

MINC-11

MNCDI DIAGNOSTIC
CVMNBA0

AH-B089A-MC
COPYRIGHT © 1978
FICHE 1 OF 1

DEC 1978
digital
MADE IN USA

This microfiche card contains a grid of 240 frames (12 columns by 20 rows). The frames are arranged in a regular grid pattern. Each frame contains a different set of data, likely representing a diagnostic report or a series of test results. The data is organized into columns and rows, with some frames containing text, some containing tables, and some containing barcodes or other graphical elements. The overall layout is structured and consistent across the grid.

IDENTIFICATION

SEQ 0001

PRODUCT CODE: AC-B088A-MC
PRODUCT NAME: CVMNBA0 MNCDI (DIGITAL IN) DIAGNOSTIC TEST
DATE: AUGUST 1978
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1978
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
4.1	PROGRAM START
5.0	SOFTWARE SWITCH REGISTER
5.1	OPTIONS
5.2	CONTROL
6.0	ERROR REPORTING
6.1	ERROR COMMENT
6.2	ERROR DATA
7.0	MISCELLANEOUS
7.1	MNC DI BUS ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	MULTIPLE MNC DI INTERFACE TESTING
7.5	RESTRICTIONS
8.0	EXECUTION TIME
9.0	PROGRAM TEST DESCRIPTIONS
9.1	LOGIC TEST
9.2	TEST MODULE WRAP-AROUND MODE
9.3	TEST MODULE SWITCH INPUT LOOP
9.4	LOGIC TEST WITH TESTER SUPPORT
10.0	LISTING

1.0 ABSTRACT

 THE MNCDI DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE. ADDITIONAL ROUTINES ARE PROVIDED TO VERIFY ADDITIONAL SECTIONS. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11/03 COMPUTER OR LSI-11 PROCESSOR
2. DLV11 WITH I/O TERMINAL (LA36, VT100, ETC.)
3. MNCDI (DIGITAL IN) OPTION

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE <XXDP>

-
1. ENSURE THAT THE DIAGNOSTIC LOAD MEDIA IS INSTALLED IN DRIVE 0.
 2. BOOT THE MEDIA BY TYPING '173000G' IF IN THE ODT MICRO-CODE STATE OR CYCLING THE POWER 'ON-OFF' SWITCH.
 3. UPON SUCCESSFUL BOOTING OF THE LOAD MEDIA, THE XXDP MONITOR WILL IDENTIFY ITSELF AND INFORM THE OPERATOR OF THE OPERATING OPTIONS THAT MAYBE SELECTED.
 4. THE OPERATOR SHOULD TYPE 'R VMNB??' FOLLOWED BY A 'RETURN' THE XXDP MONITOR WILL LOAD THE PROGRAM INTO MEMORY AND START THE PROGRAM AT LOCATION 200.

4.0 STARTING PROCEDURE

-
1. THE PROGRAM WILL RESPOND BY TYPING THE PROGRAM TITLE.
 2. THE PROGRAM WILL NOW ASK FOR AN INITIAL SWITCH REGISTER VALUE TO BE STORED IN THE SOFTWARE SWITCH REGISTER.
 3. THE PROGRAM WILL NOW DISPLAY THE MENU OF TEST OR LOOPS AVAILABLE. THE OPERATOR SELECTS THE TESTS BY TYPING THE SELECTED CHARACTER FOLLOWED BY DEPRESSING THE 'RETURN' KEY.

4.1 PROGRAM START

200	STARTING ADDRESS OF THE PROGRAM
204	RESTART ADDRESS OF THE PROGRAM
210	STARTING ADDRESS FOR THE OPTION TESTER.

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW12=1	010000	INHIBIT SIZING THE NUMBER OF MNCDI'S
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <7-0>

5.2 CONTROL

1. THE TEST OR LOOP MAYBE STOPPED BY TYPING THE "CONTROL & C" KEYS. THIS OPERATION WILL STOP THE PROGRAM AND ENABLE THE OPERATOR TO SELECT DIFFERENT PROGRAM COMMAND'S.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE "CONTROL & G" KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.
3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING 'P' (CONTINUE).
4. IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL "CONTROL & G OR C" COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE BY THE PROGRAM.

6.0 ERROR REPORTING

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF.

6.2 ERROR DATA

*UNIT	UNIT NUMBER
*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*BUSADR	MNCDI BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

*ALWAYS REPORTED

7.0 MISCELLANEOUS

7.1 MNCDI BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' (LOC. 1244) IF BASE BUS ADDRESS IS NOT 171160.
MODIFY LOCATION '\$VECT1' (LOC. 1240) IF INTERRUPT VECTOR IS NOT 130.
MODIFY LOCATION '\$CDW1' (LOC. 1250) IF THE OUTPUT BASE ADDRESS
IS NOT 171260 FOR THE WRAP-AROUND TEST.

*NOTE: USE THE 'B' PROGRAM COMMAND TO MODIFY THIS LOCATIONS
AFTER PROGRAM LOAD.

7.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REQUIRES 8K OR MORE).
THIS DIAGNOSTIC DOES SUPPORT 'APT' BUT HAS NOT BEEN RUN UNDER IT.

7.3 POWER FAIL

A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT
WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH
NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).

7.4 MULTIPLE MNCDI INTERFACE TESTING

THIS PROGRAM DOES 'AUTO-SIZE' THE NUMBER OF MNCDI'S CONNECTED.
THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 MNCDI INTERFACES
WITH CONTIGUOUS BUS ADDRESSES. THE 'AUTO-SIZE' CAN BE INHIBITED
BY THE OPERATOR SETTING BIT 15 OF LOCATION '\$ENV' (LOC. 1214) OR
SETTING SWITCH REGISTER BIT 12 TO A ONE. USE THE 'B' PROGRAM
COMMAND TO LOAD THE BASE AND VECTOR ADDRESSES.

7.5 RESTRICTIONS

ALL USER CONNECTIONS MUST BE REMOVED.

8.0 EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS
TO ABOUT 20 SECONDS WITH ITERATIONS ENABLED WITH ONE MNCDI CONNECTED.
AN END PASS MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS.
END OF PASS WILL ALSO REPORT TOTAL ERROR COUNT AND ANY UNIT'S THAT HAD ERRORED.

9.0 PROGRAM TEST DESCRIPTIONS

9.1 L = LOGIC TESTS

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE MNCDI INPUT CONTROL. A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING. THE PROGRAM WILL AUTO-SIZE UP TO 8 MNCDI'S TO BE TESTED.

9.2 W WRAP-AROUND LOGIC AND DATA TEST

THE PURPOSE IS TO VERIFY PROPER OPERATION OF THE DIGITAL INPUT SIGNALS FROM THE CABLE CONNECTOR. A MNCDO SUPPLIES DATA AND CONTROL SIGNALS TO THE MNCDI. THE ROUTINE REQUIRES THE USE OF A MNCDI, MNCDI TEST MODULE, MNCDO, MNCDO TEST MODULE AND A SHORT BERG CABLE.

9.3 T TYPEOUT LOOP FOR INPUT TEST MODULE SWITCHES

THE LOOP ENABLES VERIFICATION OF THE CABLE CONNECTOR INPUT WITHOUT THE USE OF A MNCDO. THE PROGRAM WILL WAIT FOR THE OPERATOR TO DEPRESS THE MNCDI TEST MODULE PUSH-BUTTON. WHEN THE OPERATOR DEPRESSES THE BUTTON, A SIGNAL FLAG (STROBE) IS SENSED AND THE PROGRAM READS A MNCDI REGISTER. THE CONTENTS OF THE MNCDI TEST MODULE SWITCH REGISTER CAN BE READ BY READING THE MNCDI INPUT DATA REGISTER. THE VALUE OF THE INPUT IS REPORTED AS AN OCTAL NUMBER AND A 16 BIT BINARY. THE OPERATOR MAY CHANGE THE TEST MODULE DATA SWITCHES AND DEPRESS THE PUSH-BUTTON AGAIN.

9.4 L = LOGIC TEST WITH TESTER SUPPORT (SA 210)

THE IN-HOUSE TESTER CONSISTS OF A INPUT/OUTPUT DEVICE TO SENSE THE OUTPUT CONTROL SIGNAL AND TO STIMULATE THE INPUT DATA AND CONTROL SIGNALS. AN EXPANDED LOGIC TEST IS EXECUTED INCLUDING A WRAP-AROUND DATA AND CONTROL TEST.

10.0 LISTING

20	BASIC DEFINITIONS
22	OPERATIONAL SWITCH SETTINGS
24	TRAP CATCHER
58	ACT11 HOOKS
60	APT PARAMETER BLOCK
61	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
219	INITIALIZE THE COMMON TAGS
227	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
239	KEYBOARD COMMAND DECODER
278	DETERMINE THE NUMBER OF MNCDI'S ON THE SYSTEM
399	SUBROUTINE TO HANDLE CONTROL C/G
416	
417	MNCDI TESTS
418	
420	T1 VERIFY CORRECT I.D. CODE FOR MNCDI (IN-HOUSE TESTER)
430	T2 VERIFY A MNCDI BUS ADDRESS RESPONSE
442	T3 FLOAT A 1 ACROSS THE MNCDI STIMULUS BIT REGISTER
444	T4 FLOAT A 0 ACROSS THE MNCDI STIMULUS BIT REGISTER
445	T5 ENSURE THAT 'RESET' CLEARS THE MNCDI STIMULUS BIT REGISTER
446	T6 VERIFY BYTE OPERATION ON THE MNCDI STIMULUS BIT REGISTER
448	T7 TEST THAT BIT1 OF MNCDI STATUS REGISTER IS READ-WRITE
449	T10 TEST THAT BIT2 OF MNCDI STATUS REGISTER IS READ-WRITE
450	T11 TEST THAT BIT3 OF MNCDI STATUS REGISTER IS READ-WRITE
452	T12 TEST THAT BIT4 OF MNCDI STATUS REGISTER IS READ-WRITE
453	T13 TEST THAT BITS 5 OF MNCDI STATUS REGISTER IS READ-WRITE
454	T14 TEST THAT BIT6 OF MNCDI STATUS REGISTER IS READ-WRITE
456	T15 TEST THAT BIT8 OF MNCDI STATUS REGISTER IS READ-WRITE
457	T16 TEST THAT BIT9 OF MNCDI STATUS REGISTER IS READ-WRITE
458	T17 TEST THAT BIT12 OF MNCDI STATUS REGISTER IS READ-WRITE
460	T20 TEST THAT BIT14 OF MNCDI STATUS REGISTER IS READ-WRITE
461	T21 ENSURE THAT 'RESET' CLEARS THE MNCDI STATUS REGISTER
462	T22 VERIFY HIGH BYTE OPERATION ON THE INPUT STATUS REGISTER
471	T23 VERIFY LOW BYTE OPERATION ON THE INPUT STATUS REGISTER
481	T24 VERIFY THAT MAINT. STROBE SETS 'INPUT DATA READY'
490	T25 VERIFY THAT 'INPUT DATA READY' CAN BE WRITTEN TO A ZERO
499	T26 VERIFY THAT 'INPUT DATA READY' CAN BE CLEARED BY A 'RESET'
509	T27 'INPUT DATA READY' WILL NOT SET IF IN STIMILUS MODE AND NO SBR MATCH
520	T30 VERIFY THAT 'OVERRUN ERROR' SETS
530	T31 VERIFY THAT 'OVERRUN ERROR' CAN BE WRITTEN TO A ZERO
541	T32 VERIFY THAT 'RESET' CLEARS 'OVERRUN ERROR'
553	T33 VERIFY INVERT DATA FUNCTION
576	T34 VERIFY EACH BIT OF THE MNCDI INPUT DATA REGISTER CAN BE CLEARED
593	T35 VERIFY THAT 'RESET' CLEARS MNCDI INPUT DATA REGISTER
604	T36 VERIFY THAT A 2ND STROBE PULSE WILL NOT CHANGE THE DIR DATA
616	T37 INTERRUPT TEST -- VERIFY MNCDI INTERRUPTS VIA DATA READY VECTOR
639	T40 INTERRUPT TEST -- VERIFY MNCDI INTERRUPTS VIA OVERRUN ERROR
667	
668	
669	
670	
671	

672	
673	
674	WRAPAROUND MODE TESTS
677	T41 VERIFY MODE 00 -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
692	T42 VERIFY MODE 01 -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
704	T43 VERIFY MODE 10 -- INPUT STROBE WILL NOT SET INPUT DATA READY FLAG
716	T44 VERIFY INPUT REPLY SETS OUTPUT DONE FLAG
730	T45 VERIFY THE MNCDO - WRAPAROUND - MNCDI DATA PATH
747	T46 VERIFY THE MNCDO - WRAPAROUND - MNCDI INVERTED DATA PATH
765	T47 VERIFY IN 10 MODE THAT SBR AND INPUT BITS SET INPUT READY
806	T50 TEST THE TRANSITION ENABLE AND TRANSITION DETECTION
831	T51 DETERMINE IF MORE MNCDI'S REMAIN TO BE TESTED
857	END OF PASS ROUTINE
874	MNCDI TEST MODULE SWITCH TYPEOUT LOOP
898	TTY INPUT ROUTINE
899	SCOPE HANDLER ROUTINE
916	ERROR HANDLER ROUTINE
917	ERROR MESSAGE TYPEOUT ROUTINE
919	BINARY TO OCTAL (ASCII) AND TYPE
920	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
921	APT COMMUNICATIONS ROUTINE
924	POWER DOWN AND UP ROUTINES
928	READ AN OCTAL NUMBER FROM THE TTY
932	TYPE ROUTINE
933	BINARY TO ASCII AND TYPE ROUTINE
936	TRAP DECODER
(3)	TRAP TABLE
1037	ASCII MESSAGES

```

15      .TITLE CVMNB-A MNCDI  DIAGNOSTIC
(1)    .*COPYRIGHT (C) 1978
(1)    .*DIGITAL EQUIPMENT CORP.
(1)    .*MAYNARD, MASS. 01754
(1)    .*
(1)    .*PROGRAM BY SHOOP
(1)    .*
(1)    .*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)    .*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)    .*
16
17      171160      ABASE=171160      ;DEFAULT DIGITAL INPUT ADDRESSES
18      000120      AVECT1=120      ;DEFAULT DIGITAL INPUT VECTOR
19      171260      ACDW1=171260    ;DEFAULT DIGITAL OUTPUT ADDRESSES <WRAP-AROUND>
20      .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)
(1)      100000      :* 'SWITCH REGISTER' SWITCH DEFINITIONS
(1)      040000      SW15= 100000
(1)      020000      SW14= 40000
(1)      010000      SW13= 20000
(1)      004000      SW12= 10000
(1)      002000      SW11= 4000
(1)      001000      SW10= 2000
(1)      000400      SW09= 1000
(1)      000200      SW08= 400
(1)      000100      SW07= 200
(1)      000040      SW06= 100
(1)      000020      SW05= 40
(1)      000010      SW04= 20
(1)      000004      SW03= 10
(1)      000002      SW02= 4
(1)      000001      SW01= 2
(1)                                  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)
(1)      100000      :*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)      040000      BIT15= 100000
(1)      020000      BIT14= 40000
(1)      010000      BIT13= 20000
(1)      004000      BIT12= 10000
(1)      002000      BIT11= 4000
(1)      001000      BIT10= 2000
(1)      000400      BIT09= 1000
(1)      000200      BIT08= 400
(1)      000100      BIT07= 200
(1)      000040      BIT06= 100
(1)      000020      BIT05= 40
(1)      000010      BIT04= 20
(1)      000004      BIT03= 10
(1)      000002      BIT02= 4
(1)      000001      BIT01= 2
(1)                                  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
```

```

22          .SBTTL OPERATIONAL SWITCH SETTINGS
(1)          : *
(1)          : * SWITCH                USE
(1)          : * -----
(1)          : * 15          HALT ON ERROR
(1)          : * 14          LOOP ON TEST
(1)          : * 13          INHIBIT ERROR TYPEOUTS
(1)          : * 12          INHIBIT SIZING THE # OF MNCDI'S
(1)          : * 11          INHIBIT ITERATIONS
(1)          : * 9          LOOP ON ERROR
(1)          : * 8          LOOP ON TEST IN SWR<7:0>
23
24          .SBTTL TRAP CATCHER
25
26          000000          .=0
27          : *ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ''+2''
28          : *AND ''JSR PC,R0'' SEQUENCE TO CATCH ILLEGAL INTERRUPTS.
29          : *AND INTERRUPTS TO THE WRONG VECTOR.
30          : *LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
31          : *VECTORS.
41          000004          .=4
42 000004 014570 000200    .WORD IOTRD,200          ;HANDLE BUSS ERROR.
43          000174          .=174
44 000174 000000          DISPREG: .WORD 0          ;SOFTWARE DISPLAY REGISTER
45 000176 000000          SWREG: .WORD 0          ;SOFTWARE SWITCH REGISTER
46
47          000200          .-200
48 000200 000137 001542    JMP BEGIN
49 000204 000137 001534    JMP RESTRT          ;RESTART ADDRESS
50 000210 000137 001554    JMP TESTER          ;TESTER STARTING ADDRESS
51
52
53          000100          .=100
54
55 000100 000104 000200 000002    104,200,RTI          ;LSI-11 'B EVENT'' PROTECTION

```

57
58
(1)
(2)
(1)
(1) 000046
(1) 000052
(1)
(1)
59
60
(1)
(2)
(1)
(2)
(1)
(1) 000024
(1) 000044
(1)
(1)
(1)
(1)
(1) 001000
(1) 001000 000000
(1) 001002 001170
(1) 001004 000030
(1) 001006 000030
(1) 001010 000030
(1) 001012 000032

.SBTTL ACT11 HOOKS

```
*****
:HOOKS REQUIRED BY ACT11
$SVPC=      ;SAVE PC
.=46
$ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
.=52
.WORD 0     ;;2)SET LOC.52 TO ZERO
.$SVPC      ;; RESTORE PC
.=1000
```

.SBTTL APT PARAMETER BLOCK

```
*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
*****
.$X=      ;;SAVE CURRENT LOCATION
.=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
200      ;;FOR APT START UP
.=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR   ;;POINT TO APT HEADER BLOCK
.=.$X     ;;RESET LOCATION COUNTER
*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.
```

```
$APTHD:
$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STSM: .WORD 30 ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 30 ;;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
.WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```

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

```

(2)	001204	000000	\$MSGAD: .WORD	AMSGAD	::MESSAGE ADDRESS
(2)	001206	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
(2)	001210		\$ETABLE:		::APT ENVIRONMENT TABLE
(2)	001210	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
(2)	001211	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
(2)	001212	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
(2)	001214	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
(2)	001216	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
(2)			:*		BITS 15-11=CPU TYPE
(2)			:*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)			:*		11/70=06,PDQ=07,Q=10
(2)			:*		BIT 10=REAL TIME CLOCK
(2)			:*		BIT 9=FLOATING POINT PROCESSOR
(2)			:*		BIT 8=MEMORY MANAGEMENT
(2)	001220	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001221	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
(2)			:*		MEM.TYPE BYTE -- (HIGH BYTE)
(2)			:*		900 NSEC CORE=001
(2)			:*		300 NSEC BIPOLAR=002
(2)			:*		500 NSEC MOS=003
(2)	001222	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			:*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001224	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001225	000	\$MTYP2: .BYTE	AMTYP2	::MEM. TYPE,BLK#2
(2)	001226	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001230	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001231	000	\$MTYP3: .BYTE	AMTYP3	::MEM. TYPE,BLK#3
(2)	001232	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001234	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001235	000	\$MTYP4: .BYTE	AMTYP4	::MEM. TYPE,BLK#4
(2)	001236	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001240	000120	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001242	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001244	171160	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001246	000000	\$DEV: .WORD	ADEV	::DEVICE MAP
(2)	001250	171260	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001252	000000	\$CDW2: .WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
(2)	001254		\$ETEND:		
(2)			.MEXIT		


```

(1) .SBTTL ERROR POINTER TABLE
(1)
(1) : *THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) : *THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) : *LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) : *NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) : *NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) : * EM ;;POINTS TO THE ERROR MESSAGE
(1) : * DH ;;POINTS TO THE DATA HEADER
(1) : * DT ;;POINTS TO THE DATA
(1) : * DF ;;POINTS TO THE DATA FORMAT
(1)
(1) $ERRTB:
(1) 001254
62
63 :ITEM 1
64 001254 015652 017150 020512 EM1,DH1,DT1,DF0 ;MNCDO BUS ERROR
001262 020674
65 :ITEM 2
66 001264 016513 017406 020614 EM12,DH12,DT12,DF0 ;MNCDO-MNCDI WRAP-AROUND STATUS ERROR
001272 020674
67 :ITEM 3
68 001274 016604 017451 020632 EM13,DH13,DT13,DF0 ;MNCDO-MNCDI WRAP-AROUND DATA ERROR
001302 020674
69 :ITEM 4
70 001304 016673 017275 020554 EM14,DH7,DT7,DF0 ;MNCDI INCORRECT I.D. VALUE
001312 020674
71 :ITEM 5
72 001314 016150 017201 020524 EM5,DH5,DT5,DF0 ;MNCDI BUS ERROR
001322 020674
73 :ITEM 6
74 001324 016255 017241 020540 EM6,DH6,DT6,DF0 ;MNCDI STIMULUS REGISTER ERROR
001332 020674
75 :ITEM 7
76 001334 016330 017275 020554 EM7,DH7,DT7,DF0 ;MNCDI STATUS REGISTER ERROR
001342 020674
77 :ITEM 10
78 001344 016401 017331 020570 EM10,DH10,DT10,DF0 ;MNCDI DATA REGISTER ERROR
001352 020674
79 :ITEM 11
80 001354 016450 017364 020604 EM11,DH11,DT11,DF0 ;MNCDI INTERRUPT ERROR
001362 020674
81 :ITEM 12
82 001364 016740 017514 020650 EM15,DH15,DT15,DF0 ;MNCDI ILLEGAL OR TRAP INTERRUPT
001372 020674
83 :ITEM 13
84 001374 017011 017547 020662 EM16,DH16,DT16,DF0 ;EXISTING MNCDI FAILED TO RESPOND
001402 020674

```

157	001404	000000	OCSR:	0	
158	001406	000000	OCSR1:	0	:HIGH BYTE ADDRESS
159					
160	001410	000000	DOR:	0	
161	001412	000000	DOR1:	0	:HIGH BYTE ADDRESS
162					
163	001414	000000	ICSR:	0	
164	001416	000000	ICSR1:	0	:HIGH BYTE ADDRESS
165					
166	001420	000000	DIR:	0	
167	001422	000000	DIR1:	0	
168	001424	000000	SBR:	0	
169	001426	000000	SBR1:	0	:HIGH BYTE ADDRESS
170					
171	001430	000000	DIDINV:	0	:INPUT INTERREPT VECTOR # 1
172	001432	000000	DIDINS:	0	
173					
174	001434	000000	DIEINV:	0	:INPUT INTERRUPT VECTOR # 2
175	001436	000000	DIEINS:	0	
176					
177	001440	000000	DWARF:	0	:0= NO EDGE CONNECTOR, =1 IN-HOUSE TESTER, =2 WRAP-AROUND
178	001442	000000	TEMP:	0	
179	001444	000000	TEMP1:	0	
180	001446	000000	TEMP2:	0	:RESTART INDICATOR
181	001450	167770	TSTRO:	167770	:IN-HOUSE TESTER ADDRESS
182	001452	167772	TSTR2:	167772	
183	001454	167774	TSTR4:	167774	
184	001456	167776	TSTR6:	167776	
185	001460	000000	MASKNM:	0	:DEVICE MASK
186	001462	000004	VADDR0:	4	:MULTIPLE OUTPUT UNITS ADDRESSES DIFFERENCE
187	001464	000010	VADDR:	10	:MULTIPLE INPUT UNITS, ADDRESS DIFFERENCE
188	001466	000120	VECLST:	AVECT1	:VECTOR FOR UNIT #0
189	001470	000130		AVECT1+10	: #1
190	001472	000140		AVECT1+20	: #2
191	001474	000150		AVECT1+30	: #3
192	001476	000470		AVECT1+350	: #4
193	001500	000460		AVECT1+340	: #5
194	001502	000450		AVECT1+330	: #6
195	001504	000440		AVECT1+320	: #7
196	001506	000000	VECOFF:	0	:VECTOR OFFSET FROM DEFAULT
197	001510	000010		10	: #1
198	001512	000020		20	: #2
199	001514	000030		30	: #3
200	001516	000350		350	: #4
201	001520	000340		340	: #5
202	001522	000330		330	: #6
203	001524	000320		320	: #7
204	001526	000000	EVER:	0	
205	001530	000000	BADUNT:	0	:BAD UNIT INDICATOR
206	001532	000000	UNITBD:	0	
207					
208		010000	BITDAT=	BIT12	:MAINT INPUT INHIBIT
209		004000	BITEXT	BIT11	:INPUT MAINT STROBE

```

212 001534 005237 001446      RESTRT: INC      TEMP2      ;INDICATE RESTART
213 001540 000412              BR      RBEGO
214 001542 005037 001440      BEGIN: CLR      DWARF      ;CLEAR DWARF MODE INDICATOR
215 001546 005037 001446              CLR      TEMP2
216 001552 000405              BR      RBEGO
217 001554 012737 000001 001440  TESTER: MOV     #1,DWARF    ;INDICATE MNC DI IN-HOUSE TESTER MODE
218 001562 005037 001446              CLR      TEMP2
219 001566
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001566 012706 001100      MOV     # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 001572 005026              CLR     (R6)+           ;;CLEAR MEMORY LOCATION
(1) 001574 022706 001140      CMP     #SWR,R6 ;;DONE?
(1) 001600 001374              BNE    -6              ;;LOOP BACK IF NO
(1) 001602 012706 001100      MOV     #STACK,SP      ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 001606 012737 012006 000020  MOV     # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001614 012737 000340 000022  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
(1) 001622 012737 012326 000030  MOV     # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001630 012737 000340 000032  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
(1) 001636 012737 014504 000034  MOV     # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001644 012737 000340 000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
(1) 001652 012737 013624 000024  MOV     # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 001660 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
(1) 001666 013737 010110 010102  MOV     #ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
(1) 001674 005037 001160              CLR     $TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001700 005037 001162              CLR     $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001704 112737 000001 001115  MOVB   #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
(1) 001712 012737 001712 001106  MOV     #,$SLPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001720 012737 001720 001110  MOV     #,$SLPERR       ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001726 013746 000004              MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
(2) 001732 012737 001766 000004  MOV     #64$,@#ERRVEC  ;;SET UP ERROR VECTOR
(2) 001740 012737 177570 001140  MOV     #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001746 012737 177570 001142  MOV     #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001754 022777 177777 177156  CMP     #-1,@SWR       ;;TRY TO REFERENCE HARDWARE SWR
(2) 001762 001012              BNE    66$            ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;;AND THE HARDWARE SWR IS NOT --1
(2) 001764 000403              BR     65$            ;;BRANCH IF NO TIMEOUT
(2) 001766 012716 001774      64$: MOV     #65$,(SP)     ;;SET UP FOR TRAP RETURN
(2) 001772 000002              RTI
(2) 001774 012737 000176 001140  65$: MOV     #SWREG,SWR   ;;POINT TO SOFTWARE SWR
(2) 002002 012737 000174 001142  MOV     #DISPREG,DISPLAY
(2) 002010 012637 000004      66$: MOV     (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002014 005037 001176              CLR     $PASS          ;;CLEAR PASS COUNT
(2) 002020 132737 000200 001211  BITB   #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
(2) 002026 001403              BEQ    67$            ;;YES,USE NON-APT SWITCH
(2) 002030 012737 001212 001140  MOV     # $SWREG,SWR   ;;NO,USE APT SWITCH REGISTER
(2) 002036
67$:

```

```

221 ;ROUTINE TO OVERLAY THE 'STYPE' ROUTINE
222 002036 012737 005046 014146 MOV #5046,$STYPE ;CLR -(SP)
223 002044 012737 012746 014150 MOV #12746,$STYPE+2 ;MOV #STYPE+12,-(SP)
224 002052 012737 014160 014152 MOV #STYPE+12,$STYPE+4
225 002060 012737 000002 014154 MOV #RTI,$STYPE+6 ;RTI
226 002066 004737 010524 JSR PC,$TKINT ;ENABLE TKB INTR.
227 ;SBTTL TYPE PROGRAM NAME
(1) ;:TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002072 005227 177777 INC #-1 ;:FIRST TIME?
(1) 002076 001053 BNE 68$ ;:BRANCH IF NO
(1) 002100 022737 010142 000042 CMP #SENDAD,@#42 ;:ACT-11?
(1) 002106 001447 BEQ 68$ ;:BRANCH IF YES
(1) 002110 104401 002156 TYPE ,69$ ;:TYPE ASCIZ STRING
(2) ;SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002114 005737 000042 TST @#42 ;:ARE WE RUNNING UNDER XXDP/ACT?
(2) 002120 001012 BNE 70$ ;:BRANCH IF YES
(2) 002122 123727 001210 000001 CMPEB $ENV,#1 ;:ARE WE RUNNING UNDER APT?
(2) 002130 001406 BEQ 70$ ;:BRANCH IF YES
(2) 002132 023727 001140 000176 CMP SWR,#SWREG ;:SOFTWARE SWITCH REG SELECTED?
(2) 002140 001005 BNE 71$ ;:BRANCH IF NO
(2) 002142 104407 GTSWR ;:GET SOFT-SWR SETTINGS
(2) 002144 000403 BR 71$
(2) 002146 112737 000001 001134 70$: MOVB #1,$AUTOB ;:SET AUTO-MODE INDICATOR
(2) 002154 71$:
(1) 002154 000424 BR 68$ ;:GET OVER THE ASCIZ
(1) ;:69$: .ASCIZ <CRLF>#CVMNB-A MNC DI (DIGITAL IN) DIAGNOSTIC#<CRLF>
(1) 002226 68$:
228 002226 105737 001134 TSTB $AUTOB ;:TEST IF ACT/XXDP/ACT AUTO MODE
229 002232 001402 BEQ 3$ ;:BR IF NOT
230 002234 000137 002520 JMP LOGIC ;:RUN LOGIC TEST
231 002240 005737 001446 3$: TST TEMP2 ;:TEST IF RESTART
232 002244 001015 BNE MTEST1 ;:BR IF YES
233 002246 104401 020044 TYPE, SETUP0 ;:INFORM OPERATOR ABOUT FRONT SWITCH
234 002252 022737 000001 001440 CMP #1,DWARF ;:TEST IF TESTER
235 002260 001005 BNE MTEST ;:BR IF NOT AND TELL OPERATOR THE HEADER
236 002262 104401 020145 TYPE, SETUP1 ;:INFORM OPERATOR ABOUT TESTER
237 002266 013737 001450 001250 MOV TSTRO,$CDW1 ;:LOAD TESTER DEFAULT FOR WRAPAROUND

```

```

239      .SBTTL  KEYBOARD COMMAND DECODER
240 002274 104401 015046      MTEST: TYPE, PRIME0      ;TELL THE OPER. THE TESTS AVAIL
241 002300 000005      MTEST1: RESEI
242 002302 052777 000100 176634  BIS      #BIT6,@$TKS      ;ENABLE TKB INTR.
243 002310 005037 001176      CLR      $PASS      ;PRIME THE PASS COUNT
244 002314 005037 001112      CLR      $ERTTL      ;PRIME THE TOTAL # OF ERRORS
245 002320 005037 001526      CLR      EVER      ;INIT. THE UNIT TYPEOUT
246 002324 004737 003010      JSR      PC,FIXADR      ;ENSURE BASE AND VECTOR ADDRESS IS LOADED
247 002330 104401 015562      TYPE, DOT      ;INDICATE THE REST POINT
248 002334 104412      RDLIN      ;GET OPER. INPUT
249 002336 013637 002512      MOV      @(SP)+,RUNIT      ;SAVE THE FIRST CHAR.
250 002342 142737 000040 002512  BICB      #40,RUNIT      ;ENSURE UPPER CASE
251 002350 122737 000102 002512  CMPB      #'B,RUNIT      ;TEST IF 'B'
252 002356 001002      BNE      1$      ;BR IF NOT
253 002360 000137 003252      JMP      BASEXC      ;CHANGE INPUT BASE ADDRESS
254 002364 122737 000107 002512  1$: CMPB      #'G,RUNIT      ;TEST IF 'G'
255 002372 001002      BNE      2$      ;BR IF NOT
256 002374 104407      GTSWR      ;GET SWITCH VALUE
257 002376 000740      BR      MTEST1      ;AND RETYPE THE DOT
258 002400 122737 000110 002512  2$: CMPB      #'H,RUNIT      ;TEST IF 'H'
259 002406 001732      BEQ      MTEST      ;BR IF YES
260 002410 122737 000114 002512  CMPB      #'L,RUNIT      ;TEST IF 'L'
261 002416 001005      BNE      3$      ;BR IF NOT
262 002420 042737 000002 001440  BIC      #2,DWARF      ;REMOVE DWARF CONNECTED INDICATOR
263 002426 000137 002520      JMP      LOGIC      ;RUN LOGIC TEST NON-WRAPAROUND
264 002432 122737 000117 002512  3$: CMPB      #'O,RUNIT      ;TEST IF 'O'
265 002440 001002      BNE      4$      ;BR IF NOT
266 002442 000137 003210      JMP      BASEXD      ;GET OUTPUT BASE ADDRESS
267 002446 122737 000124 002512  4$: CMPB      #'T,RUNIT      ;TEST IF 'T'
268 002454 001002      BNE      5$      ;BR IF NOT
269 002456 000137 010262      JMP      DIDATA      ;RUN INPUT SWITCH TYPEOUT
270 002462 122737 000127 002512  5$: CMPB      #'W,RUNIT      ;TEST IF 'W'
271 002470 001005      GNE      77$      ;BR IF NOT
272 002472 012737 000002 001440  MOV      #2,DWARF      ;INDICATE WRAPAROUND MODE
273 002500 000137 002514      JMP      LOGICO      ;RUN LOGIC TEST
274 002504 104401 001164      77$: TYPE, $QUES
275 002510 000673      BR      MTEST1      ;TRY AGAIN
276 002512 000000      RUNIT: 0

```

```

278 .SBTTL DETERMINE THE NUMBER OF MNCDI'S ON THE SYSTEM
279 002514 104401 020220 LOGICO: TYPE, SETUP2 ;TELL OPER. THE CABLE MUST BE THERE
280 002520 013737 001244 001126 LOGIC: MOV $BASE,$BDDAT ;GET BASE ADDRESS
281 002526 005037 001460 CLR MASKNM ;CLR DEVICE MASK
282 002532 005037 001202 CLR $UNIT ;CLR UNIT NUMBER
283 002536 012737 002612 000004 MOV #2$,ERRVEC ;LOAD RETURN ADDRESS
284 002544 005777 176356 1$: TST @SBDDAT ;TEST IF ADDRESS EXISTS
285 002550 063737 001464 001126 ADD VADDR,$BDDAT ;UPDATE BUS ADDRESS
286 002556 005237 001202 INC $UNIT ;UPDATE UNIT COUNT
287 002562 005737 001210 TST $ENV ;TEST IF 'DO NOT SIZE'
288 002566 100423 BMI 3$ ;BR IF NO SIZEING
289 002570 032777 010000 176342 BIT #SW12,@SWR ;TEST IF INHIBIT SIZING IS SET
290 002576 001017 BNE 3$ ;BR IF SET
291 002600 022737 000010 001202 CMP #8,$UNIT ;TEST IF MAX NUMBER
292 002606 001356 BNE 1$ ;BR IF NOT
293 002610 000412 BR 3$ ;BR IF MAX
294 002612 022626 2$: CMP (SP)+,(SP)+ ;RESTORE STACK
295 002614 005737 001202 TST $UNIT ;TEST IF ANY EXIST
296 002620 001006 BNE 3$ ;BR IF ANY ARE THERE
297 002622 005737 000042 TST @#42 ;TEST IF XXDP CHAIN MODE
298 002626 001003 BNE 3$ ;BR IF YES
299 002630 104001 ERROR 1 ;BASE ADDRESS CAUSED A BUS TRAP
300 002632 000137 010054 JMP $EOP
301 002636 012737 014570 000004 3$: MOV #IOTRD,ERRVEC
302 002644 012737 000200 000006 MOV #200,ERRVEC+2
303 002652 005737 001526 TST EVER ;TEST IF # HAS BEEN REPORTED
304 002656 100427 BMI 4$ ;IF YES BRANCH
305 002660 023727 001440 000001 CMP DWARF,#1 ;TEST IF IN TESTER MODE
306 002666 001414 BEQ 6$ ;BR IF TESTER
307 002670 104401 017075 TYPE ,FOUND1 ;TELL OPERATOR # OF MNCDI'S FOUND
308 002674 013746 001202 MOV $UNIT,-(SP) ;PUT # TO BE TYPED ON STACK
309 002700 104405 TYPDS
310 002702 104401 017117 TYPE ,FOUND2 ;FINISH MESSAGE
311 002706 005737 001202 TST $UNIT ;ANY UNITS
312 002712 001002 BNE 6$ ;BR IF SOME
313 002714 000137 010054 JMP $EOP ;REPORT EOP
314 002720 013737 001202 001526 6$: MOV $UNIT,EVER ;SAVE THE # OF MNCDI'S FOR LATER
315 002726 052737 100000 001526 BIS #BIT15,EVER ;SET 'REPORTED # FLAG'
316 002734 000410 BR 5$
317 002736 123737 001526 001202 4$: CMPB EVER,$UNIT ;TEST IF ANY HAVE GONE AWAY
318 002744 001404 BEQ '$ ;BR IF ALL ARE STILL THERE
319 002746 113737 001526 001442 MOVB EVER,TEMP ;SAVE FOR ERROR REPORT
320 002754 104013 ERROR 13 ;EXISTING DEVICE FAILED TO RESPOND
321 002756 005037 001202 5$: CLR $UNIT ;RESET UNIT POINTER
322 002762 004737 003010 JSR PC,FIXADR ;FIX BUS ADDRESSES
323 002766 012737 000001 001460 MOV #BIT0,MASKNM ;LOAD DEVICE MASK
324 002774 005037 001530 CLR BADUNT ;RESET BAD UNIT INDICATOR
325 003000 005046 CLR -(SP) ;LOWER PRIORITY LEVEL 0
326 003002 012746 003424 MOV #TST1,-(SP)
327 003006 000002 RTI

```

```

329                                     ;SUBROUTINE TO FIX DEVICE ADDRESS AND BUS VECTORS
330 003010 012700 001404  FIXADR: MOV #OCSR,R0 ;LOAD ADDRESS POINTER
331 003014 013701 001250  MOV $CDW1,R1 ;LOAD INITIAL BUS ADDRESS
332 003020 010120 1$: MOV R1,(R0)+ ;LOAD DEVICE ADDRESS
333 003022 005201 INC R1 ;UPDATE BUS ADDRESS VALUE
334 003024 020027 001414 CMP RO,#ICSR ;TEST IF DONE WITH BUS ADDRESSES
335 003030 001373 BNE 1$ ;BR IF NOT
336 003032 013701 001244 MOV $BASE,R1 ;LOAD INITIAL INPUT BUS ADDRESS
337 003036 010120 2$: MOV R1,(R0)+ ;LOAD THE ADDRESS
338 003040 005201 INC R1 ;UPDATE THE ADDRESS
339 003042 020027 001430 CMP RO,#DIDINV ;TEST IF AT END
340 003046 001373 BNE 2$ ;BRANCH IF NOT
341 003050 013701 001240 MOV $VECT1,R1 ;LOAD INITIAL INPUT VECTOR
342 003054 010120 4$: MOV R1,(R0)+ ;LOAD THE VECTOR
343 003056 005721 TST (R1)+ ;BUMP THE ADDRESS
344 003060 020027 001440 CMP RO,#DIEINS+2 ;TEST IF DONE
345 003064 001373 BNE 4$ ;BRANCH IF NOT
346 003066 012700 001466 MOV #VECLST,R0 ;GET ACTUAL VECTOR AREA
347 003072 012701 001506 MOV #VECOFF,R1 ;GET VECTOR OFFSET POINTER
351 003076 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003102 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
(1) 003104 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003110 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
(1) 003112 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003116 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
(1) 003120 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003124 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
(1) 003126 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003132 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
(1) 003134 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003140 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
(1) 003142 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003146 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
(1) 003150 013710 001240 MOV $VECT1,(R0) ;GET BASE
(1) 003154 062120 ADD (R1)+,(R0)+ ;ADD OFFSET
352
353                                     ;ALSO TO LOAD INTELLIGENT TRAP CATCHER
354 003156 012700 000250 MOV #250,R0 ;LOAD FIRST ADDRESS OF TRAP CATCHER
355 003162 012701 000252 MOV #252,R1 ;LOAD .+2
356 003166 012702 004700 MOV #4700,R2 ;LOAD ILLEGAL INST.
357 003172 010120 5$: MOV R1,(R0)+ ;LOAD .+2
358 003174 010220 MOV R2,(R0)+ ;LOAD ILLEGAL INST.
359 003176 022121 CMP (R1)+,(R1)+ ;BUMP R1 TWICE
360 003200 020027 001000 CMP RO,#1000 ;TEST IF DONE
361 003204 001372 BNE 5$ ;BRANCH IF NOT
362 003206 000207 RTS PC

```

```

364                                     ;SUBROUTINE TO ASK THE OPERATOR FOR OUT ADRS
365 003210 104401 BASEXD: TYPE
366 003212 020320 ADROUT                                     ;ASK FOR OUTPUT ADDR.
367 003214 013746 001250 MOV $CDW1,-(SP)           ;GET DEFAULT VALUE
368 003220 104402 TYPOC                                     ;TELL OPER.
369 003222 104401 020475 TYPE, ENDOUT                 ;ADD END
370 003226 104413 RDOCT                                     ;AND WAIT FOR INPUT
371 003230 005726 TST (SP)+                             ;WAS IT A <CR> FOR DEFAULT
372 003232 001403 BEQ 1$                                ;YES- BRANCH
373 003234 016637 177776 001250 MOV -2(SP),$CDW1       ;NO-LOAD NEW ADDR.
374 003242 004737 003010 1$: JSR PC,FIXADR             ;LOAD NEW ADDRESSES
375 003246 000137 002300 JMP MTEST1                ;RETURN
376
377                                     ;SUBROUTINE TO ASK FOR INPUT ADRS AND VEC
378 003252 104401 BASEXC: TYPE
379 003254 020364 ADRIN                                     ;ASK FOR INPUT ADDR.
380 003256 013746 001244 MOV $BASE,-(SP)           ;GET DEFAULT
381 003262 104402 TYPOC                                     ;TELL OPER. DEFAULT
382 003264 104401 020475 TYPE, ENDOUT                 ;ADD END
383 003270 104413 RDOCT                                     ;AND WAIT FOR INPUT
384 003272 005726 TST (SP)+                             ;WAS IT A <CR> FOR DEFAULT
385 003274 001403 BEQ 1$                                ;YES-BRANCH
386 003276 016637 177776 001244 MOV -2(SP),$BASE       ;NO-LOAD NEW ADDR
387 003304 104401 1$: TYPE
388 003306 020427 VECIN                                     ;ASK FOR INPUT VECTOR
389 003310 013746 001240 MOV $VECT1,-(SP)           ;GET DEFAULT
390 003314 104402 TYPOC                                     ;TELL OPER THE DEFAULT
391 003316 104401 020475 TYPE, ENDOUT                 ;ADD END
392 003322 104413 RDOCT                                     ;AND WAIT FOR INPUT
393 003324 005726 TST (SP)+                             ;WAS IT A <CR> FOR DEFAULT
394 003326 001403 BEQ 2$                                ;YES BRANCH
395 003330 016637 177776 001240 MOV -2(SP),$VECT1     ;LOAD NEW VECTOR
396 003336 004737 003010 2$: JSR PC,FIXADR             ;FIX ADDRESSES AND VECTORS
397 003342 000137 002300 JMP MTEST1                ;AND RETYPE THE DOT
398
399 .SBTTL SUBROUTINE TO HANDLE CONTROL C/G
400
401 003346 105777 175572 CTRLCG: TSTB @$TKS             ;INPUT FLAG ?
402 003352 100022 BPL 2$                                ;BR IF NOT
403 003354 017737 175566 003422 MOV @$TKB,CTRCHA       ;READ CHAR.
404 003362 042737 177640 003422 BIC #177640,CTRCHA     ;MASK OFF BITS
405 003370 022737 000003 003422 CMP #3,CTRCHA         ;TEST IF CONTROL C
406 003376 001003 BNE 1$                                ;BR IF NOT
407 003400 005726 TST (SP)+                             ;CLEAN STACK
408 003402 000137 002300 JMP MTEST1                ;AND RETYPE THE DOT
409 003406 022737 000007 003422 1$: CMP #7,CTRCHA     ;TEST IF CTRL G
410 003414 001001 BNE 2$                                ;BR IF NOT
411 003416 104407 GTSWR                                     ;GET SWITCHES
412 003420 000207 2$: RTS PC                             ;EXIT
413 003422 000000 CTRLCHA: 0 ;CHAR. THE OPER TYPED

```



```
420
(3)
(3)
(2) 003424 000004
421 003426 022737 000001 001440
422 003434 001020
423 003436 005077 176010
424 003442 017737 176006 001126
425 003450 042737 177417 001126
426 003456 012737 000140 001124
427 003464 023737 001124 001126
428 003472 001401
429 003474 104004
430
(3)
(3)
(2) 003476 000004
431 003500 012737 003524 000004
432 003506 005777 175702
433 003512 005777 175702
434 003516 005777 175702
435 003522 000411
436 003524 104005
437 003526 012737 014570 000004
438 003534 012737 000200 000006
439 003542 000137 007700
440 003546 012737 014570 000004
441 003554 012737 000200 000006
442
(4)
(4)
(3) 003562 000004
(1) 003564 012737 000001 001124
(1) 003572 012737 003600 001106
(1) 003600 013777 001124 175616
(1) 003606 017737 175612 001126
(1) 003614 023737 001124 001126
(2) 003622 001401
(1) 003624 104006
(1) 003626 006337 001124
(1) 003632 001362

:*****
:*TEST 1 VERIFY CORRECT I.D. CODE FOR MNCDI (IN-HOUSE TESTER)
:*****
TST1: SCOPE
CMP #1,DWARF ;TEST IF 'IN-HOUSE TESTER' MODE
BNE TST2 ;:BR IF NOT
CLR @TSTR2 ;ENSURE TESTER MODE
MOV @TSTR4,$BDDAT ;READ I.D. VALUE
BIC #177417,$BDDAT ;MASK TO OTHER BITS
MOV #140,$GDDAT ;LOAD EXPECTED I.D. VALUE
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST2 ;:BR IF SAME
ERROR 4 ;INCORRECT I.D. VALUE FOR MNCDI
:*****
:*TEST 2 VERIFY A MNCDI BUS ADDRESS RESPONSE
:*****
TST2: SCOPE
MOV #1$,ERRVEC ;LOAD BUS TRAP VECTOR
TST @ICSR ;TEST INPUT STATUS
TST @DIR ;TEST INPUT DATA REGISTER
TST @SBR ;TEST STIM. BUFFER REGISTER
BR 2$ ;:BR IF NO TIMEOUT
1$: ERROR 5 ;BUS TIMEOUT WHEN REFERENCING THE MNCDI
MOV #IOTRD,ERRVEC ;RESTORE TRAP
MOV #200,ERRVEC+2 ;VECTOR
JMP REMAIN ;CHECK FOR MORE UNITS
2$: MOV #IOTRD,ERRVEC ;RESTORE TRAP VECTOR
MOV #200,ERRVEC+2
:*****
:*TEST 3 FLOAT A 1 ACROSS THE MNCDI STIMULUS BIT REGISTER
:*****
TST3: SCOPE
MOV #BIT0,$GDDAT ;LOAD EXPECTED BIT
MOV #1$, $LPADR ;LOAD LOOP ADDRESS
1$: MOV $GDDAT,@SBR ;LOAD MNCDI STIMULUS BIT REGISTER
MOV @SBR,$BDDAT ;READ MNCDI STIMULUS BIT REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 2$ ;:BR IF EXPECTED
ERROR 6 ; MNCDI STIMULUS BIT REGISTER FAILED TO HOLD A FLOATING
1$: ASL $GDDAT ;CHANGE THE DATA
BNE 1$ ;BR IF MORE DATA
```

```
444
(4)
(4)
(3) 003634 000004
(1) 003636 012737 000001 001442
(1) 003644 012737 003652 001106
(1) 003652 013737 001442 001124
(1) 003660 005137 001124
(1) 003664 013777 001124 175532
(1) 003672 017737 175526 001126
(1) 003700 023737 001124 001126
(2) 003706 001401
(1) 003710 104006
(1) 003712 006337 001442
(1) 003716 001355

*****
*TEST 4      FLOAT A 0 ACROSS THE MNCDI STIMULUS BIT REGISTER
*****
TST4:  SCOPE
      MOV    #BIT0,TEMP      ;LOAD INITIAL BIT
      MOV    #1$, $LPADR     ;LOAD LOOP ADDRESS
1$:   MOV    TEMP,$GDDAT     ;LOAD EXPECTED
      COM    $GDDAT         ;COMPLEMENT
      MOV    $GDDAT,@SBR     ;LOAD MNCDI STIMULUS BIT REGISTER
      MOV    @SBR,$BDDAT     ;READ MNCDI STIMULUS BIT REGISTER
      CMP    $GDDAT,$BDDAT   ;COMPARE
      BEQ    2$             ;;BR IF EXPECTED
      ERROR  6              ;MNCDI STIMULUS BIT REGISTER FAILED TO HOLD A FLOATING 0
2$:   ASL    TEMP           ;CHANGE THE DATA
      BNE    1$            ;BR IF MORE DATA
*****

445
(4)
(4)
(3) 003720 000004
(2) 003722 012737 000040 001160
(1) 003730 012777 177777 175466
(1) 003736 005037 001124
(1) 003742 000005
(1) 003744 052777 000100 175172
(1) 003752 017737 175446 001126
(3) 003760 001401
(1) 003762 104006

*****
*TEST 5      ENSURE THAT 'RESET' CLEARS THE MNCDI STIMULUS BIT REGISTER
*****
TST5:  SCOPE
      MOV    #40,$TIMES     ;;DO 40 ITERATIONS
      MOV    #-1,@SBR       ;LOAD BITS TO BE RESET
      CLR    $GDDAT         ;CLEAR EXPECTED
      RESET                ;CLEAR THE DEVICE
      BIS    #BIT6,@$TKS    ;ENABLE TKB INTR.
      MOV    @SBR,$BDDAT    ;READ MNCDI STIMULUS BIT REGISTER
      BEQ    TST6           ;;BR IF CLEARED
      ERROR  6              ;MNCDI STIMULUS BIT REGISTER FAILED TO CLEAR WITH 'RESET'
*****

446
(4)
(4)
(3) 003764 000004
(1) 003766 012737 003774 001106
(1) 003774 012777 177777 175422
(1) 004002 012737 000377 001124
(1) 004010 105077 175412
(1) 004014 017737 175404 001126
(1) 004022 023737 001124 001126
(2) 004030 001404
(1) 004032 104006
(1) 004034 012737 004042 001106
(1) 004042 012777 177777 175354
(1) 004050 012737 177400 001124
(1) 004056 105077 175342
(1) 004062 017737 175336 001126
(1) 004070 023737 001124 001126
(2) 004076 001401
(1) 004100 104006
(1) 004102

*****
*TEST 6      VERIFY BYTE OPERATION ON THE MNCDI STIMULUS BIT REGISTER
*****
TST6:  SCOPE
      MOV    #1$, $LPADR     ;LOAD RETURN ADDRESS
1$:   MOV    #-1,@SBR       ;LOAD MNCDI STIMULUS BIT REGISTER
      MOV    #377,$GDDAT    ;LOAD EXPECTED
      CLRB  @SBR1          ;CLEAR HIGH BYTE
      MOV    @SBR,$BDDAT    ;READ MNCDI STIMULUS BIT REGISTER
      CMP    $GDDAT,$BDDAT  ;COMPARE
      BEQ    2$             ;;BR IF SAME
      ERROR  6              ;CLEARING HIGH BYTE CHANGED LOW BYTE
2$:   MOV    #2$, $LPADR     ;LOAD LOOP RETURN
      MOV    #-1,@SBR       ;LOAD MNCDI STIMULUS BIT REGISTER
      MOV    #177400,$GDDAT ;LOAD EXPECTED
      CLRB  @SBR           ;CLEAR LOW BYTE
      MOV    @SBR,$BDDAT    ;READ MNCDI STIMULUS BIT REGISTER
      CMP    $GDDAT,$BDDAT  ;COMPARE
      BEQ    3$             ;;BR IF SAME
      ERROR  6              ;CLEARING LOW BYTE CHANGED HIGH BYTE
3$:
```

```
448
(4)
(4)
(3) 004102 000004
(1) 004104 012737 000002 001124
(1) 004112 013777 001124 175274
(1) 004120 017737 175270 001126
(1) 004126 023737 001124 001126
(2) 004134 001401
(1) 004136 104007
(1)
(1) 004140 043777 001124 175246
(1) 004146 017737 175242 001126
(1) 004154 023737 001124 001126
(3) 004162 001001
(1) 004164 104007

*****
*TEST 7 TEST THAT BIT1 OF MNCDI STATUS REGISTER IS READ-WRITE
*****
TST7: SCOPE
MOV #BIT1,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@ICSR ;LOAD BIT1 INTO MNCDI STATUS REGISTER
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 7 ;BIT1 OF MNCDI STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST10 ;;BR IF CLEARED
ERROR 7 ;BIT1 OF MNCDI STATUS REGISTER FAILED TO CLEAR

449
(4)
(4)
(3) 004166 000004
(1) 004170 012737 000004 001124
(1) 004176 013777 001124 175210
(1) 004204 017737 175204 001126
(1) 004212 023737 001124 001126
(2) 004220 001401
(1) 004222 104007
(1)
(1) 004224 043777 001124 175162
(1) 004232 017737 175156 001126
(1) 004240 023737 001124 001126
(3) 004246 001001
(1) 004250 104007

*****
*TEST 10 TEST THAT BIT2 OF MNCDI STATUS REGISTER IS READ-WRITE
*****
TST10: SCOPE
MOV #BIT2,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@ICSR ;LOAD BIT2 INTO MNCDI STATUS REGISTER
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 7 ;BIT2 OF MNCDI STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST11 ;;BR IF CLEARED
ERROR 7 ;BIT2 OF MNCDI STATUS REGISTER FAILED TO CLEAR

450
(4)
(4)
(3) 004252 000004
(1) 004254 012737 000010 001124
(1) 004262 013777 001124 175124
(1) 004270 017737 175120 001126
(1) 004276 023737 001124 001126
(2) 004304 001401
(1) 004306 104007
(1)
(1) 004310 043777 001124 175076
(1) 004316 017737 175072 001126
(1) 004324 023737 001124 001126
(3) 004332 001001
(1) 004334 104007

*****
*TEST 11 TEST THAT BIT3 OF MNCDI STATUS REGISTER IS READ-WRITE
*****
TST11: SCOPE
MOV #BIT3,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@ICSR ;LOAD BIT3 INTO MNCDI STATUS REGISTER
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 7 ;BIT3 OF MNCDI STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST12 ;;BR IF CLEARED
ERROR 7 ;BIT3 OF MNCDI STATUS REGISTER FAILED TO CLEAR
```

```
452
(4)
(4)
(3) 004336 000004
(1) 004340 012737 000020 001124
(1) 004346 013777 001124 175040
(1) 004354 017737 175034 001126
(1) 004362 023737 001124 001126
(2) 004370 001401
(1) 004372 104007
(1)
(1) 004374 043777 001124 175012
(1) 004402 017737 175006 001126
(1) 004410 023737 001124 001126
(3) 004416 001001
(1) 004420 104007
453
(4)
(4)
(3) 004422 000004
(1) 004424 012737 000040 001124
(1) 004432 013777 001124 174754
(1) 004440 017737 174750 001126
(1) 004446 023737 001124 001126
(2) 004454 001401
(1) 004456 104007
(1)
(1) 004460 043777 001124 174726
(1) 004466 017737 174722 001126
(1) 004474 023737 001124 001126
(3) 004502 001001
(1) 004504 104007
454
(4)
(4)
(3) 004506 000004
(1) 004510 012737 000100 001124
(1) 004516 013777 001124 174670
(1) 004524 017737 174664 001126
(1) 004532 023737 001124 001126
(2) 004540 001401
(1) 004542 104007
(1)
(1) 004544 043777 001124 174642
(1) 004552 017737 174636 001126
(1) 004560 023737 001124 001126
(3) 004566 001001
(1) 004570 104007

::*****
:*TEST 12 TEST THAT BIT4 OF MNCDI STATUS REGISTER IS READ-WRITE
::*****
TST12: SCOPE
MOV #BIT4,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@ICSR ;LOAD BIT4 INTO MNCDI STATUS REGISTER
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 7 ;BIT4 OF MNCDI STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST13 ;;BR IF CLEARED
ERROR 7 ;BIT4 OF MNCDI STATUS REGISTER FAILED TO CLEAR

::*****
:*TEST 13 TEST THAT BITS OF MNCDI STATUS REGISTER IS READ-WRITE
::*****
TST13: SCOPE
MOV #BITS,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@ICSR ;LOAD BITS INTO MNCDI STATUS REGISTER
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 7 ;BITS OF MNCDI STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST14 ;;BR IF CLEARED
ERROR 7 ;BITS OF MNCDI STATUS REGISTER FAILED TO CLEAR

::*****
:*TEST 14 TEST THAT BIT6 OF MNCDI STATUS REGISTER IS READ-WRITE
::*****
TST14: SCOPE
MOV #BIT6,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@ICSR ;LOAD BIT6 INTO MNCDI STATUS REGISTER
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 7 ;BIT6 OF MNCDI STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST15 ;;BR IF CLEARED
ERROR 7 ;BIT6 OF MNCDI STATUS REGISTER FAILED TO CLEAR
```

```
456
(4)
(4)
(3) 004572 000004
(1) 004574 012737 000400 001124
(1) 004602 013777 001124 174604
(1) 004610 017737 174600 001126
(1) 004616 023737 001124 001126
(2) 004624 001401
(1) 004626 104007
(1)
(1) 004630 043777 001124 174556 1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
(1) 004636 017737 174552 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
(1) 004644 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THE BIT
(3) 004652 001001 BNE TST16 ;;BR IF CLEARED
(1) 004654 104007 ERROR 7 ;BIT8 OF MNCDI STATUS REGISTER FAILED TO CLEAR
457
(4)
(4)
(3) 004656 000004
(1) 004660 012737 001000 001124
(1) 004666 013777 001124 174520
(1) 004674 017737 174514 001126
(1) 004702 023737 001124 001126
(2) 004710 001401
(1) 004712 104007
(1)
(1) 004714 043777 001124 174472 1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
(1) 004722 017737 174466 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
(1) 004730 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THE BIT
(3) 004736 001001 BNE TST17 ;;BR IF CLEARED
(1) 004740 104007 ERROR 7 ;BIT9 OF MNCDI STATUS REGISTER FAILED TO CLEAR
458
(4)
(4)
(3) 004742 000004
(1) 004744 012737 010000 001124
(1) 004752 013777 001124 174434
(1) 004760 017737 174430 001126
(1) 004766 023737 001124 001126
(2) 004774 001401
(1) 004776 104007
(1)
(1) 005000 043777 001124 174406 1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
(1) 005006 017737 174402 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
(1) 005014 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THE BIT
(3) 005022 001001 BNE TST20 ;;BR IF CLEARED
(1) 005024 104007 ERROR 7 ;BIT12 OF MNCDI STATUS REGISTER FAILED TO CLEAR
```

```

460      ::*****
(4)      :*TEST 20      TEST THAT BIT14 OF MNCDI STATUS REGISTER IS READ-WRITE
(4)      :*****
(3) 005026 000004 TST20: SCOPE
(1) 005030 012737 040000 001124      MOV      #BIT14,$GDDAT      ;LOAD EXPECTED
(1) 005036 013777 001124 174350      MOV      $GDDAT,@ICSR      ;LOAD BIT14 INTO MNCDI STATUS REGISTER
(1) 005044 017737 174344 001126      MOV      @ICSR,$BDDAT      ;READ MNCDI STATUS REGISTER
(1) 005052 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;TEST THAT IT SET
(2) 005060 001401      BEQ      1$                  ;;BR IF SET
(1) 005062 104007      ERROR    7                  ;BIT14 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1) 005064 043777 001124 174322 1$:  BIC      $GDDAT,@ICSR      ;CLEAR THAT BIT
(1) 005072 017737 174316 001126      MOV      @ICSR,$BDDAT      ;READ MNCDI STATUS REGISTER AGAIN
(1) 005100 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;TEST THE BIT
(3) 005106 001001      BNE      TST21              ;;BR IF CLEARED
(1) 005110 104007      ERROR    7                  ;BIT14 OF MNCDI STATUS REGISTER FAILED TO CLEAR
461      ::*****
(4)      :*TEST 21      ENSURE THAT 'RESET' CLEARS THE MNCDI STATUS REGISTER
(4)      :*****
(3) 005112 000004 TST21: SCOPE
(2) 005114 012737 000040 001160      MOV      #40,$TIMES        ;;DO 40 ITERATIONS
(1) 005122 012777 040426 174264      MOV      #40426,@ICSR      ;LOAD BITS TO BE RESET
(1) 005130 005037 001124      CLR      $GDDAT            ;CLEAR EXPECTED
(1) 005134 000005      RESET                      ;CLEAR THE DEVICE
(1) 005136 052777 000100 174000      BIS      #BIT6,@$TKS        ;ENABLE TKB INTR.
(1) 005144 017737 174244 001126      MOV      @ICSR,$BDDAT      ;READ MNCDI STATUS REGISTER
(3) 005152 001401      BEQ      TST22              ;;BR IF CLEARED
(1) 005154 104007      ERROR    7                  ;MNCDI STATUS REGISTER FAILED TO CLEAR WITH 'RESET'
462      ::*****
(3)      :*TEST 22      VERIFY HIGH BYTE OPERATION ON THE INPUT STATUS REGISTER
(3)      :*****
(2) 005156 000004 TST22: SCOPE
463 005160 012777 040426 174226      MOV      #40426,@ICSR      ;LOAD INPUT REG. BIT
464 005166 105077 174224      CLRB     @ICSR1            ;CLEAR HIGH BYTE
465 005172 012737 000026 001124      MOV      #BIT4!BIT2!BIT1,$GDDAT ;LOAD EXPECTED
466 005200 017737 174210 001126      MOV      @ICSR,$BDDAT      ;READ INPUT STATUS REG.
467 005206 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
468 005214 001401      BEQ      TST23              ;;BR IF SAME
469 005216 104007      ERROR    7                  ;CLERING HIGH BYTE CHANGED LOW BYTE
470
471      ::*****
(3)      :*TEST 23      VERIFY LOW BYTE OPERATION ON THE INPUT STATUS REGISTER
(3)      :*****
(2) 005220 000004 TST23: SCOPE
472 005222 012777 040426 174164      MOV      #40426,@ICSR      ;LOAD INPUT REG.
473 005230 105077 174160      CLRB     @ICSR            ;CLEAR LOW BYTE
474 005234 012737 040400 001124      MOV      #40400,$GDDAT      ;LOAD EXPECTED
475 005242 017737 174146 001126      MOV      @ICSR,$BDDAT      ;READ INPUT STATUS REG.
476 005250 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
477 005256 001401      BEQ      TST24              ;;BR IF SAME
478 005260 104007      ERROR    7                  ;CLEARING LOW BYTE CHANGED HIGH BYTE
479

```

```

481                                     ::*****
(3)                                     ::*TEST 24      VERIFY THAT MAINT. STROBE SETS 'INPUT DATA READY'
(3)                                     ::*****
(2) 005262 000004                       TST24: SCOPE
482 005264 005077 174124                 CLR      @ICSR                      ;ENSURE CLEAR FLAG
483 005270 012737 000200 001124         MOV      #BIT7,$GDDAT              ;LOAD EXPECTED DATA
484 005276 012777 004200 174110         MOV      #BITEXT!BIT7,@ICSR       ;GENERATE MAINT. STROBE
485 005304 017737 174104 001126         MOV      @ICSR,$BDDAT             ;READ INPUT STATUS REGISTER
486 005312 023737 001124 001126         CMP      $GDDAT,$BDDAT           ;COMPARE RESULTS
487 005320 001401                       BEQ      TST25                    ;;BR IF SAME
488 005322 104007                       ERROR   7                        ;MAINT. STROBE FAILED TO SET 'INPUT DATA READY'
489
490                                     ::*****
(3)                                     ::*TEST 25      VERIFY THAT 'INPUT DATA READY' CAN BE WRITTEN TO A ZERO
(3)                                     ::*****
(2) 005324 000004                       TST25: SCOPE
491 005326 005037 001124                 CLR      $GDDAT                   ;LOAD EXPECTED DATA
492 005332 012777 004000 174054         MOV      #BITEXT,@ICSR           ;GENERATE MAINT. STROBE
493 005340 005077 174050                 CLR      @ICSR                   ;CLEAR DATA READY FLAG
494 005344 017737 174044 001126         MOV      @ICSR,$BDDAT           ;READ INPUT STATUS REGISTER
495 005352 023737 001124 001126         CMP      $GDDAT,$BDDAT         ;COMPARE
496 005360 001401                       BEQ      TST26                    ;;BR IF SAME
497 005362 104007                       ERROR   7                        ;'INPUT DATA READY' FAILED TO BE WRITTEN TO A ZERO
498
499                                     ::*****
(3)                                     ::*TEST 26      VERIFY THAT 'INPUT DATA READY' CAN BE CLEARED BY A 'RESET'
(3)                                     ::*****
(2) 005364 000004                       TST26: SCOPE
(1) 005366 012737 000040 001160         MOV      #40,$TIMES              ;;DO 40 ITERATIONS
500 005374 005037 001124                 CLR      $GDDAT                   ;LOAD EXPECTED DATA
501 005400 012777 004000 174006         MOV      #BITEXT,@ICSR           ;GENERATE MAINT. STROBE
502 005406 000005                       RESET
503 005410 052777 000100 173526         BIS      #BIT6,@$TKS             ;ENABLE TKB INTR.
504 005416 017737 173772 001126         MOV      @ICSR,$BDDAT           ;READ INPUT STATUS REGISTER
505 005424 023737 001124 001126         CMP      $GDDAT,$BDDAT         ;COMPARE
506 005432 001401                       BEQ      TST27                    ;;BR IF CLEARED
507 005434 104007                       ERROR   7                        ;'INPUT DATA READY' FAILED TO BE CLEARED BY 'RESET'
508
509                                     ::*****
(3)                                     ::*TEST 27      'INPUT DATA READY' WILL NOT SET IF IN STIMILUS MODE AND NO SBR MATCH
(3)                                     ::*****
(2) 005436 000004                       TST27: SCOPE
510 005440 005077 173760                 CLR      @SBR                    ;CLEAR SBR REGISTER
511 005444 012777 177777 173746         MOV      #-1,@DIR                ;CLEAR INPUT REGISTER
512 005452 012777 000004 173734         MOV      #BIT2,@ICSR            ;SET STILILUS MODE
513 005460 052777 004000 173726         BIS      #BITEXT,@ICSR          ;GENERATE MAINT. STROBE
514 005466 012737 000004 001124         MOV      #BIT2,$GDDAT           ;LOAD EXPECTED
515 005474 017737 173714 001126         MOV      @ICSR,$BDDAT          ;READ STATUS
516 005502 023737 001124 001126         CMP      $GDDAT,$BDDAT         ;COMPARE
517 005510 001401                       BEQ      TST30                    ;;BR IF CLEARED
518 005512 104007                       ERROR   7                        ;INPUT STROBE SET INPUT READY WHEN IN STIMILUS MODE

```

```

520
(3)
(3)
(2) 005514 000004
521 005516 012737 100202 001124
522 005524 012777 000002 173662
523 005532 052777 004200 173654
524 005540 052777 104200 173646
525 005546 017737 173642 001126
526 005554 023737 001124 001126
527 005562 001401
528 005564 104007
529
530
(3)
(3)
(2) 005566 000004
531 005570 012737 000202 001124
532 005576 012777 000002 173610
533 005604 052777 004200 173602
534 005612 052777 104200 173574
535 005620 105077 173572
536 005624 017737 173564 001126
537 005632 023737 001124 001126
538 005640 001401
539 005642 104007
540
541
(3)
(3)
(2) 005644 000004
(1) 005646 012737 000040 001160
542 005654 005037 001124
543 005660 012777 000002 173526
544 005666 052777 004000 173520
545 005674 052777 004000 173512
546 005702 000005
547 005704 052777 000100 173232
548 005712 017737 173476 001126
549 005720 023737 001124 001126
550 005726 001401
551 005730 104007

*****
*TEST 30 VERIFY THAT 'OVERRUN ERROR' SETS
*****
TST30: SCOPE
MOV #BIT15!BIT7!BIT1,$GDDAT ;LOAD EXPECTED
MOV #BIT1,@ICSR ;SET STROBE MODE
BIS #BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE
BIS #BIT15!BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE AGAIN
MOV @ICSR,$BDDAT ;READ INPUT STATUS REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST31 ;;BR IF SAME
ERROR 7 ;'OVER RUN' FAILED TO SET

*****
*TEST 31 VERIFY THAT 'OVERRUN ERROR' CAN BE WRITTEN TO A ZERO
*****
TST31: SCOPE
MOV #BIT7!BIT1,$GDDAT ;LOAD EXPECTED VALUE
MOV #BIT1,@ICSR ;SET STROBE MODE
BIS #BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE
BIS #BIT15!BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE AGAIN
CLRB @ICSR1 ;CLEAR HIGH BYTE OF THE INPUT STATUS REGISTER
MOV @ICSR,$BDDAT ;READ INPUT STATUS REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST32 ;;BR IF SAME
ERROR 7 ;'OVERRUN ERROR' FAILED TO BE WRITTEN TO A ZERO

*****
*TEST 32 VERIFY THAT 'RESET' CLEARS 'OVERRUN ERROR'
*****
TST32: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
CLR $GDDAT ;CLEAR EXPECTED
MOV #BIT1,@ICSR ;SET STROBE MODE
BIS #BITEXT,@ICSR ;GENERATE MAINT. STROBE
BIS #BITEXT,@ICSR ;GENERATE MAINT. STROBE AGAIN
RESET
BIS #BIT6,@$TKS ;ENABLE TKB INTR.
MOV @ICSR,$BDDAT ;READ INPUT REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST33 ;;BR IF SAME
ERROR 7 ;RESET FAILED TO CLEAR 'OVERRUN ERROR'

```



```

553          ::*****
(3)          :*TEST 33      VERIFY INVERT DATA FUNCTION
(3)          :*****
(2) 005732 000004 TST33: SCOPE
554 005734 005037 001124      CLR      $GDDAT      ;LOADEXPECTED
555 005740 012777 010000 173446  MOV     #BITDAT,@ICSR ;SET INPUT INHIBIT
556 005746 017737 173446 001126  MOV     @DIR,$BDDAT  ;READ INPUT
557 005754 023737 001124 001126  CMP     $GDDAT,$BDDAT ;COMPARE
558 005762 001401          BEQ     1$           ;;BR IF SAME
559 005764 104007          ERROR    7           ;INPUT INHIBIT FAILED TO INHIBIT INPUT
560 005766 012777 010060 173420 1$:  MOV     #BITDAT!BIT5!BIT4,@ICSR ;SET INVERT DATA AND INPUT INHIBIT
561 005774 012737 177777 001124  MOV     #-1,$GDDAT   ;LOAD EXPECTED
562 006002 017737 173412 001126  MOV     @DIR,$BDDAT  ;READ INPUT
563 006010 001001          BNE     2$           ;BR IF NON-ZERO
564 006012 104007          ERROR    7           ;INVERT DATA FUNCTION FAILED
565 006014 023737 001124 001126 2$:  CMP     $GDDAT,$BDDAT ;COMPARE DATA
566 006022 001401          BEQ     3$           ;;BR IF SAME
567 006024 104007          ERROR    7           ;INVERT DATA - DATA PATH ERROR
568
569 006026 012777 010000 173360 3$:  MOV     #BITDAT,@ICSR ;SET INPUT INHIBIT
570 006034 005037 001124          CLR     $GDDAT      ;CLEAR EXPECTED
571 006040 017737 173354 001126  MOV     @DIR,$BDDAT  ;READ INPUT
572 006046 023737 001124 001126  CMP     $GDDAT,$BDDAT ;COMPARE
573 006054 001401          BEQ     TST34       ;;BR IF SAME
574 006056 104007          ERROR    7           ;INVERT DATA FUNCTION OR INPUT INHIBIT FAILED
575

```

```

576          ::*****
(3)          :*TEST 34      VERIFY EACH BIT OF THE MNC DI INPUT DATA REGISTER CAN BE CLEARED
(3)          :*****
(2) 006060 000004 TST34: SCOPE
577 006062 012737 006076 001110  MOV     #1$,$LPERR   ;LOAD LOOP ADDRESS ON ERROR
578 006070 012737 000001 001442  MOV     #BIT0,TEMP    ;LOAD INITIAL BIT
579 006076 012777 010060 173310 1$:  MOV     #BITDAT!BIT4!BIT5,@ICSR ;LOAD INHIBIT INPUT AND INVERT DATA
580 006104 013737 001442 001124  MOV     TEMP,$GDDAT  ;LOAD EXPECTED
581 006112 005137 001124          COM     $GDDAT      ;MAKE OPPOSITE
582 006116 017700 173276          MOV     @DIR,R0     ;READ INPUT
583 006122 013777 001442 173270  MOV     TEMP,@DIR    ;CLEAR THE INPUT BIT
584 006130 042777 000040 173256  BIC     #BIT5,@ICSR  ;REM INVERT DATA BITOVE
585 006136 052777 000002 173250  BIS     #BIT1,@ICSR  ;ENABLE EXT. STROBE TO PREVENT DATA INPUT BEING
586 006144 017737 173250 001126  MOV     @DIR,$BDDAT  ;READ INPUT REG.
587 006152 023737 001124 001126  CMP     $GDDAT,$BDDAT ;COMPARE
588 006160 001401          BEQ     2$           ;;BR IF SAME
589 006162 104010          ERROR    10        ;INPUT REGISTER BIT FAILED TO CLEAR
590 006164 006337 001442 2$:  ASL     TEMP        ;SHIFT THE DATA
591 006170 001342          BNE     1$         ;TRY MORE BITS

```

593
(3)
(3)
(2) 006172 000004
(1) 006174 012737 000040 001160
594 006202 012777 004060 173204
595 006210 017737 173204 001442
596 006216 005037 001124
597 006222 000005
598 006224 052777 000100 172712
599 006232 052777 000004 173154
600 006240 017737 173154 001126
601 006246 001401
602 006250 104010
603
604
(3)
(3)
(2) 006252 000004
605 006254 012777 010062 173132
606 006262 052777 004200 173124
607 006270 042777 000040 173116
608 006276 052777 004200 173110
609 006304 012737 177777 001124
610 006312 017737 173102 001126
611 006320 023737 001124 001126
612 006326 001401
613 006330 104010
614

```
*****  
*TEST 35 VERIFY THAT 'RESET' CLEARS MNCDI INPUT DATA REGISTER  
*****  
TST35: SCOPE  
MOV #40,$TIMES ;;DO 40 ITERATIONS  
MOV #BITEXT!BITS!BIT4,@ICSR ;INVERT DATA AND INHIBIT INPUT  
MOV @DIR,TEMP ;READ REGISTER  
CLR $GDDAT ;LOAD EXPECTED  
RESET ;CLEAR THE INPUT REG.  
BIS #BIT6,@$TKS ;ENABLE TKB INTR.  
BIS #BIT2,@ICSR ;INHIBIT REG. FROM BEING CLOKED  
MOV @DIR,$BDDAT ;READ REGISTER  
BEQ TST36 ;;BR IF CLEARED  
ERROR 10 ;RESET FAILED TO CLEAR INPUT REGISTER
```

```
*****  
*TEST 36 VERIFY THAT A 2ND STROBE PULSE WILL NOT CHANGE THE DIR DATA  
*****  
TST36: SCOPE  
MOV #BIT12!BITS!BIT4!BIT1,@ICSR ;DISABLE INPUTS, ENABLE INVERT DATA, EXT  
BIS #BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE  
BIC #BITS,@ICSR ;REMOVE INVERT DATA  
BIS #BITEXT!BIT7,@ICSR ;SET MAINT. STROBE AGAIN  
MOV #-1,$GDDAT ;LOAD EXPECTED DATA  
MOV @DIR,$BDDAT ;READ REGISTER  
CMP $GDDAT,$BDDAT ;COMPARE  
BEQ TST37 ;;BR IF SAME  
ERROR 10 ;DATA READY FAILED TO INHIBIT 2ND  
;STROBE FROM CHAINING THE DIR
```

```
616 .....  
(3) *TEST 37 INTERRUPT TEST -- VERIFY MNC DI INTERRUPTS VIA DATA READY VECTOR  
(3) .....  
(2) TST37: SCOPE  
617 006332 000004 MOV #64$, $LPADR ;LOAD RETURN VECTOR  
618 006334 012737 006342 001106 64$: MOV #1$, @DIDINV ;LOAD RETURN LEVEL  
619 006342 012777 006422 173060 MOV #200, @DIDINS ;LOAD RETURN LEVEL  
620 006350 012777 000200 173054 CLR -(SP)  
621 006356 005046 MOV #10$, -(SP)  
622 006360 012746 006366 RTI  
623 006364 000002 RTI  
624 006366 012777 000102 173020 10$: MOV #BIT6!BIT1, @ICSR ;SET STROBE MODE  
625 006374 052777 004200 173012 BIS #BITEXT!BIT7, @ICSR ;GENERATE MAINT. STROBE  
626 006402 000240 NOP  
627 006404 000240 NOP  
628 006406 000240 NOP  
629 006410 000240 NOP  
629 006412 005077 172776 CLR @ICSR ;CLEAR STATUS  
630 006416 104011 ERROR 11 ;MNC DI INPUT DATA READY FAILED TO INTERRUPT  
631 006420 000401 BR 2$ ;:RESET VECTOR  
632  
633 006422 022626 1$: CMP (SP)+, (SP)+ ;CLEAN STACK  
634 006424 005077 172764 2$: CLR @ICSR ;CLEAR DEVICE  
635 006430 013777 001432 172772 MOV DIDINS, @DIDINV  
636 006436 012777 004700 172766 MOV #4700, @DIDINS  
637  
638  
639 .....  
(3) *TEST 40 INTERRUPT TEST -- VERIFY MNC DI INTERRUPTS VIA OVERRUN ERROR  
(3) .....  
(2) TST40: SCOPE  
640 006444 000004 MOV #1$, @DIEINV ;LOAD RETURN VECTOR  
641 006446 012777 006534 172760 MOV #200, @DIEINS ;LOAD RETURN STATUS  
642 006454 012777 000200 172754 CLR -(SP)  
643 006462 005046 MOV #10$, -(SP)  
644 006464 012746 006472 RTI  
645 006470 000002 RTI  
645 006472 012777 040002 172714 10$: MOV #BIT14!BIT1, @ICSR ;ENABLE INTR. AND STROBE MODE  
646 006500 052777 104200 172706 BIS #BIT15!BITEXT!BIT7, @ICSR ;GENERATE MAINT. STROBE  
647 006506 052777 104200 172700 BIS #BIT15!BITEXT!BIT7, @ICSR ;GENERATE MAINT. STROBE AGAIN TO SET OVE  
648 006514 000240 NOP  
649 006516 000240 NOP  
650 006520 000240 NOP  
651 006522 000240 NOP  
652 006524 005077 172664 CLR @ICSR ;DISABLE INTR.  
653 006530 104011 ERROR 11 ;MNC DI FAILED TO INTERRUPT ON 'OVERRUN ERROR'  
654 006532 000401 BR 2$ ;:RESET VECTOR  
655  
656 006534 022626 1$: CMP (SP)+, (SP)+ ;CLEAN THE STACK  
657 006536 005077 172652 2$: CLR @ICSR ;CLEAR DEVICE  
658 006542 013777 001436 172664 MOV DIEINS, @DIEINV  
659 006550 012777 004700 172660 MOV #4700, @DIEINS  
660 006556 005046 CLR -(SP)  
661 006560 012746 006566 MOV #11$, -(SP)  
662 006564 000002 RTI  
663 006566 11$:
```

```
665
676
677
(3)
(3)
(2) 006566 000004
678 006570 005737 001440
679 006574 001002
680 006576 000137 007700
681 006602 012777 177777 172610 1$:
682 006610 005077 172610
683 006614 005077 172574
684 006620 005077 172560
685 006624 012777 025252 172556
686 006632 012737 000200 001124
687 006640 017737 172550 001126
688 006646 023737 001124 001126
689 006654 001401
690 006656 104002
691
692
(3)
(3)
(2) 006660 000004
693 006662 012777 177777 172530
694 006670 005077 172530
695 006674 012777 000002 172512
696 006702 005077 172476
697 006706 012777 052525 172474
698 006714 012737 000202 001124
699 006722 017737 172466 001126
700 006730 023737 001124 001126
701 006736 001401
702 006740 104002
703
704
(3)
(3)
(2) 006742 000004
705 006744 012777 177777 172446
706 006752 005077 172446
707 006756 012777 000004 172430
708 006764 005077 172414
709 006770 012777 070707 172412
710 006776 012737 000004 001124
711 007004 017737 172404 001126
712 007012 023737 001124 001126
713 007020 001401
714 007022 104002

*****
*TEST 41 VERIFY MODE 00 -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
*****
TST41: SCOPE
TST DWARF ;TEST IF WRAP-AROUND OR TESTER MODE
1$ BNE 1$ ;BR IF YES
JMP REMAIN ;NO REPORT END OF PASS
MOV #-1,@DIR ;CLEAR INPUT REG.
CLR @SBR ;CLEAR STIM. REG.
CLR @ICSR ;CLEAR INPUT DATA READY FLAG
CLR @OCSR ;CLEAR OUTPUT STATUS
MOV #25252,@DOR ;WRITE TO THE OUTPUT DATA REG.
MOV #BIT7,$GDDAT ;LOAD EXPECTED
MOV @ICSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST42 ;;BR IF SET
ERROR 2 ;MODE 00 -- EXT. STROBE FAILED TO SET INPUT DAT

*****
*TEST 42 VERIFY MODE 01 -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
*****
TST42: SCOPE
MOV #-1,@DIR ;CLEAR INPUT REG.
CLR @SBR ;CLEAR STIM. REG.
MOV #BIT1,@ICSR ;CLEAR INPUT DATA READY FLAG
CLR @OCSR ;CLEAR OUTPUT STATUS
MOV #52525,@DOR ;WRITE TO THE OUTPUT DATA REG.
MOV #BIT7!BIT1,$GDDAT ;LOAD EXPECTED
MOV @ICSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST43 ;;BR IF SET
ERROR 2 ;MODE 01 -- EXT. STROBE FAILED TO SET INPUT DAT

*****
*TEST 43 VERIFY MODE 10 -- INPUT STROBE WILL NOT SET INPUT DATA READY FLAG
*****
TST43: SCOPE
MOV #-1,@DIR ;CLEAR INPUT REG.
CLR @SBR ;CLEAR STIM. REG.
MOV #BIT2,@ICSR ;CLEAR INPUT DATA READY FLAG
CLR @OCSR ;CLEAR OUTPUT STATUS
MOV #70707,@DOR ;WRITE TO THE OUTPUT DATA REG.
MOV #BIT2,$GDDAT ;LOAD EXPECTED
MOV @ICSR,$BDDAT ;READ STATUS
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST44 ;;BR IF CLEARED
ERROR 2 ;MODE 10 -- EXT. STROBE SET INPUT DATA READY FL
```

```
716 ::*****
(3) :*TEST 44 VERIFY INPUT REPLY SETS OUTPUT DONE FLAG
(3) :*****
(2) 007024 000004 TST44: SCOPE
717 007026 005077 172356 CLR @DOR ;CLEAR OUTPUT DATA
718 007032 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD EXPECTED
719 007040 022737 000001 001440 CMP #1,DWARF ;CHECK IF IN TESTER MODE
720 007046 001003 BNE 1$ ;BR IF NOT
721 007050 012737 100000 001124 MOV #BIT15,$GDDAT ;LOAD TESTER WRAPAROUND STATUS
722 007056 005077 172322 1$: CLR @OCSR ;CLEAR OUTPUT DONE FLAG
723 007062 012777 004200 172324 MOV #BITEXT!BIT7,@ICSR ;SET INPUT READY FLAG
724 007070 005077 172320 CLR @ICSR ;CLEAR INPUT READY FLAG<GEN. INPUT REPLY>
725 007074 017737 172304 001126 MOV @OCSR,$BDDAT ;READ OUTPUT STATUS
726 007102 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
727 007110 001401 BEQ TST45 ;;BR IF SET
728 007112 104002 ERROR 2 ;INPUT REPLY FAILED TO SET OUTPUT DONE FLAG
729
730 ::*****
(3) :*TEST 45 VERIFY THE MNCDO - WRAPAROUND - MNCDI DATA PATH
(3) :*****
(2) 007114 000004 TST45: SCOPE
(1) 007116 012737 000100 001160 MOV #100,$TIMES ;;DO 100 ITERATIONS
731 007124 012777 177777 172266 MOV #-1,@DIR ;CLEAR INPUT REG.
732 007132 005077 172246 CLR @OCSR ;CLEAR OUTPUT STATUS
733 007136 005077 172252 CLR @ICSR ;CLEAR INPUT STATUS
734 007142 012737 000001 001442 MOV #BIT0,TEMP ;LOAD EXPECTED
735
736 007150 013737 001442 001124 1$: MOV TEMP,$GDDAT ;LOAD TIMEOUT EXPECTED
737 007156 013777 001124 172224 MOV $GDDAT,@DOR ;LOAD OUTPUT DATA REG.
738 007164 017737 172230 001126 MOV @DIR,$BDDAT ;READ INPUT DATA REGISTER
739 007172 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
740 007200 001401 BEQ 2$ ;;BR IF SAME
741 007202 104003 ERROR 3 ;INPUT DATA PATH ERROR
742
743 007204 006337 001442 2$: ASL TEMP ;TRY NEXT BIT
744 007210 001357 BNE 1$ ;BR IF MORE BITS
745
```

```
747      ;:*****
(3)      ;*TEST 46      VERIFY THE MNCDO - WRAPAROUND - MNCDI INVERTED DATA PATH
(3)      ;:*****
(2) 007212 000004      TST46: SCOPE
(1) 007214 012737 000100 001160      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
748 007222 012777 177777 172170      MOV      #-1,@DIR      ;:CLEAR INPUT REG.
749 007230 005077 172150      CLR      @OCSR      ;:CLEAR OUTPUT STATUS
750 007234 012777 000060 172152      MOV      #BITS!BIT4,@ICSR      ;:LOAD INPUT STATUS <INVERT DATA>
751 007242 012737 000001 001442      MOV      #BIT0,TEMP      ;:LOAD INITIAL BIT
752
753 007250 013777 001442 172132 1$:      MOV      TEMP,@DOR      ;:LOAD OUTPUT DATA REG.
754 007256 017737 172136 001126      MOV      @DIR,$BDDAT      ;:READ INPUT DATA REGISTER
755 007264 013737 001442 001124      MOV      TEMP,$GDDAT      ;:GET THE BIT
756 007272 005137 001124      COM      $GDDAT      ;:INVERT EXPECTED INPUT DATA
757 007276 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;:COMPARE
758 007304 001401      BEQ      2$      ;:BR IF SAME
759 007306 104003      ERROR      3      ;:INVERTED INPUT DATA PATH ERROR
760
761 007310 006337 001442      2$:      ASL      TEMP      ;:TRY NEXT BIT
762 007314 001355      BNE      1$      ;:BR IF MORE BITS
763
```

```

765      ;*****
(3)      ;*TEST 47      VERIFY IN 10 MODE THAT SBR AND INPUT BITS SET INPUT READY
(3)      ;*****
(2) 007316 000004
(1) 007320 012737 000100 001160 TST47: SCOPE
766      MOV      #100,$TIMES      ;;DO 100 ITERATIONS
767      INPUT DATA READY FLAG
768      LOAD A FLOATING 1 ACROSS THE SBR
769      VERIFY THAT ONLY THE CORRECT BIT SET DATA READY
770      CLR      @DOR
771 007326 005077 172056      MOV      #BIT0,TEMP      ;LOAD INITIAL BIT
772 007332 012737 000001 001442
773      1$: CLR      @DOR      ;CLEAR OUTPUT BITS
774 007340 005077 172044      CLR      @SBR      ;CLEAR SBR REG.
775 007344 005077 172054      MOV      #BIT2,@ICSR      ;CLEAR INPUT READY AND SET MODE 10
776 007350 012777 000004 172036      MOV      #-1,@DIR      ;CLEAR INPUT REG.
777 007356 012777 177777 172034
778      MOV      TEMP,@SBR      ;LOAD SBR REG.
779 007364 013777 001442 172032      BIC      #BIT7,@ICSR      ;CLEAR INPUT READY BIT
780 007372 042777 000200 172014      MOV      TEMP,@DOR      ;LOAD OUTPUT REG.
781 007400 013777 001442 172002
782      MOV      #BIT15!BIT7!BIT2,$GDDAT ;LOAD EXPECTED STATUS
783 007406 012737 100204 001124      MOV      @ICSR,$BDDAT ;READ STATUS
784 007414 017737 171774 001126      CMP      $GDDAT,$BDDAT ;COMPARE
785 007422 023737 001124 001126      BEQ      2$ ;;BR IF SET
786 007430 001401      ERROR 2 ;INPUT DATA READY FLAG FAILED
787 007432 104002 ;TO SET IN MODE 10 <STIMULUS MODE>
788
789      ;NOW LOAD ALL BITS EXCEPT THE FLOATING BIT AND ENSURE INPUT DATA READY DOES NOT SET
790 007434 013737 001442 001444 2$: MOV      TEMP,TEMP1 ;COPY EXPECTED
791 007442 005137 001444      COM      TEMP1 ;USE REVERSE PATTERN
792 007446 005077 171736      CLR      @DOR ;CLEAR OUTPUT REG.
793 007452 012777 177777 171740      MOV      #-1,@DIR ;CLEAR INPUT REG.
794 007460 042777 100200 171726      BIC      #BIT15!BIT7,@ICSR ;CLEAR INPUT DATA READY
795 007466 012737 000004 001124      MOV      #BIT2,$GDDAT ;LOAD EXPECTED
796
797 007474 013777 001444 171706      MOV      TEMP1,@DOR ;LOAD ALL OTHER DATA BITS
798 007502 017737 171706 001126      MOV      @ICSR,$BDDAT ;READ INPUT STATUS
799 007510 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE
800 007516 001401      BEQ      3$ ;;BR IF CLEARED
801 007520 104002      ERROR 2 ;INPUT DATA READY FLAG SET IN ERROR
802      ;UNEXPECTED SBR BIT SET INPUT DATA READY
803 007522 006337 001442 3$: ASL      TEMP ;TRY NEXT BIT
804 007526 001304      BNE      1$ ;BR IF MORE BITS
805
806      ;*****
(3)      ;*TEST 50      TEST THE TRANSITION ENABLE AND TRANSITION DETECTION
(3)      ;*****
(2) 007530 000004
807 007532 005077 171652
808 007536 012777 177777 171654 TST50: SCOPE
809 007544 012737 010000 001442      CLR      @DOR ;CLEAR INPUT REGISTER
810 007552 012777 000404 171634      MOV      #-1,@DIR ;LOAD INITIAL TRANSITION BIT
811 007560 012737 100604 001124      MOV      #BIT12,TEMP ;SET STIM. CLEAR READY, ENABLE TRANS.
      MOV      #BIT8!BIT2,@ICSR ;LOAD EXPECTED STATUS
      MOV      #BIT15!BIT8!BIT7!BIT2,$GDDAT

```

```

812 007566 042777 100200 171620 1$: BIC #BIT15!BIT7,@ICSR ;CLEAR READY
813 007574 013777 001442 171622 MOV TEMP,@SBR ;LOAD STIMULUS REG.
814 007602 013777 001442 171600 MOV TEMP,@DOR ;LOAD INPUT REG. <VIA OUTPUT REG.>
815 007610 017737 171600 001126 MOV @ICSR,$BDDAT ;READ INPUT STATUS
816 007616 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
817 007624 001401 BEQ ERROR 2$ ;:BR IF SAME
818 007626 104002 ERROR 2 ;TRANSITION ENABLE OR TRANSITION TO A ONE FAILED
819 ;NOW REMOVE THE TRANSITION DATA BIT (THIS SHOULD CAUSE THE INPUT READY FLAG TO SET AGAIN
820 007630 012777 177777 171562 2$: MOV #-1,@DIR ;CLEAR INPUT REG
821 007636 042777 100200 171550 BIC #BIT15!BIT7,@ICSR ;CLEAR INPUT READY FLAG
822 007644 043777 001442 171536 BIC TEMP,@DOR ;REMOVE THE INPUT DATA
823 ;THIS SHOULD CAUSE THE TRANSITION TO A ZERO
824 007652 017737 171536 001126 MOV @ICSR,$BDDAT ;READ INPUT STATUS
825 007660 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
826 007666 001401 BEQ ERROR 3$ ;:BR IF SET
827 007670 104002 ERROR 2 ;TRANSITION TO A ZERO FAILED
828 007672 006337 001442 3$: ASL TEMP ;TRY ANOTHER BIT ?
829 007676 001333 BNE 1$ ;BR IF YES
830 007700 REMAIN:
831 ;*****
(3) ;*TEST 51 DETERMINE IF MORE MNCDI'S REMAIN TO BE TESTED
(3) ;*****
(2) 007700 000004 TST51: SCOPE
(1) 007702 012737 000001 001160 MOV #1,$TIMES ;:DO 1 ITERATION
832 007710 005237 001202 INC $UNIT ;UPDATE UNIT NUMBER
833 007714 123737 001202 001526 CMPB $UNIT,EVER ;TEST IF MORE
834 007722 001454 BEQ 3$ ;:BR IF NOT
835 007724 012701 001404 MOV #OCSR,R1 ;LOAD POINTER TO OUTPUT STATUS ADDRESS
836 007730 063721 001462 1$: ADD VADDR,(R1)+ ;UPDATE OUTPUT BUS ADDRESS
837 007734 020127 001414 CMP R1,#DOR1+2 ;TEST IF DONE
838 007740 001373 BNE 1$ ;BRANCH IF NOT
839 007742 063721 001464 2$: ADD VADDR,(R1)+ ;UPDATE INPUT ADDRESS
840 007746 020127 001430 CMP R1,#SBR1+2 ;TEST IF DONE
841 007752 001373 BNE 2$ ;BRANCH IF NOT
842 007754 006337 001460 ASL MASKNM ;UPDATE ERROR FLAG BIT
843 007760 004737 012272 JSR PC,WHICHU ;DETERMINE UNIT #
844 007764 013700 001532 MOV UNITBD,RO ;GET UNIT #
845 007770 006300 ASL RO ;MAKE WORD
846 007772 016037 001466 001430 MOV VECLST(RO),DIDINV ;GET VALUE
847 010000 013737 001430 001432 MOV DIDINV,DIDINS ;MAKE OTHER VALUES
848 010006 062737 000002 001432 ADD #2,DIDINS
849 010014 013737 001430 001434 MOV DIDINV,DIEINV
850 010022 062737 000004 001434 ADD #4,DIEINV
851 010030 013737 001430 001436 MOV DIDINV,DIEINS
852 010036 062737 000006 001436 ADD #6,DIEINS
853 010044 005037 001102 CLR $STNM ;RESET TEST NUMBER
854 010050 000137 003424 JMP TST1 ;TEST NEXT UNIT
855 010054 3$:

```



```

857          .SBTTL  END OF PASS ROUTINE
(1)
(2)          ::*****
(1)          ::*INCREMENT THE PASS NUMBER ($PASS)
(1)          ::*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
(1)          ::*IF THERES A MONITOR GO TO IT
(1)          ::*IF THERE ISN'T JUMP TO EXTMSG
(1)
(1) 010054   $EOP:
(1) 010054   SCOPE
(1) 010056   000004   CLR          $STNM          ;;ZERO THE TEST NUMBER
(1) 010062   005037   001102   CLR          $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
(1) 010066   005237   001176   INC          $PASS           ;;INCREMENT THE PASS NUMBER
(1) 010072   042737   100000   001176   BIC          #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
(1) 010100   005327   DEC          (PC)+          ;;LOOP?
(1) 010102   000001   $EOPCT: .WORD 1
(1) 010104   003022   BGT          $DOAGN          ;;YES
(1) 010106   012737   MOV          (PC)+,@(PC)+   ;;RESTORE COUNTER
(1) 010110   000001   $ENDCT: .WORD 1
(1) 010112   010102   $EOPCT
(1) 010114   104401   010161   TYPE        ,SENDMG          ;;TYPE 'END PASS #'
(2) 010120   013746   001176   MOV          $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
(2) 010124   104405   TYPDS       ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 010126   104401   010156   TYPE        ,SENULI         ;;TYPE A NULL CHARACTER
(1) 010132   013700   000042   $GET42: MOV   @#42,R0        ;;GET MONITOR ADDRESS
(1) 010136   001405   BEQ          $DOAGN          ;;BRANCH IF NO MONITOR
(1) 010140   000005   RESET       ;;CLEAR THE WORLD
(1) 010142   004710   $ENDAD: JSR  PC,(R0)         ;;GO TO MONITOR
(1) 010144   000240   NOP         ;;SAVE ROOM
(1) 010146   000240   NOP         ;;FOR
(1) 010150   000240   NOP         ;;ACT11
(1) 010152
(1) 010152   000137   $DOAGN: JMP   @(PC)+          ;;RETURN
(1) 010154   010176   $RTNAD: .WORD EXTMSG
(1) 010156   377      377      000   $ENULL: .BYTE -1,-1,0       ;;NULL CHARACTER STRING
(1) 010161   015      042412   042116   $ENDMG: .ASCIIZ <15><12>/END PASS #/
(1) 010166   050040   051501   020123
(1) 010174   000043
858
859 010176   052777   000100   170740   EXTMSG: BIS   #BIT6,@$TKS    ;;ENABLE TKB INTR.
860 010204   005737   001112   TST          $ERTTL         ;;TEST IF ANY ERRORS
861 010210   001416   BEQ          1$             ;;BR IF NONE
862 010212   104401   017600   TYPE        ,ERRTOT        ;;TYPE TOTAL ERRORS MESSAGE
863 010216   013746   001112   MOV          $ERTTL,-(SP)   ;;PUSH TOTAL ERRORS ON STACK
864 010222   104405   TYPDS       ;;TYPE IT
865 010224   022737   000001   001460   CMP          #1,MASKNM     ;;TEST IF MULTIPLE
866 010232   001405   BEQ          1$             ;;BR IF NOT
867 010234   104401   017627   TYPE        ,MESGD         ;;TYPE BAD UNITS
868 010240   013746   001530   MOV          BADUNT,-(SP)   ;;PUSH BAD UNITS ON STACK FOR TYPE OUT
869 010244   104406   TYPBN       ;;TYPE IT
870 010246   104401   010156   1$: TYPE    ,SENULL         ;;ENSURE ALL TEXT GOT TYPED
871 010252   004737   003346   JSR          PC,CTRLCG      ;;TEST FOR CTRL C/G
872 010256   000137   002520   JMP          LOGIC

```

```

874                                     .SBTTL  MNCDI TEST MODULE SWITCH TYPEOUT LOOP
875 010262 012706 001100          DIDATA: MOV   #STACK,SP           ;LOAD THE STACK POINTER
876 010266 004737 003010          JSR    PC,FIXADR           ;FIX DEVICE ADDRESSES
877 010272 104401 010350          TYPE,  WAITO
878 010276 005077 171112          1$:   CLR   @ICSR           ;CLEAR INPUT STATUS
879 010302 012777 177777 171110  MOV   #-1,@DIR           ;CLEAR INPUT DATA REGISTER
880 010310 004737 003346          2$:   JSR    PC,CTRLCG      ;TEST IF CTRL C/G
881 010314 105777 171074          TSTB  @ICSR           ;WAIT FOR INPUT READY
882 010320 100373                    BPL   2$
883 010322 017746 171072          MOV   @DIR,-(SP)        ;GET INPUT DATA
884 010326 104402                    TYPOC ;TYPE THE OCTAL VALUE
885 010330 104401 020502          TYPE,  ADASH           ;TYPE A DASH
886 010334 017746 171060          MOV   @DIR,-(SP)        ;GET INPUT DATA AGAIN
887 010340 104406                    TYPBN ;TYPE THE BINARY VALUE
888 010342 104401 020506          TYPE,  TCRLF           ;TYPE A CR-LF
889 010346 000753                    BR    1$
890
891 010350      015      012          WAITO: .BYTE 15,12
892 010352 040527 052111 047111     .ASCII \WAITING FOR OPERATOR TO ACTIVIAE MNCDI (D/I)\
      010360 020107 047506 020122
      010366 050117 051105 052101
      010374 051117 052040 020117
      010402 041501 044524 044526
      010410 052101 020105 047115
      010416 042103 020111 042050
      010424 044457      051
893 010427      040 042524 052123     .ASCII \ TEST MODULE BUTTON\
      010434 046440 042117 046125
      010442 020105 052502 052124
      010450 047117
894 010452      015      012      000 .BYTE 15,12,0
895      010456          .EVEN
896

```

```

898          .SBTTL  TTY INPUT ROUTINE
(1)
(2)          ;*****
(1)          .ENABL  LSB
(1) 010456 000000 $TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
(1) 010460 000000 $TKQIN: .WORD 0          ;;INPUT POINTER
(1) 010462 000000 $TKQOUT: .WORD 0         ;;OUTPUT POINTER
(1) 010464 000040 $TKQSRT: .BLKB 32.        ;;TTY KEYBOARD QUEUE
(1)          $TKQEND=.
(1)
(1)          ;*TK INITIALIZE ROUTINE
(1)          ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
(1)          ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
(1)
(1)          ;*CALL:
(1)          ;*      JSR      PC,$TKINT
(1)          ;*      RETURN
(1)
(1) 010524 005037 010456 $TKINT: CLR  $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
(1) 010530 012737 010464 010460 MOV  #TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
(1) 010536 013737 010460 010462 MOV  $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1) 010544 012737 010574 000060 MOV  #TKSRV,@TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
(1) 010552 012737 000200 000062 MOV  #200,@TKVEC+2 ;;'BR' LEVEL 4
(1) 010560 005777 170362 TST  @TKB          ;;CLEAR DONE FLAG
(1) 010564 012777 000100 170352 MOV  #100,@TKS     ;;ENABLE TTY KEYBOARD INTERRUPT
(1) 010572 000207          RTS      PC          ;;RETURN TO CALLER
(1)
(1)          ;*TK SERVICE ROUTINE
(1)          ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
(1)          ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
(1)          ;*IT IN THE QUEUE.
(1)          ;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
(1)          ;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (MTEST1)
(1)
(1) 010574 117746 170346 $TKSRV: MOVB  @TKB,-(SP) ;;PICKUP THE CHARACTER
(1) 010600 042716 177600 BIC  #^C177,(SP) ;;STRIP THE JUNK
(1) 010604 021627 000003 CMP  (SP),#3      ;;IS IT A CONTROL C?
(1) 010610 001007 BNE  1$          ;;BRANCH IF NO
(1) 010612 104401 011744 TYPE  ,CNTLC      ;;TYPE A CONTROL-C (^C)
(1) 010616 004737 010524 JSR  PC,$TKINT   ;;INIT THE KEYBOARD
(1) 010622 005726 TST  (SP)+      ;;CLEAN UP STACK
(1) 010624 000137 002300 JMP  MTEST1     ;;CONTROL C RESTART
(1) 010630 021627 000007 1$: CMP  (SP),#7      ;;IS IT A CONTROL G?
(1) 010634 001004 BNE  2$          ;;BRANCH IF NO
(1) 010636 022737 000176 001140 CMP  #SWREG,SWR  ;;IS SOFT-SWR SELECTED?
(1) 010644 001500 BEQ  6$          ;;GO TO SWR CHANGE
(1)
(1) 010646 2$:
(1) 010646 022737 000040 010456 CMP  #32.,$TKCNT ;;IS THE QUEUE FULL?
(1) 010654 001004 BNE  3$          ;;BRANCH IF NO
(1) 010656 104401 011740 TYPE  ,SBELL     ;;RING THE TTY BELL
(1) 010662 005726 TST  (SP)+      ;;CLEAN CHARACTER OFF OF STACK
(1) 010664 000451 BR   5$          ;;EXIT
(1) 010666 021627 000023 3$: CMP  (SP),#23    ;;IS IT A CONTROL-S?

```

```

(1) 010672 001021 BNE 32$ ;;BRANCH IF NO
(1) 010674 005077 170244 CLR @STKS ;;DISABLE TTY KEYBOARD INTERRUPTS
(1) 010700 005726 TST (SP)+ ;;CLEAN CHAR OFF STACK
(1) 010702 105777 170236 31$: TSTB @STKS ;;WAIT FOR A CHAR
(1) 010706 100375 BPL 31$ ;;LOOP UNTIL ITS THERE
(1) 010710 117746 170232 MOVB @STKB,-(SP) ;;GET THE CHARACTER
(1) 010714 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 010720 022627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
(1) 010724 001366 BNE 31$ ;;BRANCH IF NO
(1) 010726 012777 000100 170210 MOV #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
(1) 010734 000002 RTI ;;RETURN
(1) 010736 005237 010456 32$: INC $TKCNT ;;COUNT THIS CHARACTER
(1) 010742 021627 000140 CMP (SP),#140 ;;IS IT UPPER CASE?
(1) 010746 002405 BLT 4$ ;;BRANCH IF YES
(1) 010750 021627 000175 CMP (SP),#175 ;;IS IT A SPECIAL CHAR?
(1) 010754 003002 BGT 4$ ;;BRANCH IF YES
(1) 010756 042716 000040 BIC #40,(SP) ;;MAKE IT UPPER CASE
(1) 010762 112677 177472 4$: MOVB (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
(1) 010766 005237 010460 INC $TKQIN ;;UPDATE THE POINTER
(1) 010772 023727 010460 010524 CMP $TKQIN,$$TKQEND ;;GO OFF THE END?
(1) 011000 001003 BNE 5$ ;;BRANCH IF NO
(1) 011002 012737 010464 010460 MOV $$TKQRT,$$TKQIN ;;RESET THE POINTER
(1) 011010 000002 5$: RTI ;;RETURN
    
```

```

(1) *****
(1) *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
(1) *CALL WHEN OPERATING IN TTY INTERRUPT MODE.
    
```

```

(1) 011012 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED
(1) 011020 001124 BNE 15$ ;;EXIT IF NOT
(1) 011022 105777 170116 TSTB @STKS ;;IS A CHAR WAITING?
(1) 011026 100121 BPL 15$ ;;IF NOT, EXIT
(1) 011030 117746 170112 MOVB @STKB,-(SP) ;;YES
(1) 011034 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 011040 021627 000007 CMP (SP),#7 ;;IS IT A CONTROL-G?
(1) 011044 001300 BNE 2$ ;;IF NOT, PUT IT IN THE TTY QUEUE
(1) 2$: AND EXIT
    
```

```

(1) *****
(1) *CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
(1) *ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
(1) *CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
    
```

```

(1) 011046 123727 001134 000001 6$: CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
(1) 011054 001674 BEQ 2$ ;;BRANCH IF YES
(1) 011056 005726 TST (SP)+ ;;CLEAR CONTROL-G OFF STACK
(1) 011060 004737 010524 JSR PC,$$TKINT ;;FLUSH THE TTY INPUT QUEUE
(1) 011064 005077 170054 CLR @STKS ;;DISABLE TTY KEYBOARD INTERRUPTS
(1) 011070 112737 000001 001135 MOVB #1,$$INTAG ;;SET INTERRUPT MODE INDICATOR
(1) 011076 104401 011756 $GTSWR: TYPE ,$$CNTLG ;;ECHO THE CONTROL-G (^G)
(1) 011102 104401 011763 TYPE ,$$MSWR ;;TYPE CURRENT CONTENTS
(2) 011106 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
(2) 011112 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
    
```

```

(1) 011114 104401 011774          TYPE      .SMNEW      ;;PROMPT FOR NEW SWR
(1) 011120 005046          19$:      CLR      -(SP)      ;;CLEAR COUNTER
(1) 011122 005046          CLR      -(SP)      ;;THE NEW SWR
(1) 011124 105777 170014      7$:      TSTB     @STKS      ;;CHAR THERE?
(1) 011130 100375          BPL      7$         ;;IF NOT TRY AGAIN
(1)
(1) 011132 117746 170010          MOVB     @STKB,-(SP) ;;PICK UP CHAR
(1) 011136 042716 177600          BIC     #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1)
(1) 011142 021627 000003          CMP     (SP),#3      ;;IS IT A CONTROL-C?
(1) 011146 001015          BNE     9$          ;;BRANCH IF NOT
(1) 011150 104401 011744          TYPE     .SCNTLC     ;;YES, ECHO CONTROL-C (^C)
(1) 011154 062706 000006          ADD     #6,SP        ;;CLEAN UP STACK
(1) 011160 123727 001135 000001      CMPB     $INTAG,#1   ;;REENABLE TTY KEYBOARD INTERRUPTS?
(1) 011166 001003          BNE     8$          ;;BRANCH IF NO
(1) 011170 012777 000100 167746      MOV     #100,@STKS  ;;ALLOW TTY KEYBOARD INTERRUPTS
(1) 011176 000137 002300      8$:      JMP     MTEST1      ;;CONTROL-C RESTART
(1)
(1)
(1) 011202 021627 000025          9$:      CMP     (SP),#25     ;;IS IT A CONTROL-U?
(1) 011206 001005          BNE     10$         ;;BRANCH IF NOT
(1) 011210 104401 011751          TYPE     .SCNTLU     ;;YES, ECHO CONTROL-U (^U)
(1) 011214 062706 000006      20$:     ADD     #6,SP        ;;IGNORE PREVIOUS INPUT
(1) 011220 000737          BR      19$         ;;LET'S TRY IT AGAIN
(1)
(1)
(1) 011222 021627 000015          10$:     CMP     (SP),#15     ;;IS IT A <CR>?
(1) 011226 001022          BNE     16$         ;;BRANCH IF NO
(1) 011230 005766 000004          TST     4(SP)       ;;YES, IS IT THE FIRST CHAR?
(1) 011234 001403          BEQ     11$         ;;BRANCH IF YES
(1) 011236 016677 000002 167674      MOV     2(SP),@SWR  ;;SAVE NEW SWR
(1) 011244 062706 000006      11$:     ADD     #6,SP        ;;CLEAR UP STACK
(1) 011250 104401 001165      14$:     TYPE     .SCRLF     ;;ECHO <CR> AND <LF>
(1) 011254 123727 001135 000001      CMPB     $INTAG,#1   ;;RE-ENABLE TTY KBD INTERRUPTS?
(1) 011262 001003          BNE     15$         ;;BRANCH IF NOT
(1) 011264 012777 000100 167652      MOV     #100,@STKS  ;;RE-ENABLE TTY KBD INTERRUPTS
(1) 011272 000002          RTI     15$         ;;RETURN
(1) 011274 004737 014360      16$:     JSR     PC,$TYPEC   ;;ECHO CHAR
(1) 011300 021627 000060          CMP     (SP),#60    ;;CHAR < 0?
(1) 011304 002420          BLT     18$         ;;BRANCH IF YES
(1) 011306 021627 000067          CMP     (SP),#67    ;;CHAR > 7?
(1) 011312 003015          BGT     18$         ;;BRANCH IF YES
(1) 011314 042726 000060          BIC     #60,(SP)+   ;;STRIP-OFF ASCII
(1) 011320 005766 000002          TST     2(SP)       ;;IS THIS THE FIRST CHAR
(1) 011324 001403          BEQ     17$         ;;BRANCH IF YES
(1) 011326 006316          ASL     (SP)        ;;NO, SHIFT PRESENT
(1) 011330 006316          ASL     (SP)        ;;CHAR OVER TO MAKE
(1) 011332 006316          ASL     (SP)        ;;ROOM FOR NEW ONE.
(1) 011334 005266 000002      17$:     INC     2(SP)       ;;KEEP COUNT OF CHAR
(1) 011340 056616 177776          BIS     -2(SP),(SP) ;;SET IN NEW CHAR
(1) 011344 000667          BR      7$         ;;GET THE NEXT ONE
(1) 011346 104401 001164      18$:     TYPE     .SQUES     ;;TYPE ?<CR><LF>
(1) 011352 000720          BR      20$         ;;SIMULATE CONTROL-U
(1) .DSABL  LSB

```

```
(1)
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 011354 011646
(1) 011356 016666 000004 00000?
(1) 011364 005066 000004
(2) 011370 005046
(2) 011372 012746 011400
(2) 011376 000002
(2) 011400
(1) 011400 005737 010456
(1) 011404 001775
(1) 011406 005337 010456
(1) 011412 117766 177044 000004
(1) 011420 005237 010462
(1) 011424 023727 010462 010524
(1) 011432 001003
(1) 011434 012737 010464 010462
(1) 011442 000002
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 011444 010346
(1) 011446 005046
(1) 011450 012703 011700
(1) 011454 022703 011740
(1) 011460 101456
(1) 011462 104411
(1) 011464 112613
(1) 011466 122713 000177
(1) 011472 001022
(1) 011474 005716
(1) 011476 001007
(1) 011500 112737 000134 011676
(1) 011506 104401 011676
(1) 011512 012716 177777
(1) 011516 005303
(1) 011520 020327 011700
(1) 011524 103434
(1) 011526 111337 011676
(1) 011532 104401 011676
(1) 011536 000746
(1) 011540 005716

:*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
* RDCHR ::GET A CHARACTER FROM THE QUEUE
* RETURN HERE ::CHARACTER IS ON THE STACK
* ::WITH PARITY BIT STRIPPED OFF
:
$RDCHR: MOV (SP),-(SP) ::PUSH DOWN THE PC AND
MOV 4(SP),2(SP) ::THE PS
CLR 4(SP) ::GET READY FOR A CHARACTER
CLR -(SP) ::PUT NEW PS ON STACK
MOV #64$,-(SP) ::PUT NEW PC ON STACK
RTI ::POP NEW PC AND PS
64$:
1$: TST $TKCNT ::WAIT ON A CHARACTER
BEQ 1$
DEC $TKCNT ::DECREMENT THE COUNTER
MOVB @STKQOUT,4(SP) ::GET ONE CHARACTER
INC $TKQOUT ::UPDATE THE POINTER
CMP $TKQOUT,#$TKQEND ::DID IT GO OFF OF THE END?
BNE 2$ ::BRANCH IF NO
MOV #$TKQSRRT,$TKQOUT ::RESET THE POINTER
RTI ::RETURN
2$:
:*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
* RDLIN ::INPUT A STRING FROM THE TTY
* RETURN HERE ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
* ::TERMINATOR WILL BE A BYTE OF ALL 0'S
:
$RDLIN: MOV R3, -(SP) ::SAVE R3
CLR -(SP) ::CLEAR THE RUBOUT KEY
1$: MOV #$TTYIN,R3 ::GET ADDRESS
2$: CMP #$TTYIN+32.,R3 ::BUFFER FULL?
BLOS 4$ ::BR IF YES
RDCHR ::GO READ ONE CHARACTER FROM THE TTY
MOVB (SP)+,(R3) ::GET CHARACTER
10$: CMPB #177,(R3) ::IS IT A RUBOUT
BNE 5$ ::BR IF NO
TST (SP) ::IS THIS THE FIRST RUBOUT?
BNE 6$ ::BR IF NO
MOVB #' \,9$ ::TYPE A BACK SLASH
TYPE ,9$
MOV #-1,(SP) ::SET THE RUBOUT KEY
6$: DEC R3 ::BACKUP BY ONE
CMP R3,#$TTYIN ::STACK EMPTY?
BLO 4$ ::BR IF YES
MOVB (R3),9$ ::SETUP TO TYPEOUT THE DELETED CHAR.
TYPE ,9$ ::GO TYPE
BR 2$ ::GO READ ANOTHER CHAR.
5$: TST (SP) ::RUBOUT KEY SET?
```



```

(1) 012006          $SCOPE:
(1) 012006 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(2) 012010 004737 003346          JSR          PC,CTRLCG
(1) 012014 032777 040000 167116 1$:          BIT          #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
(1) 012022 001114          BNE          $OVER          ;;YES IF SW14=1
(1)          ;#####START OF CODE FOR THE XOR TESTER#####
(1) 012024 000416          $XTSTR: BR          6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)          ;#####END OF CODE FOR THE XOR TESTER#####
(1) 012026 013746 000004          MOV          @WERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 012032 012737 012052 000004          MOV          #5$,@WERRVEC          ;;SET FOR TIMEOUT
(1) 012040 005737 177060          TST          @#177060          ;;TIME OUT ON XOR?
(1) 012044 012637 000004          MOV          (SP)+,@WERRVEC          ;;RESTORE THE ERROR VECTOR
(1) 012050 000463          BR          $SVLAD          ;;GO TO THE NEXT TEST
(1) 012052 022626          5$:          CMP          (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
(1) 012054 012637 000004          MOV          (SP)+,@WERRVEC          ;;RESTORE THE ERROR VECTOR
(1) 012060 000423          BR          7$          ;;LOOP ON THE PRESENT TEST
(1) 012062          6$:;#####END OF CODE FOR THE XOR TESTER#####
(1) 012062 032777 000400 167050          BIT          #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
(1) 012070 001404          BEQ          2$          ;;BR IF NO
(1) 012072 127737 167042 001102          CMPB         @SWR,$STSTM          ;;ON THE RIGHT TEST? SWR<7:0>
(1) 012100 001465          BEQ          $OVER          ;;BR IF YES
(1) 012102 105737 001103          2$:          TSTB         $ERFLG          ;;HAS AN ERROR OCCURRED?
(1) 012106 001421          BEQ          3$          ;;BR IF NO
(1) 012110 123737 001115 001103          CMPB         $ERMAX,$ERFLG          ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 012116 101015          BHI          3$          ;;BR IF NO
(1) 012120 032777 001000 167012          BIT          #BIT09,@SWR          ;;LOOP ON ERROR?
(1) 012126 001404          BEQ          4$          ;;BR IF NO
(1) 012130 013737 001110 001106          7$:          MOV          $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 012136 000446          BR          $OVER
(1) 012140 105037 001103          4$:          CLRB         $ERFLG          ;;ZERO THE ERROR FLAG
(1) 012144 005037 001160          CLR          $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 012150 000415          BR          1$          ;;ESCAPE TO THE NEXT TEST
(1) 012152 032777 004000 166760          3$:          BIT          #BIT11,@SWR          ;;INHIBIT ITERATIONS?
(1) 012160 001011          BNE          1$          ;;BR IF YES
(1) 012162 005737 001176          TST          $PASS          ;;IF FIRST PASS OF PROGRAM
(1) 012166 001406          BEQ          1$          ;;INHIBIT ITERATIONS
(1) 012170 005237 001104          INC          $ICNT          ;;INCREMENT ITERATION COUNT
(1) 012174 023737 001160 001104          CMP          $TIMES,$ICNT          ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 012202 002024          BGE          $OVER          ;;BR IF MORE ITERATION REQUIRED
(1) 012204 012737 000001 001104          1$:          MOV          #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
(1) 012212 013737 012270 001160          MOV          $MXCNT,$TIMES          ;;SET NUMBER OF ITERATIONS TO DO
(1) 012220 105237 001102          $SVLAD: INCB         $STSTM          ;;COUNT TEST NUMBERS
(1) 012224 113737 001102 001174          MOVB         $STSTM,$TESTN          ;;SET TEST NUMBER IN APT MAILBOX
(1) 012232 011637 001106          MOV          (SP),$LPADR          ;;SAVE SCOPE LOOP ADDRESS
(1) 012236 011637 001110          MOV          (SP),$LPERR          ;;SAVE ERROR LOOP ADDRESS
(1) 012242 005037 001162          CLR          $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 012246 112737 000001 001115          MOVB         #1,$ERMAX          ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 012254 013777 001102 166660          $OVER:      MOV          $STSTM,@DISPLAY          ;;DISPLAY TEST NUMBER
(1) 012262 013716 001106          MOV          $LPADR,(SP)          ;;FUDGE RETURN ADDRESS
(1) 012266 000002          RTI          ;;FIXES PS
(1) 012270 003720          $MXCNT: 2000.          ;;MAX. NUMBER OF ITERATIONS

```



```

901
907      ;SUBROUTINE TO DETERMINE UNIT #
908 012272 012737 000000 001532 WHICHU: MOV #0,UNITBD      ;PRIME THE VALUE
909 012300 013737 001460 012322      MOV MASKNM,11$      ;LOAD VALUE
910 012306 006237 012322      10$: ASR 11$      ;SHIFT
911 012312 001404      BEQ 12$      ;BR WHEN DONE
912 012314 005237 001532      INC UNITBD      ;UPDATE BIT
913 012320 000772      BR 10$
914 012322 000000      11$: 0
915 012324 000207      12$: RTS PC      ;EXIT
916      .SBTTL ERROR HANDLER ROUTINE
(1)
(2)      ;*****
(1)      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1)      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1)      ;*AND GO TO $ERRTYP ON ERROR
(1)      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)      ;*SW15=1 HALT ON ERROR
(1)      ;*SW13=1 INHIBIT ERROR TYPEOUTS
(1)      ;*SW09=1 LOOP ON ERROR
(1)      ;*CALL
(1)      ;* ERROR N ;;ERROR-EMT AND N=ERROR ITEM NUMBER
(1)      $ERROR:
(1) 012326      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 012326 104410      JSR PC,CTRLCG      ;TEST FOR CTRL C/G
(3) 012330 004737 003346      BIS MASKNM,BADUNT      ;INCIDATE BAD BIT
(3) 012334 053737 001460 001530      JSR PC,WHICHU      ;DETERMINE UNIT #
(3) 012342 004737 012272      7$: INCB $ERFLG      ;;SET THE ERROR FLAG
(1) 012346 105237 001103      BEQ 7$      ;;DON'T LET THE FLAG GO TO ZERO
(1) 012352 001775      MOV $STNM,@DISPLAY      ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 012354 013777 001102 166560      INC $ERTTL      ;;INC THE ERROR COUNT
(1) 012362 005237 001112      MOV (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 012366 011637 001116      SUB #2,$ERRPC
(1) 012372 162737 000002 001116      MOVB @ $ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 012400 117737 166512 001114      BIT #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
(1) 012406 032777 020000 166524      BNE 20$      ;;SKIP TYPEOUTS
(1) 012414 001004      JSR PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(1) 012416 004737 012530      TYPE $CRLF
(1) 012422 104401 001165      20$: CMPB #APTENV,$ENV      ;;RUNNING IN APT MODE
(1) 012426 122737 000001 001210      BNE 2$      ;;NO SKIP APT ERROR REPORT
(1) 012434 001007      MOVB $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 012436 113737 001114 012450      JSR PC,$ATY4      ;;REPORT FATAL ERROR TO APT
(1) 012444 004737 013374      21$: .BYTE 0
(1) 012450 000      .BYTE 0
(1) 012451 000
(1) 012452 000777      22$: ER 22$      ;;APT ERROR LOOP
(1) 012454 005777 166460      2$: TST @SWR      ;;HALT ON ERROR
(1) 012460 100002      BPL 3$      ;;SKIP IF CONTINUE
(1) 012462 000000      HALT      ;;HALT ON ERROR!
(1) 012464 104410      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 012466 032777 001000 166444      3$: BIT #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
(1) 012474 001402      BEQ 4$      ;;BR IF NO
(1) 012476 013716 001110      MOV $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING

```

```

(1) 012502 005737 001162      4$:  TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
(1) 012506 001402              BEQ      5$              ;;BR IF NONE
(1) 012510 013716 001162      MOV      $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 012514              5$:
(1) 012514 022737 010142 000042  CMP      #SENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
(1) 012522 001001              BNE      6$              ;;BRANCH IF NO
(1) 012524 000000              HALT
(1) 012526              6$:
(1) 012526 000002              RTI              ;;RETURN
917 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 012530      $ERRTYP:
(1) 012530 104401 001165      TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 012534 010046      MOV     R0,-(SP)        ;;SAVE R0
(1) 012536 005000      CLR     R0              ;;PICKUP THE ITEM INDEX
(1) 012540 153700 001114      BISB   @#$ITEMB,R0
(1) 012544 001004      BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)
(2) 012546 013746 001116      MOV     $ERRPC,-(SP)    ;;TYPE THE PC OF THE ERROR
(2)
(2) 012552 104402      TYPCC   ;;SAVE $ERRPC FOR TYPEOUT
(1) 012554 000445      BR      10$            ;;ERROR ADDRESS
(1) 012556 005300      1$:  DEC     R0              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 012560 006300      ASL     R0              ;;GET OUT
(1) 012562 006300      ASL     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1) 012564 006300      ASL     R0              ;; WORK FOR THE ERROR TABLE
(1) 012566 062700 001254      ADD     #SERRTB,R0      ;;FORM TABLE POINTER
(1) 012572 012037 012602      MOV     (R0)+,2$        ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 012576 001404      BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1) 012600 104401      TYPE    'ERROR MESSAGE' ;;TYPE THE 'ERROR MESSAGE'
(1) 012602 000000      .WORD   0              ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 012604 104401 001165      TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 012610 012037 012620      3$:  MOV     (R0)+,4$        ;;PICKUP 'DATA HEADER' POINTER
(1) 012614 001404      BEQ     5$              ;;SKIP TYPEOUT IF 0
(1) 012616 104401      TYPE    'DATA HEADER'   ;;TYPE THE 'DATA HEADER'
(1) 012620 000000      .WORD   0              ;;'DATA HEADER' POINTER GOES HERE
(1) 012622 104401 001165      TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 012626 010146      5$:  MOV     R1,-(SP)        ;;SAVE R1
(1) 012630 012001      MOV     (R0)+,R1        ;;PICKUP 'DATA TABLE' POINTER
(1) 012632 001415      BEQ     9$              ;;BR IF NO DATA TO BE TYPED
(1) 012634 012000      MOV     (R0)+,R0        ;;PICKUP 'DATA FORMAT' POINTER
(1) 012636 105720      6$:  TSTB   (R0)+           ;;'OCTAL' OR 'DECIMAL'
(1) 012640 001003      BNE     7$              ;;BR IF DECIMAL
(2) 012642 013146      MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(2) 012644 104402      TYPCC   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 012646 000402      BR      8$
(2) 012650      7$:
(2) 012650 013146      MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(2) 012652 104405      TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN

```

```

;*****
;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.

```

```

$ERRTYP:
TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
MOV     R0,-(SP)        ;;SAVE R0
CLR     R0              ;;PICKUP THE ITEM INDEX
BISB   @#$ITEMB,R0
BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
;;TYPE THE PC OF THE ERROR
MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
;;ERROR ADDRESS
TYPCC   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
BR      10$            ;;GET OUT
1$:  DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
;; WORK FOR THE ERROR TABLE
ASL     R0
ASL     R0
ADD     #SERRTB,R0      ;;FORM TABLE POINTER
MOV     (R0)+,2$        ;;PICKUP 'ERROR MESSAGE' POINTER
BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
TYPE    'ERROR MESSAGE' ;;TYPE THE 'ERROR MESSAGE'
.WORD   0              ;;'ERROR MESSAGE' POINTER GOES HERE
TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
3$:  MOV     (R0)+,4$        ;;PICKUP 'DATA HEADER' POINTER
BEQ     5$              ;;SKIP TYPEOUT IF 0
TYPE    'DATA HEADER'   ;;TYPE THE 'DATA HEADER'
.WORD   0              ;;'DATA HEADER' POINTER GOES HERE
TYPE    ,SCLRF          ;;'CARRIAGE RETURN' & 'LINE FEED'
5$:  MOV     R1,-(SP)        ;;SAVE R1
MOV     (R0)+,R1        ;;PICKUP 'DATA TABLE' POINTER
BEQ     9$              ;;BR IF NO DATA TO BE TYPED
MOV     (R0)+,R0        ;;PICKUP 'DATA FORMAT' POINTER
6$:  TSTB   (R0)+           ;;'OCTAL' OR 'DECIMAL'
BNE     7$              ;;BR IF DECIMAL
MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
TYPCC   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
BR      8$
7$:
MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
TYPDS   ;;GO TYPE--DECIMAL ASCII WITH SIGN

```

(1)	012654	005711		8\$:	TST	(R1)	:: IS THERE ANOTHER NUMBER?
(1)	012656	001403			BEQ	9\$:: BR IF NO
(1)	012660	104401	012700		TYPE	,11\$:: TYPE TWO(2) SPACES
(1)	012664	000764			OR	6\$:: LOOP
(1)							
(1)	012666	012601		9\$:	MCV	(SP)+,R1	:: RESTORE R1
(1)	012670	012600		10\$:	MOV	(SP)+,R0	:: RESTORE R0
(1)	012672	104401	001165		TYPE	, \$CRLF	:: 'CARRIAGE RETURN' & 'LINE FEED'
(1)	012676	000207			RTS	PC	:: RETURN
(1)	012700	020040	000	11\$:	.ASCII?	/ /	:: TWO(2) SPACES
(1)		012704			.EVEN		

```

919          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)          ;*****
(1)          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)          ;*CALL:
(1)          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(1)          ;*      TYPOS          ;;CALL FOR TYPEOUT
(1)          ;*      .BYTE  N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)          ;*      .BYTE  M          ;;M=1 OR 0
(1)          ;*                               ;;1=TYPE LEADING ZEROS
(1)          ;*                               ;;0=SUPPRESS LEADING ZEROS
(1)          ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)          ;*$TYPOS OR $TYPOC
(1)          ;*CALL:
(1)          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(1)          ;*      TYPON          ;;CALL FOR TYPEOUT
(1)          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)          ;*CALL:
(1)          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(1)          ;*      TYPOC          ;;CALL FOR TYPEOUT
(1) 012704 017646 000000          $TYPOS: MOV      @ (SP),-(SP)          ;;PICKUP THE MODE
(1) 012710 116637 000001 013127  MOVB     1(SP), $OFILL          ;;LOAD ZERO FILL SWITCH
(1) 012716 112637 013131          MOVB     (SP)+, $OMODE+1          ;;NUMBER OF DIGITS TO TYPE
(1) 012722 062716 000002          ADD      #2, (SP)          ;;ADJUST RETURN ADDRESS
(1) 012726 000406          BR      $TYPON
(1) 012730 112737 000001 013127  $TYPOC: MOVB     #1, $OFILL          ;;SET THE ZERO FILL SWITCH
(1) 012736 112737 000006 013131  MOVB     #6, $OMODE+1          ;;SET FOR SIX(6) DIGITS
(1) 012744 112737 000005 013126  $TYPON: MOVB     #5, $OCNT          ;;SET THE ITERATION COUNT
(1) 012752 010346          MOV      R3, -(SP)          ;;SAVE R3
(1) 012754 010446          MOV      R4, -(SP)          ;;SAVE R4
(1) 012756 010546          MOV      R5, -(SP)          ;;SAVE R5
(1) 012760 113704 013131          MOVB     $OMODE+1, R4          ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 012764 005404          NEG      R4
(1) 012766 062704 000006          ADD      #6, R4          ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 012772 110437 013130          MOVB     R4, $OMODE          ;;SAVE IT FOR USE
(1) 012776 113704 013127          MOVB     $OFILL, R4          ;;GET THE ZERO FILL SWITCH
(1) 013002 016605 000012          MOV      12(SP), R5          ;;PICKUP THE INPUT NUMBER
(1) 013006 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
(1) 013010 006105          1$: ROL     R5          ;;ROTATE MSB INTO 'C'
(1) 013012 000404          BR      3$          ;;GO DO MSB
(1) 013014 006105          2$: ROL     R5          ;;FORM THIS DIGIT
(1) 013016 006105          ROL     R5
(1) 013020 006105          ROL     R5
(1) 013022 010503          MOV      R5, R3
(1) 013024 006103          3$: ROL     R3          ;;GET LSB OF THIS DIGIT
(1) 013026 105337 013130          DECB     $OMODE          ;;TYPE THIS DIGIT?
(1) 013032 100016          BPL     7$          ;;BR IF NO
(1) 013034 042703 177770          BIC     #177770, R3          ;;GET RID OF JUNK
(1) 013040 001002          BNE     4$          ;;TEST FOR 0

```

```

(1) 013042 005704           TST      R4           ;;SUPPRESS THIS 0?
(1) 013044 001403           BEQ      5$           ;;BR IF YES
(1) 013046 005204           4$: INC      R4           ;;DON'T SUPPRESS ANYMORE 0'S
(1) 013050 052703 000060     BIS      #'0,R3       ;;MAKE THIS DIGIT ASCII
(1) 013054 052703 000040     5$: BIS      #' ,R3       ;;MAKE ASCII IF NOT ALREADY
(1) 013060 110337 013124     MOV      R3,8$        ;;SAVE FOR TYPING
(1) 013064 104401 013124     TYPE      8$         ;;GO TYPE THIS DIGIT
(1) 013070 105337 013126     7$: DECB     $OCNT       ;;COUNT BY 1
(1) 013074 003347           BGT      2$           ;;BR IF MORE TO DO
(1) 013076 002402           BLT      6$           ;;BR IF DONE
(1) 013100 005204           INC      R4           ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 013102 000744           BR       2$           ;;GO DO THE LAST DIGIT
(1) 013104 012605           6$: MOV      (SP)+,R5     ;;RESTORE R5
(1) 013106 012604           MOV      (SP)+,R4     ;;RESTORE R4
(1) 013110 012603           MOV      (SP)+,R3     ;;RESTORE R3
(1) 013112 016666 000002 000004 MOV      2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
(1) 013120 012616           MOV      (SP)+,(SP)
(1) 013122 000002           RTI                   ;;RETURN
(1) 013124 000           8$: .BYTE     0           ;;STORAGE FOR ASCII DIGIT
(1) 013125 000           .BYTE     0           ;;TERMINATOR FOR TYPE ROUTINE
(1) 013126 000           $OCNT: .BYTE    0           ;;OCTAL DIGIT COUNTER
(1) 013127 000           $OFILL: .BYTE    0           ;;ZERO FILL SWITCH
(1) 013130 000000           $OMODE: .WORD     0           ;;NUMBER OF DIGITS TO TYPE
.920 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

(1)
(2)
(1)
(1) *****
(1) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1) *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1) *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1) *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1) *REPLACED WITH SPACES.
(1) *CALL:
(1) *   MOV      NUM,-(SP)           ;;PUT THE BINARY NUMBER ON THE STACK
(1) *   TYPDS                      ;;GO TO THE ROUTINE

(1)
(2) $TYPDS:
(3) 013132 010046           MOV      R0,-(SP)       ;;PUSH R0 ON STACK
(3) 013134 010146           MOV      R1,-(SP)       ;;PUSH R1 ON STACK
(3) 013136 010246           MOV      R2,-(SP)       ;;PUSH R2 ON STACK
(3) 013140 010346           MOV      R3,-(SP)       ;;PUSH R3 ON STACK
(3) 013142 010546           MOV      R5,-(SP)       ;;PUSH R5 ON STACK
(1) 013144 012746 020200     MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1) 013150 016605 000020     MOV      20(SP),R5      ;;GET THE INPUT NUMBER
(1) 013154 100004           BPL      1$           ;;BR IF INPUT IS POS.
(1) 013156 005405           NEG      R5           ;;MAKE THE BINARY NUMBER POS.
(1) 013160 112766 000055 000001 MOV      #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1) 013166 005000           1$: CLR      R0           ;;ZERO THE CONSTANTS INDEX
(1) 013170 012703 013346     MOV      #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
(1) 013174 112723 000040     MOV      #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
(1) 013200 005002           2$: CLR      R2           ;;CLEAR THE BCD NUMBER
(1) 013202 016001 013336     MOV      $DTBL(R0),R1   ;;GET THE CONSTANT
(1) 013206 160105           3$: SUB      R1,R5        ;;FORM THIS BCD DIGIT
(1) 013210 002402           BLT      4$           ;;BR IF DONE
(1) 013212 005202           INC      R2           ;;INCREASE THE BCD DIGIT BY 1

```

```

(1) 013214 000774          BR      3$
(1) 013216 060105          4$:  ADD    R1,R5      ;;ADD BACK THE CONSTANT
(1) 013220 005702          TST    R2          ;;CHECK IF BCD DIGIT=0
(1) 013222 001002          BNE    5$          ;;FALL THROUGH IF 0
(1) 013224 105716          TSTB   (SP)        ;;STILL DOING LEADING J'S?
(1) 013226 100407          BMI    7$          ;;BR IF YES
(1) 013230 106316          5$:  ASLB   (SP)        ;;MSD?
(1) 013232 103003          BCC    6$          ;;BR IF NO
(1) 013234 116663 000001 177777  MOVB   1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 013242 052702 000060          6$:  BIS    #'0,R2    ;;MAKE THE BCD DIGIT ASCII
(1) 013246 052702 000040          7$:  BIS    #' ,R2    ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 013252 110223          MOVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 013254 005720          TST    (R0)+       ;;JUST INCREMENTING
(1) 013256 020027 000010          CMP    R0,#10     ;;CHECK THE TABLE INDEX
(1) 013262 002746          BLT    2$          ;;GO DO THE NEXT DIGIT
(1) 013264 003002          BGT    8$          ;;GO TO EXIT
(1) 013266 010502          MOV    R5,R2      ;;GET THE LSD
(1) 013270 000764          BR     6$          ;;GO CHANGE TO ASCII
(1) 013272 105726          8$:  TSTB   (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 013274 100003          BPL    9$          ;;BR IF NO
(1) 013276 116663 177777 177776  MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 013304 105013          9$:  CLRB   (R3)      ;;SET THE TERMINATOR
(3) 013306 012605          MOV    (SP)+,R5   ;;POP STACK INTO R5
(3) 013310 012603          MOV    (SP)+,R3   ;;POP STACK INTO R3
(3) 013312 012602          MOV    (SP)+,R2   ;;POP STACK INTO R2
(3) 013314 012601          MOV    (SP)+,R1   ;;POP STACK INTO R1
(3) 013316 012600          MOV    (SP)+,R0   ;;POP STACK INTO R0
(1) 013320 104401 013346          TYPE  ,SDBLK      ;;NOW TYPE THE NUMBER
(1) 013324 016666 000002 000004  MOV    2(SP),4(SP) ;;ADJUST THE STACK
(1) 013332 012616          MOV    (SP)+,(SP)
(1) 013334 000002          RTI                    ;;RETURN TO USER
(1) 013336 023420          $DTBL: 10000.
(1) 013340 001750          1000.
(1) 013342 000144          100.
(1) 013344 000012          10.
(1) 013346 000004          $DBLK: .BLKW 4
          .SBTTL APT COMMUNICATIONS ROUTINE

(1)
(2)
(1) 013356 112737 000001 013622 $ATY1: MOVB   #1,$FFLG ;;TO REPORT FATAL ERROR
(1) 013364 112737 000001 013620 $ATY3: MOVB   #1,$MFLG ;;TO TYPE A MESSAGE
(1) 013372 000403          BR     $ATYC
(1) 013374 112737 000001 013622 $ATY4: MOVB   #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(2) 013402          $ATYC:
(3) 013402 010046          MOV    R0,-(SP)   ;;PUSH R0 ON STACK
(3) 013404 010146          MOV    R1,-(SP)   ;;PUSH R1 ON STACK
(1) 013406 105737 013620          TSTB   $MFLG     ;;SHOULD TYPE A MESSAGE?
(1) 013412 001450          BEQ    5$          ;;IF NOT: BR
(1) 013414 122737 000001 001210  CMPB   #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 013422 001031          BNE    3$          ;;IF NOT: BR
(1) 013424 132737 000100 001211  BITB   #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 013432 001425          BEQ    3$          ;;IF NOT: BR
(1) 013434 017600 000004          MOV    @4(SP),R0  ;;GET MESSAGE ADDR.
(1) 013440 062766 000002 000004  ADD    #2,4(SP)    ;;BUMP RETURN ADDR.

```

```

(1) 013446 005737 001170      1$:  TST      $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
(1) 013452 001375              BNE      1$              ;;IF NOT: WAIT
(1) 013454 010037 001204      MOV      R0,$MSGAD      ;;PUT ADDR IN MAILBOX
(1) 013460 105720              TSTB    (R0)+           ;;FIND END OF MESSAGE
(1) 013462 001376              BNE      2$              ;;
(1) 013464 163700 001204      SUB      $MSGAD,R0      ;;SUB START OF MESSAGE
(1) 013470 006200              ASR      R0              ;;GET MESSAGE LNGTH IN WORDS
(1) 013472 010037 001206      MOV      R0,$MSGGLT     ;;PUT LENGTH IN MAILBOX
(1) 013476 012737 000004 001170  MOV      #4,$MSGTYPE    ;;TELL APT TO TAKE MSG.
(1) 013504 000413              BR       5$              ;;
(1) 013506 017637 000004 013532 3$:  MOV      @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
(1) 013514 062766 000002 000004      ADD     #2,4(SP)       ;;BUMP RETURN ADDRESS
(3) 013522 013746 177776      MOV     177776,-(SP)   ;;PUSH 177776 ON STACK
(1) 013526 004737 014146      JSR     PC,$TYPE       ;;CALL TYPE MACRO
(1) 013532 000000              4$:  .WORD    0              ;;
(1) 013534              5$:  .WORD    0              ;;
(1) 013534 105737 013622      10$: TSTB    $FFLG          ;;SHOULD REPORT FATAL ERROR?
(1) 013540 001416              BEQ     12$             ;;IF NOT: BR
(1) 013542 005737 001210      TST     $ENV            ;;RUNNING UNDER APT?
(1) 013546 001413              BEQ     12$             ;;IF NOT: BR
(1) 013550 005737 001170      11$: TST     $MSGTYPE       ;;FINISHED LAST MESSAGE?
(1) 013554 001375              BNE     11$             ;;IF NOT: WAIT
(1) 013556 017637 000004 001172  MOV     @4(SP),$FATAL   ;;GET ERROR #
(1) 013564 062766 000002 000004      ADD     #2,4(SP)       ;;BUMP RETURN ADDR.
(1) 013572 005237 001170      INC     $MSGTYPE       ;;TELL APT TO TAKE ERROR
(1) 013576 105037 013622      12$: CLRB   $FFLG          ;;CLEAR FATAL FLAG
(1) 013602 105037 013621      CLRB   $LFLG           ;;CLEAR LOG FLAG
(1) 013606 105037 013620      CLRB   $MFLG           ;;CLEAR MESSAGE FLAG
(3) 013612 012601              MOV     (SP)+,R1        ;;POP STACK INTO R1
(3) 013614 012600              MOV     (SP)+,R0        ;;POP STACK INTO R0
(1) 013616 000207              RTS     PC              ;;RETURN
(1) 013620      000      $MFLG: .BYTE    0        ;;MESSG. FLAG
(1) 013621      000      $LFLG: .BYTE    0        ;;LOG FLAG
(1) 013622      000      $FFLG: .BYTE    0        ;;FATAL FLAG
(1)      013624      .EVEN
(1)      000200      APTSIZE=200
(1)      000001      APTENV=001
(1)      000100      APTSPOOL=100
(1)      000040      APTCSUP=040

```

923
924

.SBTTL POWER DOWN AND UP ROUTINES

(1)
(2)

::*****

:POWER DOWN ROUTINE

```
(1) 013624 012737 013770 000024 $PWRDN: MOV $IILLUP,@PWRVEC ;;SET FOR FAST UP
(1) 013632 012737 000340 000026 MOV #340,@PWRVEC+2 ;;PRIO:7
(3) 013640 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 013642 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 013644 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 013646 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 013650 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
(3) 013652 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(3) 013654 017746 165260 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
(1) 013660 010637 013774 MOV SP,$SAVR6 ;;SAVE SP
(1) 013664 012737 013676 000024 MOV $PWRUP,@PWRVEC ;;SET UP VECTOR
(1) 013672 000000 HALT
(1) 013674 000776 BR -.2 ;;HANG UP
```

(1)
(2)

::*****

:POWER UP ROUTINE

```
(1) 013676 012737 013770 000024 $PWRUP: MOV $IILLUP,@PWRVEC ;;SET FOR FAST DOWN
(1) 013704 013706 013774 MOV $SAVR6,SP ;;GET SP
(1) 013710 005037 013774 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
(1) 013714 005237 013774 1$: INC $SAVR6 ;;WAIT FOR THE INC
(1) 013720 001375 BNE 1$ ;;OF WORD
(3) 013722 012677 165212 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
(3) 013726 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
(3) 013730 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
(3) 013732 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 013734 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 013736 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 013740 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 013742 012737 013624 000024 MOV $PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 013750 012737 000340 000026 MOV #340,@PWRVEC+2 ;;PRIO:7
(1) 013756 104401 TYPE ;;REPORT THE POWER FAILURE
(1) 013760 013776 $PWRMSG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
(1) 013762 012716 MOV (PC)+,(SP) ;;RESTART AT BEGIN
(1) 013764 001542 $PWRAD: .WORD BEGIN ;;RESTART ADDRESS
(1) 013766 000002 RTI
(1) 013770 000000 $IILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
(1) 013772 000776 BR -.2 ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 013774 000000 $SAVR6: 0 ;;PUT THE SP HERE
```

```
925 013776 005015 042522 052123 PWRMSG: .ASCIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
014004 051101 044524 043516
014012 040440 052106 051105
014020 040440 050040 053517
014026 051105 043040 044501
014034 052514 042522 005015
014042 000
```

926 014044 .EVEN


```

928      .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)      ::*****
(1)      ::THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)      ::CHANGE IT TO BINARY.
(1)      ::CALL:
(1)      ::*      RDOCT          ;;READ AN OCTAL NUMBER
(1)      ::*      RETURN HERE      ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1)      ::*                               ;;HIGH ORDER BITS ARE IN $HIOCT
(1)
(1) 014044 011646      $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1) 014046 016666 000004 000002      MOV      4(SP),2(SP)      ;;INPUT NUMBER
(3) 014054 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 014056 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 014060 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(1) 014062 104412      1$:  RDLIN          ;;READ AN ASCII LINE
(1) 014064 012600      MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
(1) 014066 005001      CLR      R1          ;;CLEAR DATA WORD
(1) 014070 005002      CLR      R2
(1) 014072 112046      2$:  MOVB      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
(1) 014074 001412      BEQ      3$          ;;IF ZERO GET OUT
(1) 014076 006301      ASL      R1          ;;*2
(1) 014100 006102      ROL      R2
(1) 014102 006301      ASL      R1          ;;*4
(1) 014104 006102      ROL      R2
(1) 014106 006301      ASL      R1          ;;*8
(1) 014110 006102      ROL      R2
(1) 014112 042716 177770      BIC      #^C7,(SP)      ;;STRIP THE ASCII JUNK
(1) 014116 062601      ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
(1) 014120 000764      BR       2$          ;;LOOP
(1) 014122 005726      3$:  TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
(1) 014124 010166 000012      MOV      R1,12(SP)      ;;SAVE THE RESULT
(1) 014130 010237 014144      MOV      R2,$HIOCT
(3) 014134 012602      MCV      (SP)+,R2      ;;POP STACK INTO R2
(3) 014136 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
(3) 014140 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
(1) 014142 000002      RTI          ;;RETURN
(1) 014144 000000      $HIOCT: .WORD 0      ;;HIGH ORDER BITS GO HERE

```

```

930      ;CAUTION      THE FIRST 4 LOC. ARE OVERLAYED TO LOWER INTERRUPT LEVEL
931      ;              THE OVERLAY OCCURS AFTER THE 'SETUP' CODE
932      ;SBTTL  TYPE ROUTINE
(1)
(2)
(1)      ;*****
(1)      ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FFED.
(1)      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)      ;*
(1)      ;*CALL:
(1)      ;*1) USING A TRAP INSTRUCTION
(1)      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)      ;*OR
(1)      ;*      TYPE
(1)      ;*      MESADR
(1)      ;*
(1)      $TYPE:  TSTB      $TPFLG      ;;IS THERE A TERMINAL?
(1)      BPL      1$      ;;BR IF YES
(1)      HALT      ;;HALT HERE IF NO TERMINAL
(1)      BR      3$      ;;LEAVE
(1)      1$:  MOV      R0,-(SP)      ;;SAVE R0
(1)      MOV      @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
(1)      CMPB     #APTENV,$ENV      ;;RUNNING IN APT MODE
(1)      BNE      62$      ;;NO,GO CHECK FOR APT CONSOLE
(1)      BITB     #APTPOOL,$ENVM    ;;SPOOL MESSAGE TO APT
(1)      BEQ      62$      ;;NO,GO CHECK FOR CONSOLE
(1)      MOV      R0,61$      ;;SETUP MESSAGE ADDRESS FOR APT
(1)      JSR      PC,$ATY3      ;;SPOOL MESSAGE TO APT
(1)      61$:  .WORD    0      ;;MESSAGE ADDRESS
(1)      62$:  BITB     #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
(1)      BNE      60$      ;;YES,SKIP TYPE OUT
(1)      2$:  MOVB     (R0)+,-(SP)    ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1)      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
(1)      TST      (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
(1)      60$:  MOV      (SP)+,R0      ;;RESTORE R0
(1)      3$:  ADD      #2,(SP)      ;;ADJUST RETURN PC
(1)      RTI      ;;RETURN
(1)      4$:  CMPB     #HT,(SP)      ;;BRANCH IF <HT>
(1)      BEQ      8$      ;;BRANCH IF NOT <CRLF>
(1)      CMPB     #CRLF,(SP)
(1)      BNE      5$      ;;POP <CR><LF> EQUIV
(1)      TST      (SP)+      ;;TYPE A CR AND LF
(1)      TYPE
(1)      $CRLF
(1)      CLRB     $CHARCNT      ;;CLEAR CHARACTER COUNT
(1)      BR      2$      ;;GET NEXT CHARACTER
(1)      5$:  JSR      PC,$TYPEPC      ;;GO TYPE THIS CHARACTER
(1)      6$:  CMPB     $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
(1)      BNE      2$      ;;IF NO GO GET NEXT CHAR.
(1)      MOV      $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
(1)      ;;AND THE NULL CHAR.

```

```

(1) 014314 105366 000001      7$:  DECB    1(SP)      ;; DOES A NULL NEED TO BE TYPED?
(1) 014320 002770              BLT     6$           ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 014322 004737 014360      JSR     PC,$TYPEC   ;; GO TYPE A NULL
(1) 014326 105337 014424      DECB    $CHARCNT   ;; DO NOT COUNT AS A COUNT
(1) 014332 000770              BR      7$         ;; LOOP
(1)
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1) 014334 112716 000040      8$:  MOVB    #' (SP)  ;; REPLACE TAB WITH SPACE
(1) 014340 004737 014360      9$:  JSR     PC,$TYPEC ;; TYPE A SPACE
(1) 014344 132737 000007 014424 BITB    #7,$CHARCNT ;; BRANCH IF NOT AT
(1) 014352 001372              BNE     9$         ;; TAB STOP
(1) 014354 005726              TST     (SP)+     ;; POP SPACE OFF STACK
(1) 014356 000724              BR      2$         ;; GET NEXT CHARACTER
(1) 014360 105777 164564      $TYPEC: TSTB   @$TSPS ;; WAIT UNTIL PRINTER IS READY
(1) 014364 100375              BPL     $TYPEC
(1) 014366 116677 000002 164556 MOVB    2(SP),@$TPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 014374 122766 000015 000002 CMPB    #CR,2(SP)  ;; IS CHARACTER A CARRIAGE RETURN?
(1) 014402 001003              BNE     1$         ;; BRANCH IF NO
(1) 014404 105037 014424      CLRB    $CHARCNT  ;; YES--CLEAR CHARACTER COUNT
(1) 014410 000406              BR      $TYPEX    ;; EXIT
(1) 014412 122766 000012 000002 1$:  CMPB    #LF,2(SP)  ;; IS CHARACTER A LINE FEED?
(1) 014420 001402              BEQ     $TYPEX    ;; BRANCH IF YES
(1) 014422 105227              INCB    (PC)+     ;; COUNT THE CHARACTER
(1) 014424 000000      $CHARCNT: .WORD  0 ;; CHARACTER COUNT STORAGE
(1) 014426 000207      $TYPEX: RTS     PC
(1)
933 ;SBTTL BINARY TO ASCII AND TYPE ROUTINE
(1)
(2) ;*****
(1) ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
(1) ;*BINARY-ASCII NUMBER AND TYPE IT.
(1) ;*CALL:
(1) ;*      MOV     NUMBER,-(SP)  ;;NUMBER TO BE TYPED
(1) ;*      TYPBN  ;;TYPE IT
(1)
(1) $TYPBN: MOV     R1,-(SP)  ;;SAVE R1 ON THE STACK
(1) 014432 016601 000006      MOV     6(SP),R1  ;;GET THE INPUT NUMBER
(1) 014436 000261              SEC                ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1) 014440 112737 000060 014502 1$:  MOVB    #'0,$BIN  ;;SET CHARACTER TO AN ASCII '0'.
(1) 014446 006101              ROL     R1         ;;GET THIS BIT
(1) 014450 001406              BEQ     2$         ;;DONE?
(1) 014452 105537 014502      ADCB    $BIN      ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1) 014456 104401 014502      TYPE    , $BIN   ;;GO TYPE THIS BIT
(1) 014462 000241              CLC                ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
(1) 014464 000765              BR      1$         ;;GO DO THE NEXT BIT
(1) 014466 012601              2$:  MOV     (SP)+,R1 ;;POP THE STACK INTO R1
(1) 014470 016666 000002 000004      MOV     2(SP),4(SP) ;;ADJUST THE STACK
(1) 014476 012616              MOV     (SP)+,(SP)
(1) 014500 000002              RTI                ;;RETURN TO USER
(1) 014502 000 000      $BIN:  .BYTE  0,0 ;;STORAGE FOR ASCII CHAR. AND TERMINATOR

```

```

935
936      .SBTTL  TRAP DECODER
(1)
(2)      ;:*****
(1)      ;:THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
(1)      ;:AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)      ;:OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)      ;:GO TO THAT ROUTINE.
(1)
(1) 014504 010046      $TRAP:  MOV    R0,-(SP)      ;;SAVE R0
(1) 014506 016600 000002      MOV    2(SP),R0      ;;GET TRAP ADDRESS
(1) 014512 005740      TST    -(R0)         ;;BACKUP BY 2
(1) 014514 111000      MOVB   (R0),R0       ;;GET RIGHT BYTE OF TRAP
(1) 014516 006300      ASL    R0            ;;POSITION FOR INDEXING
(1) 014520 016000 014540      MOV    $TRPAD(R0),R0 ;;INDEX TO TABLE
(1) 014524 000200      RTS    R0            ;;GO TO ROUTINE
(1)
(1)
(1)      ;:THIS IS USE TO HANDLE THE 'GETPRI' MACRO
(1)
(1) 014526 011646      $TRAP2: MOV   (SP),-(SP) ;;MOVE THE PC DOWN
(1) 014530 016666 000004 000002      MOV   4(SP),2(SP) ;;MOVE THE PSW DOWN
(1) 014536 000002      RTI                    ;;RESTORE THE PSW
(1)
(3)      .SBTTL  TRAP TABLE
(3)
(3)      ;:THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)      ;:BY THE 'TRAP' INSTRUCTION.
(3)
(3)      ;      ROUTINE
(3)      ;      -----
(3) 014540 014526      $TRPAD: .WORD   $TRAP2
(3) 014542 014146      $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3) 014544 012730      $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 014546 012704      $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 014550 012744      $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 014552 013132      $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
(3) 014554 014430      $TYPBN  ;;CALL=TYPBN     TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
(1)
(3) 014556 011102      $GTSWR  ;;CALL=GTSWR     TRAP+7(104407)  GET SOFT-SWR SETTING
(1)
(3) 014560 011012      $CKSWR  ;;CALL=CKSWR     TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
(3) 014562 011354      $RDCHR  ;;CALL=RDCHR     TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
(3) 014564 011444      $RDLIN  ;;CALL=RDLIN     TRAP+12(104412) TTY TYPEIN STRING ROUTINE
(3) 014566 014044      $RDOCT  ;;CALL=RDOCT     TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY

```


1037
1038 015046 015 012
1039 015050 020114 020075 047514
015056 044507 020103 042524
015064 052123 053440 052111
015072 020110 051117 053440
015100 052111 047510 052125
015106 052040 051505 020124
015114 047515 052504 042514
015122 041440 047117 042516
015130 052103 042105
1040 015134 015 012
1041 015136 020127 020075 051127
015144 050101 051101 052517
015152 042116 046040 043517
015160 041511 052040 051505
015166 020124 044527 044124
015174 046440 041516 047504
015202 041440 047117 042516
015210 052103 042105 052040
015216 020117 047115 042103
015224 111
1042 015225 015 012
1043 015227 124 036440 052040
015234 050131 047505 052125
015242 046040 047517 020120
015250 047506 020122 047115
015256 042103 020111 042524
015264 052123 046440 042117
015272 046125 020105 053523
015300 052111 044103 051505
1044 015306 015 012
1045 015310 020107 020075 042507
015316 020124 042516 020127
015324 053523 052111 044103
015332 051040 043505 051511
015340 042524 020122 040526
015346 052514 105
1046 015351 015 012
1047 015353 102 036440 024040
015360 044504 044507 040524
015366 020114 047111 020051
015374 040502 042523 047440
015402 020122 042526 052103
015410 051117 040440 042104
015416 042522 051523 041440
015424 040510 043516 051505
1048 015432 015 012
1049 015434 020117 020075 042050
015442 043511 052111 046101
015450 047440 052125 020051
015456 040502 042523 040440
015464 042104 042522 051523
015472 041440 040510 043516

.SBTTL ASCII MESSAGES

PRIMEO: .BYTE 15,12

.ASCII \L = LOGIC TEST WITH OR WITHOUT TEST MODULE CONNECTED\

.BYTE 15,12

.ASCII \W = WRAPAROUND LOGIC TEST WITH MNCDO CONNECTED TO MNCDI\

.BYTE 15,12

.ASCII \T = TIMEOUT LOOP FOR MNCDI TEST MODULE SWITCHES\

.BYTE 15,12

.ASCII \G = GET NEW SWITCH REGISTER VALUE\

.BYTE 15,12

.ASCII \B = (DIGITAL IN) BASE OR VECTOR ADDRESS CHANGES\

.BYTE 15,12

.ASCII \O = (DIGITAL OUT) BASE ADDRESS CHANGE\

1050	015500	105				
	015501	015	012		.BYTE	15,12
1051	015503	110	036440	044040	.ASCIZ	\H = HELP THE OPERATOR AND RETYPE THIS LIST \
	015510	046105	020120	044124		
	015516	020105	050117	051105		
	015524	052101	051117	040440		
	015532	042116	051040	052105		
	015540	050131	020105	044124		
	015546	051511	046040	051511		
	015554	020124	020040	000040		
1052	015562	015	012		DOT:	.BYTE 15,12
1053	015564	054524	042520	052040		.ASCIZ /TYPE THE 'TEST CHARACTER' THEN DEPRESS 'RETURN KEY' /
	015572	042510	021040	042524		
	015600	052123	041440	040510		
	015606	040522	052103	051105		
	015614	020042	044124	047105		
	015622	042040	050105	042522		
	015630	051523	021040	042522		
	015636	052524	047122	045440		
	015644	054505	020042	000040		
1054						
1055	015652	047115	042103	020117	EM1:	.ASCIZ \MNCDO (DIGITAL OUT) DOES NOT EXIST <BUS ERROR> CHECK ADDRESS SWITC
	015660	042050	043511	052111		
	015666	046101	047440	052125		
	015674	004451	047504	051505		
	015702	047040	052117	042440		
	015710	044530	052123	020040		
	015716	041074	051525	042440		
	015724	051122	051117	020076		
	015732	041440	042510	045503		
	015740	040440	042104	042522		
	015746	051523	051440	044527		
	015754	041524	042510	000123		
1056	015762	047115	042103	020117	EM2:	.ASCIZ \MNCDO (DIGITAL OUT) DATA REGISTER ERROR\
	015770	042050	043511	052111		
	015776	046101	047440	052125		
	016004	004451	040504	040524		
	016012	051040	043505	051511		
	016020	042524	020122	051105		
	016026	047522	000122			
1057	016032	047115	042103	020117	EM3:	.ASCIZ \MNCDO (DIGITAL OUT) STATUS REGISTER ERROR\
	016040	042050	043511	052111		
	016046	046101	047440	052125		
	016054	004451	052123	052101		
	016062	051525	051040	043505		
	016070	051511	042524	020122		
	016076	051105	047522	000122		
1058	016104	047115	042103	020117	EM4:	.ASCIZ \MNCDO (DIGITAL OUT) INTERRUPT ERROR\
	016112	042050	043511	052111		
	016120	046101	047440	052125		
	016126	004451	047111	042524		
	016134	051122	050125	020124		
	016142	051105	047522	000122		
1059	016150	047115	042103	020111	EM5:	.ASCIZ \MNC DI (DIGITAL IN) DOES NOT EXIST <BUS ERROR> CHECK ADDRESS SWITCH

	016156	042050	043511	052111			
	016164	046101	044440	024516			
	016172	042011	042517	020123			
	016200	047516	020124	054105			
	016206	051511	004524	041074			
	016214	051525	042440	051122			
	016222	051117	020076	044103			
	016230	041505	020113	042101			
	016236	051104	051505	020123			
	016244	053523	052111	044103			
	016252	051505	000				
1060	016255	115	041516	044504	EM6:	.ASCIZ \MNC DI (DIGITAL IN)	STIMULUS REGISTER ERROR\
	016262	024040	044504	044507			
	016270	040524	020114	047111			
	016276	004451	052123	046511			
	016304	046125	051525	051040			
	016312	043505	051511	042524			
	016320	020122	051105	047522			
	016326	000122					
1061	016330	047115	042103	020111	EM7:	.ASCIZ \MNC DI (DIGITAL IN)	STATUS REGISTER ERROR\
	016336	042050	043511	052111			
	016344	046101	044440	024516			
	016352	051411	040524	052524			
	016360	020123	042522	044507			
	016366	052123	051105	042440			
	016374	051122	051117	000			
1062	016401	115	041516	044504	EM10:	.ASCIZ \MNC DI (DIGITAL IN)	DATA REGISTER ERROR\
	016406	024040	044504	044507			
	016414	040524	020114	047111			
	016422	004451	040504	040524			
	016430	051040	043505	051511			
	016436	042524	020122	051105			
	016444	047522	000122				
1063	016450	047115	042103	020111	EM11:	.ASCIZ \MNC DI (DIGITAL IN)	INTERRUPT ERROR\
	016456	042050	043511	052111			
	016464	046101	044440	024516			
	016472	044411	052116	051105			
	016500	052522	052120	042440			
	016506	051122	051117	000			
1064	016513	115	041516	044504	EM12:	.ASCIZ /MNC DI MNC DO	DIGITAL INPUT-OUTPUT WRAPAROUND STATUS ERROR/
	016520	046440	041516	047504			
	016526	042011	043511	052111			
	016534	046101	044440	050116			
	016542	052125	047455	052125			
	016550	052520	020124	051127			
	016556	050101	051101	052517			
	016564	042116	051440	040524			
	016572	052524	020123	051105			
	016600	047522	000122				
1065	016604	047115	042103	020111	EM13:	.ASCIZ /MNC DI MNC DO	DIGITAL INPUT-OUTPUT WRAPAROUND DATA ERROR/
	016612	047115	042103	004517			
	016620	044504	044507	040524			
	016626	020114	047111	052520			
	016634	026524	052517	050124			

	016642	052125	053440	040522					
	016650	040520	047522	047125					
	016656	020104	040504	040524					
	016664	042440	051122	051117					
	016672	000							
1066	016673	115	041516	044504	EM14:	.ASCIIZ	/MNC DI	INCORRECT I.D. VALUE FOR MNC DI/	
	016700	044411	041516	051117					
	016706	042522	052103	044440					
	016714	042056	020056	040526					
	016722	052514	020105	047506					
	016730	020122	047115	042103					
	016736	000111							
1067	016740	047115	042103	020111	EM15:	.ASCIIZ	/MNC DI (DIGITAL IN)	ILLEGAL INTR. OR TRAP/	
	016746	042050	043511	052111					
	016754	046101	044440	024516					
	016762	044411	046114	043505					
	016770	046101	044440	052116					
	016776	027122	047440	020122					
	017004	051124	050101	000					
1068	017011	115	041516	044504	EM16:	.ASCIIZ	/MNC DI (DIGITAL IN)	EXISTING MNC DI FAILED TO RESPOND/	
	017016	024040	044504	044507					
	017024	040524	020114	047111					
	017032	004451	054105	051511					
	017040	044524	043516	046440					
	017046	041516	044504	043040					
	017054	044501	042514	020104					
	017062	047524	051040	051505					
	017070	047520	042116	000					
1069	017075	200	051120	043517	FOUND1:	.ASCIIZ	<200>/PROGRAM DETECTED/		
	017102	040522	020115	042504					
	017110	042524	052103	042105					
	017116	000							
1070	017117	040	047115	042103	FOUND2:	.ASCIIZ	\ MNC DI (DIGITAL IN)'S \		
	017124	020111	042050	043511					
	017132	052111	046101	044440					
	017140	024516	051447	020040					
	017146	000040							
1071	017150	047125	052111	042411	DH1:	.ASCIIZ	/UNIT ERRPC OUTCSR OUTDAT/		
	017156	051122	041520	047411					
	017164	052125	051503	004522					
	017172	052517	042124	052101					
	017200	000							
1072	017201	125	044516	004524	DH5:	.ASCIIZ	/UNIT ERRPC IN CSR IN DIR IN SBR/		
	017206	051105	050122	004503					
	017214	047111	041440	051123					
	017222	044411	020116	044504					
	017230	004522	047111	051440					
	017236	051102	000						
1073	017241	125	044516	004524	DH6:	.ASCIIZ	/UNIT ERRPC IN SBR GOOD BAD/		
	017246	051105	050122	004503					
	017254	047111	051440	051102					
	017262	043411	047517	004504					
	017270	041040	042101	000					
1074	017275	125	044516	004524	DH7:	.ASCIIZ	/UNIT ERRPC IN CSR GOOD BAD/		

	017750	052120	040440	020124	
	017756	000			
1086	017757	200	046120	040505	VTMSG2: .ASCII <200>/PLEASE CHECK VECTOR SWITCHES/<200>
	017764	042523	041440	042510	
	017772	045503	053040	041505	
	020000	047524	020127	053523	
	020006	052111	044103	051505	
	020014	200			
1087	020015	011	042522	052123	.ASCIZ / RESTARTING LOGIC TEST/
	020022	051101	044524	043516	
	020030	046040	043517	041511	
	020036	052040	051505	000124	
1088	020044	015	012		SETUP0: .BYTE 15,12
1089	020046	047105	052523	042522	.ASCII \ENSURE MNCDI (DIGITAL IN) DATA SWITCH IS IN THE '-' POSITION\
	020054	046440	041516	044504	
	020062	024040	044504	044507	
	020070	040524	020114	047111	
	020076	020051	040504	040524	
	020104	051440	044527	041524	
	020112	020110	051511	044440	
	020120	020116	044124	020105	
	020126	026442	020042	047520	
	020134	044523	044524	047117	
1090	020142	015	012	000	SETUP1: .BYTE 15,12,0
1091	020145	103	047117	042516	.ASCII \CONNECT MNCDI (DIGITAL IN) TO THE TESTER\
	020152	052103	046440	041516	
	020160	044504	024040	044504	
	020166	044507	040524	020114	
	020174	047111	020051	047524	
	020202	052040	042510	052040	
	020210	051505	042524	122	
1092	020215	015	012	000	SETUP2: .BYTE 15,12,0
1093	020220	015	012		.BYTE 15,12
1094	020222	047115	042103	020111	.ASCII \MNCDI (DIGITAL IN) MUST BE CONNECTED TO MNCDO (DIGITAL OUT)\
	020230	042050	043511	052111	
	020236	046101	044440	024516	
	020244	046440	051525	020124	
	020252	042502	041440	047117	
	020260	042516	052103	042105	
	020266	052040	020117	047115	
	020274	042103	020117	042050	
	020302	043511	052111	046101	
	020310	047440	052125	051	
1095	020315	015	012	000	ADROUT: .BYTE 15,12,0
1096	020320	005015	047115	042103	.ASCIZ <15><12>/MNCDO (DIGITAL OUT) BUS ADDRESS </
	020326	020117	042050	043511	
	020334	052111	046101	047440	
	020342	052125	020051	052502	
	020350	020123	042101	051104	
	020356	051505	020123	000074	
1097	020364	005015	047115	042103	ADRIN: .ASCIZ <15><12>/MNCDI (DIGITAL IN) BUS ADDRESS </
	020372	020111	042050	043511	
	020400	052111	046101	044440	
	020406	024516	041040	051525	

```
1098 020414 040440 042104 042522
      020422 051523 036040 000
      020427 015 046412 041516 VECIN: .ASCIZ <15><12>/MNCDI (DIGITAL IN) VECTOR ADDRESS </
      020434 044504 024040 044504
      020442 044507 040524 020114
      020450 047111 020051 042526
      020456 052103 051117 040440
      020464 042104 042522 051523
      020472 036040 000
1099 020475 076 037440 000040 ENDOUT: .ASCIZ /> ? /
1100 020502 026440 000040 ADASH: .ASCIZ / - /
1101 020506 015 012 000 TCRLF: .BYTE 15,12,0
1102 .EVEN
1103 020512 020512 DT1: UNITBD,$ERRPC,OCSR,DOR,0
      020520 001410 000000
1104 020524 001532 001116 001414 DT5: UNITBD,$ERRPC,ICSR,DIR,SBR,0
      020532 001420 001424 000000
1105 020540 001532 001116 001424 DT6: UNITBD,$ERRPC,SBR,$GDDAT,$BDDAT,0
      020546 001124 001126 000000
1106 020554 001532 001116 001414 DT7: UNITBD,$ERRPC,ICSR,$GDDAT,$BDDAT,0
      020562 001124 001126 000000
1107 020570 001532 001116 001420 DT10: UNITBD,$ERRPC,DIR,$GDDAT,$BDDAT,0
      020576 001124 001126 000000
1108 020604 001532 001116 001414 DT11: UNITBD,$ERRPC,ICSR,0
      020612 000000
1109 020614 001532 001116 001404 DT12: UNITBD,$ERRPC,OCSR,ICSR,$GDDAT,$BDDAT,0
      020622 001414 001124 001126
      020630 000000
1110 020632 001532 001116 001410 DT13: UNITBD,$ERRPC,DOR,DIR,$GDDAT,$BDDAT,0
      020640 001420 001124 001126
      020646 000000
1111 020650 001532 001116 015042 DT15: UNITBD,$ERRPC,TRTO,TRFRO,0
      020656 015044 000000
1112 020662 001532 001116 001442 DT16: UNITBD,$ERRPC,TEMP,$UNIT,0
      020670 001202 000000
1113 020674 000 000 000 DFO: .BYTE 0,0,0,0,0,0,0,0,0,0,0
      020677 000 000 000
      020702 000 000 000
      020705 000 000 000
1114
1115 000001 .END
```

ABASE = 171160	17#	61													
ACDW1 = 171260	19#	61													
ACDW2 = 000000	61														
ACPUOP= 000000	61														
ADASH 020502	885	1100#													
ADDW0 = 000000	61														
ADDW1 = 000000	61														
ADDW10= 000000	61														
ADDW11= 000000	61														
ADDW12= 000000	61														
ADDW13= 000000	61														
ADDW14= 000000	61														
ADDW15= 000000	61														
ADDW2 = 000000	61														
ADDW3 = 000000	61														
ADDW4 = 000000	61														
ADDW5 = 000000	61														
ADDW6 = 000000	61														
ADDW7 = 000000	61														
ADDW8 = 000000	61														
ADDW9 = 000000	61														
ADEVCT= 000000	61														
ADEVN = 000000	61														
ADRIN 020364	379	1097#													
ADROUT 020320	366	1096#													
AENV = 000000	61														
AENVN = 000000	61														
AFATAL= 000000	61														
AMADR1= 000000	61														
AMADR2= 000000	61														
AMADR3= 000000	61														
AMADR4= 000000	61														
AMAMS1= 000000	61														
AMAMS2= 000000	61														
AMAMS3= 000000	61														
AMAMS4= 000000	61														
AMSGAD= 000000	61														
AMSGLG= 000000	61														
AMSGTY= 000000	61														
AMTYP1= 000000	61														
AMTYP2= 000000	61														
AMTYP3= 000000	61														
AMTYP4= 000000	61														
APASS = 000000	61														
APRIOR= 000000	61														
APTCSU= 000040	921#	932													
APTENV= 000001	916	921#	932												
APTSIZ= 000200	219	921#													
APTSP0= 000100	921#	932													
ASWREG= 000000	61														
AESTN= 000000	61														
AUNIT = 000000	61														
AUSWR = 000000	61														
AVECT1= 000120	18#	61	188	189	190	191	192	193	194	195					

\$NWTST= 000001

420#	430#	442#	444#	445#	446#	448#	449#	450#	452#	453#	454#	456#
457#	458#	460#	461#	462#	471#	481#	490#	499#	509#	520#	530#	541#
553#	576#	593#	604#	616#	639#	677#	692#	704#	716#	730#	747#	765#
806#	831#											

\$SOCNT 013126

919#*

\$SOMODE 013130

919#*

\$SOVER 012254

899#

\$SPASS 001176

61# 219* 243* 857* 899 1008

\$SPASTM 001006

60#

\$SPWRAD 013764

924#

\$SPWRDN 013624

219 924#

\$SPWRMG 013760

924#

\$SPWRUP 013676

924#

\$SQUES 001164

61# 274 898 916 932

\$SRDCHR 011354

898#

\$SRDDEC= ***** U

936

\$SRDLIN 011444

898#

\$SRDOCT 014044

928#

\$SRDSZ = 000040

936

\$SRTNAD 010154

898#

\$SR2A = ***** U

857#

\$SSAVRE= ***** U

936

\$SSAVR6 013774

936

\$SSCOPE 012006

924#*

\$SETUP= 000137

219 899#

\$STUP = 177777

211#

\$SSLAD 012220

211#

\$SSVPC = 000106

899#

\$SSWR = 165400

58#	15	22	61	219	420	430	442	444	445	446	448	449
8#	450	452	453	454	456	457	458	460	461	462	471	481
	499	509	520	530	541	553	576	593	604	616	639	677
	704	716	730	747	765	806	831	857	899	916	924	692

\$SSWREG 001212

61#

\$SSWRMK= 000000

219

\$STESTN 001174

22 899

\$STIMES 001160

61# 899#

\$STKB 001146

61# 219*

\$STKCNT 010456

61# 403

\$STKINT 010524

61# 445*

\$STKQEN= 010524

61# 461*

\$STKQIN 010460

61# 499*

\$STKQOU 010462

61# 541*

\$STKQSR 010464

61# 593*

\$STKS 001144

61# 730*

\$STKSRV 010574

61# 747*

\$STN = 000052

61# 765*

9#	15	420#	422	428	430#	442#	444#	445#	446#	448#	449#	450#
452#	453#	454#	456#	457#	458#	460#	461#	462#	468	471#	477	481#
487	490#	496	499#	506	509#	517	520#	527	530#	538	541#	550
553#	573	576#	593#	601	604#	612	616#	639#	677#	689	692#	701
704#	713	716#	727	730#	747#	765#	806#	831#				

\$STPB 001152

61# 932*

\$STPFLG 001157

61# 932

\$STPS 001150

61# 932

\$STRAP 014504

219 936#

CVMNB-A MNC DI DIAGNOSTIC
CVMNBA.P11

MACY11 27(654) 19-SEP-78 08:54 ^{G 7} PAGE 35-22

SEQ 0084

*CVMNBA,CVMNBA/CRF=CVMNBA
RUN-TIME: 26 13 2 SECONDS
CORE USED: 26K