

DR11K

DR11K DIGITAL I/O
CZDRGE0

AH-8662E-MC

COPYRIGHT 75-80

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

This microfiche card contains a grid of frames on the left side, each containing digital data. The data is organized into columns and rows, with some frames containing headers or labels. The right side of the card is mostly blank, with a few faint markings and a small vertical bar near the bottom right corner.

.REM%

IDENTIFICATION

PRODUCT CODE: AC-8660E-MC
PRODUCT NAME: CZDRGE0 DR11K DIGITAL I/O
DATE: AUG. 1979
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, 1979 BY DIGITAL EQUIPMENT CORPORATION.

TABLE OF CONTENTS

1. ABSTRACT
2. EQUIPMENT REQUIREMENTS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESSES
5. OPERATING PROCEDURE
6. ERRORS
7. RESTRICTIONS
8. MISCELANEOUS
 - 8.1 EXECUTION TIMES
 - 8.2 DEVICE ADDRESSES
9. PROGRAM DESCRIPTION
 - 9.1 LOGIC TEST
 - 9.2 CONTROL LINE LOOP
 - 9.3 COULTER FUNCTION LOOP
 - 9.4 LOGIC TEST WITH COULTER JUMPER'S
 - 9.5 LOGIC TEST WITH H321 LOC-BOX JUMPER'S
 - 9.6 H321 FUNCTION LOOP
 - 9.7 XL01 ADJUSTMENT LOOP
10. LISTING

1. ABSTRACT

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

THIS PROGRAM ALSO INCLUDES TWO ROUTINES TO VERIFY PROPER
CONNECTION'S TO THE H321 LOC-BOX AND ADJUSTMENT OF THE XL01'S.
THE ROUTINES SERVE THE SAME FUNCTION AS THE
'PDL#3 AND PDL#4' ONLINE TEST PROGRAMS.

THE PROGRAM ALSO SUPPLY'S A SOFTWARE ROUTINE TO VERIFY PROPER
CONNECTION'S TO DIGITAL DATA INSTRUMENT'S LIKE THE
COULTER MODEL 'S' COUNTER (REF 9.3).

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

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

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERMINAL AND 8K OF MEMORY
DR11-K OPTION INSTALLED
BC08R-1 ONE FOOT OUTPUT TO INPUT WRAPAROUND CABLE

3. LOADING PROCEDURE

THE PROCEDURE FOR LOADING BINARY TAPES SHOULD BE FOLLOWED.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS (LOGIC TEST)

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

CONTROL:

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

- 1) TYPE CONTROL G <^G>; THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXX NEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE ''NEW='' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED) IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U <^U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

WILL BE USED AS THE SOFTWARE DISPLAY REGISTER.

SW 15 = 1	100000	HALT ON ERROR
SW 14 = 1	040000	LOOP ON CURRENT SUB-TEST
SW 13 = 1	020000	INHIBIT ERROR TYPEDS
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 XL01 ADJUSTMENT LOOP

5. OPERATING PROCEDURE

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

6. ERRORS

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

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

7. RESTRICTIONS

1. IF SEQUENTIAL DR11-K'S, ALL DR11-K'S MUST BE IN THE SAME INTERRUPT SWITCHES AND DATA PATH JUMPER 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.
5. AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC MUST BE RUN FIRST

8. MISCELANEOUS

8.1 EXECUTION TIME

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

8.2 DEVICE ADDRESS PROGRAM LOCATIONS

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

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

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

9.1 LOGIC TEST <SA 200 AND 204>

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

W21A, W22A AND W23A CAN BE DIAGNOSED.

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

9.2 CONTROL LINE LOOP <SA 210>

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

9.3 COULTER FUNCTION LOOP <SA 214>

THE COULTER INTERFACE LOOP PROVIDES THE OPERATOR WITH SOFTWARE INTERFACE WITH THE COULTER HARDWARE. "SBASE" (LOC. 1244) CONTAINS THE DR11K BUS ADDRESS. THE OPERATOR IS INSTRUCTED TO RUN A SAMPLE ON THE COULTER. THE PROGRAM WILL WAIT FOR AN INTERRUPT FROM THE DR11K. UPON AN INTERRUPT, THE PROGRAM DELAYS 70 MSEC. TO ALLOW THE DATA LINES TO SETTLE. THE 70 MSEC. DELAY IS A FACTOR OF CPU TYPE AND MAY REQUIRE THE OPERATOR TO MODIFY THE VALUE OF "CPU" (LOC. 1370) FOR THE PROPER DELAY ON A DIFFERENT CPU TYPE:

11/05	240
11/40	620

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

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

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

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

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

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

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

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

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

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

9.6 H321 LOC-BOX FUNCTION LOOP <SA 230>

THIS LOOP ENABLES THE OPERATOR TO VERIFY PROPER OPERATION OF THE H321 LOC-BOX LAMP'S, SWITCHES AND THE WIRE CONNECTION. THIS LOOP SERVES THE SAME FUNCTION AS THE "PDL DIAG #3 -- PDL3" THE OPERATOR SELECTS THE LOC-BOX'S, VIA DR11K BUS ADDRESS, AND THEN DEPRESSES EACH SWITCH. THE PROGRAM WILL COMPLEMENT THE LAMP ABOVE THE SWITCH. THE OPERATOR SHOULD DEPRESS THE SWITCHES SLOWLY TO VERIFY THAT ONLY DEPRESSING A SWITCH WILL CAUSE THE LAMP TO CHANGE AND THAT RELEASING THE SWITCH DOES NOT. IF THE LATER IS DETECTED THE DR11-K JUMPER CONFIGURATION SHOULD BE QUESTIONED? THE PROGRAM MAY OCCASIONLY MISS A SWITCH CLOSURE.

THE BUS ADDRESSES OF THE DR11K'S ARE 167770, 167760 AND 167750. THE OPERATOR MAY CHANGE LOCATIONS LOC1, LOC2, LOC3 (LOC. 1402, 1404, 1406) TO RUN ON OTHER DR11K'S.

9.7 XL01 ADJUSTMENT LOOP <SA 234>

THIS LOOP ENABLES THE OPERATOR TO VERIFY OR ADJUST THE XL01/CC55A/CC55C POT'S. THIS LOOP SERVES THE SAME FUNCTION AS "PDL DIAGNOSTIC #4 -- PDL4" THE H321 LAMP'S OPERATE AS SHOWN BELOW. REFER TO THE PDL-11 INSTALLATION MANUAL FOR ADJUSTMENT DETAILS.

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

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

10. LISTING

ATTACHED.

z


```
390 .TITLE DR11-K LOGIC TEST MAINDEC-11-CZDRG-E
(1) ;*COPYRIGHT (C) 1979
(1) ;*DIGITAL EQUIPMENT CORP.
(1) ;*MAYNARD, MASS. 01754
(1) ;*
(1) ;*PROGRAM BY R. SHOOP
(1) ;*
(1) ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1) ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1) ;*
391
392 167770 ABASE=167770
393 100300 AVECT1=100300
394 .SBTTL BASIC DEFINITIONS
(1)
(1) ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1) 001100 STACK= 1100
(1) .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
(1) .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1) ;*MISCELLANEOUS DEFINITIONS
(1) 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
(1) 000012 LF= 12 ;;CODE FOR LINE FEED
(1) 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
(1) 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1) 177776 PS= 177776 ;;PROCESSOR STATUS WORD
(1) .EQUIV PS,PSW
(1) 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
(1) 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1) 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1) 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)
(1) ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1) 000000 R0= %0 ;;GENERAL REGISTER
(1) 000001 R1= %1 ;;GENERAL REGISTER
(1) 000002 R2= %2 ;;GENERAL REGISTER
(1) 000003 R3= %3 ;;GENERAL REGISTER
(1) 000004 R4= %4 ;;GENERAL REGISTER
(1) 000005 R5= %5 ;;GENERAL REGISTER
(1) 000006 R6= %6 ;;GENERAL REGISTER
(1) 000007 R7= %7 ;;GENERAL REGISTER
(1) 000006 SP= %6 ;;STACK POINTER
(1) 000007 PC= %7 ;;PROGRAM COUNTER
(1)
(1) ;*PRIORITY LEVEL DEFINITIONS
(1) 000000 PR0= 0 ;;PRIORITY LEVEL 0
(1) 000040 PR1= 40 ;;PRIORITY LEVEL 1
(1) 000100 PR2= 100 ;;PRIORITY LEVEL 2
(1) 000140 PR3= 140 ;;PRIORITY LEVEL 3
(1) 000200 PR4= 200 ;;PRIORITY LEVEL 4
(1) 000240 PR5= 240 ;;PRIORITY LEVEL 5
(1) 000300 PR6= 300 ;;PRIORITY LEVEL 6
(1) 000340 PR7= 340 ;;PRIORITY LEVEL 7
(1)
(1) ;*'SWITCH REGISTER' SWITCH DEFINITIONS
(1) 100000 SW15= 100000
```



```

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


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

(1)
(1) 001000
(1) 001000 000000
(1) 001002 001170
(1) 001004 000030
(1) 001006 000010
(1) 001010 000030
(1) 001012 000031

\$APTHD:
\$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
\$MBADR: .WORD \$MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
\$STIM: .WORD 30 ;;RUN TIM OF LONGEST TEST
\$PASTM: .WORD 10 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 30 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
\$ETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

409

.SBTTL COMMON TAGS

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

(1) 001100 001100
(1) 001100 000000
(1) 001102 000
(1) 001103 000
(1) 001104 000000
(1) 001106 000000
(1) 001110 000000
(1) 001112 000000
(1) 001114 000
(1) 001115 001
(1) 001116 000000
(1) 001120 000000
(1) 001122 000000
(1) 001124 000000
(1) 001126 000000
(1) 001130 000000
(1) 001132 000000
(1) 001134 000
(1) 001135 000
(1) 001136 000000
(1) 001140 177570
(1) 001142 177570
(1) 001144 177560
(1) 001146 177562
(1) 001150 177564
(1) 001152 177566
(1) 001154 000
(1) 001155 002
(1) 001156 012
(1) 001157 000
(1) 001160 000000
(1) 001162 000000
(1) 001164 077
(1) 001165 015
(1) 001166 000012

SCMTAG: .=1100
 .WORD 0
\$TSTNM: .BYTE 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
 .WORD 0
 .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
 .WORD 0
SWR: .WORD DSWR
DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$TPFLG: .BYTE 0
\$TIMES: 0
\$ESCAPE: 0
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCII <12>

:::START OF COMMON TAGS
:::CONTAINS THE TEST NUMBER
:::CONTAINS ERROR FLAG
:::CONTAINS SUBTEST ITERATION COUNT
:::CONTAINS SCOPE LOOP ADDRESS
:::CONTAINS SCOPE RETURN FOR ERRORS
:::CONTAINS TOTAL ERRORS DETECTED
:::CONTAINS ITEM CONTROL BYTE
:::CONTAINS MAX. ERRORS PER TEST
:::CONTAINS PC OF LAST ERROR INSTRUCTION
:::CONTAINS ADDRESS OF 'GOOD' DATA
:::CONTAINS ADDRESS OF 'BAD' DATA
:::CONTAINS 'GOOD' DATA
:::CONTAINS 'BAD' DATA
:::RESERVED--NOT TO BE USED
:::AUTOMATIC MODE INDICATOR
:::INTERRUPT MODE INDICATOR
:::ADDRESS OF SWITCH REGISTER
:::ADDRESS OF DISPLAY REGISTER
:::TTY KBD STATUS
:::TTY KBD BUFFER
:::TTY PRINTER STATUS REG. ADDRESS
:::TTY PRINTER BUFFER REG. ADDRESS
:::CONTAINS NULL CHARACTER FOR FILLS
:::CONTAINS # OF FILLER CHARACTERS REQUIRED
:::INSERT FILL CHARS. AFTER A 'LINE FEED'
:::"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
:::MAX. NUMBER OF ITERATIONS
:::ESCAPE ON ERROR ADDRESS
:::QUESTION MARK
:::CARRIAGE RETURN
:::LINE FEED

.SBTTL APT MAILBOX-ETABLE

.EVEN
\$MAIL: :::APT MAILBOX
\$MSGTY: .WORD AMSGTY :::MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL :::FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN :::TEST NUMBER
\$PASS: .WORD APASS :::PASS COUNT
\$DEVCT: .WORD ADEVCT :::DEVICE COUNT
\$UNIT: .WORD AUNIT :::I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD :::MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG :::MESSAGE LENGTH

(2) 001170
(2) 001170 000000
(2) 001172 000000
(2) 001174 000000
(2) 001176 000000
(2) 001200 000000
(2) 001202 000000
(2) 001204 000000
(2) 001206 000000


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


449					
450	001332	014475	:ITEM 7	EM7	:OPERATOR INTERVENTION ERROR
451	001334	015055		DH4	:ERRPC DRADD
452	001336	015176		DT4	:SERRPC DRADD
453	001340	015224		DF0	
454					
455			:ITEM 10	EM10	:INTERRUPT INPUT BIT FAILED TO SET INPUT READY
456	001342	014531		DH10	:ERRPC DRADD STATUS EXPECTED INPUT BIT
457	001344	015102		DT10	:SERRPC DRADD \$BDDAT \$GDDAT BRLEV3
458	001346	015206		DF0	
459	001350	015224			
460					
461			:ITEM 11	EM11	:NON-INTERRUPTING INPUT BIT SET INPUT READY
462	001352	014614		DH10	:ERRPC DRADD STATUS EXPECTED INPUT BIT
463	001354	015102		DT10	:SERRPC DRADD \$BDDAT \$GDDAT BRLEV3
464	001356	015206		DF0	
465	001360	015224			
466					
467	001362	167770	BASEBA: 167770		:STARTING BASE BUS ADDRESS
468	001364	000300	BASEIV: 300		:STARTING BASE INTERRUPT ADDRESS
469	001366	000200	BASEBR: 200		
470	001370	000240	CPU: 240		:1 MS. CPU DELAY FACTOR 240 FOR 11/05
471					: 620 FOR 11/40
472	001372	000000	NMBEXT: 0		:ADDITIONAL DR-11-K
473	001374	000000	NBEXT: 0		
474					
475	001376	167770	DRADD: 167770		:CURRENT DR11-K STARTING ADDRESS
476	001400	000300	DRIV: 300		:CURRENT DR11-K STARTING INTERRUPT VECTOR
477					
478					
479	001402	167770	LOC1: 167770		:LOC BOX #1 ADDR
480	001404	167760	LOC2: 167760		:LOC BOX #2 ADDR
481	001406	167750	LOC3: 167750		:LOC BOX #3 ADDR
482	001410	167770	GRSTAT: 167770		:DR STATUS
483	001412	167772	GRDAI: 167772		:INPUT REG.
484	001414	167774	GRDIO: 167774		:OUTPUT REG.
485	001416	167775	GRBHIO: 167775		:OUTPUT REG HIGH BYTE
486	001420	000000	NOTLCH: 0		
487	001422	000000	INTBIT: 0		
488	001424	000300	GRIVA: 300		
489	001426	000302	GRIVSA: 302		
490	001430	000304	GRIVB: 304		
491	001432	000306	GRIVSB: 306		
492	001434	000200	DIOBRL: 200		
493	001436	000000	BRLEV1: 0		
494	001440	000000	BRLEV2: 0		
495	001442	000000	BRLEV3: 0		
496	001444	000000	ODDJMP: 0		
497	001446	000000	MINSIN: 0		
498	001450	000000	TRANST: 0		
499	001452	000000	JUMPER: 0		:1 IF RUNNING H322 JUMPER CONFIG ;2 IF COULTER JUMPER CONFIG
500					

```

508
509
510
511 001454 005000
512 001456 005037 001452
513 001462 000414
514 001464 012737 000002 001452 COULTJ:
515 001472 005000
516 001474 000407
517 001476 012737 000001 001452 H322J:
518 001504 005000
519 001506 000402
520 001510 012700 177777
521 001514 000005
522
(1)
(1) 001516 012706 001100
(1) 001522 005026
(1) 001524 022706 001140
(1) 001530 001374
(1) 001532 012706 001100
(1)
(1) 001536 012737 015244 000020
(1) 001544 012737 000340 000022
(1) 001552 012737 015524 000030
(1) 001560 012737 000340 000032
(1) 001566 012737 020254 000034
(1) 001574 012737 000340 000036
(1) 001602 012737 016300 000024
(1) 001610 012737 000340 000026
(1) 001616 013737 011272 011264
(1) 001624 005037 001160
(1) 001630 005037 001162
(1) 001634 112737 000001 001115
(1) 001642 012737 001642 001106
(1) 001650 012737 001650 001110
(2)
(2)
(2) 001656 013746 000004
(2) 001662 012737 001716 000004
(2) 001670 012737 177570 001140
(2) 001676 012737 177570 001142
(2) 001704 022777 177777 177226
(2) 001712 001012
(2)
(2) 001714 000403
(2) 001716 012716 001724
(2) 001722 000002
(2) 001724 012737 000176 001140
(2) 001732 012737 000174 001142
(2) 001740 012637 000004
(1)
(2) 001744 005037 001176
(2) 001750 132737 000200 001211
(2) 001756 001403
(2) 001760 012737 001212 001140

```

```

*****
DIGITAL I-O LOGIC TEST
*****
BEGIN: CLR R0 ;CLEAR R0
        CLR JUMPER ;INDICATE NORMAL FACTORY BUILD
        BR RBEG
COULTJ: MOV #2,JUMPER ;INDICATE COULTER JUMPER CONFIG
        CLR R0
        BR RBEG
H322J:  MOV #1,JUMPER ;INDICATE LOC BOX JUMPER CONFIG
        CLR R0
        BR RBEG
BEGIN1: MOV #-1,R0 ;LOAD R0
RBEG:   RESET
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV # $CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;:CLEAR MEMORY LOCATION
CMP #SWR,R6 ;:DONE?
BNE -6 ;:LOOP BACK IF NO
MOV #STACK,SP ;:SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV # $SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@#IOTVEC+2 ;:LEVEL 7
MOV # $ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
MOV #340,@#EMTVEC+2 ;:LEVEL 7
MOV # $TRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
MOV #340,@#TRAPVEC+2 ;:LEVEL 7
MOV # $PWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
MOV #340,@#PWRVEC+2 ;:LEVEL 7
MOV $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
MOV #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,$SLPERR ;:SETUP THE ERROR LOOP ADDRESS
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
MOV #64,$@#ERRVEC ;:SET UP ERROR VECTOR
MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
;:AND THE HARDWARE SWR IS NOT = -1
BR 65$ ;:BRANCH IF NO TIMEOUT
64$: MOV #65$,(SP) ;:SET UP FOR TRAP RETURN
RTI
65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY
66$: MOV (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
CLR $PASS ;:CLEAR PASS COUNT
BITB #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
BEQ 67$ ;:YES,USE NON-APT SWITCH
MOV # $SWREG,SWR ;:NO,USE APT SWITCH REGISTER

```



```

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

```

568 002402 005037 001450      CLR      TRANST
569 002406 000137 003102      JMP      IOTEST                ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
570 002412 022737 000002 001452 1$:  CMP      #2,JUMPER            ;TEST IF COULTER CONFIG.
571 002420 001015                BNE      3$                    ;BR IF NOT
572 002422 012737 177777 001420  MOV      #-1,NOTLCH           ;LOAD COULTER JUMPER CONFIG.
573 002430 012737 170000 001422  MOV      #170000,INTBIT
574 002436 005037 001446      CLR      MINSIN
575 002442 012737 170000 001450  MOV      #170000,TRANST
576 002450 000137 003102      JMP      IOTEST                ;RUN THE LOGIC TEST WITH COULTER JUMPERS
577 002454 004537 003016      JSR      R5,TALK              ;TALK TO THE ANIMALS
578 002460 013554                SWNLB
579 002462 001420                NOTLCH
580 002464 004537 003016      JSR      R5,TALK              ;ABOUT NON LATCH INPUT BITS
581 002470 013652                SWINTB
582 002472 001422                INTBIT
583 002474 004537 003016      JSR      R5,TALK              ;TALK TO THE ANIMAL AGAIN
584 002500 013750                SWPOSB
585 002502 001446                MINSIN
586 002504 004537 003016      JSR      R5,TALK              ;ABOUT THE INTERRUPTING BITS
587 002510 014044                SWTRAB
588 002512 001450                TRANST
589
590                                ;AGAIN-ABOUT THE TRANSITION BITS
591 002514 012737 010000 001124  .SBTTL  DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
592 002522 033737 001124 001446 10$:  MOV      #BIT12,$GDDAT        ;LOAD TEST BIT
593 002530 001407                BEQ      11$                    ;TEST IF NEG INPUT BIT
594 002532 033737 001124 001450  BIT      $GDDAT,TRANST        ;BR IF NOT
595 002540 001403                BEQ      11$                    ;TEST IF TRANS. INPUT
596 002542 104007                ERROR   7                      ;BR IF NOT
597                                ;LOGICAL OPERATOR ERROR
598 002544 000137 001454      JMP      BEGIN                ;INPUT BIT CANNOT BE NEG. AND TRANS. AT THE SAME
599
600 002550 006137 001124      ROL      $GDDAT                ;
601 002554 103362                BCC     10$                    ;MOVE LEFT
602                                ;BR UNTIL DONE
603 002556 004537 003016      JSR      R5,TALK              ;ASK THE OPERATOR
604 002562 014142                SWDPOB
605 002564 001442                BRLEV3
606 002566 000137 003102      JMP      IOTEST                ;ABOUT PROGRAM OPTIONS
607
608                                ;INTERRUPT TO UNEXPECTED VECTOR
609 002572 021627 001000  VECTR:  CMP      (SP),#1000        ;TEST IF IN PROGRAM CODE
610 002576 103402                BLO     1$                      ;BR IF NOT
611 002 00 000000                HALT
612 002602 000776                BR      70$                    ;FATAL BUS TRAP IN PROGRAM AREA
613 002604 021627 000200 1$:  CMP      (SP),#200            ;TEST IF IN SACRED VECTOR AREA
614 002610 101002                BHI     2$                      ;BR IF NOT
615 002612 000000                HALT
616 002614 000776                BR      71$                    ;FATAL VECTOR TRAP
617 002616 012637 003014 2$:  MOV      (SP)+,72$            ;GET ADDR THE TRAP OCCURRED TO
618 002622 162737 000004 003014  SUB      #4,72$                ;MAKE REAL ADDRESS
619 002630 032777 020000 176302  BIT      #SW13,@SWR           ;TEST IF TYPE ERROR INHIBIT
620 002636 001050                BNE     3$                      ;BR IF INHIBIT
621 002640 104401 002646      TYPE    ,65$                  ;:TYPE ASCIZ STRING
(1) 002644 000417                BR      64$                    ;:GET OVER THE ASCIZ
(1)                                ;:65$: .ASCIZ <15><12>/DR11K INTERRUPTED TO LOC. /

```



```

(1) 002704
622 002704 013746 003014
623 002710 104402
624 002712 104401 002720
(1) 002716 000420
(1)
(1) 002760
625 002760 042737 000007 003014
626 002766 032777 100000 176144
627 002774 001401
628 002776 000000
629 003000 022626
630 003002 013737 003014 001400
631 003010 000137 003126
632 003014 000000
633
634
635
636 003016 012537 003030
637 003022 012537 003074
638 003026 104401
639 003030 013554
640 003032 023727 001140 000176
641 003040 001006
642 003042 104401 017763
643 003046 104412
644 003050 012677 176064
645 003054 000403
646 003056 104401
647 003060 014227
648 003062 000000
649 003064 017777 176050 000002
650 003072 000205
651 003074 001420
652
653
654 003076 104006
655 003100 000002
656
657 003102 000005
658 003104 013737 001362 001376
659 003112 013737 001364 001400
660 003120 013737 001372 001374
661 003126 000005
662 003130 012701 000240
663 003134 012702 000242
664 003140 010221
665 003142 012721 004700
666 003146 022222
667 003150 020227 001076
668 003154 001371
669
670 003156 000005
671 003160 004737 003166
672 003164 000453
673 003166 013737 001376 001410

```

```

64$: MOV 72$,-(SP) ;LOAD VALUE TO BE TYPED
      TYPOC
      TYPE ,67$ ;:TYPE ASCIZ STRING
      BR 66$ ;:GET OVER THE ASCIZ
;:67$: .ASCIZ / AND WILL NOW USE THAT VECTOR/<15><12>
66$:
3$: BIC #7,72$ ;:MASK OFF LOWER 3 BITS
     BIT #SW15,@SWR ;:TEST IF HALT ON ERROR
     BEQ 4$ ;:BR IF NOT
     HALT ;DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
4$: CMP (SP)+,(SP)+ ;:CLEAN THE STACK
     MOV 72$,DRIV ;:LOAD NEW VECTOR ADDRESS
     JMP RBEG2 ;:TEST THE DR11K AT THE NEW VECTOR
72$: 0

```

```

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

```

```

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

```

```

IOTEST: RESET
        MOV BASEBA,DRADD ;:LOAD INITIAL STARTING ADDRESS
        MOV BASEIV,DRIV ;:LOAD INITIAL STARTING VECTOR
        MOV NMBEXT,NBEXT ;:RELOAD # AVAILABLE
RBEG2: RESET
        MOV #240,R1 ;:LOAD R1
        MOV #242,R2
5$: MOV R2,(R1)+ ;:SAVE THE ADDRESS
     MOV #4700,(R1)+ ;:LOAD "JSR PC,R0"
     CMP (R2)+,(R2)+ ;:BUMP R2
     CMP R2,#SCMTAG-2 ;:TEST FOR LAST
     BNE 5$ ;:BR UNTIL DONE

```

```

IOTST1: RESET
        JSR PC,SETADD ;:SET UP BUS ADDRESS AND VECTOR
        BR IOTTS1
SETADD: MOV DRADD,GRSTAT ;:LOAD 1ST ADDRESS

```

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


706
(3)
(3)
(2) 003446 000004
(1) 003450 012737 000040 001160
707 003456 005037 001124
708 003462 012777 177777 175724
709 003470 000005
710 003472 017737 175716 001126
711 003500 001401
712 003502 104003
713
714
(3)
(3)
(2) 003504 000004
715 003506 012737 052525 001124
716 003514 012777 052525 175672
717 003522 017737 175666 001126
718 003530 023737 001124 001126
719 003536 001401
720 003540 104003
721
722
(3)
(3)
(2) 003542 000004
723 003544 012737 125252 001124
724 003552 012777 125252 175634
725 003560 017737 175630 001126
726 003566 023737 001124 001126
727 003574 001401
728 003576 104003
729
730
(3)
(3)
(2) 003600 000004
(1) 003602 012737 000004 001160
731 003610 012737 003622 001110
732 003616 005037 001124
733 003622 013777 001124 175564
734 003630 017737 175560 001126
735 003636 023737 001124 001126
736 003644 001401
737 003646 104003
738 003650 005237 001124
739 003654 001362

```
*****
*TEST 3      TEST THAT RESET CLEARS OUTPUT REG.
*****
TST3:  SCOPE
        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
        CLR     $GDDAT
        MOV     #-1,@GRDIO
        RESET
        MOV     @GRDIO,$BDDAT   ;SET DATA TO ALL ONES
        BEQ     TST4            ;READ OUTPUT REG.
        ERROR   3              ;;BR IF EQUAL
                                ;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,$TIMES      ;;DO 4 ITERATIONS
        MOV     #2$,$LPERR     ;LOAD SCOPE ERROR RETURN
        CLR     $GDDAT         ;CLEAR PATTERN
        MOV     $GDDAT,@GRDIO  ;LOAD THE OUTPUT REG.
        MOV     @GRDIO,$BDDAT  ;READ THE REG.
        CMP     $GDDAT,$BDDAT  ;COMPARE
        BEQ     1$            ;;BR IF EQUAL
        ERROR   3              ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
        INC     $GDDAT         ;UPDATE THE PATTERN
        BNE    2$            ;TRY AGAIN
```

```
741
(3)
(3)
(2) 003656 000004
742 003660 012737 003674 001110
743 003666 012737 000001 001124
744
745 003674 005077 175514
746 003700 053777 001124 175506
747 003706 017737 175502 001126
748 003714 023737 001124 001126
749 003722 001401
750 003724 104003
751
752 003726 006337 001124
753 003732 001360
754
(3)
(3)
(2) 003734 000004
755 003736 012737 003752 001110
756 003744 012737 000001 001124
757
758 003752 012777 177777 175434
759 003760 043777 001124 175426
760 003766 017737 175422 001126
761 003774 005137 001126
762 004000 023737 001124 001126
763 004006 001401
764 004010 104003
765
766 004012 006337 001124
767 004016 001355
768
769
(3)
(3)
(2) 004020 000004
770 004022 012737 125252 001124
771 004030 013777 001124 175356
774 004036 005177 175352
(1) 004042 005177 175346
(1) 004046 005177 175342
(1) 004052 005177 175336
(1) 004056 005177 175332
(1) 004062 005177 175326
(1) 004066 005177 175322
(1) 004072 005177 175316
775 004076 017737 175312 001126
776 004104 023737 001124 001126
777 004112 001401
778 004114 104003

*****
*TEST 7 FLOAT A 1 ACROSS THE OUTPUT REGISTER
*****
TST7: SCOPE
MOV #1,$LPERR ;LOAD SCOPE ERROR RETURN
MOV #BIT0,$GDDAT ;LOAD EXPECTED VALUE
1$: CLR @GRDIO ;CLEAR OUTPUT
BIS $GDDAT,@GRDIO ;SET THAT BIT
MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;TEST RESULTS
BEQ 2$ ;BR IF EQUAL ?
ERROR 3

2$: ASL $GDDAT ;SHIFT EXPECTED DATA
BNE 1$ ;BR UNTIL DONE

*****
*TEST 10 FLOAT A 0 ACROSS THE OUTPUT REGISTER
*****
TST10: SCOPE
MOV #1,$LPERR ;LOAD SCOPE ERROR RETURN
MOV #BIT0,$GDDAT ;LOAD EXPECTED VALUE
1$: MOV #-1,@GRDIO ;LOAD OUTPUT TO A ONE
BIC $GDDAT,@GRDIO ;CLEAR A BIT
MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
COM $BDDAT ;COMPLEMENT IT
CMP $GDDAT,$BDDAT ;EQUAL ?
BEQ 2$ ;BR IF EQUAL
ERROR 3

2$: ASL $GDDAT ;SHIFT LEFT
BNE 1$ ;BRANCH UNTIL DONE

*****
*TEST 11 TEST FOR SLOW OUTPUT GATES WITH #125252
*****
TST11: SCOPE
MOV #125252,$GDDAT ;LOAD EXPECTED VALUE
MOV $GDDAT,@GRDIO ;LOAD OUTPUT
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
COM @GRDIO ;COMPLEMENT OUTPUT REG.
MOV @GRDIO,$BDDAT ;READ OUTPUT REG.
CMP $GDDAT,$BDDAT ;TEST REGISTER
BEQ TST12 ;;BR IF CORRECT
ERROR 3
```


780
(3)
(3)
(2) 004116 000004
781 004120 012737 052525 001124
782 004126 013777 001124 175260
785 004134 005177 175254
(1) 004140 005177 175250
(1) 004144 005177 175244
(1) 004150 005177 175240
(1) 004154 005177 175234
(1) 004160 005177 175230
(1) 004164 005177 175224
(1) 004170 005177 175220
786 004174 017737 175214 001126
787 004202 023737 001124 001126
788 004210 001401
789 004212 104003
790
791
(3)
(3)
(2) 004214 000004
(1) 004216 012737 000400 001160
792 004224 012737 004246 001110
793 004232 012737 052400 001124
794 004240 013777 001124 175146
795 004246 113777 001124 175140
796 004254 017737 175134 001126
797 004262 023737 001124 001126
798 004270 001401
799 004272 104003
800 004274 105237 001124
801 004300 001362
802
803
(3)
(3)
(2) 004302 000004
(1) 004304 012737 000400 001160
804 004312 012737 004334 001110
805 004320 012737 000252 001124
806 004326 013777 001124 175060
807 004334 113777 001125 175054
808 004342 017737 175046 001126
809 004350 023737 001124 001126
810 004356 001401
811 004360 104003
812 004362 105237 001125
813 004366 001362

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

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

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

815
(3)
(3)
(2) 004370 000004
816 004372 005077 175016
817 004376 012777 177777 175006
818 004404 012737 100000 001124
819 004412 012777 100000 174770
820 004420 017737 174764 001126
821 004426 033737 001124 001126
822 004434 001001
823 004436 104001
824
825
(3)
(3)
(2) 004440 000004
826 004442 005077 174742
827 004446 012737 040000 001124
828 004454 012777 040000 174726
829 004462 017737 174722 001126
830 004470 033737 001124 001126
831 004476 001001
832 004500 104001
833
834
(3)
(3)
(2) 004502 000004
835 004504 012737 000200 001124
836 004512 012777 000200 174670
837 004520 017737 174664 001126
838 004526 023737 001124 001126
839 004534 001401
840 004536 104001
841
842
(3)
(3)
(2) 004540 000004
843 004542 005077 174642
844 004546 012737 000100 001124
845 004554 012777 000100 174626
846 004562 017737 174622 001126
847 004570 023737 001124 001126
848 004576 001401
849 004600 104001
850

```
*****  
: *TEST 15 TEST OUTPUT DATA ACCEPT FLAG  
*****  
TST15: SCOPE  
CLR @GRDIO ;CLEAR STATUS  
MOV #-1,@GRDAI ;LOAD EXPECTED  
MOV #BIT15,$GDDAT ;SET BIT 15  
MOV @GRSTAT,@GRSTAT ;READ STATUS  
BIT $GDDAT,$BDDAT ;READ STATUS  
BNE TST16 ;;BR IF SET  
ERROR 1 ;ERROR, BIT 15 FAILED TO SET  
  
*****  
: *TEST 16 TEST OUTPUT INTERRUPT ENABLE  
*****  
TST16: SCOPE  
CLR @GRSTAT ;CLEAR STATUS  
MOV #BIT14,$GDDAT ;LOAD EXPECTED  
MOV #BIT14,@GRSTAT ;LOAD BIT 14  
MOV @GRSTAT,$BDDAT ;READ STATUS  
BIT $GDDAT,$BDDAT ;READ STATUS  
BNE TST17 ;;BR IF SET  
ERROR 1 ;ERROR BIT 14 FAILED TO SET  
  
*****  
: *TEST 17 TEST INPUT DATA READY FLAG  
*****  
TST17: SCOPE  
MOV #BIT7,$GDDAT ;LOAD EXPECTED  
MOV #BIT7,@GRSTAT ;SET BIT 7  
MOV @GRSTAT,$BDDAT ;READ RESULT  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST20 ;;BR IF EQUAL  
ERROR 1 ;ERROR, BIT 7 FAILED TO SET  
  
*****  
: *TEST 20 TEST INPUT INTERRUPT ENABLE  
*****  
TST20: SCOPE  
CLR @GRSTAT ;CLEAR STATUS  
MOV #BIT6,$GDDAT ;LOAD EXPECTED  
MOV #BIT6,@GRSTAT ;SET BIT 6  
MOV @GRSTAT,$BDDAT ;READ RESULT  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST21 ;;BR IF EQUAL  
ERROR 1 ;ERROR, BIT 6 FAILED TO SET
```



```

852
(3)
(3)
(2) 004602 000004
(1) 004604 012737 000040 001160
853 004612 005077 174572
854 004616 005037 001124
855 004622 012777 140300 174560
856 004630 000005
857 004632 017737 174552 001126
858 004640 001401
859 004642 104001
860
861
865
869
(3)
(3)
(2) 004644 000004
870 004646 032777 002000 174264
871 004654 001405
872 004656 112737 000036 001102
873 004664 000137 007200
874 004670
(1) 004670 005077 174520
875 004674 012777 177777 174510
876 004702 005037 001124
877 004706 012777 000000 174500
878 004714 017737 174472 001126
879 004722 043737 001444 001126
880 004730 023737 001124 001126
881 004736 001401
882 004740 104002
883
884
(3)
(3)
(2) 004742 000004
885 004744 005077 174444
886 004750 012777 177777 174434
887 004756 012737 177777 001124
888 004764 043737 001444 001124
889 004772 013777 001124 174414
890 005000 017737 174406 001126
891 005006 043737 001444 001126
892 005014 023737 001124 001126
893 005022 001401
894 005024 104002
895
  
```

```

*****
*TEST 21 TEST THAT RESET CLEARS THE DIGITAL STATUS REGISTER
*****
TST21: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
CLR @GRSTAT ;CLEAR STATUS
CLR $GDDAT ;LOAD EXPECTED
MOV #BIT15!BIT14!BIT7!BIT6,@GRSTAT
RESET
MOV @GRSTAT,$BDDAT ;READ RESULT
BEQ TST22 ;;BR IF EQUAL
ERROR 1 ;ERROR, RESET FAILED TO CLEAR DIGITAL
;STATUS REGISTER
  
```

```

*****
*TEST 22 TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
*****
TST22: SCOPE
BIT #BIT10,@SWR ;TEST SWITCH BIT
BEQ 1$ ;BRANCH IF DOWN
MOVB #36,$TSTNM ;LOAD NEW TEST NUMBER
JMP DRT21 ;BYPASS SOME TEST USING THE EXTERNAL CABLE
1$:
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
CLR $GDDAT ;CLEAR EXPECTED
MOV #0,@GRDIO ;LOAD THE OUTPUT
MOV @GRDAI,$BDDAT ;READ THE INPUT
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST23 ;;BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL THE OUTPUT REG.
  
```

```

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

897
(3)
(3)
(2) 005026 000004
898 005030 005077 174360
899 005034 012777 177777 174350
900 005042 012737 052525 001124
901 005050 043737 001444 001124
902 005056 013777 001124 174330
903 005064 017737 174322 001126
904 005072 043737 001444 001126
905 005100 023737 001124 001126
906 005106 001401
907 005110 104002

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

908
909
(3)
(3)
(2) 005112 000004
910 005114 005077 174274
911 005120 012777 177777 174264
912 005126 012737 125252 001124
913 005134 043737 001444 001124
914 005142 013777 001124 174244
915 005150 017737 174236 001126
916 005156 043737 001444 001126
917 005164 023737 001124 001126
918 005172 001401
919 005174 104002

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



```
921
(3)
(3)
(2) 005176 000004
(1) 005200 012737 000001 001160
922 005206 005737 001444
923 005212 001461
924 005214 012737 010000 001124
925 005222 033737 001444 001124 1$:
926 005230 001447
927 005232 033737 001420 001124
928 005240 001043
929 005242 013777 001124 174144
930 005250 012777 177777 174134
931 005256 043777 001124 174130
932 005264 017737 174122 001126
933 005272 023737 001124 001126
934 005300 001401
935 005302 104002
936
937
938 005304 033737 001124 001446 3$:
939 005312 001016
940 005314 012777 177777 174070
941 005322 053777 001124 174064
942 005330 017737 174056 001126
943 005336 023737 001124 001126
944 005344 001401
945 005346 104002
946
947
948 005350 006137 001124 2$:
949 005354 103322
```

```
*****
*TEST 26 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
*****
TST26: SCOPE
MOV #1,$TIMES ;;DO 1 ITERATION
TST ODDJMP ;TEST IF ANY ODD JUMPERS
BEQ TST27 ;;BR IF NONE
MOV #BIT12,$GDDAT ;LOAD TEST BIT
BIT ODDJMP,$GDDAT ;TEST IF ODD JUMPER BIT
BEQ 2$ ;BR IF NOT
BIT NOTLCH,$GDDAT ;TEST IF LATCHING INPUT BIT
BNE 2$ ;BR IF NOT
MOV $GDDAT,@GRDIO ;LOAD OUTPUT
MOV #-1,@GRDAI ;CLEAR INPUT
BIC $GDDAT,@GRDIO ;CLEAR OUTPUT BIT
MOV @GRDAI,$BDDAT ;READ INPUT REG
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 3$ ;BR IF EQUAL
ERROR 2 ;ERROR, NEG. INPUT OR NEG. TRANSITION
; INPUT BIT FAILED TO SET INPUT REGISTER

BIT $GDDAT,MINSIN ;TEST IF NEG. INPUT BIT
BNE 2$ ;BR IF
MOV #-1,@GRDAI ;CLEAR INPUT
BIS $GDDAT,@GRDIO ;LOAD OUTPUT
MOV @GRDAI,$BDDAT ;READ INPUT
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ;BR IF EQUAL
ERROR 2 ;ERROR, POSITIVE INPUT TRANSITION
;LOGIC FAILED TO SET INPUT REGISTER BIT

ROL $GDDAT ;CHANGE DATA PATTERN
BCC 1$ ;BR IF MORE DATA
```

```

951
(3)
(3)
(2) 005356 000004
952 005360 012737 005406 001110
953
954 005366 012737 000001 001440
955 005374 013737 001420 001442
956 005402 005137 001442
957 005406 013737 001440 001124
958 005414 033737 001124 001444
959 005422 001042
960 005424 033737 001124 001420
961 005432 001436
962
963 005434 013777 001124 173752
964 005442 017737 173744 001126
965 005450 043737 001442 001126
966 005456 023737 001124 001126
967 005464 001401
968 005466 104002
969
970
971
972
973 005470 043777 001124 173716
974 005476 017737 173710 001126
975 005504 043737 001442 001126
976 005512 005037 001124
977 005516 033737 001440 001126
978 005524 001401
979 005526 104002
980
981
982 005530 006337 001440
983 005534 001324
984

```

```

*****
*TEST 27      FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
*****
TST27: SCOPE
MOV      #2$, $LPERR          ;LOAD ERROR SCOPE RETURN
MOV      #BIT0, BRLEV2        ;LOAD EXPECTED
MOV      NOTLCH, BRLEV3       ;GET NON-LATCH.
COM      BRLEV3               ;COMPLEMENT
2$: MOV   BRLEV2, $GDDAT       ;LOAD GOOD
BIT      $GDDAT, ODDJMP       ;TEST IF ODD JUMPER
BNE      1$                   ;BYPASS IF ODD JUMPER
BIT      $GDDAT, NOTLCH       ;TEST FOR NON-LATCH
BEQ      1$                   ;BR IF LATCHING
MOV      $GDDAT, @GRDIO        ;LOAD OUTPUT
MOV      @GRDA1, $BDDAT        ;READ INPUT
BIC      BRLEV3, $BDDAT        ;MASK TO LATCH BITS
CMP      $GDDAT, $BDDAT        ;COMPARE
BEQ      3$                   ;;BR IF EQUAL
ERROR    2                    ;INPUT REGISTER IN ERROR
                               ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
3$: BIC   $GDDAT, @GRDIO        ;CLEAR OUTPUT BIT
MOV      @GRDA1, $BDDAT        ;READ INPUT
BIC      BRLEV3, $BDDAT        ;MASK TO LATCH BITS
CLR      $GDDAT                ;CLEAR EXPECTED
BIT      BRLEV2, $BDDAT        ;TEST FOR BIT
BEQ      1$                   ;;BR IF CLEARED
ERROR    2                    ;INPUT BIT LATCHED IN ERROR
                               ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
1$: ASL   BRLEV2                ;CHANGE PATTERN
BNE      2$                   ;BR UNTIL DONE

```



```

986
(3)
(3)
(2) 005536 000004
987 005540 012737 005566 001110
988 005546 012737 000001 001124
989 005554 005077 173634
990 005560 012777 177777 173624
991
992 005566 033737 001124 001420 2$: BIT $GDDAT,NOTLCH ;TEST FOR NON-LATCHING
993 005574 001021 BNE 1$ ;BR IF NON-LATCH
994 005576 033737 001124 001444 BIT $GDDAT,ODDJMP ;TEST IF ODD JUMPER
995 005604 001015 BNE 1$ ;BYPASS IF ODD JUMPER
996
997 005606 013777 001124 173600 MOV $GDDAT,@GRDIO ;LOAD OUTPUT
998 005614 005077 173574 CLR @GRDIO ;CLEAR OUTPUT REGISTER
999 005620 017737 173566 001126 MOV @GRDAI,$BDDAT ;READ INPUT REG.
1000 005626 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1001 005634 001401 BEQ 1$ ;;BR IF EQUAL
1002 005636 104002 ERROR 2 ;INPUT REGISTER FAILED TO LATCH DATA
1003
1004 005640 053777 001124 173544 1$: BIS $GDDAT,@GRDAI ;CLEAR INPUT BIT
1005 005646 006337 001124 ASL $GDDAT ;CHANGE PATTERN
1006 005652 001345 BNE 2$ ;BR UNTIL DONE
1007
1008
(3)
(3)
(2) 005654 000004
1009 005656 012737 005704 001110
1010 005664 012737 000001 001442
1011 005672 005077 173516
1012 005676 012777 177777 173506
1013
1014 005704 033737 001442 001420 2$: BIT BRLEV3,NOTLCH ;TEST FOR LATCHING
1015 005712 001032 BNE 1$ ;BR IF NOT
1016 005714 033737 001124 001444 BIT $GDDAT,ODDJMP ;TEST IF ODD JUMPER
1017 005722 001026 BNE 1$ ;BYPASS IF ODD JUMPER BIT
1018
1019 005724 012737 177777 001124 MOV #-1,$GDDAT ;LOAD
1020 005732 043737 001420 001124 BIC NOTLCH,$GDDAT
1021 005740 043737 001442 001124 BIC BRLEV3,$GDDAT ;MAKE BRLEV3
1022 005746 013777 001124 173440 MOV $GDDAT,@GRDIO ;LOAD OUTPUT
1023 005754 005077 173434 CLR @GRDIO ;CLEAR OUTPUT REGISTER
1024
1025 005760 017737 173426 001126 MOV @GRDAI,$BDDAT ;READ INPUT
1026 005766 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1027 005774 001401 BEQ 1$ ;;BR IF EQUAL
1028 005776 104002 ERROR 2 ;INPUT REGISTER FAILED TO LATCH DATA
1029
1030 006000 053777 001442 173404 1$: BIS BRLEV3,@GRDAI ;CLEAR INPUT BIT
1031 006006 006337 001442 ASL BRLEV3 ;CHANGE PATTERN
1032 006012 001334 BNE 2$ ;BR UNTIL DONE

```

```

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



```
(1) 006316 005100 COM RO
(1) 006320 010077 173070 MOV RO,@GRDIO ;LOAD OUTPUT
(2) 006324 005077 173064 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006330 017701 173056 MOV @GRDAI,R1 ;READ INPUT
(1) 006334 050177 173052 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006340 005100 COM RO
(1) 006342 010077 173046 MOV RO,@GRDIO ;LOAD OUTPUT
(2) 006346 005077 173042 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006352 017701 173034 MOV @GRDAI,R1 ;READ INPUT
(1) 006356 050177 173030 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006362 005100 COM RO
1050
1051 006364 010137 001126 MOV R1,$BDDAT ;LOAD READ
1052 006370 000240 NOP
1053 006372 000240 NOP
1054 006374 000240 NOP
1055 006376 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
1056 006404 001401 BEQ TST33 ;;BR IF EQUAL
1057 006406 104002 ERROR 2 ;INPUT GATE SLOW
1058
1059
```

```
*****
;*TEST 33 TEST FOR SLOW INPUT GATES WITH #52525
*****
```

```
TST33: SCOPE
(2) 006410 000004
1060
1061 006412 012737 052525 001124 MOV #52525,$GDDAT ;SETUP EXPECTED
1062 006420 043737 001444 001124 BIC ODDJMP,$GDDAT ;MASK ODD JUMPER BITS
1063 006426 043737 001420 001124 BIC NOTLCH,$GDDAT ;CONVERT
1064 006434 013700 001124 MOV $GDDAT,RO
1065 006440 005077 172750 CLR @GRDIO ;CLEAR OUTPUT REGISTER
1066 006444 012777 177777 172740 MOV #-1,@GRDAI ;CLEAR INPUT
1067
1074 006452 010077 172736 MOV RO,@GRDIO ;LOAD OUTPUT
(2) 006456 005077 172732 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006462 017701 172724 MOV @GRDAI,R1 ;READ INPUT
(1) 006466 050177 172720 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006472 005100 COM RO
(1) 006474 010077 172714 MOV RO,@GRDIO ;LOAD OUTPUT
(2) 006500 005077 172710 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006504 017701 172702 MOV @GRDAI,R1 ;READ INPUT
(1) 006510 050177 172676 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006514 005100 COM RO
(1) 006516 010077 172672 MOV RO,@GRDIO ;LOAD OUTPUT
(2) 006522 005077 172666 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006526 017701 172660 MOV @GRDAI,R1 ;READ INPUT
(1) 006532 050177 172654 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006536 005100 COM RO
(1) 006540 010077 172650 MOV RO,@GRDIO ;LOAD OUTPUT
(2) 006544 005077 172644 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006550 017701 172636 MOV @GRDAI,R1 ;READ INPUT
(1) 006554 050177 172632 BIS R1,@GRDAI ;CLEAR INPUT
(1) 006560 005100 COM RO
(1) 006562 010077 172626 MOV RO,@GRDIO ;LOAD OUTPUT
(2) 006566 005077 172622 CLR @GRDIO ;CLEAR OUTPUT REGISTER
(1) 006572 017701 172614 MOV @GRDAI,R1 ;READ INPUT
(1) 006576 050177 172610 BIS R1,@GRDAI ;CLEAR INPUT
```

```

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


1084
(3)
(3)
(2) 007004 000004
(1) 007006 012737 000040 001160
1085 007014 005077 172374
1086 007020 012777 177777 172364
1087 007026 012777 177777 172360
1088 007034 005037 001124
1089 007040 000005
1090 007042 017737 172344 001126
1091 007050 043737 001444 001126
1092 007056 001401
1093 007060 104002
1094
1095
(3)
(3)
(2) 007062 000004
1096 007064 012737 000200 001124
1097 007072 005077 172312
1098 007076 012777 000000 172310
1099 007104 022727 000000 000000
1100 007112 017737 172272 001126
1101 007120 023737 001124 001126
1102 007126 001401
1103 007130 104001
1104
1105
1106
1107
(3)
(3)
(2) 007132 000004
1108 007134 012737 100000 001124
1109 007142 005077 172242
1110 007146 012777 000000 172240
1111 007154 022727 000000 000000
1112 007162 017700 172224
1113 007166 017737 172216 001126
1114 007174 100401
1115 007176 104001
1116

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

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

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

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

```

1118 007200
(4)
(3)
(3)
(2) 007200 000004
(1) 007202 012737 000040 001160
1119 007210 000005
1120 007212 005077 172172
1121 007216 005037 177776
1122 007222 012777 007302 172174
1123 007230 012777 000340 172170
1124 007236 012777 007306 172164
1125 007244 012777 000340 172160
1126 007252 012777 000000 172134
1127 007260 017700 172126
1128 007264 000240
1129 007266 000240
1130 007270 000240
1131 007272 000240
1132 007274 005077 172110
1133 007300 000404
1134 007302 104006
1135 007304 000002
1136 007306 104006
1137 007310 000002
1138
(3)
(3)
(2) 007312 000004
1139 007314 005077 172070
1140 007320 012777 007372 172076
1141 007326 012777 000340 172072
1142 007334 012777 007414 172066
1143 007342 012777 000340 172062
1144 007350 052777 000040 172032
1145 007356 005037 177776
1146 007362 000240
1147 007364 000240
1148 007366 104004
1149 007370 000415
1150
1151
1152
1153 007372 012777 007414 172024
1154 007400 022626
1155 007402 005037 177776
1156 007406 000240
1157 007410 000240
1158 007412 000404
1159 007414 022626
1160 007416 104006
1161 007420 005037 177776
  
```

```

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


1163
(3)
(3)
(2) 007424 000004
1164 007426 005077 171756
1165 007432 012777 007510 171764
1166 007440 012777 000340 171760
1167 007446 012777 007552 171754
1168 007454 012777 000340 171750
1169 007462 012777 000100 171720
1170 007470 052777 000040 171712
1171 007476 005037 177776
1172 007502 000240
1173 007504 000240
1174 007506 104004
1175 007510 012777 007552 171706 1\$:
1176 007516 022626
1177 007520 005037 177776
1178 007524 005037 001124
1179 007530 017737 171654 001126
1180 007536 032737 000100 001126
1181 007544 001406
1182 007546 104001
1183 007550 000404
1184 007552 022626 2\$:
1185 007554 104006
1186 007556 005037 177776
1187
(3)
(3)
(2) 007562 000004
1188 007564 005077 171620
1189 007570 012777 007664 171626
1190 007576 012777 000340 171622
1191 007604 012777 007642 171616
1192 007612 012777 000340 171612
1193 007620 052777 020000 171562
1194 007626 005037 177776
1195 007632 000240
1196 007634 000240
1197 007636 104005
1198 007640 000415
1199
1200
1201 007642 012777 007664 171560
1202 007650 022626
1203 007652 005037 177776
1204 007656 000240
1205 007660 000240
1206 007662 000404
1207 007664 022626 2\$:
1208 007666 104006
1209 007670 005037 177776
1210

*TEST 41 TEST THAT THE INPUT INTR. CLEARS INT. ENABLE VIA MAINT. BIT

TST41: SCOPE
CLR @GRSTAT ;CLEAR STATUS
MOV #1\$,@GRIVA ;LOAD RETURN VECTOR
MOV #340,@GRIVSA ;
MOV #2\$,@GRIVB ;LOAD OUTPUT VECTOR
MOV #340,@GRIVSB ;
MOV #BIT6,@GRSTAT ;LOAD INPUT INTR. ENABLE
BIS #BIT5,@GRSTAT ;MAINT. INT C
CLR PSW ;LOWER PSW
NOP
NOP
ERROR 4 ;ERROR, INPUT FAILED TO INTERRUPT
MOV #2\$,@GRIVA ;LOAD INPUT VECTOR
CMP (SP)+,(SP)+ ;POP THE STACK *4
CLR PSW ;LOWER PRIOR.
CLR \$GDDAT ;CLEAR EXPECTED
MOV @GRSTAT,\$BDDAT ;READ STATUS
BIT #BIT6,\$BDDAT ;TEST BIT 6
BEQ TST42 ;;BR IF CLEARED
ERROR 1 ;INPUT INTR. FAILED TO CLEAR
BR TST42 ;;<NEXT TEST>
2\$:
CMP (SP)+,(SP)+ ;POP THE STACK
ERROR 6 ;UNEXPECTED INTERRUPT
CLR PSW

*TEST 42 TEST THAT THE OUTPUT CAN INTR. USING THE MAINT. BIT

TST42: SCOPE
CLR @GRSTAT ;CLEAR STATUS
MOV #2\$,@GRIVA ;LOAD INPUT VECTOR
MOV #340,@GRIVSA ;
MOV #1\$,@GRIVB ;LOAD OUTPUT VECTOR
MOV #340,@GRIVSB ;
BIS #BIT13,@GRSTAT ;MAINT. INTERRUPT
CLR PSW ;LOWER PSW
NOP
NOP
ERROR 5 ;OUTPUT FAILED TO INTERRUPT
BR TST43 ;;BR TO NEXT TEST
;SUB-TEST, TEST THAT IF PSW IS LOWERED AGAIN, NO INTERRUPT WILL OCCUR
; IF IT DOES, 'INT DONE D HIGH' FAILED TO CLEAR 'INT D' FLOP
1\$:
MOV #2\$,@GRIVB ;LOAD OUTPUT VECTOR
CMP (SP)+,(SP)+ ;POP THE STACK *4
CLR PSW
NOP
NOP
BR TST43 ;;<NEXT TEST>
2\$:
CMP (SP)+,(SP)+ ;POP THE STACK
ERROR 6 ;UNEXPECTED INTERRUPT
CLR PSW

```

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



```
1241 (3)
1242 (3)
1243 (2) 010062 000004
1244 010064 032777 002000 171046
1245 010072 001067
1246 010074 012737 010110 001110
1247 010102 012737 000001 001442
1248 010110 005037 001126
1249 010114 012737 000200 001124
1250 010122 033737 001442 001422
1251 010130 001445
1252 010132 005077 171256
1253 010136 012777 177777 171246
1254 010144 005077 171240
1255 010150 053777 001442 171236
1256 010156 043777 001442 171230
1257 010164 053777 001442 171222
1258 010172 005077 171212
1259
1260
1261 010176 105777 171206
1262 010202 100401
1263 010204 104010
1264
1265
1266
1267 010206 043777 001442 171200 2$:
1268 010214 053777 001442 171170
1269 010222 005037 001124
1270 010226 005077 171156
1271 010232 117737 171152 001126
1272 010240 100001
1273 010242 104001
1274
1275 010244 006337 001442 3$:
1276 010250 001317
```

```
*****
*TEST 44 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
*****
TST44: SCOPE
BIT #BIT10,@SWR ;TEST CABLE SWITCH
BNE TST45 ;:BYPASS IF NO I/O CABLE
MOV #1$,$LPERR ;LOAD ERROR SCOPE RETURN
MOV #BIT0,BRLEV3 ;LOAD INTERRUPT BIT
1$: CLR $BDDAT ;CLEAR BAD DATA
MOV #BIT7,$GDDAT ;LOAD GOOD DATA
BIT BRLEV3,INTBIT ;TEST IF THIS BIT WILL INTERRUPT
BEQ 3$ ;:NO TRY NEXT BIT
CLR @GRDIO ;CLEAR OUTPUT REGISTER
MOV #-1,@GRDAI ;CLEAR INPUT
CLR @GRSTAT ;CLEAR STATUS
BIS BRLEV3,@GRDIO ;LOAD OUTPUT
BIC BRLEV3,@GRDIO ;CLEAR OUTPUT
BIS BRLEV3,@GRDIO ;LOAD OUTPUT
CLR @GRSTAT ;CLEAR FLAG FROM DATA READY
;SHOULD REMAIN SET VIA DIRECT SET SIDE
;IF INTERRUPT INPUT SWITCH IS ON
;TEST READY BIT
TSTB @GRSTAT ;:BR IF SET
BMI 2$
ERROR 10 ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
;?? DID OPERATOR GIVE CORRECT
;INPUT INTERRUPT BITS ??
2$: BIC BRLEV3,@GRDIO ;CLEAR OUTPUT
BIS BRLEV3,@GRDAI ;CLEAR INPUT
CLR $GDDAT ;CLEAR EXPECTED
CLR @GRSTAT ;CLEAR STATUS
MOVB @GRSTAT,$BDDAT ;READ STATUS
BPL 3$ ;:BR IF CLEARED
ERROR 1 ;INPUT READY FAILED TO CLEAR
3$: ASL BRLEV3 ;CHANGE BIT
BNE 1$ ;BR IF NOT DONE
```

```
1278  
(3)  
(3)  
(2) 010252 000004  
1279 010254 032777 002000 170656  
1280 010262 001050  
1281  
1282 010264 012737 010300 001110  
1283 010272 012737 000001 001442  
1284 010300 012737 000200 001126 1$:  
1285 010306 005037 001124  
1286  
1287 010312 033737 001442 001422  
1288 010320 001026  
1289 010322 005077 171066  
1290 010326 012777 177777 171056  
1291 010334 005077 171050  
1292 010340 053777 001442 171046  
1293 010346 043777 001442 171040  
1294 010354 053777 001442 171032  
1295 010362 005077 171022  
1296  
1297 010366 105777 171016  
1298 010372 100001  
1299 010374 104011  
1300  
1301  
1302  
1303  
1304 010376 006337 001442 3$:  
1305 010402 001336  
*****  
*TEST 45 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG  
*****  
TST45: SCOPE  
BIT #BIT10,@SWR ;TEST CABLE SWITCH  
BNE TST46 ;;BYPASS IF NO I/O CABLE  
MOV #1$,$LPERR ;LOAD ERROR SCOPE RETURN  
MOV #BIT0,BRLEV3 ;LOAD NON-INTERRUPT BIT  
MOV #200,$BDDAT ;LOAD BAD DATA  
CLR $GDDAT ;CLEAR GOOD DATA  
BIT BRLEV3,INTBIT ;TEST IF THIS BIT WILL INTERRUPT  
BNE 3$ ;;YES SKIP AND TRY NEXT BIT  
CLR @GRDIO ;CLEAR OUTPUT REGISTER  
MOV #-1,@GRDAI ;CLEAR INPUT  
CLR @GRSTAT ;CLEAR STATUS  
BIS BRLEV3,@GRDIO ;LOAD OUTPUT  
BIC BRLEV3,@GRDIO ;CLEAR OUTPUT  
BIS BRLEV3,@GRDIO ;SET OUTPUT  
CLR @GRSTAT ;CLEAR FLAG FROM DATA READY  
;SHOULD REMAIN SET VIA DIRECT SET SIDE  
TSTB @GRSTAT ;TEST READY BIT  
BPL 3$ ;;BR IF CLEAR  
ERROR 11 ;INPUT NON-INTERRUPT BIT SET INPUT READY  
;?? DID OPERATOR GIVE CORRECT  
;INPUT INTERRUPT BITS ??  
ASL BRLEV3 ;CHANGE BIT  
BNE 1$ ;BR IF NOT DONE
```



```

1307
(3)
(3)
(2) 010404 000004
1308 010406 000240
1309 010410 000240
1310 010412 032777 002000 170520
1311 010420 001401
1312 010422 000461
1313 010424 005077 170760
1314 010430 012777 010524 170772
1315 010436 012777 000340 170766
1316 010444 013777 001426 170752
1317 010452 005077 170750
1318 010456 012777 040000 170724
1319 010464 012777 000000 170722
1320 010472 017700 170714
1321 010476 005037 177776
1322 010502 000240
1323 010504 000240
1324 010506 000240
1325 010510 042777 040000 170672
1326 010516 000240
1327 010520 104005
1328 010522 000401
1329 010524 022626
1330 010526
(1) 010526 012777 003076 170670
(1) 010534 012777 000340 170664
(1) 010542 012777 003076 170660
(1) 010550 012777 000340 170654
1331 010556 005037 177776
1332 010562 005077 170622
  
```

```

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

```

1334
(3)
(3)
(2) 010566 000004
(1) 010570 012737 000001 001160
1335 010576 000005
1336 010600 042737 177437 001434
1337 010606 001001
1338 010610 104007
1339 010612 022737 000340 001434
1340 010620 001001
1341 010622 104007
1342 010624 013737 001434 001436
1343 010632 162737 000040 001436
1344 010640 013737 001434 001440
  
```

```

*****
:*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
1$: CMP #340,DIOBRL
BNE 2$
ERROR 7 ;BR LEVEL INDICATED FOR DIGITAL I/O WAS 7
2$: MOV DIOBRL,BRLEV1
SUB #40,BRLEV1
MOV DIOBRL,BRLEV2
  
```

```

1346
(3)
(3)
(2) 010646 000004
1347 010650 005077 170534
1348 010654 012777 010726 170542
1349 010662 013737 001436 177776
1350 010670 005737 001452
1351 010674 001403
1352 010676 012737 000200 177776
1353 010704 052777 000040 170476 2$:
1354 010712 000240
1355 010714 000240
1356 010716 000240
1357 010720 000240
1358 010722 104004
1359 010724 000401
1360 010726 022626
1361 010730
(1) 010730 012777 003076 170465
(1) 010736 012777 000340 170462
(1) 010744 012777 003076 170456
(1) 010752 012777 000340 170452
1362 010760 005037 177776
1363
(3)
(3)
(2) 010764 000004
1364 010766 005077 170416
1365 010772 012777 011060 170424
1366 011000 013737 001440 177776
1367 011006 005737 001452
1368 011012 001403
1369 011014 012737 000240 177776
1370 011022 052777 000040 170360 4$:
1371 011030 000240
1372 011032 000240
1373 011034 000240
1374
;SUB-TEST, NOW LOWER THE PSW AND ALLOW THE INTERRUPT
1375 011036 012777 011072 170360
1376 011044 005037 177776
1377 011050 000240
1378 011052 000240
1379 011054 000240
1380 011056 000403
1381 011060 022626
1382 011062 005037 177776
1383 011066 104006
1384 011070 000401
1385 011072 022626
1386 011074
(1) 011074 012777 003076 170322
(1) 011102 012777 000340 170316
(1) 011110 012777 003076 170312
(1) 011116 012777 000340 170306
1387 011124 005037 177776

```

```

*****
*TEST 50 TEST FOR INTR. FROM DRA INPUT ON LEVEL INDICATED -1 VIA MAINT. INT
*****
TST50: SCOPE
CLR @GRSTAT ;CLEAR ENABLES
MOV #1$,@GRIVA ;SET UP VECTORS
MOV BRLEV1,PSW ;CHANGE PSW
TST JUMPER ;TEST IF FACTORY MODE
BEQ 2$ ;BR IF YES
MOV #200,PSW ;LOAD BR LEVEL #5-1 IF LOC-BOX OR COULTER MODE
BIS #BIT5,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
NOP
NOP
NOP
NOP
ERROR 4 ;ERROR, NO INTERRUPT FROM DIGITAL I/O INPUT
BR 3$ ;;BR TO NEXT TEST
1$: CMP (SP)+,(SP)+
3$: MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
MOV #340,@GRIVSA
MOV #UNEXPT,@GRIVB ;RESET OUTPUT VECTOR
MOV #340,@GRIVSB
CLR PSW ;LOWER PSW
*****
*TEST 51 TEST FOR NO INTR. FROM DRA INPUT ON LEVEL INDICATED VIA MAINT. INT
*****
TST51: SCOPE
CLR @GRSTAT ;CLEAR ENABLES
MOV #1$,@GRIVA ;SET UP VECTORS
MOV BRLEV2,PSW ;CHANGE PSW
TST JUMPER ;TEST IF FACTORY MODE
BEQ 4$ ;BR IF YES
MOV #240,PSW ;LOAD BR LEVEL #5 IF LOC-BOX OR COULTER MODE
BIS #BIT5,@GRSTAT ;GENERATE INPUT MAINT. INTERRUPT
NOP
NOP
NOP
;SUB-TEST, NOW LOWER THE PSW AND ALLOW THE INTERRUPT
MOV #2$,@GRIVA ;RESET VECTOR
CLR PSW ;LOWER PSW
NOP
NOP
BR 3$ ;ERROR
1$: CMP (SP)+,(SP)+
CLR PSW
3$: ERROR 6 ;UNEXPECTED INTERRUPT <IS BR LEVEL PLUG CORRECT>
BR 5$ ;;
2$: CMP (SP)+,(SP)+ ;POP THE STACK
5$: MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
MOV #340,@GRIVSA
MOV #UNEXPT,@GRIVB ;RESET OUTPUT VECTOR
MOV #340,@GRIVSB
CLR PSW

```


1388
 1389
 (3)
 (3)
 (2) 011130 000004
 (1) 011132 012737 000001 001160
 1390 011140 005737 001374
 1391 011144 001415
 1392 011146 032777 010000 167764
 1393 011154 001024
 1394 011156 162737 000010 001376
 1395 011164 062737 000010 001400
 1396 011172 005337 001374
 1397 011176 000413
 1398 011200 013737 001362 001376
 1399 011206 013737 001364 001400
 1400 011214 013737 001372 001374
 1401 011222 000137 011236
 1402 011226 012700 177777
 1403 011232 000137 003126
 1404
 1405
 (1)
 (2)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1) 011236
 (1) 011236 000004
 (1) 011240 005037 001102
 (1) 011244 005037 001160
 (1) 011250 005237 001176
 (1) 011254 042737 100000 001176
 (1) 011262 005327
 (1) 011264 000001
 (1) 011266 003022
 (1) 011270 012737
 (1) 011272 000001
 (1) 011274 011264
 (1) 011276 104401 011343
 (2) 011302 013746 001176
 (2) 011306 104405
 (1) 011310 104401 011340
 (1) 011314 013700 000042
 (1) 011320 001405
 (1) 011322 000005
 (1) 011324 004710
 (1) 011326 000240
 (1) 011330 000240
 (1) 011332 000240
 (1) 011334
 (1) 011334 000137
 (1) 011336 011226

```

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

.SBTTL  END OF PASS ROUTINE

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

$EOP:   SCOPE
        CLR     $STNM        ;;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
        $EOPCT
        TYPE    ,SENDMG     ;;TYPE 'END PASS #'
        MOV     $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
        TYPDS   ;:GO TYPE--DECIMAL ASCII WITH SIGN
        TYPE    ,SENULL    ;:TYPE A NULL CHARACTER
$GET42: MOV     @#42,R0     ;:GET MONITOR ADDRESS
        BEQ     $DOAGN     ;:BRANCH IF NO MONITOR
        RESET  ;:CLEAR THE WORLD
$ENDAD: JSR     PC,(R0)    ;:GO TO MONITOR
        NOP    ;:SAVE ROOM
        NOP    ;:FOR
        NOP    ;:ACT11
$DOAGN: JMP     @(PC)+     ;;RETURN
$RTNAD: .WORD  BYPAS1
  
```

(1) 011340 377 377 000
(1) 011343 015 042412 042116
(1) 011350 050040 051501 020123
(1) 011356 000043

\$ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
\$ENDMG: .ASCIZ <15><12>/END PASS #/

1406

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

*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) ;:PUT THE BINARY NUMBER ON THE STACK
* TYPDS ;:GO TO THE ROUTINE

(1) 011360
(3) 011360 010046
(3) 011362 010146
(3) 011364 010246
(3) 011366 010346
(3) 011370 010546
(1) 011372 012746 020200
(1) 011376 016605 000020
(1) 011402 100004
(1) 011404 005405
(1) 011406 112766 000055 000001
(1) 011414 005000
(1) 011416 012703 011574
(1) 011422 112723 000040
(1) 011426 005002
(1) 011430 016001 011564
(1) 011434 160105
(1) 011436 002402
(1) 011440 005202
(1) 011442 000774
(1) 011444 060105
(1) 011446 005702
(1) 011450 001002
(1) 011452 105716
(1) 011454 100407
(1) 011456 106316
(1) 011460 103003
(1) 011462 116663 000001 177777
(1) 011470 052702 000060
(1) 011474 052702 000040
(1) 011500 110223
(1) 011502 005720
(1) 011504 020027 000010
(1) 011510 002746
(1) 011512 003002
(1) 011514 010502
(1) 011516 000764
(1) 011520 105726
(1) 011522 100003
(1) 011524 116663 177777 177776

\$TYPDS:
MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH 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 1\$;:BR IF INPUT IS POS.
NEG R5 ;: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 \$DTBL(R0),R1 ;:GET THE CONSTANT
3\$: SUB 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=0
BNE 5\$;:FALL THROUGH IF 0
TSTB (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
MOVB 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
BGT 8\$;:GO TO EXIT
MOV R5,R2 ;:GET THE LSD
BR 6\$;:GO CHANGE TO ASCII
8\$: TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
BPL 9\$;:BR IF NO
MOVB -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING

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

```

1439
1440 011766 012706 001100          COULTR: .SBTTL COULTER INTERFACE TEST
1441 011772 004737 011712          MOV      #STACK,SP
1442 011776 004737 003166          JSR      PC,SETTRP      ;LOAD TRAP AND POWER FAIL VECTORS
1443 012002 005077 167402          JSR      PC,SETADD      ;SETUP BUS ADDRESS AND VECTOR
1444 012006 012777 012050 167410  CLR      @GRSTAT        ;CLEAR INPUT STATUR REG
1445 012014 012777 000340 167404  MOV      #20$,@GRIVA    ;LOAD INPUT VECTOR
1446 012022 104401 013524          MOV      #340,@GRIVSA
1447 012026 012701 020336          1$:     TYPE,RUNMSG      ;TELL OPERATOR TO "RUN SAMPLE"
1448 012032 012777 000100 167350  MOV      #BUFFER,R1     ;LOAD POINTER FOR RESULTS
1449 012040 005037 177776          4$:     MOV      #BIT6,@GRSTAT ;ENABLE INTERRUPT
1450 012044 000001          CLR      PS             ;LOWER PSW
1451 012046 000771          WAIT
1452 012050 022626          BR       4$            ;WAIT FOR OPERATOR
1453 012052 012737 000106 011706  20$:    CMP      (SP)+,(SP)+    ;POP STACK
1454 012060 013700 001370          MOV      #70,,EXTTMP    ;LOAD # OF 1 MSEC DELAYS
1455 012064 005300          6$:     MOV      CPU,RO    ;LOAD 1 MSEC. DELAY WEIGHT ;240 FOR 11/05 ;620 FOR 11/40
1456 012066 001376          3$:     DEC      RO         ;DELAY
1457 012070 005337 011706          BNE     3$
1458 012074 001371          DEC     EXTTMP          ;FINISHED ALL MSEC. DELAY ?
1459 012076 017700 167310          BNE     6$            ;BR IF NOT
1460 012102 010021          MOV     @GRDA1,RO      ;SAVE RESULTS
1461 012104 012777 177777 167300  MOV     RO,(R1)+        ;SAVE IN BUFFER
1462 012112 042700 007777          MOV     #-1,@GRDA1     ;CLEAR INPUT
1463 012116 001345          BIC     #7777,RO       ;MASK
1464 012120 010103          BNE     4$            ;BR IF NOT LAST SAMPLE
1465 012122 014300          MOV     R1,R3          ;COPY R1
1466 012124 004737 012140          5$:     MOV     -(R3),RO    ;GET RESULTS
1467 012130 022703 020336          JSR     PC,40$         ;CONVERT AND TYPE
1468 012134 001732          CMP     #BUFFER,R3     ;FINISHED ?
1469 012136 000771          BEQ     1$            ;BR IF YES
1470          BR       5$          ;CONT. TYPING UNTIL DONE
1471 012140 012737 000055 013546  ;SUBROUTINE FOR THE COULTER TEST
1472 012146 012702 013553          40$:    MOV     #55,MSGRUS+2 ;FIX ASCII MESSAGE
1473 012152 012737 000004 011706  MOV     #MSPNT1,R2     ;LOAD DEST. POINTER
1474 012160 000404          MOV     #4,EXTTMP      ;LOAD COUNT
1475 012162 006000          BR       12$
1476 012164 006000          10$:    ROR     RO
1477 012166 006000          ROR     RO
1478 012170 006000          ROR     RO
1479 012172 010001          ROR     RO
1480 012174 042701 177760          12$:    MOV     RO,R1        ;LOAD R1
1481 012200 052701 000060          BIC     #177760,R1     ;MASK
1482 012204 110142          BIS     #60,R1         ;MAKE DIGIT
1483 012206 005337 011706          MOV     R1,-(R2)       ;SAVE DIGIT
1484 012212 001363          DEC     EXTTMP         ;FINISHED ?
1485 012214 000337 013546          BNE     10$           ;BR IF NOT
1486 012220 104401 013544          SWAB   MSGRUS+2       ;ADJUST MESSAGE
1487 012224 000207          TYPE, MSGRUS
          RTS      PC
          ;EXIT

```


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

```

1541
1542
1543 012460 013537 012526
1544 012464 001001
1545 012466 000205
1546 012470 013702 012526
1547 012474 062702 000002
1548 012500 010203
1549 012502 062703 000002
1550 012506 011200
1551 012510 011301
1552 012512 040100
1553 012514 041213
1554 012516 060013
1555 012520 012712 177777
1556 012524 000205
1557
1558 012526 000000
1559 012530 000000
1560 012532 000000
1561 012534 000000
1562 012536 000000
1563
1564
1565 012540 012706 001100
1566 012544 004737 011712
1567 012550 104401 013426
1568 012554 004537 012346
1569 012560 001402 012532
1570 012564 000412
1571 012566 004537 012346
1572 012572 001404 012534
1573 012576 000405
1574 012600 004537 012346
1575 012604 001406 012536
1576 012610 000400
1577
1584 012612 005737 012532
1585 012616 001431
1586 012620 004537 013066
(1) 012624 000 000
(1) 012626 000007
(1) 012630 001402
1587 012632 004537 013066
(1) 012636 001 003
(1) 012640 000070
(1) 012642 001402
1588 012644 004537 013066
(1) 012650 002 006
(1) 012652 000700
(1) 012654 001402
1589 012656 004537 013066
(1) 012662 003 011
(1) 012664 007000
(1) 012666 001402
1590 012670 004537 013066

;SUBROUTINE TO LOAD THE LAMPS FROM THE LOC BOX SWITCHES
LOUP1: MOV @ (R5)+,40$ ;GET ARG.
      BNE 1$ ;BR IF YES
      RTS R5 ;EXIT
1$: MOV 40$,R2
   ADD #2,R2
   MOV R2,R3
   ADD #2,R3
   MOV (R2),R0
   MOV (R3),R1
   BIC R1,R0 ;MASK OFF BITS
   BIC (R2),(R3) ;LOAD OUTPUT BITS
   ADD R0,(R3) ;LOAD
   MOV #-1,(R2) ;CLEAR INPUT
   RTS R5 ;EXIT

40$: 0
41$: 0
LOC1Y: 0 ;YES/NO ANSWER TO LOC BOX #1
LOC2Y: 0
LOC3Y: 0

.SBTTL XL01 ADJUSTMENT ROUTINE
XL01AD: MOV #STACK,SP
        JSR PC,SETTRP
        TYPE ,XLOHDR ;TELL OPERATOR
        JSR R5,INADR ;GET BUS ADDRESS
        LOC1,LOC1Y
        BR XL010
        JSR R5,INADR
        LOC2,LOC2Y
        BR XL010
        JSR R5,INADR
        LOC3,LOC3Y
        BR XL010

XL010: TST LOC1Y ;TEST IF SELECTED
        BEQ 1$ ;BR IF NOT SELECTED
        JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
        .BYTE 0,0 ;LOAD CH# AND SHIFT COUNT
        7 ;LOAD DATA MASK
        LOC1 ;DR11K BUS ADDRESS
        JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
        .BYTE 1,3 ;LOAD CH# AND SHIFT COUNT
        70 ;LOAD DATA MASK
        LOC1 ;DR11K BUS ADDRESS
        JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
        .BYTE 2,6 ;LOAD CH# AND SHIFT COUNT
        700 ;LOAD DATA MASK
        LOC1 ;DR11K BUS ADDRESS
        JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
        .BYTE 3,11 ;LOAD CH# AND SHIFT COUNT
        7000 ;LOAD DATA MASK
        LOC1 ;DR11K BUS ADDRESS
        JSR R5,XLOADJ ;NOW CONVERT THE DATA AND DISPLAY IN LAMPS
    
```


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

```

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


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

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

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

	015070	042104	052011	052123						
	015076	052516	000115							
1687	015102	051105	050122	020103	DM10:	.ASCIZ	/ERRPC	DRADD	TSTNUM	STATUS EXPECT INPUT BIT/
	015110	020040	051104	042101						
	015116	004504	051524	047124						
	015124	046525	051411	040524						
	015132	052524	020123	042440						
	015140	050130	041505	020124						
	015146	044440	050116	052125						
	015154	041040	052111	000						
1688		015162								
1689	015162	001116	001376	015242	DT1:	.EVEN	\$ERRPC,DRADD,TSTNUM,\$BDDAT,\$GDDAT,0			
	015170	001126	001124	000000						
1690	015176	001116	001376	015242	DT4:	\$ERRPC,DRADD,TSTNUM,0				
	015204	000000								
1691	015206	001116	001376	015242	DT10:	\$ERRPC,DRADD,TSTNUM,\$BDDAT,\$GDDAT,BRLEV3,0				
	015214	001126	001124	001442						
	015222	000000								
1692	015224	000000	000000	000000	DF0:	0,0,0,0,0,0,0				
	015232	000000	000000	000000						
	015240	000000								
1693	015242	000000			TSTNUM:	0				


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

1699
1700

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

(1)
(2)
(1)
(1)
(1)
(1)
(1) 015716
(1) 015716 104401 001165
(1) 015722 010046
(1) 015724 005000
(1) 015726 153700 001114
(1) 015732 001004
(1)
(2) 015734 013746 001116
(2)
(2) 015740 104402
(1) 015742 000426
(1) 015744 005300
(1) 015746 006300
(1) 015750 006300
(1) 015752 006300
(1) 015754 062700 001252
(1) 015760 012037 015770
(1) 015764 001404
(1) 015766 104401
(1) 015770 000000
(1) 015772 104401 001165
(1) 015776 012037 016006
(1) 016002 001404
(1) 016004 104401
(1) 016006 000000
(1) 016010 104401 001165
(1) 016014 011000
(1) 016016 001004
(1) 016020 012600
(1) 016022 104401 001165
(1) 016026 000207
(1) 016030
(2) 016030 013046
(2) 016032 104402
(1) 016034 005710
(1) 016036 001770
(1) 016040 104401 016046
(1) 016044 000771
(1) 016046 020040 000
(1) 016052

```

*****
*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
$ERRTYP:
      TYPE      , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
      MOV      RO, -(SP)    ;; SAVE RO
      CLR      RO          ;; PICKUP THE ITEM INDEX
      BISB     @#$ITEMB, RO
      BNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
                          ;; TYPE THE PC OF THE ERROR
                          ;; SAVE $ERRPC FOR TYPEOUT
                          ;; ERROR ADDRESS
                          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
                          ;; GET OUT
                          ;; ADJUST THE INDEX SO THAT IT WILL
                          ;; WORK FOR THE ERROR TABLE
      MOV      $ERRPC, -(SP)
      TYPOC
      BR      6$
1$:   DEC      RO
      ASL     RO
      ASL     RO
      ASL     RO
      ADD     #$ERRTB, RO   ;; FORM TABLE POINTER
      MOV     (RO)+, 2$    ;; PICKUP "ERROR MESSAGE" POINTER
      BEQ     3$          ;; SKIP TYPEOUT IF NO POINTER
      TYPE   0           ;; TYPE THE "ERROR MESSAGE"
                          ;; "ERROR MESSAGE" POINTER GOES HERE
2$:   .WORD  0           ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPE   , $CRLF     ;; PICKUP "DATA HEADER" POINTER
3$:   MOV     (RO)+, 4$   ;; SKIP TYPEOUT IF 0
      BEQ     5$          ;; TYPE THE "DATA HEADER"
                          ;; "DATA HEADER" POINTER GOES HERE
4$:   .WORD  0           ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPE   , $CRLF     ;; PICKUP "DATA TABLE" POINTER
5$:   MOV     (RO), RO    ;; GO TYPE THE DATA
      BNE     7$          ;; RESTORE RO
6$:   MOV     (SP)+, RO   ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPE   , $CRLF     ;; RETURN
      RTS    PC
7$:   MOV     @ (RO)+, -(SP) ;; SAVE @ (RO)+ FOR TYPEOUT
      TYPOC
      TST    (RO)        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
                          ;; IS THERE ANOTHER NUMBER?
      BEQ     6$         ;; BR IF NO
      TYPE   , 8$        ;; TYPE TWO(2) SPACES
      BR     7$         ;; LOOP
8$:   .ASCIZ  ' / '      ;; TWO(2) SPACES
      .EVEN

```


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


```

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

```

1713
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 017002 011646
(1) 017004 016666 000004 000002
(3) 017012 010046
(3) 017014 010146
(3) 017016 010246
(1) 017020 104411
(1) 017022 012600
(1) 017024 005001
(1) 017026 005002
(1) 017030 112046
(1) 017032 001412
(1) 017034 006301
(1) 017036 006102
(1) 017040 006301
(1) 017042 006102
(1) 017044 006301
(1) 017046 006102
(1) 017050 042716 177770
(1) 017054 062601
(1) 017056 000764
(1) 017060 005726
(1) 017062 010166 000012
(1) 017066 010237 017102
(3) 017072 012602
(3) 017074 012601
(3) 017076 012600
(1) 017100 000002
(1) 017102 000000
1714
(1)
(2)
(1)
(1)
(2)
(1)
(1)
(1)
(1)
(1) 017104 022737 000176 001140
(1) 017112 001074
(1) 017114 105777 162024
(1) 017120 100071
(1) 017122 117746 162020
(1) 017126 042716 177600
(1) 017132 022726 000007
(1) 017136 001062

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

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

$RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
MOV      4(SP),2(SP)          ;;INPUT NUMBER
MOV      R0,-(SP)             ;;PUSH R0 ON STACK
MOV      R1,-(SP)             ;;PUSH R1 ON STACK
MOV      R2,-(SP)             ;;PUSH R2 ON STACK
1$: RDLIN                    ;;READ AN ASCII LINE
MOV      (SP)+,R0              ;;GET ADDRESS OF 1ST CHARACTER
CLR      R1                    ;;CLEAR DATA WORD
CLR      R2
2$: MOVB      (R0)+,-(SP)        ;;PICKUP THIS CHARACTER
BEQ      3$                    ;;IF ZERO GET OUT
ASL      R1                    ;;*2
ROL      R2
ASL      R1                    ;;*4
ROL      R2
ASL      R1                    ;;*8
ROL      R2
BIC      #^C7,(SP)            ;;STRIP THE ASCII JUNK
ADD      (SP)+,R1              ;;ADD IN THIS DIGIT
BR       2$                    ;;LOOP
3$: TST      (SP)+              ;;CLEAN TERMINATOR FROM STACK
MOV      R1,12(SP)            ;;SAVE THE RESULT
MOV      R2,$HIOCT
MOV      (SP)+,R2              ;;POP STACK INTO R2
MOV      (SP)+,R1              ;;POP STACK INTO R1
MOV      (SP)+,R0              ;;POP STACK INTO R0
RTI                               ;;RETURN
$HIOCT: .WORD      0           ;;HIGH ORDER BITS GO HERE

.SBTTL TTY INPUT ROUTINE

*****
.ENABL  LSB

*****
*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
*WHEN OPERATING IN TTY FLAG MODE.
$CKSWR: CMP      #SWREG,SWR     ;;IS THE SOFT-SWR SELECTED?
BNE      15$                    ;;BRANCH IF NO
TSTB     @TKS                    ;;CHAR THERE?
BPL      15$                    ;;IF NO, DON'T WAIT AROUND
MOVB     @TKB,-(SP)              ;;SAVE THE CHAR
BIC      #^C177,(SP)            ;;STRIP-OFF THE ASCII
CMP      #7,(SP)+                ;;IS IT A CONTROL G?
BNE      15$                    ;;NO, RETURN TO USER
```



```

(1) ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) ;*CALL:
(1) ;* RDCHR ;:INPUT A SINGLE CHARACTER FROM THE TTY
(1) ;* RETURN HERE ;:CHARACTER IS ON THE STACK
(1) ;* ;:WITH PARITY BIT STRIPPED OFF
(1) ;
(1) SRDCHR: MOV (SP),-(SP) ;:PUSH DOWN THE PC
(1) 017366 011646 MOV 4(SP),2(SP) ;:SAVE THE PS
(1) 017370 016666 000004 000002 1$: TST @STKS ;:WAIT FOR
(1) 017376 105777 161542 BPL 1$ ;:A CHARACTER
(1) 017402 100375 MOVB @STKB,4(SP) ;:READ THE TTY
(1) 017404 117766 161536 000004 BIC #^C<177>,4(SP) ;:GET RID OF JUNK IF ANY
(1) 017412 042766 177600 000004 CMP 4(SP),#23 ;:IS IT A CONTROL-S?
(1) 017420 026627 000004 000023 BNE 3$ ;:BRANCH IF NO
(1) 017426 001013 2$: TST @STKS ;:WAIT FOR A CHARACTER
(1) 017430 105777 161510 BPL 2$ ;:LOOP UNTIL ITS THERE
(1) 017434 100375 MOVB @STKB,-(SP) ;:GET CHARACTER
(1) 017436 117746 161504 BIC #^C177,(SP) ;:MAKE IT 7-BIT ASCII
(1) 017442 042716 177600 CMP (SP)+,#21 ;:IS IT A CONTROL-Q?
(1) 017446 022627 000021 BNE 2$ ;:IF NOT DISCARD IT
(1) 017452 001366 BR 1$ ;:YES, RESUME
(1) 017454 000750 3$: CMP 4(SP),#140 ;:IS IT UPPER CASE?
(1) 017456 026627 000004 000140 BLT 4$ ;:BRANCH IF YES
(1) 017464 002407 000004 000175 CMP 4(SP),#175 ;:IS IT A SPECIAL CHAR?
(1) 017466 026627 000004 000175 BGT 4$ ;:BRANCH IF YES
(1) 017474 003003 BIC #40,4(SP) ;:MAKE IT UPPER CASE
(1) 017476 042766 000040 000004 4$: RTI ;:GO BACK TO USER
(1) 017504 000002
(2) ;*****
(1) ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1) ;*CALL:
(1) ;* RDLIN ;:INPUT A STRING FROM THE TTY
(1) ;* RETURN HERE ;:ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1) ;* ;:TERMINATOR WILL BE A BYTE OF ALL 0'S
(1) ;
(1) SRDLIN: MOV R3,-(SP) ;:SAVE R3
(1) 017506 010346 CLR -(SP) ;:CLEAR THE RUBOUT KEY
(1) 017510 0C5046 1$: MOV #STTYIN,R3 ;:GET ADDRESS
(1) 017512 012703 017742 2$: CMP #STTYIN+7,R3 ;:BUFFER FULL?
(1) 017516 022703 017751 BLOS 4$ ;:BR IF YES
(1) 017522 101456 RDCHR ;:GO READ ONE CHARACTER FROM THE TTY
(1) 017524 104410 MOVB (SP)+,(R3) ;:GET CHARACTER
(1) 017526 112613 10$: CMPB #177,(R3) ;:IS IT A RUBOUT
(1) 017530 122713 000177 BNE 5$ ;:BR IF NO
(1) 017534 001022 TST (SP) ;:IS THIS THE FIRST RUBOUT?
(1) 017536 005716 BNE 6$ ;:BR IF NO
(1) 017540 001007 MOVB #' \,9$ ;:TYPE A BACK SLASH
(1) 017542 112737 000134 017740 TYPE ,9$
(1) 017550 104401 017740 MOV #-1,(SP) ;:SET THE RUBOUT KEY
(1) 017554 012716 177777 6$: DEC R3 ;:BACKUP BY ONE
(1) 017560 005303 CMP R3,#STTYIN ;:STACK EMPTY?
(1) 017562 020327 017742 BLO 4$ ;:BR IF YES
(1) 017566 103434 MOVB (R3),9$ ;:SETUP TO TYPEOUT THE DELETED CHAR.
(1) 017570 111337 017740 TYPE ,9$ ;:GO TYPE
(1) 017574 104401 017740 BR 2$ ;:GO READ ANOTHER CHAR.
(1) 017600 000746 5$: TST (SP) ;:RUBOUT KEY SET?
(1) 017602 005716
    
```


(1)	017604	001406			BEG	7\$::BR IF NO
(1)	017606	112737	000134	017740	MOVB	#'\,9\$::TYPE A BACK SLASH
(1)	017614	104401	017740		TYPE	.9\$	
(1)	017620	005016			CLR	(SP)	::CLEAR THE RUBOUT KEY
(1)	017622	122713	000025	7\$:	CMPB	#25,(R3)	::IS CHARACTER A CTRL U?
(1)	017626	001003			BNE	8\$::BR IF NO
(1)	017630	104401	017751		TYPE	,\$CNTLU	::TYPE A CONTROL "U"
(1)	017634	000726			BR	1\$::GO START OVER
(1)	017636	122713	000022	8\$:	CMPB	#22,(R3)	::IS CHARACTER A "R"?
(1)	017642	001011			BNE	3\$::BRANCH IF NO
(1)	017644	105013			CLRB	(R3)	::CLEAR THE CHARACTER
(1)	017646	104401	001165		TYPE	,\$CRLF	::TYPE A "CR" & "LF"
(1)	017652	104401	017742		TYPE	,\$TTYIN	::TYPE THE INPUT STRING
(1)	017656	000717			BR	2\$::GO PICKUP ANOTHER CHACTER
(1)	017660	104401	001164	4\$:	TYPE	,\$QUES	::TYPE A "?"
(1)	017664	000712			BR	1\$::CLEAR THE BUFFER AND LOOP
(1)	017666	111337	017740	3\$:	MOVB	(R3),9\$::ECHO THE CHARACTER
(1)	017672	104401	017740		TYPE	.9\$	
(1)	017676	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
(1)	017702	001305			BNE	2\$::LOOP IF NOT RETURN
(1)	017704	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
(1)	017710	104401	001166		TYPE	,\$LF	::TYPE A LINE FEED
(1)	017714	005726			TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
(1)	017716	012603			MOV	(SP)+,R3	::RESTORE R3
(1)	017720	011646			MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
(1)	017722	016666	000004	000002	MOV	4(SP),2(SP)	:: FIRST ASCII CHARACTER ON IT
(1)	017730	012766	017742	000004	MOV	#\$TTYIN,4(SP)	
(1)	017736	000002			RTI		::RETURN
(1)	017740	000		9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
(1)	017741	000			.BYTE	0	::TERMINATOR
(1)	017742	000007			\$TTYIN: .BLKB	7	::RESERVE 7 BYTES FOR TTY INPUT
(1)	017751	136	006525	000012	\$CNTLU: .ASCIZ	/'^U/<15><12>	::CONTROL "U"
(1)	017756	043536	005015	000	\$CNTLG: .ASCIZ	/'^G/<15><12>	::CONTROL "G"
(1)	017763	015	051412	051127	\$MSWR: .ASCIZ	<15><12>/SWR = /	
(1)	017770	036440	000040				
(1)	017774	020040	042516	020127	\$MNEW: .ASCIZ	/ NEW = /	
(1)	020002	020075	000				
(1)		020006			.EVEN		

1715

1717
 1718

.SBTTL APT COMMUNICATIONS ROUTINE

```

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



```

(1)          000100
(1)          000040
1719
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 020254 010046
(1) 020256 016600 000002
(1) 020262 005740
(1) 020264 111000
(1) 020266 006300
(1) 020270 016000 020310
(1) 020274 000200
(1)
(1)
(1)
(1)
(1) 020276 011646
(1) 020300 016666 000004 000002
(1) 020306 000002
(1)
(3)
(3)
(3)
(3)
(3)
(3)
(3)
(3) 020310 020276
(3) 020312 016520
(3) 020314 016076
(3) 020316 016052
(3) 020320 016112
(3) 020322 011360
(1)
(3) 020324 017154
(1)
(3) 020326 017104
(3) 020330 017366
(3) 020332 017506
(3) 020334 017002
1720
1721 020336 000000
1722          000001

```

```

APTSPool=100
APTCSUP=040
.SBTTL TRAP DECODER

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

$TRAP:  MOV    RO,-(SP)          ;;SAVE RO
        MOV    2(SP),RO        ;;GET TRAP ADDRESS
        TST    -(RO)           ;;BACKUP BY 2
        MOVB   (RO),RO         ;;GET RIGHT BYTE OF TRAP
        ASL    RO              ;;POSITION FOR INDEXING
        MOV    $TRPAD(RO),RO    ;;INDEX TO TABLE
        RTS    RO              ;;GO TO ROUTINE

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

$TRAP2: MOV    (SP),-(SP)       ;;MOVE THE PC DOWN
        MOV    4(SP),2(SP)     ;;MOVE THE PSW DOWN
        RTI                    ;;RESTORE THE PSW

.SBTTL TRAP TABLE

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

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

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

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

BUFFER: 0
.END

;10 WORD BUFFER FOR THE COULTER INTERFACE TEST

```

ABASE = 167770	392#	409			
ACDW1 = 000000	409				
ACDW2 = 000000	409				
ACPUOP= 000000	409				
ADBR 013302	1619	1655#			
ADCS 013276	1609*	1614*	1617	1653#	
ADCS1 013300	1608*	1654#			
ADDW0 = 000000	409				
ADDW1 = 000000	409				
ADDW10= 000000	409				
ADDW11= 000000	409				
ADDW12= 000000	409				
ADDW13= 000000	409				
ADDW14= 000000	409				
ADDW15= 000000	409				
ADDW2 = 000000	409				
ADDW3 = 000000	409				
ADDW4 = 000000	409				
ADDW5 = 000000	409				
ADDW6 = 000000	409				
ADDW7 = 000000	409				
ADDW8 = 000000	409				
ADDW9 = 000000	409				
ADEVCT= 000000	409				
ADEVN = 000000	409				
AENV = 000000	409				
AENVN = 000000	409				
AFATAL= 000000	409				
AMADR1= 000000	409				
AMADR2= 000000	409				
AMADR3= 000000	409				
AMADR4= 000000	409				
AMAMS1= 000000	409				
AMAMS2= 000000	409				
AMAMS3= 000000	409				
AMAMS4= 000000	409				
AMSGAD= 000000	409				
AMSGLG= 000000	409				
AMSGTY= 000000	409				
AMTYP1= 000000	409				
AMTYP2= 000000	409				
AMTYP3= 000000	409				
AMTYP4= 000000	409				
APASS = 000000	409				
APRIOR= 000000	409				
APTCSU= 000040	171	1718#			
APTENV= 000001	1697	1711	1718#		
APTSIZ= 000200	522	1718#			
APTSPO= 000100	1711	1718#			
ASWREG= 000000	409				
ATESTN= 000000	409				
AUNIT = 000000	409				
AUSWR = 000000	409				
AVECT1= 100300	393#	409			
AVECT2= 000000	409				
BASEBA 001362	467#	523*	529	658	1398

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

i

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

	918	923	1056	1081	1092	1102	1114	1133	1149	1158	1181	1183	1198	1206	1217
	1243	1280	1312												
.EQUAT	385#	394													
.HEADE	385#	390													
.SETUP	386#	447													
.SWRHI	387#	396													
.SWRLO	396#														
.SACT1	388#	406													
.SAPT8	388#	409#													
.SAPTH	388#	408													
.SAPTY	388#	1718													
.SCATC	385#	398													
.SCATA	385#	409													
.SEOP	385#	1405													
.SERRO	385#	1697													
.SERRT	387#	1700													
.SPARM	386#														
.SPOWE	386#	1706													
.SRDOC	387#	1713													
.SREAD	386#	1714													
.SSAVE	386#														
.SSCOP	386#	1696													
.SSPAC	386#														
.SSWDO	386#														
.STRAP	386#	1719													
.STYPD	387#	1406													
.STYPE	385#	386#	1711												
.STYPO	385#	1703													

. ABS. 020340 000 CON RW REL GBL D

ERRORS DETECTED: 0

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