CLEAR INPUT
1452 011740 042700 0077 BIC #7777,RO ;MASK
1453 011744 001345 BNE 4$ ;BR IF NOT LAST SAMPLE
1454 011746 010103 MOV R1,R3 ;COPY R1
1455 011750 014300 5$: MOV -(R3),RO ;GET RESULTS
1456 011752 004737 011766 JSR PC,40$ ;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$: ROR RO
1466 012012 006000 ROR RO
1467 012014 006000 ROR RO
1468 012016 006000 ROR RO
1469 012020 010001 12$: MOV RO,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 LOCBOX: MOV #STACK,SP
1481 JSR PC,SETTRP
1482 TYPE ,LOCHDR ;TELL OPERATOR WHAT THIS IS
1483 JSR R5,INADR ;GET THE ADDRESSES
1484 LOC1,LOC1Y
1485 BR LOCA ;BR IF DONE
1486 JSR R5,INADR ;GET NEXT ADDRESS
1487 LOC2,LOC2Y
1488 BR LOCA ;BR IF DONE
1489 JSR R5,INADR ;GET NEXT ADDRESS
1490 LOC3,LOC3Y
1491 BR LOCA
1492
1493 LOCA: JSR R5,LOUP1 ;TEST FOR SWITCHES AND LOAD LAMPS
1494 LOC1
1495 JSR R5,LOUP1 ;LOAD NEXT SET
1496 LOC2
1497 JSR R5,LOUP1 ;LOAD NEXT SET
1498 LOC3
1499
1500 BR LOCA ;LOOP BACK
1501
1502 INADR: MOV @R5+,10$ ;GET BUS ADDRESS
1503 MOV @R5+,12$ ;GET INDICATOR
1504 1$: TYPE, INADRH ;ASK FOR INPUT
1505 MOV 10$,-(SP) ;TYPE CURRENT ADDRESS
1506 TYPOC
1507 TYPE, INADR ;ADD YES NO
1508 RDLIN ;READ CHAR. AND ECHC
1509 NOP
1510 NOP
1511 MOV @SP+,11$ ;GET THE ANSWER
1512 BIC #177640,11$ ;MASK OFF BITS
1513 BEQ 3$ ;BR IF CR
1514 CMP #'N,11$ ;TEST IF NO
1515 BEQ 2$ ;BR IF NO
1516 CMP #'Y,11$ ;TEST IF YES
1517 BNE 1$ ;BR IF OTHER
1518 INC @12$ ;SET YES FLAG
1519 2$: TST R5+
1520 3$: RTS ;EXIT
1521 10$: 0
1522 11$: 0
1523 12$: 0
    
```

```

1525
1526 ;SUBROUTINE TO LOAD THE LAMPS FROM THE LOC BOX SWITCHES
1527 LOUP1: MOV 2(R5)+,405 ;GET ARG.
1528 BNE 15 ;BR IF YES
1529 RTS R5 ;EXIT
1530 15: MOV 405,R2
1531 ADD #2,R2
1532 MOV R2,R3
1533 ADD #2,R3
1534 MOV (R2),R0
1535 MOV (R3),R1
1536 BIC R1,R0 ;MASK OFF BITS
1537 BIC (R2),(R3) ;LOAD OUTPUT BITS
1538 ADD R0,(R3) ;LOAD
1539 MOV #-1,(R2) ;CLEAR INPUT
1540 RTS R5 ;EXIT
1541
1542 405: 0
1543 415: 0
1544 LOC1Y: 0 ;YES/NO ANSWER TO LOC BOX #1
1545 LOC2Y: 0
1546 LOC3Y: 0
1547
1548 .SBTTL XLO1 ADJUSTMENT ROUTINE
1549 XLO1AD: MOV #STACK,SP
1550 JSR PC,SETTRP
1551 TYPE XLOHDR ;TELL OPERATOR
1552 JSR R5,INADR ;GET BUS ADDRESS
1553 LOC1,LOC1Y
1554 BR XLO10
1555 JSR R5,INADR ;
1556 LOC2,LOC2Y
1557 BR XLO10
1558 JSR R5,INADR
1559 LOC3,LOC3Y
1560 BR XLO10
1561
1562 XLO10:
1563 (1) 012412 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
1564 (1) 012416 000 000 .BYTE 0,0 ;LOAD CH# AND SHIFT COUNT
1565 (1) 012420 000007 7 ;LOAD DATA MASK
1566 (1) 012422 001326 LOC1 ;DR11K BUS ADDRESS
1567 1569 012424 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
1568 (1) 012430 001 003 .BYTE 1,3 ;LOAD CH# AND SHIFT COUNT
1569 (1) 012432 000070 70 ;LOAD DATA MASK
1570 (1) 012434 001326 LOC1 ;DR11K BUS ADDRESS
1571 1570 012436 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
1572 (1) 012442 002 006 .BYTE 2,6 ;LOAD CH# AND SHIFT COUNT
1573 (1) 012444 000700 700 ;LOAD DATA MASK
1574 (1) 012446 001326 LOC1 ;DR11K BUS ADDRESS
1575 1571 012450 004537 012644 JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
1576 (1) 012454 003 011 .BYTE 3,11 ;LOAD CH# AND SHIFT COUNT
1577 (1) 012456 007000 7000 ;LOAD DATA MASK
1578 (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 XLC10
1584
1585 ;ANALOG CONVERSION AND LOC BOX DRIVER
1586 012644 112577 000206 XLOADJ: MOVB (R5)+,2ADCS1 ;LOAD CHANNEL #
1587 012650 052777 020000 000176 BIS #BIT13,2ADCS ;SET UNIPOLAR
1588 012656 112537 013046 MOVB (R5)+,40$ ;GET SHIFT COUNT
1589 012662 012537 013052 MOV (R5)+,42$ ;GET DATA MASK
1590 012666 013537 013050 MOV 2(R5)+,41$ ;GET DR11K BUS ADDRESS
1591 012672 001464 BEQ 70$ ;BR IF NONE
1592 012674 105277 00C154 INCB 2ADCS ;CONVERT THE DATA

```

1593	012700	005001			CLR	R1	
1594	012702	062737	000004	013050	ADD	#4, R1	:GET LAMP ADDRESS
1595	012710	105777	000140		TSTB	ADCS	:WAIT FOR DONE
1596	012714	100375			BPL	15	
1597	012716	017703	000136		MOV	ADBR, R0	:READ THE DATA
1598	012722	162700	000004		SUB	#4, R0	:TEST IF AT UPPER END
1599	012726	100432			BMI	75	:BR IF YES
1600	012730	162700	000014		SUB	#14, R0	:TEST IF NEAR UPPER END
1601	012734	100425			BMI	65	:BR IF YES
1602	012736	162700	000060		SUB	#60, R0	:TEST IF GETTING NEAR UPPER END
1603	012742	100420			BMI	55	:BR IF YES
1604	012744	162700	001530		SUB	#1530, R0	:TEST IF GETTING NEAR LOWER END
1605	012750	100407			BMI	25	:BR IF YES
1606	012752	162700	000060		SUB	#60, R0	:TEST IF NEAR LOWER END
1607	012756	100406			BMI	35	:BR IF YES
1608	012760	162700	000014		SUB	#14, R1	:TEST IF AT LOWER END
1609	012764	100405			BMI	45	:BR IF YES
1610	012766	000412			BR	75	:BR IF AT STOP
1611	012770	012701	000004	25:	MOV	#4, R1	:LOAD #
1612	012774	062701	000002	35:	ADD	#2, R1	
1613	013000	005201		45:	INC	R1	
1614	013002	000404			BR	75	:BR AND LOAD LAMP'S
1615	013004	012701	000002	55:	MOV	#2, R1	:LOAD #
1616	013010	062701	000004	65:	ADD	#4, R1	
1617	013014	013700	013046	75:	MOV	40\$, R0	:LOAD SHIFT COUNT
1618	013020	006301		105:	ASL	R1	:SHUFFEL THE DATA
1619	013022	005300			DEC	R0	
1620	013024	100375			BPL	105	:BR IF NOT DONE
1621	013026	006201			ASR	R1	:RE ADJUST
1622	013030	043777	013052	000012	BIC	42\$, R1	:CLEAR OLD DATA IN LAMPS
1623	013036	050177	000006		BIS	R1, #415	:LOAD THE LAMPS
1624	013042	005001			CLR	R1	
1625	013044	000205		705:	RTS	R5	:EXIT
1626							
1627	013046	000000		405:	0		
1628	013050	000000		415:	0		
1629	013052	000000		425:	0		

1631	013054	170400			ADCS:	170400	
1632	013056	170401			ADCS1:	170401	
1633	013060	170402			ADDR:	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			
	013166	037040	020040	020131	INDADR:	.ASCIZ	/ > Y OR N ? /
	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	013242	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	SWINTE:	.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							

1673
1674

.SBTTL SCOPE HANDLER ROUTINE

```

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

```

```

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

```

```

(1) 015222 013737 015272 001166      MOV      $MXCNT,$TIMES      ;; SET NUMBER OF ITERATIONS TO DO
(1) 015230 105237 001102      $SVLAD: INCB      $TSTNM      ;; COUNT TEST NUMBERS
(1) 015234 011637 001106      MOV      (SP),$LPADR      ;; SAVE SCOPE LOOP ADDRESS
(1) 015240 011637 001110      MOV      (SP),$LPERR      ;; SAVE ERROR LOOP ADDRESS
(1) 015244 005037 001170      CLR      $ESCAPE      ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 015250 112737 000001 001115      MOVVB   #1,$ERRMAX      ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 015256 013777 001102 163656      $OVER:  MOV      $TSTNM,$DISPLAY ;; DISPLAY TEST NUMBER
(1) 015264 013716 001106      MOV      $LPADR,(SP)      ;; FUDGE RETURN ADDRESS
(1) 015270 000002      RTI      ;; FIXES PS
(1) 015272 003720      $MXCNT: 2000.      ;; MAX. NUMBER OF ITERATIONS
1675 .SBTTL ERROR HANDLER ROUTINE

::*****
::*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
::*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
::*AND GO TO $ERRTYP ON ERROR
::*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
::*SW15=1      HALT ON ERROR
::*SW13=1      INHIBIT ERROR TYPEOUTS
::*SW09=1      LOOP ON ERROR
::*CALL
::*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER

(1) 015274      $ERROR:
(1) 015274 104407      CKSWR      ;; TEST FOR CHANGE IN SOFT-SWR
(2) 015276 113737 001102 015020      MOVVB   $TSTNM,$TSTNUM
(1) 015304 105237 001103      7$:      INCB      $ERFLG      ;; SET THE ERROR FLAG
(1) 015310 001775      BEQ      7$      ;; DON'T LET THE FLAG GO TO ZERO
(1) 015312 013777 001102 163622      MOV      $TSTNM,$DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
(1) 015320 005237 001112      INC      $ERTTL      ;; INC THE ERROR COUNT
(1) 015324 011637 001116      MOV      (SP),$ERRPC      ;; GET ADDRESS OF ERROR INSTRUCTION
(1) 015330 162737 000002 001116      SUB      #2,$ERRPC
(1) 015336 117737 163554 001114      MOVVB   #2,$ERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
(1) 015344 032777 020000 163566      BIT      #BIT13,$SWR      ;; SKIP TYPEOUT IF SET
(1) 015352 001004      BNE      20$      ;; SKIP TYPEOUTS
(1) 015354 004737 015440      JSR      PC,$ERRTYP      ;; GO TO USER ERROR ROUTINE
(1) 015360 104401 001173      TYPE      $CRLF

(1) 015364      20$:
(1) 015364 005777 163550      2$:      TST      @SWR      ;; HALT ON ERROR
(1) 015370 100002      BPL      3$      ;; SKIP IF CONTINUE
(1) 015372 000000      HALT
(1) 015374 104407      CKSWR      ;; TEST FOR CHANGE IN SOFT-SWR
(1) 015376 032777 001000 163534      3$:      BIT      #BIT09,@SWR      ;; LOOP ON ERROR SWITCH SET?
(1) 015404 001402      BEQ      4$      ;; BR IF NO
(1) 015406 013716 001110      MOV      $LPERR,(SP)      ;; FUDGE RETURN FOR LOOPING
(1) 015412 005737 001170      4$:      TST      $ESCAPE      ;; CHECK FOR AN ESCAPE ADDRESS
(1) 015416 001402      BEQ      5$      ;; BR IF NONE
(1) 015420 013716 001170      MOV      $ESCAPE,(SP)      ;; FUDGE RETURN ADDRESS FOR ESCAPE
(1) 015424      5$:
(1) 015424 022737 011174 000042      CMP      #SENDAD,@#42      ;; ACT-11 AUTO-ACCEPT?
(1) 015432 001001      BNE      6$      ;; BRANCH IF NO
(1) 015434 000000      HALT      ;; YES
(1) 015436      6$:
(1) 015436 000002      RTI      ;; RETURN

```


168C
1681

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
* MOV NUM,-(SP) ::NUMBER TO BE TYPED
* TYPOS ::CALL FOR TYPEOUT
* .BYTE N ::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
* .BYTE M ::M=1 OR 0
* ::1=TYPE LEADING ZEROS
* ::0=SUPPRESS LEADING ZEROS

*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC

*CALL:
* MOV NUM,-(SP) ::NUMBER TO BE TYPED
* TYPON ::CALL FOR TYPEOUT

*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

*CALL:
* MOV NUM,-(SP) ::NUMBER TO BE TYPED
* TYPOC ::CALL FOR TYPEOUT

(1)	015574	017646	000000		STYPOS:	MOV	2,SP),-(SP,	::PICKUP THE MODE
(1)	015600	116637	000001	016017		MOV	1(SP),%OFILL	::LOAD ZERO FILL SWITCH
(1)	015606	112637	016021			MOV	(SP)+,%OMODE+1	::NUMBER OF DIGITS TO TYPE
(1)	015612	062716	000002			ADD	#2,(SP)	::ADJUST RETURN ADDRESS
(1)	015616	000406				BR	STYPON	
(1)	015620	112737	000001	016017	STYPOC:	MOV	#1,%OFILL	::SET THE ZERO FILL SWITCH
(1)	015626	112737	000006	016021		MOV	#6,%OMODE+1	::SET FOR SIX(6) DIGITS
(1)	015634	112737	000005	016016	STYPON:	MOV	#5,%OCNT	::SET THE ITERATION CCUNT
(1)	015642	010346				MOV	R3,-(SP)	::SAVE R3
(1)	015644	010446				MOV	R4,-(SP)	::SAVE R4
(1)	015646	010546				MOV	R5,-(SP)	::SAVE R5
(1)	015650	113704	016021			MOV	%OMODE+1,R4	::SET THE NUMBER OF DIGITS TO TYPE
(1)	015654	005404				NEG	R4	
(1)	015656	062704	000006			ADD	#6,R4	::SUBTRACT IT FOR MAX. ALLOWED
(1)	015662	110437	016020			MOV	R4,%OMODE	::SAVE IT FOR USE
(1)	015656	113704	016017			MOV	%OFILL,R4	::GET THE ZERO FILL SWITCH
(1)	015672	016605	000012			MOV	12(SP),R5	::PICKUP THE INPUT NUMBER
(1)	015676	005003				CLR	R3	::CLEAR THE OUTPUT WORD
(1)	015700	006105			1\$:	ROL	R5	::ROTATE MSB INTO "C"
(1)	015702	000404				BR	3\$::GO DO MSB
(1)	015704	006105			2\$:	ROL	R5	::FORM THIS DIGIT
(1)	015706	006105				ROL	R5	
(1)	015710	006105				ROL	R5	
(1)	015712	010503				MOV	R5,R3	
(1)	015714	006103			3\$:	ROL	R3	::GET LSB OF THIS DIGIT
(1)	015716	105337	016020			DECB	%OMODE	::TYPE THIS DIGIT
(1)	015722	100016				BPL	7\$::BR IF NO
(1)	015724	042703	177770			BIC	#177770,R3	::GET RID OF JUNK

(1)	015730	001002		BNE	4\$:: TEST FOR 0
(1)	015732	005704		TST	R4	:: SUPPRESS THIS 0'
(1)	015734	001403		BEQ	5\$:: BR IF YES
(1)	015736	005204		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	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		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		.BYTE	0	:: STORAGE FOR ASCII DIGIT
(1)	016015	000		.BYTE	00	:: TERMINATOR FOR TYPE ROUTINE
(1)	016016	000		.BYTE	000	:: OCTAL DIGIT COUNTER
(1)	016017	000		.BYTE	000	:: ZERO FILL SWITCH
(1)	016020	000000		.WORD	0	:: NUMBER OF DIGITS TO TYPE

1683
1684

.SBTTL POWER DOWN AND UP ROUTINES

::*****

:POWER DOWN ROUTINE

(1)	016022	012737	016166	000024	\$PWRDN: MOV	#\$ILLUP, 2#PWRVEC	::SET FOR FAST UP
(1)	016030	012737	000340	000026	MOV	#340, 2#PWRVEC+2	::PRIO:7
(3)	016036	010046			MOV	R0, -(SP)	::PUSH R0 ON STACK
(3)	016040	010146			MOV	R1, -(SP)	::PUSH R1 ON STACK
(3)	016042	010246			MOV	R2, -(SP)	::PUSH R2 ON STACK
(3)	016044	010346			MOV	R3, -(SP)	::PUSH R3 ON STACK
(3)	016046	010446			MOV	R4, -(SP)	::PUSH R4 ON STACK
(3)	016050	010546			MOV	R5, -(SP)	::PUSH R5 ON STACK
(3)	016052	017746	163062		MOV	2SWR, -(SP)	::PUSH 2SWR ON STACK
(1)	016056	010637	016172		MOV	SP, \$SAVR6	::SAVE SP
(1)	016062	012737	016074	000024	MOV	#\$PWRUP, 2#PWRVEC	::SET UP VECTOR
(1)	016070	000000			HALT		
(1)	016072	000776			BR	.-2	::HANG UP

::*****

:POWER UP ROUTINE

(1)	016074	012737	016166	000024	\$PWRUP: MOV	#\$ILLUP, 2#PWRVEC	::SET FOR FAST DOWN
(1)	016102	013706	016172		MOV	\$SAVR6, SP	::GET SP
(1)	016106	005037	016172		CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
(1)	016112	005237	016172		1\$: INC	\$SAVR6	::WAIT FOR THE INC
(1)	016116	001375			BNE	1\$::OF WORD
(3)	016120	012677	163014		MOV	(SP)+, 2SWR	::POP STACK INTO 2SWR
(3)	016124	012605			MOV	(SP)+, R5	::POP STACK INTO R5
(3)	016126	012604			MOV	(SP)+, R4	::POP STACK INTO R4
(3)	016130	012603			MOV	(SP)+, R3	::POP STACK INTO R3
(3)	016132	012602			MOV	(SP)+, R2	::POP STACK INTO R2
(3)	016134	012601			MOV	(SP)+, R1	::POP STACK INTO R1
(3)	016136	012600			MOV	(SP)+, R0	::POP STACK INTO R0
(1)	016140	012737	016022	000024	MOV	#\$PWRDN, 2#PWRVEC	::SET UP THE POWER DOWN VECTOR
(1)	016146	012737	000340	000026	MOV	#340, 2#PWRVEC+2	::PRIO:7
(1)	016154	104401			TYPE		::REPORT THE POWER FAILURE
(1)	016156	016174			\$PWRMG: .WORD	PWRMSG	::POWER FAIL MESSAGE POINTER
(1)	016160	012716			MOV	(PC)+, (SP)	::RESTART AT IOTEST
(1)	016162	002752			\$PWRAD: .WORD	IOTEST	::RESTART ADDRESS
(1)	016164	000002			RTI		
(1)	016166	000000			\$ILLUP: HALT		::THE POWER UP SEQUENCE WAS STARTED
(1)	016170	000776			BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
(1)	016172	000000			\$SAVR6: 0		::PUT THE SP HERE
1685	016174	005015	042522	052123	PWRMSG: .ASCIZ	<15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>	
	016202	051101	044524	043516			
	016210	040440	052106	051105			
	016216	040440	050040	053517			
	016224	051105	043040	044501			
	016232	052514	042522	005015			
	016240	030					

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) 016242 105737 001157 $TYPE: TSTB $TPFLG ;: IS THERE A TERMINAL?
(1) 016246 100002 BPL 1$ ;: BR IF YES
(1) 016250 000000 HALT ;: HALT HERE IF NO TERMINAL
(1) 016252 000407 BR 3$ ;: LEAVE
(1) 016254 010046 1$: MOV RO,-(SP) ;: SAVE RO
(1) 016256 C17600 000002 MOV #2(SP),RO ;: GET ADDRESS OF ASCIZ STRING
(1) 016262 112046 2$: MOVB (RO)+,-(SP) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 016264 001005 BNE 4$ ;: BR IF IT ISN'T THE TERMINATOR
(1) 016266 005726 TST (SP)+ ;: IF TERMINATOR POP IT OFF THE STACK
(1) 016270 012600 60$: MOV (SP)+,RO ;: RESTORE RO
(1) 016272 062716 000002 3$: ADD #2,(SP) ;: ADJUST RETURN PC
(1) 016276 000302 RTI ;: RETURN
(1) 016300 122716 000011 4$: CMPB #HT,(SP) ;: BRANCH IF <HT>
(1) 016304 001430 BEQ 8$ ;:
(1) 016306 122716 000200 CMPB #CRLF,(SP) ;: BRANCH IF NOT <CRLF>
(1) 016312 001006 BNE 5$ ;:
(1) 016314 005726 TST (SP)+ ;: POP <CR><LF> EQUIV
(1) 016316 104401 TYPE ;: TYPE A CR AND LF
(1) 016320 001173 $CRLF ;:
(1) 016322 105037 016456 CLRB $CHARCNT ;: CLEAR CHARACTER COUNT
(1) 016326 000755 BR 2$ ;: GET NEXT CHARACTER
(1) 016330 004737 016412 5$: JSR PC,$TYPEC ;: GO TYPE THIS CHARACTER
(1) 016334 123726 001156 6$: CMPB $FILLC,(SP)+ ;: IS IT TIME FOR FILLER CHARS.?
(1) 016340 001350 BNE 2$ ;: IF NO GO GET NEXT CHAR.
(1) 016342 013746 001154 MOV $NULL,-(SP) ;: GET # OF FILLER CHARS. NEEDED
(1) ;: AND THE NULL CHAR.
(1) 016346 105366 000001 7$: 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) C16354 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) 016366 112716 000040 8$: MOVB #' ,(SP) ;: REPLACE TAB WITH SPACE
(1) 016372 004737 016412 9$: JSR PC,$TYPEC ;: TYPE A SPACE

```

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


```

(1) 016612 022726 000007      CMP      #7,(SP)+      ;; IS IT A CONTROL G?
(1) 016616 001062              BNE      15$          ;; NO RETURN TO JSER
(1) 016620 123727 001134 000001  CMPB     $AUTOB,#1    ;; ARE WE RUNNING IN ALTO-MCDE?
(1) 016626 001456              BEQ      15$          ;; BRANCH IF YES
(1)
(1) 016630 104401 017436      TYPE     , $CNTLG     ;; ECHO THE CONTROL-G (↑G)
(1) 016634 104401 017443      $GTSWR: TYPE     , $1SWR   ;; TYPE CURRENT CONTENTS
(2) 016640 013746 000176      MOV      SWREG,-(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$:    TSTB     2$TKS   ;; CHAR THERE?
(1) 016662 100375              BPL      7$          ;; IF NOT TRY AGAIN
(1)
(1) 016664 117746 162256      MOVB     2$TKB,-(SP)  ;; PICK UP CHAR
(1) 016670 042716 177600      BIC      #1C177,(SP) ;; MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1) 016674 021627 000025      9$:    CMP      (SP),#25  ;; IS IT A CONTROL-?
(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),2$SWR  ;; SAVE NEW SWR
(1) 016736 062706 000006      11$:   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      #100,2$TKS  ;; RE-ENABLE TTY KBD INTERRUPTS
(1) 016764 000002      15$:   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$:   INC      2(SP)     ;; KEEP COUNT OF CHAR
(1) 017032 056616 177776      BIS      -2(SP),(SP) ;; SET IN NEXT CHAR
(1) 017036 000707              BR       7$         ;; 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              BLC      4$              ;; BR IF YES
(1) 017250 111337 017420      MOVVB   (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 MOVVB   #' \,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$: MOVVB   (R3),9$         ;; 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      $MNEW: .ASCIZ  / NEW = /
(1) 017454 020040 042516 020127
(1) 017462 020075 000
(1) 017466 .EVEN
1693

```


SW6	=	000100	390#						
SW7	=	000200	390#						
SW8	=	000400	390#						
SW9	=	001000	390#						
ALK	=	002866	568	571	574	577	594	627#	
TRIVE	=	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	584#						
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	1094#					
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#						

TYPDS = 104405	1400	1696*												
TYPE = 104401	531	548	549	553	612	615	629	633	637	1400	1401	1436	147E	
	1482	1504	1507	1551	1675	1678	1681	1684	1689	1692	1696*			
TYPOC = 104402	614	1506	1678	1692	1696*									
TYPON = 104404	1696*													
TYPOS = 104403	551	1696*												
UNEXPT 002746	645*	683	1233	1325	1356	1381								
VECTOP 002442	540	600*												
XLOADJ 012644	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	
	1581	1582	1586*											
XLOHDR 013204	1551	1637*												
XLO1AD 012340	401	1549*												
XLO10 012412	1554	1557	1560	1568*	1583									
\$ALTOB 001134	402*	1692												
\$BDADR 001122	402*													
\$BDDAT 001126	402*	692*	693	701*	708*	709	716*	717	725*	726	738*	739	751*	
	752*	753	766*	767	777*	778	787*	788	799*	800	911*	812	820*	
	821	828*	829	837*	838	848*	869*	870*	871	881*	882*	883	894*	
	895*	896	906*	907*	908	923*	924	933*	934	955*	956*	957	965*	
	966*	968	990*	991	1016*	1017	1042*	1046	1067*	1071	1081*	1082*	1091*	
	1092	1104*	1174*	1175	1242*	1266*	1279*	1667	1669					
\$CHARC 016456	1689**													
\$CKSWR 016564	1692*	1696												
\$CMTAG 001100	402*	515	658											
\$CM1 = 000002	402*													
\$CM2 = 000004	402*													
\$CM3 = 000002	402*													
\$CNTLG 017436	1692*													
\$CNTLU 017431	1692*													
\$CRLF 001173	402*	1675	1678	1689	1692									
\$DBLK 011444	1401*													
\$DCAGN 011204	1400*													
\$DTBL 011434	1401*													
\$ENDAD 011174	1400*	1675												
\$ENDCT 011142	515	1400*												
\$ENDMG 011213	1400*													
\$ENULL 011210	1400*													
\$ECP 011106	1396	1400*												
\$ECPCT 011134	515*	1400*												
\$ERFLG 001103	402*	1674*	1675*											
\$ERMAX 001115	402*	515*	1674*											
\$ERROR 015274	515	1675*												
\$ERRPC 001116	402*	1667	1668	1669	1675*	1678								
\$ERRTB 001176	402*	1678												
\$ERRTY 015440	1675	1678*												
\$ERTTL 001112	402*	1675*												
\$ESCAP 001170	402*	515*	1674*	1675										
\$FILLC 00115E	402*	1689												
\$FILLS 001155	402*	1689												
\$GDADR 001120	402*													
\$GDDAT 001124	402*	582*	583	595	591*	690*	693	698*	706*	709	714*	717	723*	
	724	726*	729*	734*	737	739	743*	747*	750	753	757*	761*	762	
	767	772*	773	778	784*	785	786	788	791*	796*	797	798	800	
	803*	809*	812	818*	821	826*	829	835*	838	845*	867*	871	878*	

		879*	880	883	891*	892*	893	896	903*	904*	905	908	915*	916
		918	920	922	924	929	932	934	939*	948*	949	951	954	957
		964	967*	979*	983	985	988	991	995	996*	1007	1010*	1011*	1012*
		1013	1017	1027*	1028*	1029*	1030	1046	1052*	1053*	1054*	1055	1071	1079*
		1087*	1092	1099*	1173*	1243*	1264*	1280*	1667	1669				
\$GET42	011164	1400#												
\$GTSWR	016634	1692#	1696											
\$HD =	000000	388												
\$HIOCT	016562	1691#*												
\$ICNT	001104	402#	1674*											
\$ILLUP	016166	1684#												
\$INTAG	001135	402#	1692											
\$ITEMB	001114	402#	1675*	1678										
\$LF	001174	402#	1675	1689	1692									
\$LPADR	001106	402#	515*	1674*										
\$LPEAR	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	850	875	888	900
		912	942	977	999	1025	1050	1075	1086	1098	1109	1131	1158	1182

N06

DR11-K LOGIC
DZDRGC.P11

TEST MAINDEC-11-DZDRGD
CROSS REFERENCE TABLE

MACY11 27(663)

19-DEC-76 09:22

PHCE 42-11

SEQ 0080

ADD	666 1681 743	668 1689 757	670 1691 973	673 1692 946	675	677	1390	1401	1531	1533	1538	1594	1612	1616	1678
ASL	1401				1022	1270	1299	1618	1678	1691	1692	1696			
ASLB	1621														
ASR	592	940	1401												
BCC	533	546	531	526	618	694	702	710	718	727	740	754	768	779	789
BEG	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	1276	1346	1363	1386	1400
	1458	1513	1515	1591	1674	1675	1678	1691	1689	1631	1632				
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						
BISE	1678														
BIT	529	532	583	585	610	617	812	821	861	916	918	929	949	951	968
	983	985	1005	1007	1175	1210	1237	1245	1274	1292	1305	1387	1674	1675	
BITB	1679														
BLO	601	1692													
BLOS	1692														
BMT	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	1272	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	983	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	1296	1290	1308	1312	1316	1323	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	703	717	725	739
	753	767	778	788	800	829	838	871	883	896	908	924	934	957	957
	1017	1046	1071	1090	1092	1102	1148	1154	1171	1179	1197	1232	1230	1324	1334
	1355	1376	1380	1401	1442	1457	1514	1516	1674	1675	1692				
	1674	1689	1692												
CMP2	752	765	776	947	1040	1065									
DEC	538	1391	1400	1415	1445	1447	1472	1619	1678	1692					
DEC2	1681	1689													
FEAT	390														
FEAT2	394	524	602	606	619	639	1675	1684	1689						
FEAT3	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