

MNC DI

MNC DI DIAG
CVMNBBO

AH-B089B-MC

COPYRIGHT '78-80

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

IDENTIFICATION

SEQ 0001

PRODUCT CODE: AC-B0888-MC
PRODUCT NAME: CVMNSBC MNCDI (DIGITAL IN) DIAGNOSTIC TEST
DATE: JULY 1979
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1978, 1979
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. ABSTRACT

THE 'B' VERSION CORRECTS A PROGRAM BUG IN THE TESTER SECTION. 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
 EXPLT DATA THAT WAS EXPECTED
 RCVD DATA THAT WAS RECEIVED

*ALWAYS REPORTED

MISCELLANEOUS

CA
CA

7.1 MNC DI 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 120.
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 MNC DI INTERFACE TESTING

THIS PROGRAM DOES 'AUTO-SIZE' THE NUMBER OF MNC DI'S CONNECTED. THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 MNC DI 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. EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS TO ABOUT 20 SECONDS WITH ITERATIONS ENABLED WITH ONE MNC DI 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.

9.5 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 BIT5 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
693	T42 VERIFY MODE 01 -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
706	T43 VERIFY MODE 10 -- INPUT STROBE WILL NOT SET INPUT DATA READY FLAG
719	T44 VERIFY INPUT REPLY SETS OUTPUT DONE FLAG
734	T45 VERIFY THE MNCDO - WRAPAROUND - MNCDI DATA PATH
754	T46 VERIFY THE MNCDO - WRAPAROUND - MNCDI INVERTED DATA PATH
775	T47 VERIFY IN 10 MODE THAT SBR AND INPUT BITS SET INPUT READY
818	T50 TEST THE TRANSITION ENABLE AND TRANSITION DETECTION
845	T51 DETERMINE IF MORE MNCDI'S REMAIN TO BE TESTED
871	END OF PASS ROUTINE
888	MNCDI TEST MODULE SWITCH TIMEOUT LOOP
918	TTY INPUT ROUTINE
919	SCOPE HANDLER ROUTINE
936	ERROR HANDLER ROUTINE
937	ERROR MESSAGE TIMEOUT ROUTINE
938	BINARY TO OCTAL (ASCII) AND TYPE
939	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
940	APT COMMUNICATIONS ROUTINE
943	POWER DOWN AND UP ROUTINES
947	READ AN OCTAL NUMBER FROM THE TTY
951	TYPE ROUTINE
952	BINARY TO ASCII AND TYPE ROUTINE
955	TRAP DECODER
(3)	TRAP TABLE
1056	ASCII MESSAGES

```

15      .TITLE CVMNB-B MNCIDI  DIAGNOSTIC
(1)    : *COPYRIGHT (C) 1979
(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  EM,ERROR      ;;BASIC DEFINITION OF ERROR CALL
(1)    .EQUIV  IOT,SCOPE    ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)    : *MISCELLANEOUS DEFINITIONS
(1)    000011      HT= 11          ;;CODE FOR HORIZONTAL TAB
(1)    000012      LF= 12          ;;CODE FOR LINE FEED
(1)    000015      CR= 15          ;;CODE FOR CARRIAGE RETURN
(1)    000200      CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)    177776      PS= 177776     ;;PROCESSOR STATUS WORD
(1)    .EQUIV  PS,PSW
(1)    177774      STKLMT= 177774  ;;STACK LIMIT REGISTER
(1)    177772      PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)    177570      DSWR= 177570   ;;HARDWARE SWITCH REGISTER
(1)    177570      DDISP= 177570  ;;HARDWARE DISPLAY REGISTER
(1)
(1)    : *GENERAL PURPOSE REGISTER DEFINITIONS
(1)    000000      R0= %0          ;;GENERAL REGISTER
(1)    000001      R1= %1          ;;GENERAL REGISTER
(1)    000002      R2= %2          ;;GENERAL REGISTER
(1)    000003      R3= %3          ;;GENERAL REGISTER
(1)    000004      R4= %4          ;;GENERAL REGISTER
(1)    000005      R5= %5          ;;GENERAL REGISTER
(1)    000006      R6= %6          ;;GENERAL REGISTER
(1)    000007      R7= %7          ;;GENERAL REGISTER
(1)    000006      SP= %6          ;;STACK POINTER
(1)    000007      PC= %7          ;;PROGRAM COUNTER
(1)
(1)    : *PRIORITY LEVEL DEFINITIONS
(1)    000000      PR0= 0          ;;PRIORITY LEVEL 0
(1)    000040      PR1= 40         ;;PRIORITY LEVEL 1
(1)    000100      PR2= 100        ;;PRIORITY LEVEL 2
(1)    000140      PR3= 140        ;;PRIORITY LEVEL 3
(1)    000200      PR4= 200        ;;PRIORITY LEVEL 4
(1)    000240      PR5= 240        ;;PRIORITY LEVEL 5
(1)    000300      PR6= 300        ;;PRIORITY LEVEL 6
(1)    000340      PR7= 340        ;;PRIORITY LEVEL 7
(1)
(1)    : *'SWITCH REGISTER' SWITCH DEFINITIONS

```

```
(1) 100000 SW15= 100000
(1) 040000 SW14= 40000
(1) 020000 SW13= 20000
(1) 010000 SW12= 10000
(1) 004000 SW11= 4000
(1) 002000 SW10= 2000
(1) 001000 SW09= 1000
(1) 000400 SW08= 400
(1) 000200 SW07= 200
(1) 000100 SW06= 100
(1) 000040 SW05= 40
(1) 000020 SW04= 20
(1) 000010 SW03= 10
(1) 000004 SW02= 4
(1) 000002 SW01= 2
(1) 000001 SW00= 1
(1) .EQUIV SW09,SW9
(1) .EQUIV SW08,SW8
(1) .EQUIV SW07,SW7
(1) .EQUIV SW06,SW6
(1) .EQUIV SW05,SW5
(1) .EQUIV SW04,SW4
(1) .EQUIV SW03,SW3
(1) .EQUIV SW02,SW2
(1) .EQUIV SW01,SW1
(1) .EQUIV SW00,SW0
(1)
(1) ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1) 100000 BIT15= 100000
(1) 040000 BIT14= 40000
(1) 020000 BIT13= 20000
(1) 010000 BIT12= 10000
(1) 004000 BIT11= 4000
(1) 002000 BIT10= 2000
(1) 001000 BIT09= 1000
(1) 000400 BIT08= 400
(1) 000200 BIT07= 200
(1) 000100 BIT06= 100
(1) 000040 BIT05= 40
(1) 000020 BIT04= 20
(1) 000010 BIT03= 10
(1) 000004 BIT02= 4
(1) 000002 BIT01= 2
(1) 000001 BIT00= 1
(1) .EQUIV BIT09,BIT9
(1) .EQUIV BIT08,BIT8
(1) .EQUIV BIT07,BIT7
(1) .EQUIV BIT06,BIT6
(1) .EQUIV BIT05,BIT5
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0
(1)
(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
```

(1)	000004	ERRVEC= 4	::TIME OUT AND OTHER ERRORS
(1)	000010	RESVEC= 10	::RESERVED AND ILLEGAL INSTRUCTIONS
(1)	000014	TBITVEC=14	::" " 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)      :
(1)      :
(1)      :
(1)      :
(1)      :
(1)      :
(1)      :
(1)      :
(1)      :
23      :
24      .SBTTL  TRAP CATCHER
25      :
26      000000      .=0
27      :
28      :
29      :
30      :
31      :
41      000004      .=4
42      000004 014702 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     RESTR      ;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 .SBTTL ACT11 HOOKS  
(1)  
(2) :*****  
(1) :HOOKS REQUIRED BY ACT11  
(1) 000106 $SVPC=. ;SAVE PC  
(1) 000046 . =46  
(1) 000046 010234 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEGP  
(1) 000052 000052 . -52  
(1) 000052 000000 .WORD 0 ;;2)SET LOC.52 TO ZERO  
(1) 000106 .=$SVPC ;; RESTORE PC  
59 001000 . =1000  
60 .SBTTL APT PARAMETER BLOCK  
(1)  
(2) :*****  
(1) :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT  
(2) :*****  
(1) 001000 .SX=. ;;SAVE CURRENT LOCATION  
(1) 000024 . 24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM  
(1) 000024 000200 200 ;;FOR APT START UP  
(1) 000044 . =44 ;;POINT TO APT INDIRECT ADDRESS PNTR.  
(1) 000044 001000 $APTHDR ;;POINT TO APT HEADER BLOCK  
(1) 001000 . =.SX ;;RESET LOCATION COUNTER  
(2) :*****  
(1) :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC  
(1) :INTERFACE SPEC.  
(1)  
(1) 001000 $APTHD:  
(1) 001000 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.  
(1) 001002 001170 $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)  
(1) 001004 000030 $TSTM: .WORD 30 ;;RUN TIM OF LONGEST TEST  
(1) 001006 000030 $PASTM: .WORD 30 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)  
(1) 001010 000030 $UNITM: .WORD 30 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT  
(1) 001012 000032 .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 001100 . =1100 ;; START OF COMMON TAGS
(1) 001100 000000 $CMTAG: .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 $LPEPR: .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 KED STATUS
(1) 001146 177562 $TKB: 177562 ;; TTY KED 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: .ASCIIZ <12> ;; LINE FEED
(2) :*****
(2) .SBTTL APT MAILBOX-ETABLE
(3) :*****
(2) .EVEN
(2) 001170 $MAIL: .WORD 0 ;; 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 AMGLG ;; MESSAGE LENGTH
```



```
(2) 001210          SETABLE:          ;; APT ENVIRONMENT TABLE
(2) 001210          $ENV: .BYTE   AENV          ;; ENVIRONMENT BYTE
(2) 001211          $ENVM: .BYTE   AENVM         ;; ENVIRONMENT MODE BITS
(2) 001212          $SWREG: .WORD  ASWREG        ;; APT SWITCH REGISTER
(2) 001214          $USWR:  .WORD  AUSWR         ;; USER SWITCHES
(2) 001216          $CPUOP: .WORD  ACPUOP        ;; CPU TYPE, OPTIONS
(2)                : *                               ;; CPU TYPE
(2)                : *                               ;; 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
(2)                : *                               ;; 11/70=06, PDQ=07, Q=10
(2)                : *                               ;; BIT 10=REAL TIME CLOCK
(2)                : *                               ;; BIT 9=FLOATING POINT PROCESSOR
(2)                : *                               ;; BIT 8=MEMORY MANAGEMENT
(2) 001220          $MAMS1: .BYTE   AMAMS1        ;; HIGH ADDRESS, M.S. BYTE
(2) 001221          $MTYP1: .BYTE   AMTYP1        ;; MEM. TYPE, BLK#1
(2)                : *                               ;; MEM. TYPE BYTE -- (HIGH BYTE)
(2)                : *                               ;; 900 NSEC CORE=001
(2)                : *                               ;; 300 NSEC BIPOLAR=002
(2)                : *                               ;; 500 NSEC MOS=003
(2) 001222          $MADR1: .WORD  AMADR1        ;; HIGH ADDRESS, BLK#1
(2)                : *                               ;; MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF 'TYPE' ABOVE
(2) 001224          $MAMS2: .BYTE   AMAMS2        ;; HIGH ADDRESS, M.S. BYTE
(2) 001225          $MTYP2: .BYTE   AMTYP2        ;; MEM. TYPE, BLK#2
(2) 001226          $MADR2: .WORD  AMADR2        ;; MEM. LAST ADDRESS, BLK#2
(2) 001230          $MAMS3: .BYTE   AMAMS3        ;; HIGH ADDRESS, M.S. BYTE
(2) 001231          $MTYP3: .BYTE   AMTYP3        ;; MEM. TYPE, BLK#3
(2) 001232          $MADR3: .WORD  AMADR3        ;; MEM. LAST ADDRESS, BLK#3
(2) 001234          $MAMS4: .BYTE   AMAMS4        ;; HIGH ADDRESS, M.S. BYTE
(2) 001235          $MTYP4: .BYTE   AMTYP4        ;; MEM. TYPE, BLK#4
(2) 001236          $MADR4: .WORD  AMADR4        ;; MEM. LAST ADDRESS, BLK#4
(2) 001240          $VECT1: .WORD  AVECT1        ;; INTERRUPT VECTOR#1, BUS PRIORITY#1
(2) 001242          $VECT2: .WORD  AVECT2        ;; INTERRUPT VECTOR#2, BUS PRIORITY#2
(2) 001244          $BASE:  .WORD  ABASE         ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001246          $DEVN:  .WORD  ADEVN         ;; DEVICE MAP
(2) 001250          $CDW1:  .WORD  ACDW1         ;; CONTROLLER DESCRIPTION WORD#1
(2) 001252          $CDW2:  .WORD  ACDW2         ;; CONTROLLER DESCRIPTION WORD#2
(2) 001254          $E^END:
(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 : ITEM 1
63 : EM1,DH1,DT1,DF0 ;MNCDO BUS ERROR
64 001254 015764 017262 020624
65 001262 021006
66 : ITEM 2
66 001264 016625 017520 020726 ;MNCDO-MNCDI WRAP-AROUND STATUS ERROR
67 001272 021006
68 : ITEM 3
68 001274 016716 017563 020744 ;MNCDO-MNCDI WRAP-AROUND DATA ERROR
69 001302 021006
70 : ITEM 4
70 001304 017005 017407 020666 ;MNCDI INCORRECT I.D. VALUE
71 001312 021006
72 : ITEM 5
72 001314 016262 017313 020636 ;MNCDI BUS ERROR
73 001322 021006
74 : ITEM 6
74 001324 016367 017353 020652 ;MNCDI STIMULUS REGISTER ERROR
75 001332 021006
76 : ITEM 7
76 001334 016442 017407 020666 ;MNCDI STATUS REGISTER ERROR
77 001342 021006
78 : ITEM 10
78 001344 016513 017443 020702 ;MNCDI DATA REGISTER ERROR
79 001352 021006
80 : ITEM 11
80 001354 016562 017476 020716 ;MNCDI INTERRUPT ERROR
81 001362 021006
82 : ITEM 12
82 001364 017052 017626 020762 ;MNCDI ILLEGAL OR TRAP INTERRUPT
83 001372 021006
84 : ITEM 13
84 001374 017123 017661 020774 ;EXISTING MNCDI FAILED TO RESPOND
85 001402 021006
  
```

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	TSTR0:	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	UNITSD:	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 MNCDI IN-HOUSE TESTER MODE
218 001562 005037 001446 CLR TEMP2
219 001566 RBEGO:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001566 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
(1) 001572 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 001574 022706 001140 CMP #SWR,R6 ;;DONE?
(1) 001600 001374 BNE -6 ;;LOOP BACK IF NO
(1) 001602 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 001606 012737 012120 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001614 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
(1) 001622 012737 012440 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001630 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
(1) 001636 012737 014616 000034 MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001644 012737 000340 000036 MOV #340,@TRAPVEC+2 ;;LEVEL 7
(1) 001652 012737 013736 000024 MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
(1) 001660 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
(1) 001666 013737 010202 010174 MOV SENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
(1) 001674 005037 001160 CLR STIMES ;;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 #,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001720 012737 001720 001110 MOV #,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 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) BR 65$ ;;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 014260      MOV      #5046,$TYPE      ;CLR -(SP)
223 002044 012737 012746 014262      MOV      #12746,$TYPE+2  ;MOV #$TYPE+12,-(SP)
224 002052 012737 014272 014264      MOV      #$TYPE+12,$TYPE+4
225 002060 012737 000002 014266      MOV      #RTI,$TYPE+6   ;RTI
226 002066 004737 010636      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 010234 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      CMPB    $ENV,#1        ;:ARE WE RUNNING UNDER APT?
(2) 002130 001406      BEQ      70$           ;: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$: .ASCIZ <CRLF>#CVMNB-B MNC DI (DIGITAL IN) DIAGNOSTIC#<CRLF>
(1) 68$:
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$:    TST     TEMP2         ;:TEST IF RESTART
232 002244 001015      BNE     MTEST1        ;:BR IF YES
233 002246 104401 020156      TYPE    ,SETUP0       ;:INFORM OPERATOR ABOUT FRONT SWITCH
234 002252 022737 000001 001440      CMP     #1,DWARF      ;:TEST IF TESTER
235 002260 001005      BNE     MTEST         ;:BR IF NOT AND TELL OPERATOR THE HEADER
236 002262 104401 020257      TYPE    ,SETUP1       ;:INFORM OPERATOR ABOUT TESTER
237 002266 013737 001450 001250      MOV     TSTRO,$CDW1   ;:LOAD TESTER DEFAULT FOR WRAPAROUND
  
```

```
239 .SBTTL KEYBOARD COMMAND DECODER
240 002274 104401 015160 MTEST: TYPE, PRIMEO ;TELL THE OPER. THE TESTS AVAIL
241 002300 000240 MTEST1: NOP
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 $ERTTL ;PRIME THE TOTAL # OF ERRORS
245 002320 005037 001526 CLR EVER ;INIT. THE UNIT TYPEOUT
246 002324 004737 003010 JSR PC,FIXADR ;ENSURE BASE AND VECTOR ADDRESS IS LOADED
247 002330 104401 015674 TYPE, DOT ;INDICATE THE REST POINT
248 002334 104412 RDLIN ;GET OPER. INPUT
249 002336 013637 002512 MOV @(SP)+,RUNIT ;SAVE THE FIRST CHAR.
250 002342 142737 000040 002512 BICB #40,RUNIT ;ENSURE UPPER CASE
251 002350 122737 000102 002512 CMPB #'B,RUNIT ;TEST IF 'B'
252 002356 001002 BNE 1$ ;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 BIT #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 010354 JMP DIDATA ;RUN INPUT SWITCH TYPEOUT
270 002462 122737 000127 002512 5$: CMPB #'W,RUNIT ;TEST IF 'W'
271 002470 001005 BNE 77$ ;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 020332 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 010146 JMP $EOP
301 002636 012737 014702 000004 3$: 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 017207 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 017231 TYPE ,FOUND2 ; FINISH MESSAGE
311 002706 005737 001202 TST $UNIT ; ANY UNITS
312 002712 001002 BNE 6$ ; BR IF SOME
313 002714 000137 010146 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 5$ ; 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 INDILATOR
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 020432 ADROUT
367 003214 013746 001250 MOV $CDW1,-(SP) :ASK FOR OUTPUT ADDR.
368 003220 104402 TYPOC :GET DEFAULT VALUE
369 003222 104401 020607 TYPE, ENDOUT :TELL OPER.
370 003226 104413 RDOCT :ADD END
371 003230 005726 TST (SP)+ :AND WAIT FOR INPUT
372 003232 001403 BEQ 1$ :WAS IT A <CR> FOR DEFAULT
373 003234 016637 177776 001250 MOV -2(SP),$CDW1 :YES-BRANCH
374 003242 004737 003010 1$: JSR PC,FIXADR :NO-LOAD NEW ADDR.
375 003246 000137 002300 JMP MTEST1 :LOAD NEW ADDRESSES
                                     :RETURN
  
```

```

376
377                                     :SUBROUTINE TO ASK FOR INPUT ADRS AND VEC
378 003252 104401 BASEXC: TYPE
379 003254 020476 ADRIN
380 003256 013746 001244 MOV $BASE,-(SP) :ASK FOR INPUT ADDR.
381 003262 104402 TYPOC :GET DEFAULT
382 003264 104401 020607 TYPE, ENDOUT :TELL OPER. DEFAULT
383 003270 104413 RDOCT :ADD END
384 003272 005726 TST (SP)+ :AND WAIT FOR INPUT
385 003274 001403 BEQ 1$ :WAS IT A <CR> FOR DEFAULT
386 003276 016637 177776 001244 MOV -2(SP),$BASE :YES-BRANCH
387 003304 104401 1$: TYPE :NO-LOAD NEW ADDR
388 003306 020541 VECIN
389 003310 013746 001240 MOV $VECT1,-(SP) :ASK FOR INPUT VECTOR
390 003314 104402 TYPOC :GET DEFAULT
391 003316 104401 020607 TYPE, ENDOUT :TELL OPER THE DEFAULT
392 003322 104413 RDOCT :ADD END
393 003324 005726 TST (SP)+ :AND WAIT FOR INPUT
394 003326 001403 BEQ 2$ :WAS IT A <CR> FOR DEFAULT
395 003330 016637 177776 001240 MOV -2(SP),$VECT1 :YES BRANCH
396 003336 004737 003010 2$: JSR PC,FIXADR :LOAD NEW VECTOR
397 003342 000137 002300 JMP MTEST1 :FIX ADDRESSES AND VECTORS
                                     :AND RETYPE THE DOT
  
```

```

398
399 .SB*TL SUBROUTINE TO HANDLE CONTROL C/G
400
401 003346 105777 175572 CTRLCG: TSTB @STKS :INPUT FLAG ?
402 003352 100022 BPL 2$ :BR IF NOT
403 003354 017737 175566 003422 MOV @STKB,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$: CMP #7,CTRCHA :TEST IF CTRL G
410 003414 001001 BNE 2$
411 003416 104407 GTSWR :GET SWITCHES
412 003420 000207 2$: RTS PC :EXIT
413 003422 000000 CTRCHA: 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 014702 000004
438 003534 012737 000200 000006
439 003542 000137 007772
440 003546 012737 014702 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,$GDDAT ;LOAD EXPECTED BIT
MOV #1$,SLPADR ;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
2$: 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, @STKS ;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
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
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 7 TEST THAT BIT1 OF MNC DI STATUS REGISTER IS READ-WRITE

TST7: SCOPE
MOV #BIT1,\$GDDAT ;LOAD EXPECTED
MOV \$GDDAT,@ICSR ;LOAD BIT1 INTO MNC DI STATUS REGISTER
MOV @ICSR,\$BDDAT ;READ MNC DI STATUS REGISTER
CMP \$GDDAT,\$BDDAT ;TEST THAT IT SET
BEQ 1\$;;BR IF SET
ERROR 7 ;BIT1 OF MNC DI STATUS REGISTER FAILED TO SET
1\$: BIC \$GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,\$BDDAT ;READ MNC DI STATUS REGISTER AGAIN
CMP \$GDDAT,\$BDDAT ;TEST THE BIT
BNE TST10 ;;BR IF CLEARED
ERROR 7 ;BIT1 OF MNC DI STATUS REGISTER FAILED TO CLEAR

*TEST 10 TEST THAT BIT2 OF MNC DI STATUS REGISTER IS READ-WRITE

TST10: SCOPE
MOV #BIT2,\$GDDAT ;LOAD EXPECTED
MOV \$GDDAT,@ICSR ;LOAD BIT2 INTO MNC DI STATUS REGISTER
MOV @ICSR,\$BDDAT ;READ MNC DI STATUS REGISTER
CMP \$GDDAT,\$BDDAT ;TEST THAT IT SET
BEQ 1\$;;BR IF SET
ERROR 7 ;BIT2 OF MNC DI STATUS REGISTER FAILED TO SET
1\$: BIC \$GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,\$BDDAT ;READ MNC DI STATUS REGISTER AGAIN
CMP \$GDDAT,\$BDDAT ;TEST THE BIT
BNE TST11 ;;BR IF CLEARED
ERROR 7 ;BIT2 OF MNC DI STATUS REGISTER FAILED TO CLEAR

*TEST 11 TEST THAT BIT3 OF MNC DI STATUS REGISTER IS READ-WRITE

TST11: SCOPE
MOV #BIT3,\$GDDAT ;LOAD EXPECTED
MOV \$GDDAT,@ICSR ;LOAD BIT3 INTO MNC DI STATUS REGISTER
MOV @ICSR,\$BDDAT ;READ MNC DI STATUS REGISTER
CMP \$GDDAT,\$BDDAT ;TEST THAT IT SET
BEQ 1\$;;BR IF SET
ERROR 7 ;BIT3 OF MNC DI STATUS REGISTER FAILED TO SET
1\$: BIC \$GDDAT,@ICSR ;CLEAR THAT BIT
MOV @ICSR,\$BDDAT ;READ MNC DI STATUS REGISTER AGAIN
CMP \$GDDAT,\$BDDAT ;TEST THE BIT
BNE TST12 ;;BR IF CLEARED
ERROR 7 ;BIT3 OF MNC DI STATUS REGISTER FAILED TO CLEAR

```
452
(4) *****
:*TEST 12 TEST THAT BIT4 OF MNCDI STATUS REGISTER IS READ-WRITE
(4) *****
(3) 004336 000004 TST12: SCOPE
(1) 004340 012737 000020 001124 MOV #BIT4,$GDDAT ;LOAD EXPECTED
(1) 004346 013777 001124 175040 MOV $GDDAT,@ICSR ;LOAD BIT4 INTO MNCDI STATUS REGISTER
(1) 004354 017737 175034 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
(1) 004362 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THAT IT SET
(2) 004370 001401 BEQ 1$ ;:BR IF SET
(1) 004372 104007 ERROR 7 ;BIT4 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1) 004374 043777 001124 175012 1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
(1) 004402 017737 175006 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
(1) 004410 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THE BIT
(3) 004416 001001 BNE 1ST13 ;:BR IF CLEARED
(1) 004420 104007 ERROR 7 ;BIT4 OF MNCDI STATUS REGISTER FAILED TO CLEAR
453
(4) *****
:*TEST 13 TEST THAT BIT5 OF MNCDI STATUS REGISTER IS READ-WRITE
(4) *****
(3) 004422 000004 TST13: SCOPE
(1) 004424 012737 000040 001124 MOV #BIT5,$GDDAT ;LOAD EXPECTED
(1) 004432 013777 001124 174754 MOV $GDDAT,@ICSR ;LOAD BIT5 INTO MNCDI STATUS REGISTER
(1) 004440 017737 174750 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
(1) 004446 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THAT IT SET
(2) 004454 001401 BEQ 1$ ;:BR IF SET
(1) 004456 104007 ERROR 7 ;BIT5 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1) 004460 043777 001124 174726 1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
(1) 004466 017737 174722 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
(1) 004474 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THE BIT
(3) 004502 001001 BNE TST14 ;:BR IF CLEARED
(1) 004504 104007 ERROR 7 ;BIT5 OF MNCDI STATUS REGISTER FAILED TO CLEAR
454
(4) *****
:*TEST 14 TEST THAT BIT6 OF MNCDI STATUS REGISTER IS READ-WRITE
(4) *****
(3) 004506 000004 TST14: SCOPE
(1) 004510 012737 000100 001124 MOV #BIT6,$GDDAT ;LOAD EXPECTED
(1) 004516 013777 001124 174670 MOV $GDDAT,@ICSR ;LOAD BIT6 INTO MNCDI STATUS REGISTER
(1) 004524 017737 174664 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
(1) 004532 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THAT IT SET
(2) 004540 001401 BEQ 1$ ;:BR IF SET
(1) 004542 104007 ERROR 7 ;BIT6 OF MNCDI STATUS REGISTER FAILED TO SET
(1)
(1) 004544 043777 001124 174642 1$: BIC $GDDAT,@ICSR ;CLEAR THAT BIT
(1) 004552 017737 174636 001126 MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER AGAIN
(1) 004560 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST THE BIT
(3) 004566 001001 BNE TST15 ;:BR IF CLEARED
(1) 004570 104007 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) 004636 017737 174552 001126
(1) 004644 023737 001124 001126
(3) 004652 001001
(1) 004654 104007
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) 004722 017737 174466 001126
(1) 004730 023737 001124 001126
(3) 004736 001001
(1) 004740 104007
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) 005006 017737 174402 001126
(1) 005014 023737 001124 001126
(3) 005022 001001
(1) 005024 104007
```

*TEST 15 TEST THAT BIT8 OF MNCDI STATUS REGISTER IS READ-WRITE

TST15: SCOPE
MOV #BIT8,\$GDDAT ;LOAD EXPECTED
MOV \$GDDAT,@ICSR ;LOAD BIT8 INTO MNCDI STATUS REGIS
MOV @ICSR,\$BDDAT ;READ MNCDI STATUS REGISTER
CMP \$GDDAT,\$BDDAT ;TEST THAT IT SET
BEQ 1\$;;BR IF SET
ERROR 7 ;BIT8 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 TST16 ;;BR IF CLEARED
ERROR 7 ;BIT8 OF MNCDI STATUS REGISTER FAILED TO CLEAR

*TEST 16 TEST THAT BIT9 OF MNCDI STATUS REGISTER IS READ-WRITE

TST16: SCOPE
MOV #BIT9,\$GDDAT ;LOAD EXPECTED
MOV \$GDDAT,@ICSR ;LOAD BIT9 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 ;BIT9 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 TST17 ;;BR IF CLEARED
ERROR 7 ;BIT9 OF MNCDI STATUS REGISTER FAILED TO CLEAR

*TEST 17 TEST THAT BIT12 OF MNCDI STATUS REGISTER IS READ-WRITE

TST17: SCOPE
MOV #BIT12,\$GDDAT ;LOAD EXPECTED
MOV \$GDDAT,@ICSR ;LOAD BIT12 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 ;BIT12 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 TST20 ;;BR IF CLEARED
ERROR 7 ;BIT12 OF MNCDI STATUS REGISTER FAILED TO CLEAR


```
460
(4)
(4)
(3) 005026 000004
(1) 005030 012737 040000 001124
(1) 005036 013777 001124 174350
(1) 005044 017737 174344 001126
(1) 005052 023737 001124 001126
(2) 005060 001401
(1) 005062 104007
(1)
(1) 005064 043777 001124 174322
(1) 005072 017737 174316 001126
(1) 005100 023737 001124 001126
(3) 005106 001001
(1) 005110 104007
```

```
*****
*TEST 20 TEST THAT BIT 14 OF MNCDI STATUS REGISTER IS READ-WRITE
*****
TST20: SCOPE
MOV #BIT14,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@ICSR ;LOAD BIT14 INTO MNCDI STATUS REGISTER
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 7 ;BIT14 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 TST21 ;;BR IF CLEARED
ERROR 7 ;BIT14 OF MNCDI STATUS REGISTER FAILED TO CLEAR
```

```
461
(4)
(4)
(3) 005112 000004
(2) 005114 012737 000040 001160
(1) 005122 012777 040426 174264
(1) 005130 005037 001124
(1) 005134 000005
(1) 005136 052777 000100 174000
(1) 005144 017737 174244 001126
(3) 005152 001401
(1) 005154 104007
```

```
*****
*TEST 21 ENSURE THAT 'RESET' CLEARS THE MNCDI STATUS REGISTER
*****
TST21: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #40426,@ICSR ;LOAD BITS TO BE RESET
CLR $GDDAT ;CLEAR EXPECTED
RESET ;CLEAR THE DEVICE
BIS #BIT6,@STKS ;ENABLE TKB INTR.
MOV @ICSR,$BDDAT ;READ MNCDI STATUS REGISTER
BEQ TST22 ;;BR IF CLEARED
ERROR 7 ;MNCDI STATUS REGISTER FAILED TO CLEAR WITH 'RESET'
```

```
462
(3)
(3)
(2) 005156 000004
463 005160 012777 040426 174226
464 005166 105077 174224
465 005172 012737 000026 001124
466 005200 017737 174210 001126
467 005206 023737 001124 001126
468 005214 001401
469 005216 104007
470
```

```
*****
*TEST 22 VERIFY HIGH BYTE OPERATION ON THE INPUT STATUS REGISTER
*****
TST22: SCOPE
MOV #40426,@ICSR ;LOAD INPUT REG. BIT
CLRB @ICSR1 ;CLEAR HIGH BYTE
MOV #BIT4!BIT2!BIT1,$GDDAT ;LOAD EXPECTED
MOV @ICSR,$BDDAT ;READ INPUT STATUS REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST23 ;;BR IF SAME
ERROR 7 ;CLEARING HIGH BYTE CHANGED LOW BYTE
```

```
471
(3)
(3)
(2) 005220 000004
472 005222 012777 040426 174164
473 005230 105077 174160
474 005234 012737 040400 001124
475 005242 017737 174146 001126
476 005250 023737 001124 001126
477 005256 001401
478 005260 104007
479
```

```
*****
*TEST 23 VERIFY LOW BYTE OPERATION ON THE INPUT STATUS REGISTER
*****
TST23: SCOPE
MOV #40426,@ICSR ;LOAD INPUT REG.
CLRB @ICSR ;CLEAR LOW BYTE
MOV #40400,$GDDAT ;LOAD EXPECTED
MOV @ICSR,$BDDAT ;READ INPUT STATUS REG.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST24 ;;BR IF SAME
ERROR 7 ;CLEARING LOW BYTE CHANGED HIGH BYTE
```

```
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 ;;DC 40 ITERATIONS  
500 005374 005037 001124 CLR $GDDAT ;LOAD EXPECTED DATA  
501 005400 012777 004000 174006 MOV #BITEXT,@ICSR ;GENERATE MAINT. STROBE  
502 005406 000005 RESET  
503 005410 052777 000100 173526 BIS #BIT6,@STKS ;ENABLE TKB INTR.  
504 005416 017737 173772 001126 MOV @ICSR,$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) *TEST 30 VERIFY THAT 'OVERRUN ERROR' SETS  
(3) .....  
(2) 005514 000004  
521 005516 012737 100202 001124 TST30: SCOPE  
522 005524 012777 000002 173662 MOV #BIT15!BIT7,BIT1,$GDDAT ;LOAD EXPECTED  
523 005532 052777 004200 173654 MOV #BIT1,@ICSR ;SET STROBE MODE  
524 005540 052777 104200 173646 BIS #BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE  
525 005546 017737 173642 001126 BIS #BIT15!BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE AGAIN  
526 005554 023737 001124 001126 MOV @ICSR,$BDDAT ;READ INPUT STATUS REGISTER  
527 005562 001401 BEQ $GDDAT,$BDDAT ;COMPARE  
528 005564 104007 ERROR TST31 ;;BR IF SAME  
529 ;'OVER RUN' FAILED TO SET
```

```
530 .....  
(3) *TEST 31 VERIFY THAT 'OVERRUN ERROR' CAN BE WRITTEN TO A ZERO  
(3) .....  
(2) 005566 000004  
531 005570 012737 000202 001124 TST31: SCOPE  
532 005576 012777 000002 173610 MOV #BIT7!BIT1,$GDDAT ;LOAD EXPECTED VALUE  
533 005604 052777 004200 173602 MOV #BIT1,@ICSR ;SET STROBE MODE  
534 005612 052777 104200 173574 BIS #BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE  
535 005620 105077 173572 BIS #BIT15!BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE AGAIN  
536 005624 017737 173564 001126 CLRB @ICSR1 ;CLEAR HIGH BYTE OF THE INPUT STATUS REGISTER  
537 005632 023737 001124 001126 MOV @ICSR,$BDDAT ;READ INPUT STATUS REGISTER  
538 005640 001401 BEQ $GDDAT,$BDDAT ;COMPARE  
539 005642 104007 ERROR TST32 ;;BR IF SAME  
540 ;'OVERRUN ERROR' FAILED TO BE WRITTEN TO A ZERO
```

```
541 .....  
(3) *TEST 32 VERIFY THAT 'RESET' CLEARS 'OVERRUN ERROR'  
(3) .....  
(2) 005644 000004  
(1) 005646 012737 000040 001160 TST32: SCOPE  
542 005654 005037 001124 MOV #40,$TIMES ;;DO 40 ITERATIONS  
543 005660 012777 000002 173526 CLR $GDDAT ;CLEAR EXPECTED  
544 005666 052777 004000 173520 MOV #BIT1,@ICSR ;SET STROBE MODE  
545 005674 052777 004000 173512 BIS #BITEXT,@ICSR ;GENERATE MAINT. STROBE  
546 005702 000005 BIS #BITEXT,@ICSR ;GENERATE MAINT. STROBE AGAIN  
547 005704 052777 000100 173232 RESET ;ENABLE TKB INTR.  
548 005712 017737 173476 001126 MOV @ICSR,$BDDAT ;READ INPUT REGISTER  
549 005720 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE  
550 005726 001401 BEQ TST33 ;;BR IF SAME  
551 005730 104007 ERROR TST33 ;RESET FAILED TO CLEAR 'OVERRUN ERROR'
```

```

553
(3)
(3)
(2) 005732 000004
554 005734 005037 001124
555 005740 012777 010000 173446
556 005746 017737 173446 001126
557 005754 023737 001124 001126
558 005762 001401
559 005764 104007
560 005766 012777 010060 173420 1$:
561 005774 012737 177777 001124
562 006002 017737 173412 001126
563 006010 001001
564 006012 104007
565 006014 023737 001124 001126 2$:
566 006022 001401
567 006024 104007
568
569 006026 012777 010000 173360 3$:
570 006034 005037 001124
571 006040 017737 173354 001126
572 006046 023737 001124 001126
573 006054 001401
574 006056 104007
575
575
(3)
(3)
(2) 006060 000004
577 006062 012737 006076 001110
578 006070 012737 000001 001442
579 006076 012777 010060 173310
580 006104 013737 001442 001124
581 006112 005137 001124
582 006116 017700 173276
583 006122 013777 001442 173270
584 006130 042777 000040 173256
585 006136 052777 000002 173250
586 006144 017737 173250 001126
587 006152 023737 001124 001126
588 006160 001401
589 006162 104010
590 006164 006337 001442 2$:
591 006170 001342
  
```

```

*****
*TEST 33 VERIFY INVERT DATA FUNCTION
*****
TST33: SCOPE
      CLR $GDDAT ;LOAD EXPECTED
      MOV #BITDAT,@ICSR ;SET INPUT INHIBIT
      MOV @DIR,$BDDAT ;READ INPUT
      CMP $GDDAT,$BDDAT ;COMPARE
      BEQ 1$ ;;BR IF SAME
      ERROR 7 ;INPUT INHIBIT FAILED TO INHIBIT INPUT
1$: MOV #BITDAT!BIT5!BIT4,@ICSR ;SET INVERT DATA AND INPUT INHIBIT
      MOV #-1,$GDDAT ;LOAD EXPECTED
      MOV @DIR,$BDDAT ;READ INPUT
      BNE 2$ ;BR IF NON-ZERO
      ERROR 7 ;INVERT DATA FUNCTION FAILED
2$: CMP $GDDAT,$BDDAT ;COMPARE DATA
      BEQ 3$ ;;BR IF SAME
      ERROR 7 ;INVERT DATA - DATA PATH ERROR
3$: MOV #BITDAT,@ICSR ;SET INPUT INHIBIT
      CLR $GDDAT ;CLEAR EXPECTED
      MOV @DIR,$BDDAT ;READ INPUT
      CMP $GDDAT,$BDDAT ;COMPARE
      BEQ TST34 ;BR IF SAME
      ERROR 7 ;INVERT DATA FUNCTION OR INPUT INHIBIT FAILED
  
```

```

*****
*TEST 34 VERIFY EACH BIT OF THE MNCI INPUT DATA REGISTER CAN BE CLEARED
*****
TST34: SCOPE
      MOV #1$,SLPERR ;LOAD LOOP ADDRESS ON ERROR
      MOV #BIT0,TEMP ;LOAD INITIAL BIT
1$: MOV #BITDAT!BIT4!BIT5,@ICSR ;LOAD INHIBIT INPUT AND INVERT DATA
      MOV TEMP,$GDDAT ;LOAD EXPECTED
      COM $GDDAT ;MAKE OPPOSITE
      MOV @DIR,R0 ;READ INPUT
      MOV TEMP,@DIR ;CLEAR THE INPUT BIT
      RLC #BIT5,@ICSR ;REM INVERT DATA BITOVE
      L.S #BIT1,@ICSR ;ENABLE EXT. STROBE TO PREVENT DATA INPUT BEING
      MOV @DIR,$BDDAT ;READ INPUT REG.
      CMP $GDDAT,$BDDAT ;COMPARE
      BEQ 2$ ;;BR IF SAME
      ERROR 10 ;INPUT REGISTER BIT FAILED TO CLEAR
2$: ASL TEMP ;SHIFT THE DATA
      BNE 1$ ;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!BIT5.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!BIT5!BIT4!BIT1,@ICSR ;DISABLE INPUTS, ENABLF INVERT DATA, EXT
BIS #BITEXT!BIT7,@ICSR ;GENERATE MAINT. STROBE
BIC #BIT5,@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) *****  
617 (3) *TEST 37 INTERRUPT TEST -- VERIFY MNC DI INTERRUPTS VIA DATA READY VECTOR  
618 (2) *****  
619 TST37: SCOPE  
620 MOV #64$, $LPADR  
621 64$: MOV #1$, @DIDINV ;LOAD RETURN VECTOR  
622 MOV #200, @DIDINS ;LOAD RETURN LEVEL  
623 CLR -(SP)  
624 MOV #10$, -(SP)  
625 RTI  
626 10$: MOV #BIT6!BIT1, @ICSR ;SET STROBE MODE  
627 BIS #BITEXT!BIT7, @ICSR ;GENERATE MAINT. STROBE  
628 NOP  
629 NOP  
630 NOP  
631 CLR @ICSR ;CLEAR STATUS  
632 ERROR 11 ;MNC DI INPUT DATA READY FAILED TO INTERRUPT  
633 BR 2$ ;;RESET VECTOR  
634 1$: CMP (SP)+, (SP)+ ;CLEAN STACK  
635 2$: CLR @ICSR ;CLEAR DEVICE  
636 MOV DIDINS, @DIDINV  
637 MOV #4700, @DIDINS
```

```
638 (3) *****  
639 (3) *TEST 40 INTERRUPT TEST -- VERIFY MNC DI INTERRUPTS VIA OVERRUN ERROR  
640 (2) *****  
641 TST40: SCOPE  
642 MOV #1$, @DIEINV ;LOAD RETURN VECTOR  
643 MOV #200, @DIEINS ;LOAD RETURN STATUS  
644 CLR -(SP)  
645 MOV #10$, -(SP)  
646 RTI  
647 10$: MOV #BIT14!BIT1, @ICSR ;ENABLE INTR. AND STROBE MODE  
648 BIS #BIT15!BITEXT!BIT7, @ICSR ;GENERATE MAINT. STROBE  
649 BIS #BIT15!BITEXT!BIT7, @ICSR ;GENERATE MAINT. STROBE AGAIN TO SET OVE  
650 NOP  
651 NOP  
652 NOP  
653 CLR @ICSR ;DISABLE INTR.  
654 ERROR 11 ;MNC DI FAILED TO INTERRUPT ON 'OVERRUN ERROR'  
655 BR 2$ ;;RESET VECTOR  
656 1$: CMP (SP)+, (SP)+ ;CLEAN THE STACK  
657 2$: CLR @ICSR ;CLEAR DEVICE  
658 MOV DIEINS, @DIEINV  
659 MOV #4700, @DIEINS  
660 CLR -(SP)  
661 MOV #11$, -(SP)  
662 RTI  
663 11$:
```

665
676
677
(3)
(3)
(2)
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
(3)
(3)
(2)
694
695
696
697
698
699
700
701
702
703
704
705
706
(3)
(3)
(2)
707
708
709
710
711
712
713
714
715
716
717

006566 000004
006570 005737 001440
006574 001002
006576 000137 007772
006602 012777 172610 172610 1\$:
006610 005077 172610
006614 005077 172574
006620 005077 172560
006624 012777 025252 172556
006632 012737 000200 001124
006640 004737 010550
006644 017737 172544 001126
006652 023737 001124 001126
006660 001401
006662 10400?

006664 000004
006666 012777 177777 172524
006674 005077 172524
006700 012777 000002 172506
006706 005077 172472
006712 012777 052525 172470
006720 012737 000202 001124
006726 004737 010550
006732 017737 172456 001126
006740 023737 001124 001126
006746 001401
006750 104002

006752 000004
006754 012777 177777 172436
006762 005077 172436
006766 012777 000004 172420
006774 005077 172404
007000 012777 070707 172402
007006 012737 000004 001124
007014 004737 010550
007020 017737 172370 001126
007026 023737 001124 001126
007034 001401
007036 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
        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,$GDDAT         ;LOAD EXPECTED
        JSR    PC,DELAYO             ;DELAY A SHORT TIME
        MOV    @ICSR,$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,$GDDAT   ;LOAD EXPECTED
        JSR    PC,DELAYO             ;DELAY A SHORT TIME
        MOV    @ICSR,$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,$GDDAT         ;LOAD EXPECTED
        JSR    PC,DELAYO             ;DELAY A SHORT TIME
        MOV    @ICSR,$BDDAT         ;READ STATUS
        CMP    $GDDAT,$BDDAT        ;COMPARE
        BEQ    TST44                ;:BR IF CLEARED
        ERROR 2                      ;MODE 10 -- EXT. STROBE SET INPUT DATA READY FL
```



```

719      ::*****
(3)      :*TEST 44      VERIFY INPUT REPLY SETS OUTPUT DONE FLAG
(3)      :*****
(2) 007040 000004
720 007042 005077 172342
721 007046 012737 000200 001124
722 007054 022737 000001 001440
723 007062 001003
724 007064 012737 100000 001124
725 007072 005077 172306
726 007076 012777 004200 172310
727 007104 005077 172304
728 007110 004737 010550
729 007114 017737 172264 001126
730 007122 023737 001124 001126
731 007130 001401
732 007132 104002
733
734      ::*****
(3)      :*TEST 45      VERIFY THE MNCDO - WRAPAROUND - MNC DI DATA PATH
(3)      :*****
(2) 007134 000004
(1) 007136 012737 000100 001160
735 007144 012777 177777 172246
736 007152 005077 172226
737 007156 012777 000002 172230
738 007164 012737 000001 001440
739
740 007172 013737 001442 001124
741 007200 012777 000002 172206
742 007206 013777 001124 172174
743 007214 105777 172174
744 007220 100375
745 007222 017737 172172 001126
746 007230 023737 001124 001126
747 007236 001401
748 007240 104003
749
750 007242 006337 001442
751 007246 001351
752

```

```

TST44: SCOPE
        CLR @DOR ;CLEAR OUTPUT DATA
        MOV #BIT7,$GDDAT ;LOAD EXPECTED
        CMP #1,DWARF ;CHECK IF IN TESTER MODE
        BNE 1$ ;BR IF NOT
        MOV #BIT15,$GDDAT ;LOAD TESTER WRAPAROUND STATUS
1$:     CLR @OCSR ;CLEAR OUTPUT DONE FLAG
        MOV #BITTEXT!BIT7,@ICSR ;SET INPUT READY FLAG
        CLR @ICSR ;CLEAR INPUT READY FLAG<GEN. INPUT REPLY>
        JSR PC,DELAYO ;DELAY A SHORT TIME
        MOV @OCSR,$BDDAT ;READ OUTPUT STATUS
        CMP $GDDAT,$BDDAT ;COMPARE
        BEQ TST45 ;:BR IF SET
        ERROR 2 ;INPUT REPLY FAILED TO SET OUTPUT DONE FLAG

TST45: SCOPE
        MOV #100,$TIMES ;:DO 100 ITERATIONS
        MOV #-1,@DIR ;CLEAR INPUT REG.
        CLR @OCSR ;CLEAR OUTPUT STATUS
        MOV #BIT1,@ICSR ;INITILIZE THE INPUT STATUS REGISTER
        MOV #BIT0,TEMP ;LOAD EXPECTED
1$:     MOV TEMP,$GDDAT ;LOAD TYPEOUT EXPECTED
        MOV #BIT1,@ICSR ;INITILIZE THE INPUT STATUS REG.
        MOV $GDDAT,@DOR ;LOAD OUTPUT DATA REG.
2$:     TSTB @ICSR ;WAIT FOR STROBE PULSE
        BPL 2$ ; TO OCCUR
        MOV @DIR,$BDDAT ;READ INPUT DATA REGISTER
        CMP $GDDAT,$BDDAT ;COMPARE
        BEQ 3$ ;:BR IF SAME
        ERROR 3 ;INPUT DATA PATH ERROR
3$:     ASL TEMP ;TRY NEXT BIT
        BNE 1$ ;BR IF MORE BITS

```

```

754
(3)
(3)
(2) 007250 000004
(1) 007252 012737 000100 001160
755 007260 012777 177777 172132
756 007266 005077 172112
757 007272 012777 000062 172114
758 007300 012737 000001 001442
759
760 007306 013777 001442 172074 1$: MOV TEMP,@DOR ;LOAD OUTPUT DATA REG.
761 007314 013737 001442 001124 MOV TEMP,$GDDAT ;GET THE BIT
762 007322 005137 001124 COM $GDDAT ;INVERT EXPECTED INPUT DATA
763 007326 105777 172062 2$: TSTB @ICSR ;WAIT FOR INPUT READY
764 007332 100375 BPL 2$ ; TO OCCUR
765 007334 017737 172060 001126 MOV @DIR,$BDDAT ;READ INPUT DATA REG.
766 007342 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
767 007350 001401 BEQ 3$ ;:BR IF SAME
768 007352 104003 ERROR 3 ;INVERTED INPUT DATA PATH ERROR
769
770 007354 042777 000200 172032 3$: BIC #BIT7,@ICSR ;CLEAR INPUT READY
771 007362 006337 001442 ASL TEMP ;TRY NEXT BIT
772 007366 001347 BNE 1$ ;BR IF MORE BITS
773

```

CV
CV

```

775      ::*****
(3)      ::*TEST 47      VERIFY IN 10 MODE THAT SBR AND INPUT BITS SET INPUT READY
(3)      ::*****
(2) 007370 000004
(1) 007372 012737 000100 001160 TST47: SCOPE
776      :          MOV      #100,$TIMES      ;;DO 100 ITERATIONS
777      :          INPUT DATA READY FLAG
778      :          LOAD A FLOATING 1 ACROSS THE SBR
779      :          VERIFY THAT ONLY THE CORRECT BIT SET DATA READY
780      :          CLR      @DOR
781 007400 005077 172004          MOV      #BIT0,TEMP      ;LOAD INITIAL BIT
782 007404 012737 000001 001442
783      :          CLR      @DOR      ;CLEAR OUTPUT BITS
784 007412 005077 171772 1$:      CLR      @SBR      ;CLEAR SBR REG.
785 007416 005077 172002          MOV      #BIT2,@ICSR      ;CLEAR INPUT READY AND SET MODE 10
786 007422 012777 000004 171764          MOV      #-1,@DIR      ;CLEAR INPUT REG.
787 007430 012777 177777 171762
788      :          MOV      TEMP,@SBR      ;LOAD SBR REG.
789 007436 013777 001442 171760          BIC      #BIT7,@ICSR      ;CLEAR INPUT READY BIT
790 007444 042777 000200 171742          MOV      TEMP,@DOR      ;LOAD OUTPUT REG.
791 007452 013777 001442 171730
792      :          MOV      #BIT15!BIT7!BIT2,$GDDAT ;LOAD EXPECTED STATUS
793 007460 012737 100204 001124          JSR      PC,DELAY0      ;DELAY A SHORT TIME
794 007466 004737 010550          MOV      @ICSR,$BDDAT ;READ STATUS
795 007472 017737 171716 001126          CMP      $GDDAT,$BDDAT ;COMPARE
796 007500 023737 001124 001126          BEQ      2$              ;:BR IF SET
797 007506 001401          ERROR      2              ;INPUT DATA READY FLAG FAILED
798      :          ;TO SET IN MODE 10 <S* MILUS MODE>
799
800      :NOW LOAD ALL BITS EXCEPT THE FLOATING BIT AND ENSURE INPUT DATA READY DOES NOT SET
801 007512 013737 001442 001444 2$:      MOV      TEMP,TEMP1      ;COPY EXPECTED
802 007520 005137 001444          COM      TEMP1          ;USE REVERSE PATTERN
803 007524 005077 171660          CLR      @DOR          ;CLEAR OUTPUT REG.
804 007530 012777 177777 171662          MOV      #-1,@DIR      ;CLEAR INPUT REG.
805 007536 042777 100200 171650          BIC      #BIT15!BIT7,@ICSR ;CLEAR INPUT DATA READY
806 007544 012737 000004 001124          MOV      #BIT2,$GDDAT ;LOAD EXPECTED
807
808      :          MOV      TEMP1,@DOR      ;LOAD ALL OTHER DATA BITS
809 007552 013777 001444 171630          JSR      PC,DELAY0      ;DELAY FOR A SHORT TIME
810 007560 004737 010550          MOV      @ICSR,$BDDAT ;READ INPUT STATUS
811 007564 017737 171624 001126          CMP      $GDDAT,$BDDAT ;COMPARE
812 007572 023737 001124 001126          BEQ      3$              ;:BR IF CLEARED
813 007600 001401          ERROR      2              ;INPUT DATA READY FLAG SET IN ERROR
814      :          ;UNEXPECTED SBR BIT SET INPUT DATA READY
815 007604 006337 001442 3$:      ASL      TEMP          ;TRY NEXT BIT
816 007610 001300          BNE      1$              ;BR IF MORE BITS
817
818      ::*****
(3)      ::*TEST 50      TEST THE TRANSITION ENABLE AND TRANSITION DETECTION
(3)      ::*****
(2) 007612 000004
819 007614 005077 171570
820 007620 012777 177777 171572 TST50: SCOPE
821 007626 012737 010000 001442          CLR      @DOR          ;CLEAR INPUT REGISTER
822 007634 012777 000404 171552          MOV      #-1,@DIR      ;LOAD INITIAL TRANSITION BIT
823 007642 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
    
```

```

824 007650 042777 100200 171536 1$: BIC #BIT15!BIT7,@ICSR ;CLEAR READY
825 007656 013777 001442 171540 MOV TEMP,@SBR ;LOAD STIMULUS REG.
826 007664 013777 001442 171516 MOV TEMP,@DOR ;LOAD INPUT REG. <VIA OUTPUT REG.>
827 007672 004737 010550 JSR PC,DELAYO ;DELAY FOR A SHORT TIME
828 007676 017737 171512 001126 MOV @ICSR,$BDDAT ;READ INPUT STATUS
829 007704 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
830 007712 001401 BEQ 2$ ;;BR IF SAME
831 007714 104002 ERROR 2 ;TRANSITION ENABLE OR TRANSITION TO A ONE FAILED
832 ;NOW REMOVE THE TRANSITION DATA BIT (THIS SHOULD CAUSE THE INPUT READY FLAG TO SET AGAIN
833 007716 012777 177777 171474 2$: MOV #-1,@DIR ;CLEAR INPUT REG
834 007724 042777 100200 171462 BIC #BIT15!BIT7,@ICSR ;CLEAR INPUT READY FLAG
835 007732 043777 001442 171450 BIC TEMP,@DOR ;REMOVE THE INPUT DATA
836 ;THIS SHOULD CAUSE THE TRANSITION TO A ZERO
837 007740 004737 010550 JSR PC,DELAYO ;DO A SHORT DELAY
838 007744 017737 171444 001126 MOV @ICSR,$BDDAT ;READ INPUT STATUS
839 007752 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
840 007760 001401 BEQ 3$ ;;BR IF SET
841 007762 104002 ERROR 2 ;TRANSITION TO A ZERO FAILED
842 007764 006337 001442 3$: ASL TEMP ;TRY ANOTHER BIT ?
843 007770 001327 BNE 1$ ;BR IF YES
844 007772
845
(3)
(3)
(2) 007772 000004
(1) 007774 012737 000001 001160
846 010002 005237 001202
847 010006 123737 001202 001526
848 010014 001454
849 010016 012701 001404
850 010022 063721 001462 1$: ADD VADDR,(R1)+ ;UPDATE OUTPUT BUS ADDRESS
851 010026 020127 001414 CMP R1,#DOR1+2 ;TEST IF DONE
852 010032 001373 BNE 1$ ;BRANCH IF NOT
853 010034 063721 001464 2$: ADD VADDR,(R1)+ ;UPDATE INPUT ADDRESS
854 010040 020127 001430 CMP R1,#SBR1+2 ;TEST IF DONE
855 010044 001373 BNE 2$ ;BRANCH IF NOT
856 010046 006337 001460 ASL MASKNM ;UPDATE ERROR FLAG BIT
857 010052 004737 012404 JSR PC,WHICHU ;DETERMINE UNIT #
858 010056 013700 001532 MOV UNITBD,R0 ;GET UNIT #
859 010062 006300 ASL R0 ;MAKE WORD
860 010064 016037 001466 001430 MOV VECLST(R0),DIDINV ;GET VALUE
861 010072 013737 001430 001432 MOV DIDINV,DIDINS ;MAKE OTHER VALUES
862 010100 062737 000002 001432 ADD #2,DIDINS
863 010106 013737 001430 001434 MOV DIDINV,DIEINV
864 010114 062737 000004 001434 ADD #4,DIEINV
865 010122 013737 001430 001436 MOV DIDINV,DIEINS
866 010130 062737 000006 001436 ADD #6,DIEINS
867 010136 005037 001102 CLR $STNM ;RESET TEST NUMBER
868 010142 000137 003424 JMP TST1 ;TEST NEXT UNIT
869 010146 3$:
    
```

CV
CV


```
888 .SBTTL MNCDI TEST MODULE SWITCH TYPEOUT LOOP
889 010354 012706 001100 DIDATA: MOV #STACK,SP ;LOAD THE STACK POINTER
890 010360 004737 003010 JSR PC,FIXADR ;FIX DEVICE ADDRESSES
891 010364 104401 010442 TYPE, WAITO
892 010370 005077 171020 1$: CLR @ICSR ;CLEAR INPUT STATUS
893 010374 012777 177777 171016 MOV #-1,@DIR ;CLEAR INPUT DATA REGISTER
894 010402 004737 003346 2$: JSR PC,CTRLCG ;TEST IF CTRL C/G
895 010406 105777 171002 TSTB @ICSR ;WAIT FOR INPUT READY
896 010412 100373 BPL 2$
897 010414 017746 171000 MOV @DIR,-(SP) ;GET INPUT DATA
898 010420 104402 TYPE,ADASH ;TYPE THE OCTAL VALUE
899 010422 104401 020614 TYPE,ADASH ;TYPE A DASH
900 010426 017746 170766 MOV @DIR,-(SP) ;GET INPUT DATA AGAIN
901 010432 104406 TYPE,BN ;TYPE THE BINARY VALUE
902 010434 104401 020620 TYPE,TCRLF ;TYPE A CR-LF
903 010440 000753 BR 1$
904
905 010442 015 012 WAITO: .BYTE 15,12
906 010444 040527 052111 047111 .ASCII \WAITING FOR OPERATOR TO ACTIVIAE MNCDI (D/I)\
010452 020107 047506 020122
010460 050117 051105 052101
010466 051117 052040 020117
010474 041501 044524 044526
010502 052101 020105 047115
010510 042103 020111 042050
010516 044457 051
907 010521 040 042524 052123 .ASCII \ TEST MODULE BUTTON\
010526 046440 042117 046125
010534 020105 052502 052124
010542 047117
908 010544 015 012 000 .BYTE 15,12,0
909 010550 .EVEN
910 ;SUBROUTINE TO DELAY A SHORT AMOUNT OF TIME
911 010550 012737 000012 010566 DELAYO: MOV #10.,10$ ;LOAD DELAY COUNTER
912 010556 005337 010566 1$: DEC 10$ ;DELAY
913 010562 100375 BPL 1$ ; A WHILE
914 010564 000207 RTS PC ;EXIT
915 010566 000000 10$: 0
916
```

```
918 .SBTTL TTY INPUT ROUTINE
(1)
(2) ;*****
(1) .ENABL LSB
(1) 010570 000000 $TKCNT: .WORD 0 ;:NUMBER OF ITEMS IN QUEUE
(1) 010572 000000 $TKQIN: .WORD 0 ;:INPUT POINTER
(1) 010574 000000 $TKQOUT: .WORD 0 ;:OUTPUT POINTER
(1) 010576 000040 $TKQSRT: .BLKB 32. ;:TTY KEYBOARD QUEUE
(1) 010636 010636 $TKQEND=.
(1)
(1) *TK INITIALIZE ROUTINE
(1) ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
(1) ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
(1)
(1) ;*CALL:
(1) ;* JSR PC,$TKINT
(1) ;* RETURN
(1)
(1) 010636 005037 010570 $TKINT: CLR $TKCNT ;:CLEAR COUNT OF ITEMS IN QUEUE
(1) 010642 012737 010576 010572 MOV # $TKQSRT,$TKQIN ;:MOVE THE STARTING ADDRESS OF THE
(1) 010650 013737 010572 010574 MOV $TKQIN,$TKQOUT ;:QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1) 010656 012737 010706 000060 MOV # $TKSRV,@TKVEC ;:INITIALIZE THE KEYBOARD VECTOR
(1) 010664 012737 000200 000062 MOV #200,@TKVEC+2 ;:'BR' LEVEL 4
(1) 010672 005777 170250 TST @TKB ;:CLEAR DONE FLAG
(1) 010676 012777 000100 170240 MOV #100,@TKS ;:ENABLE TTY KEYBOARD INTERRUPT
(1) 010704 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) 010706 117746 170234 $TKSRV: MOVB @TKB,-(SP) ;:PICKUP THE CHARACTER
(1) 010712 042716 177600 BIC #^C177,(SP) ;:STRIP THE JUNK
(1) 010716 021627 000003 CMP (SP),#3 ;:IS IT A CONTROL C?
(1) 010722 001007 BNE 1$ ;:BRANCH IF NO
(1) 010724 104401 012056 TYPE ,SCNTLC ;:TYPE A CONTROL-C (^C)
(1) 010730 004737 010636 JSR PC,$TKINT ;:INIT THE KEYBOARD
(1) 010734 005726 TST (SP)+ ;:CLEAN UP STACK
(1) 010736 000137 002300 JMP MTEST1 ;:CONTROL C RESTART
(1) 010742 021627 000007 1$: CMP (SP),#7 ;:IS IT A CONTROL G?
(1) 010746 001004 BNE 2$ ;:BRANCH IF NO
(1) 010750 022737 000176 001140 CMP #SWREG,SWR ;:IS SOFT-SWR SELECTED?
(1) 010756 001500 BEQ 6$ ;:GO TO SWR CHANGE
(1)
(1) 010760 2$:
(1) 010760 022737 000040 010570 CMP #32,,$TKCNT ;:IS THE QUEUE FULL?
(1) 010766 001004 BNE 3$ ;:BRANCH IF NO
(1) 010770 104401 012052 TYPE ,SBELL ;:RING THE TTY BELL
(1) 010774 005726 TST (SP)+ ;:CLEAN CHARACTER OFF OF STACK
(1) 010776 000451 BR 5$ ;:EXIT
(1) 011000 021627 000023 3$: CMP (SP),#23 ;:IS IT A CONTROL-S?
(1) 011004 001021 BNE 32$ ;:BRANCH IF NO
(1) 011006 005077 170132 CLR @TKS ;:DISABLE TTY KEYBOARD INTERRUPTS
```



```

(1)          : *      RETURN HERE          : : CHARACTER IS ON THE STACK
(1)          : *          : : WITH PARITY BIT STRIPPED OFF
(1)          : *          :
(1)          : *          :
(1) 011466 011646 $RDCHR: MOV (SP),-(SP) : : PUSH DOWN THE PC AND
(1) 011470 016666 000004 000002      MOV 4(SP),2(SP) : : THE PS
(1) 011476 005066 000004      CLR 4(SP) : : GET READY FOR A CHARACTER
(2) 011502 005046      CLR -(SP) : : PUT NEW PS ON STACK
(2) 011504 012746 011512      MOV #64$,-(SP) : : PUT NEW PC ON STACK
(2) 011510 000002      RTI : : POP NEW PC AND PS
(2) 011512      64$:
(1) 011512 005737 010570      1$: TST $STKCNT : : WAIT ON A CHARACTER
(1) 011516 001775      BEQ 1$
(1) 011520 005337 010570      DEC $STKCNT : : DECREMENT THE COUNTER
(1) 011524 117766 177044 000004      MOVB @STKQOUT,4(SP) : : GET ONE CHARACTER
(1) 011532 005237 010574      INC $STKQOUT : : UPDATE THE POINTER
(1) 011536 023727 010574 010636      CMP $STKQOUT,#$STKQEND : : DID IT GO OFF OF THE END?
(1) 011544 001003      BNE 2$ : : BRANCH IF NO
(1) 011546 012737 010576 010574      MOV #$STKQSRST,$STKQOUT : : RESET THE POINTER
(1) 011554 000002      2$: RTI : : RETURN
(2)          : *****
(1)          : * THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)          : * CALL:
(1)          : *      RDLIN          : : INPUT A STRING FROM THE TTY
(1)          : *      RETURN HERE   : : ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)          : *          : : TERMINATOR WILL BE A BYTE OF ALL 0'S
(1) 011556 010346 $RDLIN: MOV R3, -(SP) : : SAVE R3
(1) 011560 005046      CLR -(SP) : : CLEAR THE RUBOUT KEY
(1) 011562 012703 012012      1$: MOV $TTYIN,R3 : : GET ADDRESS
(1) 011566 022703 012052      2$: CMP $TTYIN+32.,R3 : : BUFFER FULL?
(1) 011572 101456      BLOS 4$ : : BR IF YES
(1) 011574 104411      RDCHR : : GO READ ONE CHARACTER FROM THE TTY
(1) 011576 112613      MOV (SP)+,(R3) : : GET CHARACTER
(1) 011600 122713 000177      10$: CMPB #177,(R3) : : IS IT A RUBOUT
(1) 011604 001022      BNE 5$ : : BR IF NO
(1) 011606 005716      TST (SP) : : IS THIS THE FIRST RUBOUT?
(1) 011610 001007      BNE 6$ : : BR IF NO
(1) 011612 112737 000134 012010      MOVB #' \,9$ : : TYPE A BACK SLASH
(1) 011620 104401 012010      TYPE ,9$
(1) 011624 012716 177777      MOV #-1,(SP) : : SET THE RUBOUT KEY
(1) 011630 005303      6$: DEC R3 : : BACKUP BY ONE
(1) 011632 020327 012012      CMP R3,$TTYIN : : STACK EMPTY?
(1) 011636 103434      BLO 4$ : : BR IF YES
(1) 011640 111337 012010      MOV (R3),9$ : : SETUP TO TYPEOUT THE DELETED CHAR.
(1) 011644 104401 012010      TYPE ,9$ : : GO TYPE
(1) 011650 000746      BR 2$ : : GO READ ANOTHER CHAR.
(1) 011652 005716      5$: TST (SP) : : RUBOUT KEY SET?
(1) 011654 001406      BEQ 7$ : : BR IF NO
(1) 011656 112737 000134 012010      MOVB #' \,9$ : : TYPE A BACK SLASH
(1) 011664 104401 012010      TYPE ,9$
(1) 011670 005016      CLR (SP) : : CLEAR THE RUBOUT KEY
(1) 011672 122713 000025      7$: CMPB #25,(R3) : : IS CHARACTER A CTRL U?
(1) 011676 001003      BNE 8$ : : BR IF NO
(1) 011700 104401 012063      TYPE $CNTLU : : TYPE A CONTROL 'U'
(1) 011704 000726      BR 1$ : : GO START OVER

```

```
(1) 011706 122713 000022 8$: CMPB #22,(R3) ;; IS CHARACTER A '^R'?  
(1) 011712 001011 BNE 3$ ;; BRANCH IF NO  
(1) 011714 105013 CLRB (R3) ;; CLEAR THE CHARACTER  
(1) 011716 104401 001165 TYPE ,SCLF ;; TYPE A 'CR' & 'LF'  
(1) 011722 104401 012012 TYPE ,STTYIN ;; TYPE THE INPUT STRING  
(1) 011726 000717 BR 2$ ;; GO PICKUP ANOTHER CHACTER  
(1) 011730 104401 001164 4$: TYPE ,SQUES ;; TYPE A '?'  
(1) 011734 000712 BR 1$ ;; CLEAR THE BUFFER AND LOOP  
(1) 011736 111337 012010 3$: MOVB (R3),9$ ;; ECHO THE CHARACTER  
(1) 011742 104401 012010 TYPE ,9$  
(1) 011746 122723 000015 CMPB #15,(R3)+ ;; CHECK FOR RETURN  
(1) 011752 001305 BNE 2$ ;; LOOP IF NOT RETURN  
(1) 011754 105063 177777 CLRB -1(R3) ;; CLEAR RETURN (THE 15)  
(1) 011760 104401 001166 TYPE ,SLF ;; TYPE A LINE FEED  
(1) 011764 005726 TST (SP)+ ;; CLEAN RUBOUT KEY FROM THE STACK  
(1) 011766 012603 MOV (SP)+,R3 ;; RESTORE R3  
(1) 011770 011646 MOV (SP)-,(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE  
(1) 011772 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT  
(1) 012000 012766 012012 000004 MOV #STTYIN,4(SP)  
(1) 012006 000002 RTI ;; RETURN  
(1) 012010 000 9$: .BYTE 0 ;; STORAGE FOR ASCII CHAR. TO TYPE  
(1) 012011 000 .BYTE 0 ;; TERMINATOR  
(1) 012012 000040 $TTYIN: .BLKB 32 ;; RESERVE 32. BYTES FOR TTY INPUT  
(1) 012052 177607 000377 $BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL  
(1) 012056 041536 005015 000 $CNTLC: .ASCIZ /^C/<15><12> ;; CONTROL 'C'  
(1) 012063 136 006525 000012 $CNTLU: .ASCIZ /^U/<15><12> ;; CONTROL 'U'  
(1) 012070 043536 005015 000 $CNTLG: .ASCIZ /^G/<15><12> ;; CONTROL 'G'  
(1) 012075 015 051412 051127 $MSWR: .ASCIZ <15><12>/SWR - /  
(1) 012102 036440 000040 $MNEW: .ASCIZ / NEW - /  
(1) 012106 020040 042516 020127  
(1) 012114 020075 000  
(1) 012120  
919 .EVEN  
(1) .SBTTL SCOPE HANDLER ROUTINE  
(2) *****  
(1) *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT  
(1) *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)  
(1) *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>  
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:  
(1) *SW14=1 LOOP ON TEST  
(1) *SW11=1 INHIBIT ITERATIONS  
(1) *SW09=1 LOOP ON ERROR  
(1) *SW08=1 LOOP ON TEST IN SWR<7:0>  
(1) *CALL  
(1) * SCOPE ;; SCOPE=IOT  
(1) $SCOPE:  
(1) 012120 CKSWR ;; TEST FOR CHANGE IN SOFT-SWR  
(2) 012122 104410 JSR PC,CTRLCG  
(1) 012126 004737 003346 167004 1$: BIT #BIT14,@SWR ;; LOOP ON PRESENT TEST?  
(1) 012134 032777 040000 BNE $OVER ;; YES IF SW14=1  
(1) *#####START OF CODE FOR THE XOR TESTER#####  
(1) 012136 000416 $XTSTR: BR 6$ ;; IF RUNNING ON THE 'XOR' TESTER CHANGE  
(1) THIS INSTRUCTION TO A 'NOP' (NOP=240)  
(1) 012140 013746 000004 MOV @ERRVEC,-(SP) ;; SAVE THE CONTENTS OF THE ERROR VECTOR  
(1) 012144 012737 012164 000004 MOV #5$,@ERRVEC ;; SET FOR TIMEOUT
```

```
(1) 012152 005737 177060          TST    @#177060          ;;TIME OUT ON XOR?
(1) 012156 012637 000004          MOV    (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1) 012162 000463                   BR     $$VLAD           ;;GO TO THE NEXT TEST
(1) 012164 022626          5$:   CMP    (SP)+,(SP)+    ;;CLEAR THE STACK AFTER A TIME OUT
(1) 012166 012637 000004          MOV    (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1) 012172 000423                   BR     7$              ;;LOOP ON THE PRESENT TEST
(1) 012174                   6$:   ;#####END OF CODE FOR THE XOR TESTER#####
(1) 012174 032777 000400 166736          BIT    #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
(1) 012202 001404                   BEQ    2$              ;;BR IF NO
(1) 012204 127737 166730 001102          CMPB  @SWR,$STSTM      ;;ON THE RIGHT TEST? SWR<7:0>
(1) 012212 001465                   BEQ    $OVER           ;;BR IF YES
(1) 012214 105737 001103          2$:   TSTB  $ERFLG         ;;HAS AN ERROR OCCURRED?
(1) 012220 001421                   BEQ    3$              ;;BR IF NO
(1) 012222 123737 001115 001103          CMPB  $ERMAX,$ERFLG    ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 012230 101015                   BHI    3$              ;;BR IF NO
(1) 012232 032777 001000 166700          BIT    #BIT09,@SWR     ;;LOOP ON ERROR?
(1) 012240 001404                   BEQ    4$              ;;BR IF NO
(1) 012242 013737 001110 001106          7$:   MOV    $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 012250 000446                   BR     $OVER           ;;
(1) 012252 105037 001103          4$:   CLRB  $ERFLG         ;;ZERO THE ERROR FLAG
(1) 012256 005037 001160          CLR   $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 012262 000415                   BR     1$              ;;ESCAPE TO THE NEXT TEST
(1) 012264 032777 004000 166646          3$:   BIT    #BIT11,@SWR    ;;INHIBIT ITERATIONS?
(1) 012272 001011                   BNE    1$              ;;BR IF YES
(1) 012274 005737 001176          TST   $PASS           ;;IF FIRST PASS OF PROGRAM
(1) 012300 001406                   BEQ    1$              ;;INHIBIT ITERATIONS
(1) 012302 005237 001104          INC   $ICNT           ;;INCREMENT ITERATION COUNT
(1) 012306 023737 001160 001104          CMP   $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 012314 002024                   BGE   $OVER           ;;BR IF MORE ITERATION REQUIRED
(1) 012316 012737 000001 001104          1$:   MOV   #1,$ICNT       ;;REINITIALIZE THE ITERATION COUNTER
(1) 012324 013737 012402 001160          MOV   $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
(1) 012332 105237 001102          $$VLAD: INCB  $STSTM     ;;COUNT TEST NUMBERS
(1) 012336 113737 001102 001174          MOVB  $STSTM,$STSTN   ;;SET TEST NUMBER IN APT MAILBOX
(1) 012344 011637 001106          MOV   (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1) 012350 011637 001110          MOV   (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1) 012354 005037 001162          CLR   $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 012360 112737 000001 001115          MOVB  #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 012366 013777 001102 166546          $OVER: MOV   $STSTM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 012374 013716 001106          MOV   $LPADR,(SP)    ;;FUDGE RETURN ADDRESS
(1) 012400 000002          RTI                   ;;FIXES PS
(1) 012402 003720          $MXCNT: 2000.        ;;MAX. NUMBER OF ITERATIONS
```



```

(1) 012622 013716 001162      MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 012626                    5$:      CMP      #SENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
(1) 012626 022737 010234 000042      BNE     6$                ;;BRANCH IF NO
(1) 012634 001001                    HALT                       ;;YES
(1) 012636 000000
(1) 012640                    6$:      RTI                       ;;RETURN
(1) 012640 000002
937 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 012642
(1) 012642 104401 001165      TYPE    , $CRLF           ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 012646 010046            MOV     R0,-(SP)          ;;SAVE R0
(1) 012650 005000            CLR     R0                ;;PICKUP THE ITEM INDEX
(1) 012652 153700 001114      BISB   @#$ITEMB,R0
(1) 012656 001004            BNE     1$                ;;IF ITEM NUMBER IS ZERO, JUST
(1)
(2) 012660 013746 001116      MOV     $ERRPC,-(SP)      ;;SAVE $ERRPC FOR TYPEOUT
(2)
(2) 012664 104402            TYPOC                    ;;ERROR ADDRESS
(1) 012666 000445            BR      10$               ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 012670 005300            1$:    DEC     R0            ;;GET OUT
(1) 012672 006300            ASL    R0                ;;ADJUST THE INDEX SO THAT IT WILL
(1) 012674 006300            ASL    R0                ;;      WORK FOR THE ERROR TABLE
(1) 012676 006300            ASL    R0
(1) 012700 062700 001254      ADD    #$ERRTB,R0        ;;FORM TABLE POINTER
(1) 012704 012037 012714      MOV    (R0)+,2$          ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 012710 001404            BEQ    3$                ;;SKIP TYPEOUT IF NO POINTER
(1) 012712 104401            TYPE                    ;;TYPE THE 'ERROR MESSAGE'
(1) 012714 000000            2$:    .WORD   0             ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 012716 104401 001165      TYPE    , $CRLF           ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 012722 012037 012732      3$:    MOV     (R0)+,4$        ;;PICKUP 'DATA HEADER' POINTER
(1) 012726 001404            BEQ    5$                ;;SKIP TYPEOUT IF 0
(1) 012730 104401            TYPE                    ;;TYPE THE 'DATA HEADER'
(1) 012732 000000            4$:    .WORD   0             ;;'DATA HEADER' POINTER GOES HERE
(1) 012734 104401 001165      TYPE    , $CRLF           ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 012740 010146            5$:    MOV     R1,-(SP)        ;;SAVE R1
(1) 012742 012001            MOV    (R0)+,R1          ;;PICKUP 'DATA TABLE' POINTER
(1) 012744 001415            BEQ    9$                ;;BR IF NO DATA TO BE TYPED
(1) 012746 012000            MOV    (R0)+,R0          ;;PICKUP 'DATA FORMAT' POINTER
(1) 012750 105720            6$:    TSTB   (R0)+          ;;'OCTAL' OR 'DECIMAL'
(1) 012752 001003            BNE    7$                ;;BR IF DECIMAL
(2) 012754 013146            MOV    @(R1)+,-(SP)      ;;SAVE @(R1)+ FOR TYPEOUT
(2) 012756 104402            TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 012760 000402            BR      8$
(1) 012762
(2) 012762 013146            7$:    MOV    @(R1)+,-(SP)      ;;SAVE @(R1)+ FOR TYPEOUT
(2) 012764 104405            TYPDS                    ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 012766 005711            8$:    TST    (R1)            ;;IS THERE ANOTHER NUMBER?
(1) 012770 001403            BEQ    9$                ;;BR IF NO
(1) 012772 104401 013012      TYPE    ,11$             ;;TYPE TWO(2) SPACES
(1) 012776 000764            BR      6$                ;;LOOP

```



```

(1) 013320 160105      3$:  SUB   R1,R5      ;;FORM THIS BCD DIGIT
(1) 013322 002402      BLT   4$          ;;BR IF DONE
(1) 013324 005202      INC   R2          ;;INCREASE THE BCD DIGIT BY 1
(1) 013326 000774      BR    3$
(1) 013330 060105      4$:  ADD   R1,R5      ;;ADD BACK THE CONSTANT
(1) 013332 005702      TST   R2          ;;CHECK IF BCD DIGIT=0
(1) 013334 001002      BNE   5$          ;;FALL THROUGH IF 0
(1) 013336 105716      TSTB (SP)        ;;STILL DOING LEADING 0'S?
(1) 013340 100407      BMI   7$          ;;BR IF YES
(1) 013342 106316      5$:  ASLB (SP)        ;;MSD?
(1) 013344 103003      BCC   6$          ;;BR IF NO
(1) 013346 116663 000001 177777 MOVB  1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 013354 052702 000060      6$:  BIS   #'0,R2     ;;MAKE THE BCD DIGIT ASCII
(1) 013360 052702 000040      7$:  BIS   #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 013364 110223      MOVB  R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 013366 005720      TST   (R0)+      ;;JUST INCREMENTING
(1) 013370 020027 000010      CMP   R0,#10     ;;CHECK THE TABLE INDEX
(1) 013374 002746      BLT   2$          ;;GO DO THE NEXT DIGIT
(1) 013376 003002      BGT   8$          ;;GO TO EXIT
(1) 013400 010502      MOV   R5,R2      ;;GET THE LSD
(1) 013402 000764      BR    6$          ;;GO CHANGE TO ASCII
(1) 013404 105726      8$:  TSTB (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 013406 100003      BPL   9$          ;;BR IF NO
(1) 013410 116663 177777 177776 MOVB  -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 013416 105013      9$:  CLRB (R3)       ;;SET THE TERMINATOR
(3) 013420 012605      MOV   (SP)+,R5   ;;POP STACK INTO R5
(3) 013422 012603      MOV   (SP)+,R3   ;;POP STACK INTO R3
(3) 013424 012602      MOV   (SP)+,R2   ;;POP STACK INTO R2
(3) 013426 012601      MOV   (SP)+,R1   ;;POP STACK INTO R1
(3) 013430 012600      MOV   (SP)+,R0   ;;POP STACK INTO R0
(1) 013432 104401 013460      TYPE  ,SDBLK     ;;NOW TYPE THE NUMBER
(1) 013436 016666 000002 000004 MOV   2(SP),4(SP) ;;ADJUST THE STACK
(1) 013444 012616      MOV   (SP)+,(SP)
(1) 013446 000002      RTI                    ;;RETURN TO USER
(1) 013450 023420      $DTBL: 10000.
(1) 013452 001750      1000.
(1) 013454 000144      100.
(1) 013456 000012      10.
(1) 013460 000004      $DBLK: .BLKW 4
          .SBTTL APT COMMUNICATIONS ROUTINE
    
```

```

(2)
(1) 013470 112737 000001 013734 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
(1) 013476 112737 000001 013732 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
(1) 013504 000403      BR    $ATYC
(1) 013506 112737 000001 013734 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) 013514      $ATYC:
(3) 013514 010046      MOV   R0,-(SP)   ;;PUSH R0 ON STACK
(3) 013516 010146      MOV   R1,-(SP)   ;;PUSH R1 ON STACK
(1) 013520 105737 013732      TSTB $MFLG      ;;SHOULD TYPE A MESSAGE?
(1) 013524 001450      BEQ   5$          ;;IF NOT: BR
(1) 013526 122737 000001 001210 CMPB  #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 013534 001031      BNE   3$          ;;IF NOT: BR
(1) 013536 132737 000100 001211 BITB  #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 013544 001425      BEQ   3$          ;;IF NOT: BR
(1) 013546 017600 000004      MOV   @4(SP),R0  ;;GET MESSAGE ADDR.
    
```

CV
CV

```
(1) 013552 062766 000002 000004      ADD      #2,4(SP)          ;;BUMP RETURN ADDR.
(1) 013560 005737 001170      1$:     TST      $MSGTYPE    ;;SEE IF DONE w/ LAST XMISSION?
(1) 013564 001375              BNE      1$              ;;IF NOT: WAIT
(1) 013566 010037 001204      MOV      R0,$MSGAD      ;;PUT ADDR IN MAILBOX
(1) 013572 105720      2$:     TSTB     (R0)+     ;;FIND END OF MESSAGE
(1) 013574 001376              BNE      2$
(1) 013576 163700 001204      SUB      $MSGAD,R0      ;;SUB START OF MESSAGE
(1) 013602 006200              ASR      R0              ;;GET MESSAGE LNTH IN WORDS
(1) 013604 010037 001206      MOV      R0,$MSGGLGT    ;;PUT LENGTH IN MAILBOX
(1) 013610 012737 000004 001170      MOV      #4,$MSGTYPE    ;;TELL APT TO TAKE MSG.
(1) 013616 000413              BR       5$
(1) 013620 017637 000004 013644 3$:     MOV      @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
(1) 013626 062766 000002 000004      ADD      #2,4(SP)          ;;BUMP RETURN ADDRESS
(3) 013634 013746 177776      MOV      177776,-(SP)    ;;PUSH 177776 ON STACK
(1) 013640 004737 014260      JSR      PC,$TYPE       ;;CALL TYPE MACRO
(1) 013644 000000      4$:     .WORD    0
(1) 013646              5$:
(1) 013646 105737 013734      10$:    TSTB     $FFLG          ;;SHOULD REPORT FATAL ERROR?
(1) 013652 001416              BEQ      12$            ;;IF NOT: BR
(1) 013654 005737 001210      TST      $ENV           ;;RUNNING UNDER APT?
(1) 013660 001413              BEQ      12$            ;;IF NOT: BR
(1) 013662 005737 001170      11$:    TST      $MSGTYPE      ;;FINISHED LAST MESSAGE?
(1) 013666 001375              BNE      11$            ;;IF NOT: WAIT
(1) 013670 017637 000004 001172      MOV      @4(SP),$FATAL   ;;GET ERROR #
(1) 013676 062766 000002 000004      ADD      #2,4(SP)          ;;BUMP RETURN ADDR.
(1) 013704 005237 001170      INC      $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1) 013710 105037 013734      12$:    CLRB     $FFLG          ;;CLEAR FATAL FLAG
(1) 013714 105037 013733      CLRB     $LFLG          ;;CLEAR LOG FLAG
(1) 013720 105037 013732      CLRB     $MFLG          ;;CLEAR MESSAGE FLAG
(3) 013724 012601      MOV      (SP)+,R1        ;;POP STACK INTO R1
(3) 013726 012600      MOV      (SP)+,R0        ;;POP STACK INTO R0
(1) 013730 000207      RTS      PC              ;;RETURN
(1) 013732 000          $MFLG: .BYTE    0        ;;MESSG. FLAG
(1) 013733 000          $LFLG: .BYTE    0        ;;LOG FLAG
(1) 013734 000          $FFLG: .BYTE    0        ;;FATAL FLAG
(1) 013736              .EVEN
(1) 000200      APTSIZE=200
(1) 000001      APTENV=001
(1) 000100      APTSPool=100
(1) 000040      APTCSUP=040
```



```
949      :CAUTION      THE FIRST 4 LOC. ARE OVERLAYED TO LOWER INTERRUPT LEVEL
950      :          THE OVERLAY OCCURS AFTER THE 'SETUP' CODE
951      :SBTTL  TYPE ROUTINE
(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 FEED.
(1)      :*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)      :*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)      :*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)      :*
(1)      :*CALL:
(1)      :*1) USING A TRAP INSTRUCTION
(1)      :*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)      :*OR
(1)      :*      TYPE
(1)      :*      MESADR
(1)      :*
(1)      :
(1) 014260 10537 001157 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
(1) 014264 100002      BPL 1$      ;; BR IF YES
(1) 014266 000030      HALT      ;; HALT HERE IF NO TERMINAL
(1) 014270 000430      BR 3$      ;; LEAVE
(1) 014272 010046      1$: MOV R0,-(SP)      ;; SAVE R0
(1) 014274 017600 000002      MOV @2(SP),R0      ;; GET ADDRESS OF ASCIZ STRING
(1) 014300 122737 000001 001210      CMPB #APTENV,$ENV      ;; RUNNING IN APT MODE
(1) 014306 001011      BNE 62$      ;; NO,GO CHECK FOR APT CONSOLE
(1) 014310 132737 000100 001211      BITB #APTPOOL,$ENVM      ;; SPOOL MESSAGE TO APT
(1) 014316 001405      BEQ 62$      ;; NO,GO CHECK FOR CONSOLE
(1) 014320 010037 014330      MOV R0,61$      ;; SETUP MESSAGE ADDRESS FOR APT
(1) 014324 004737 013476      JSR PC,$ATY3      ;; SPOOL MESSAGE TO APT
(1) 014330 000000      61$: .WORD 0      ;; MESSAGE ADDRESS
(1) 014332 132737 000040 001211      62$: BITB #APTCSUP,$ENVM      ;; APT CONSOLE SUPPRESSED
(1) 014340 001003      BNE 60$      ;; YES,SKIP TYPE OUT
(1) 014342 112046      2$: MOVB (R0)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 014344 001005      BNE 4$      ;; BR IF IT ISN'T THE TERMINATOR
(1) 014346 005726      TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
(1) 014350 012600      60$: MOV (SP)+,R0      ;; RESTORE R0
(1) 014352 062716 000002      3$: ADD #2,(SP)      ;; ADJUST RETURN PC
(1) 014356 000002      RTI      ;; RETURN
(1) 014360 122716 000011      4$: CMPB #HT,(SP)      ;; BRANCH IF <HT>
(1) 014364 001430      BEQ 8$
(1) 014366 122716 000200      CMPB #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
(1) 014372 001006      BNE 5$
(1) 014374 005726      TST (SP)+      ;; POP <CR><LF> EQUIV
(1) 014376 104401      TYPE      ;; TYPE A CR AND LF
(1) 014400 001165      $CRLF
(1) 014402 105037 014536      CLRB $CHARCNT      ;; CLEAR CHARACTER COUNT
(1) 014406 000755      BR 2$      ;; GET NEXT CHARACTER
(1) 014410 004737 014472      5$: JSR PC,$TYPEC      ;; GO TYPE THIS CHARACTER
(1) 014414 123726 001156      6$: CMPB $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
(1) 014420 001350      BNE 2$      ;; IF NO GO GET NEXT CHAR.
(1) 014422 013746 001154      MOV $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
(1)      ;; AND THE NULL CHAR.
(1) 014426 105366 000001      7$: DECB 1(SP)      ;; DOES A NULL NEED TO BE TYPED?
(1) 014432 002770      BLT 6$      ;; BR IF NO--GO POP THE NULL OFF OF STACK
```


1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
(1)
(1)
(1)
1036
1037
(1)
(1)
(1)
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054

014754 000002
014756 022626
014760 005737 001176
014764 001025
014766 104401 017756
014772 013746 001202
014776 104405
015000 000240
015002 000240
015004 104401 020011
015010 013746 001430
015014 104403
015016 003
015017 001
015020 104401 020041
015024 013746 015154
015030 104403
015032 003
015033 001
015034 104401 020071
015040 013777 001432 164362
015046 013777 001436 164360
015054 012777 004700 164354
015062 012777 004700 164342
015070 013737 015154 001430
015076 042737 000007 001430
015104 013737 001430 001432
015112 013737 001430 001434
015120 013737 001430 001436
015126 062737 000002 001432
015134 062737 000004 001434
015142 062737 000006 001436
015150 000177 163732
015154 000000
015156 000000

: \$ >>> ERROR <<< \$

RTI
3\$: CMP (SP)+,(SP)+
TST \$PASS
BNE 4\$
TYPE,VTMSG
MOV \$UNIT,-(SP)
TYPDS
NOP
NOP
TYPE,VTMSG3
MOV DIDINV,-(SP)
TYPOS
.BYTE 3
.BYTE 1
TYPE,VTMSG1
MOV TRTO,-(SP)
TYPOS
.BYTE 3
.BYTE 1
TYPE,VTMSG2
4\$: MOV DIDINS,@DIDINV
MOV DIEINS,@DIEINV
MOV #4700,@DIEINS
MOV #4700,@DIDINS
MOV TRTO,DIDINV
BIC #7,DIDINV
MOV DIDINV,DIDINS
MOV DIDINV,DIEINV
ADD #2,DIDINS
ADD #4,DIEINV
ADD #6,DIEINS
JMP @SLPADR

TRTO: .WORD 0
TRFRC: .WORD 0

:PROCESS.
:NOTE:
:FOR THIS ERROR - DON'T USE
: 'LOOP ON ERROR' OPTION.
:ALSO EXPECT THAT THE INTERRUPT TEST TO
:WILL REPORT THAT THE DEVICE DIDN'T
:INTERRUPT.
:FOLLOW THE RECOMMENDED PROCEDURE
: IN THE DOCUMENT (ON THIS DIAGNOSTIC)
:FOR LOOPING ON TEST.
:CLEAN OFF STACK
:IS THIS THE FIRST PASS?
:NO, DON'T REPORT
:TELL OPER.
:GET UNIT #
:TYPE 'EXPECTED INTR. AT ''
:SAVE DIDINV FOR TYPEOUT
:GO TYPE--OCTAL ASCII
:TYPE 3 DIGIT(S)
:TYPE LEADING ZEROS
:TYPE '' RECEIVED INTR. AT ''
:SAVE TRTO FOR TYPEOUT
:GO TYPE--OCTAL ASCII
:TYPE 3 DIGIT(S)
:TYPE LEADING ZEROS
:TYPE 'RESTARTING TEST'
:RESTORE INCORRECT VECTOR
:LOCATION WITH .+2 + 4700
:SET UP ACTUAL VECTOR LOCATION
:CLR LOW BITS
:SET UP NEW VECTOR VALUES
:
:START TEST OVER AGAIN.
:CONTAINS ADDR. WE TRAPPED OR INTERRUPTED TO.
:CONTAINS ADDR. WE TRAPPED OR INTR. FROM.

1056
1057 015160 015 012
1058 015162 020114 020075 047514
015170 044507 020103 042524
015176 052123 053440 052111
015204 020110 051117 053440
015212 052111 047510 052125
015220 052040 051505 020124
015226 047515 052504 042514
015234 041440 047117 042516
015242 052103 042105
1059 015246 015 012
1060 015250 020127 020075 051127
015256 050101 051101 052517
015264 042116 046040 043517
015272 041511 052040 051505
015300 020124 044527 044124
015306 046440 041516 047504
015314 041440 047117 042516
015322 052103 042105 052040
015330 020117 047115 042103
015336 111
1061 015337 015 012
1062 015341 124 036440 052040
015346 050131 047505 052125
015354 046040 047517 020120
015362 047506 020122 047115
015370 042103 020111 042524
015376 052123 046440 042117
015404 046125 020105 053523
015412 052111 044103 051505
1063 015420 015 012
1064 015422 020107 020075 042507
015430 020124 042516 020127
015436 053523 052111 044103
015444 051040 043505 051511
015452 042524 020122 040526
015460 052514 105
1065 015463 015 012
1066 015465 102 036440 024040
015472 044504 044507 040524
015500 020114 047111 020051
015506 040502 042523 047440
015514 020122 042526 052103
015522 051117 040440 042104
015530 042522 051523 041440
015536 040510 043516 051505
1067 015544 015 012
1068 015546 020117 020075 042050
015554 043511 052111 046101
015562 047440 052125 020051
015570 040502 042523 040440
015576 042104 042522 051523
015604 041440 040510 043516
015612 105
1069 015613 015 012

.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 - TYPEOUT 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\

.BYTE 15,12

1070	015615	110	036440	044040		.ASCIZ	\H - HELP THE OPERATOR AND RETYPE THIS LIST	\
	015622	046105	020120	044124				
	015630	020105	050117	051105				
	015636	052101	051117	040440				
	015644	042116	051040	052105				
	015652	050131	020105	044124				
	015660	051511	046040	051511				
	015666	020124	020040	000040				
1071	015674	015	012		DOT:	.BYTE	15,12	
1072	015676	054524	042520	052040		.ASCIZ	/TYPE THE 'TEST CHARACTER' THEN DEPRESS 'RETURN KEY' /	/
	015704	042510	021040	042524				
	015712	052123	041440	040510				
	015720	040522	052103	051105				
	015726	020042	044124	047105				
	015734	042040	050105	042522				
	015742	051523	021040	042522				
	015750	052524	047122	045440				
	015756	054505	020042	000040				
1073								
1074	015764	047115	042103	020117	EM1:	.ASCIZ	\MNCDO (DIGITAL OUT)	DOES NOT EXIST <BUS ERROR> CHECK ADDRESS SWITC
	015772	042050	043511	052111				
	016000	046101	047440	052125				
	016006	004451	047504	051505				
	016014	047040	052117	042440				
	016022	044530	052123	020040				
	016030	041074	051525	042440				
	016036	051122	051117	020076				
	016044	041440	042510	045503				
	016052	040440	042104	042522				
	016060	051523	051440	044527				
	016066	041524	042510	000123				
1075	016074	047115	042103	020117	EM2:	.ASCIZ	\MNCDO (DIGITAL OUT)	DATA REGISTER ERROR\
	016102	042050	043511	052111				
	016110	046101	047440	052125				
	016116	004451	040504	040524				
	016124	051040	043505	051511				
	016132	042524	020122	051105				
	016140	047522	000122					
1076	016144	047115	042103	020117	EM3:	.ASCIZ	\MNCDO (DIGITAL OUT)	STATUS REGISTER ERROR\
	016152	042050	043511	052111				
	016160	046101	047440	052125				
	016166	004451	052123	052101				
	016174	051525	051040	043505				
	016202	051511	042524	020122				
	016210	051105	047522	000122				
1077	016216	047115	042103	020117	EM4:	.ASCIZ	\MNCDO (DIGITAL OUT)	INTERRUPT ERROR\
	016224	042050	043511	052111				
	016232	046101	047440	052125				
	016240	004451	047111	042524				
	016246	051122	050125	020124				
	016254	051105	047522	000122				
1078	016262	047115	042103	020111	EM5:	.ASCIZ	\MNCDI (DIGITAL IN)	DOES NOT EXIST <BUS ERROR> CHECK ADDRESS SWITCH
	016270	042050	043511	052111				
	016276	046101	044440	024516				
	016304	042011	042517	020123				
	016312	047516	020124	054105				

CVMNB-B MNC DI	DIAGNOSTIC	MACY11	30G(1063)	08-AUG-79	10:48	L 5	PAGE 34-2	SEQ 0063
CVMNBB.P11	08-AUG-79 10:38						ASCII MESSAGES	
	016320	051511	004524	041074				
	016326	051525	042440	051122				
	016334	051117	020076	044103				
	016342	041505	020113	042101				
	016350	051104	051505	020123				
	016356	053523	052111	044103				
	016364	051505	000					
1079	016367	115	041516	044504	EM6:	.ASCIZ	MNC DI (DIGITAL IN)	STIMULUS REGISTER ERROR\
	016374	024040	044504	044507				
	016402	040524	020114	047111				
	016410	004451	052123	046511				
	016416	046125	051525	051040				
	016424	043505	051511	042524				
	016432	020122	051105	047522				
	016440	000122						
1080	016442	047115	042103	020111	EM7:	.ASCIZ	MNC DI (DIGITAL IN)	STATUS REGISTER ERROR\
	016450	042050	043511	052111				
	016456	046101	044440	024516				
	016464	051411	040524	052524				
	016472	020123	042522	044507				
	016500	052123	051105	042440				
	016506	051122	051117	000				
1081	016513	115	041516	044504	EM10:	.ASCIZ	MNC DI (DIGITAL IN)	DATA REGISTER ERROR\
	016520	024040	044504	044507				
	016526	040524	020114	047111				
	016534	004451	040504	040524				
	016542	051040	043505	051511				
	016550	042524	020122	051105				
	016556	047522	000122					
1082	016562	047115	042103	020111	EM11:	.ASCIZ	MNC DI (DIGITAL IN)	INTERRUPT ERROR\
	016570	042050	043511	052111				
	016576	046101	044440	024516				
	016604	044411	052116	051105				
	016612	052522	052120	042440				
	016620	051122	051117	000				
1083	016625	115	041516	044504	EM12:	.ASCIZ	/MNC DI MNC DO	DIGITAL INPUT-OUTPUT WRAPAROUND STATUS ERROR/
	016632	046440	041516	047504				
	016640	042011	043511	052111				
	016646	046101	044440	050116				
	016654	052125	047455	052125				
	016662	052520	020124	051127				
	016670	050101	051101	052517				
	016676	042116	051440	040524				
	016704	052524	020123	051105				
	016712	047522	000122					
1084	016716	047115	042103	020111	EM13:	.ASCIZ	/MNC DI MNC DO	DIGITAL INPUT-OUTPUT WRAPAROUND DATA ERROR/
	016724	047115	042103	004517				
	016732	044504	044507	040524				
	016740	020114	047111	052520				
	016746	026524	052517	050124				
	016754	052125	053440	040522				
	016762	040520	047522	047125				
	016770	020104	040504	040524				
	016776	042440	051122	051117				
	017004	000						
1085	017005	115	041516	044504	EM14:	.ASCIZ	/MNC DI	INCORRECT I.D. VALUE FOR MNC DI/

1095	017472	040502	000104	042411	DH11:	.ASCIZ	/UNIT	ERRPC	IN CSR/		
	017476	047125	052111	044411							
	017504	051122	041520	000122							
1096	017512	020116	051503	042411	DH12:	.ASCIZ	/UNIT	ERRPC	OUTCSR	IN CSR	GOOD
	017520	047125	052111	047411							BAD/
	017526	051122	041520	004522							
	017534	052125	051503	051123							
	017542	047111	041440	004504							
	017550	043411	047517	000							
1097	017556	041040	042101	004524	DH13:	.ASCIZ	/UNIT	ERROC	OUTDOR	IN DIR	GOOD
	017563	125	044516	004503							BAD/
	017570	051105	047522	051117							
	017576	052517	042124	044504							
	017604	044411	020116	042117							
	017612	004522	047507	000104							
1098	017620	020011	040502	042411	DH15:	.ASCIZ	/UNIT	ERRPC	TO	FROM	ADRS./
	017626	047125	052111	020011							
	017634	051122	041520	051106							
	017642	052040	004517	051104							
	017650	046517	040440	000							
1099	017656	027123	000	004524	DH16:	.ASCIZ	/UNIT	ERRPC	WERE	ARE/	
	017661	125	044516	020103							
	017666	051105	050122	051105							
	017674	020040	053440	051101							
	017702	020105	020040								
	017710	000105									
1100	017712	035440	047524	040524	ERRTOT:	.ASCIZ	/	;TOTAL ERROR COUNT = /			
	017720	020114	051105	047522							
	017726	020122	047503	047125							
	017734	020124	020075	000							
1101	017741	040	041073	042101	MESGD:	.ASCIZ	/	;BAD UNITS /			
	017746	052440	044516	051524							
	017754	000040									
1102	017756	046600	041516	044504	VTMSG:	.ASCIZ	<200>/MNC DI	(DIGITAL IN) UNIT #/			
	017764	024040	044504	044507							
	017772	040524	020114	047111							
	020000	020051	047125	052111							
	020006	021440	000								
1103	020011	200	054105	042520	VTMSG3:	.ASCIZ	<200>/EXPECTED	INTERRUPT AT /			
	020016	052103	042105	044440							
	020024	052116	051105	052522							
	020032	052120	040440	020124							
	020040	000									
1104	020041	040	042522	042503	VTMSG1:	.ASCIZ	/	RECEIVED INTERRUPT AT /			
	020046	053111	042105	044440							
	020054	052116	051105	052522							
	020062	052120	040440	020124							
	020070	000									
1105	020071	200	046120	040505	VTMSG2:	.ASCII	<200>/PLEASE	CHECK VECTOR SWITCHES/<200>			
	020076	042523	041440	042510							
	020104	045503	053040	041505							
	020112	047524	020122	053523							
	020120	052111	044103	051505							
	020126	200									
1106	020127	011	042522	052123		.ASCIZ	/	RESTARTING LOGIC TEST/			
	020134	051101	044524	043516							

1107	020142	046040	043517	041511	
1108	020150	052040	051505	J00124	
	020156	015	012		SETUP0: .BYTE 15,12
	020160	047105	052523	042522	.ASCII \ENSURE MNC DI (DIGITAL IN) DATA SWITCH IS IN THE '-' POSITION\
	020166	046440	041516	044504	
	020174	024040	044504	044507	
	020202	040524	020114	047111	
	020210	020051	040504	040524	
	020216	051440	044527	041524	
	020224	020110	051511	044440	
	020232	020116	044124	020105	
	020240	026442	020042	047520	
1109	020246	044523	044524	047117	
	020254	015	012	000	SETUP1: .BYTE 15,12,0
1110	020257	103	047117	042516	.ASCII \CONNECT MNC DI (DIGITAL IN) TO THE TESTER\
	020264	052103	046440	041516	
	020272	044504	024040	044504	
	020300	044507	040524	020114	
	020306	047111	020051	047524	
	020314	052040	042510	052040	
1111	020322	051505	042524	122	
	020327	015	012	000	SETUP2: .BYTE 15,12,0
1112	020332	015	012		.BYTE 15,12
1113	020334	047115	042103	020111	.ASCII \MNC DI (DIGITAL IN) MUST BE CONNECTED TO MNC DO (DIGITAL OUT)\
	020342	042050	043511	052111	
	020350	046101	044440	024516	
	020356	046440	051525	020124	
	020364	042502	041440	047117	
	020372	042516	052103	042105	
	020400	052040	020117	047115	
	020406	042103	020117	042050	
	020414	043511	052111	046101	
	020422	047440	052125	051	
1114	020427	015	012	000	ADR0UT: .BYTE 15,12,0
1115	020432	005015	047115	042103	.ASCIZ <15><12>/MNC DO (DIGITAL OUT) BUS ADDRESS </
	020440	020117	042050	043511	
	020446	052111	046101	047440	
	020454	052125	020051	052502	
	020462	020123	042101	051104	
1116	020470	051505	020123	000074	ADRIN: .ASCIZ <15><12>/MNC DI (DIGITAL IN) BUS ADDRESS </
	020476	005015	047115	042103	
	020504	020111	042050	043511	
	020512	052111	046101	044440	
	020520	024516	041040	051525	
	020526	040440	042104	042522	
	020534	051523	036040	000	
1117	020541	015	046412	041516	VECIN: .ASCIZ <15><12>/MNC DI (DIGITAL IN) VECTOR ADDRESS </
	020546	044504	024040	044504	
	020554	044507	040524	020114	
	020562	047111	020051	042526	
	020570	052103	051117	040440	
	020576	042104	042522	051523	
	020604	036040	000		
1118	020607	076	037440	000040	ENDOUT: .ASCIZ /> ? /
1119	020614	026440	000040		ADASH: .ASCIZ / - /
1120	020620	015	012	000	TCRLF: .BYTE 15,12,0

```
1121      020624      020624
1122 020624 001532 001116 001404 DT1:  .EVEN
      020632 001410 000000      UNITBD,$ERRPC,OCSR,DOR,0
1123 020636 001532 001116 001414 DT5:  UNITBD,$ERRPC,ICSR,DIR,SBR,0
      020644 001420 001424 000000
1124 020652 001532 001116 001424 DT6:  UNITBD,$ERRPC,SBR,$GDDAT,$BDDAT,0
      020660 001124 001126 000000
1125 020666 001532 001116 001414 DT7:  UNITBD,$ERRPC,ICSR,$GDDAT,$BDDAT,0
      020674 001124 001126 000000
1126 020702 001532 001116 001420 DT10: UNITBD,$ERRPC,DIR,$GDDAT,$BDDAT,0
      020710 001124 001126 000000
1127 020716 001532 001116 001414 DT11: UNITBD,$ERRPC,ICSR,0
      020724 000000
1128 020726 001532 001116 001404 DT12: UNITBD,$ERRPC,OCSR,ICSR,$GDDAT,$BDDAT,0
      020734 001414 001124 001126
      020742 000000
1129 020744 001532 001116 001410 DT13: UNITBD,$ERRPC,DOR,DIR,$GDDAT,$BDDAT,0
      020752 001420 001124 001126
      020760 000000
1130 020762 001532 001116 015154 DT15: UNITBD,$ERRPC,TRTO,TRFRO,0
      020770 015156 000000
1131 020774 001532 001116 001442 DT16: UNITBD,$ERRPC,TEMP,$UNIT,0
      021002 001202 000000
1132 021006      000      000      000 DFO:  .BYTE  0,0,0,0,0,0,0,0,0,0,0
      021011      000      000      000
      021014      000      000      000
      021017      000      000
1133
1134      000001      .END
```


BASEXC	003252	253	378#												
BASEXD	003210	266	365#												
BEGIN	001542	48	214#	943											
BITDAT=	010000	208#	555	560	569	579									
BITEXT=	004000	209#	484	492	501	513	523	524	533	534	544	545	594	606	
		608	624	646	647	726									
BIT0 =	000001	20#	323	442	444	578	738	758	781						
BIT00 =	000001	20#													
BIT01 =	000002	20#													
BIT02 =	000004	20#													
BIT03 =	000010	20#													
BIT04 =	000020	20#													
BIT05 =	000040	20#													
BIT06 =	000100	20#													
BIT07 =	000200	20#													
BIT08 =	000400	20#	919												
BIT09	001000	20#	919	936											
BIT1	000002	20#	448	465	521	522	531	532	543	585	605	623	645	696	
		699	737	741	757										
BIT10 =	002000	20#													
BIT11 =	004000	20#	209	919											
BIT12 =	010000	20#	208	458	605	821									
BIT13 =	020000	20#	936												
BIT14	040000	20#	460	645	919										
BIT15 =	100000	20#	315	521	524	534	646	647	724	792	805	823	824	834	
BIT2 =	000004	20#	449	465	512	514	599	709	712	785	792	806	822	823	
BIT3 =	000010	20#	450												
BIT4 =	000020	20#	452	465	560	579	594	605	757						
BIT5 =	000040	20#	453	560	579	584	594	605	607	757					
BIT6 =	000100	20#	242	445	454	461	503	547	598	623	873				
BIT7	000200	20#	483	484	521	523	524	531	533	534	606	608	624	646	
		647	686	699	721	726	770	789	792	805	823	824	834		
BIT8 =	000400	20#	456	822	823										
BIT9	001000	20#	457												
BPTVEC	000014	20#													
CKSWR =	104410	919	936	955#											
CR =	000015	20#	951												
CRLF =	000200	20#	227	951											
CTRCHA	003422	403*	404*	405	409	413#									
CTRLCG	003346	401#	885	894	919	936									
DDISP	177570	20#	61	219											
DELAYO	010550	687	700	713	728	793	809	827	837	911#					
DF0	021006	64	66	68	70	72	74	76	78	80	82	84	1132#		
DH1	017262	64	1090#												
DH10	017443	78	1094#												
DH11	017476	80	1095#												
DH12	017520	66	1096#												
DH13	017563	68	1097#												
DH15	017626	82	1098#												
DH16	017661	84	1099#												
DH5	017313	72	1091#												
DH6	017353	74	1092#												
DH7	017407	70	76	1093#											
DIDATA	010354	269	889#												
DIDINS	001432	172#	619*	635	636*	861*	862*	1039	1042*	1045*	1048*				
DIDINV	001430	171#	339	618*	635*	860*	861	863	865	1035	1039*	1043*	1044*	1045	

DIEINS	001436	1046	1047											
DIEINV	001434	175#	344	641*	658	659*	865*	866*	1040	1041*	1047*	1050*		
DIR	001420	174#	640*	658*	863*	864*	1040*	1046*	1049*					
		166#	433	511*	556	562	571	582	583*	586	595	600	610	681*
		694*	707*	735*	745	755*	765	786*	804*	820*	833*	893*	897	900
		1123	1126	1129										
DIR1	001422	167#												
DISPLA	001142	61#	219*	919*	936*									
DISPRE	000174	44#	219											
DOR	001410	160#	685*	698*	711*	720*	742*	760*	780*	783*	790*	803*	808*	819*
		826*	835*	1122	1129									
DOR1	001412	161#	851											
DUT	015674	247	1071#											
DSWR =	177570	20#	61	219										
DT1	020624	64	1122#											
DT10	020702	78	1126#											
DT11	020716	80	1127#											
DT12	020726	66	1128#											
DT13	020744	68	1129#											
DT15	020762	82	1130#											
DT16	020774	84	1131#											
DT5	020636	72	1123#											
DT6	020652	74	1124#											
DT7	020666	70	76	1125#										
DWARF	001440	177#	214*	217*	234	262*	272*	305	421	678	722			
EMTVEC=	000030	20#	219*											
EM1	015764	64	1074#											
EM10	016513	78	1081#											
EM11	016562	80	1082#											
EM12	016625	66	1083#											
EM13	016716	68	1084#											
EM14	017005	70	1085#											
EM15	017052	82	1086#											
EM16	017123	84	1087#											
EM2	016074	1075#												
EM3	016144	1076#												
EM4	016216	1077#												
EM5	016262	72	1078#											
EM6	016367	74	1079#											
EM7	016442	76	1080#											
ENDOUT	020607	369	382	391	1118#									
ERRTOT	017712	876	1100#											
ERRVEC=	000004	20#	219*	283*	301*	302*	431*	437*	438*	440*	441*	919*		
EVER	001526	204#	245*	303	314*	315*	317	319	847					
EXTMSG	010270	871	873#											
FIXADR	003010	246	322	330#	374	396	890							
FOUND1	017207	307	1088#											
FOUND2	017231	310	1089#											
GNS =	***** U	227	955											
GTSWR =	104407	227	256	411	955#									
HT -	000011	20#	951											
ICSR	001414	163#	334	432	448*	449*	450*	452*	453*	454*	456*	457*	458*	460*
		461*	463*	466	472*	473*	475	482*	484*	485	492*	493*	494	501*
		504	512*	513*	515	522*	523*	524*	525	532*	533*	534*	536	543*
		544*	545*	548	555*	560*	569*	579*	584*	585*	594*	599*	605*	606*
		607*	608*	623*	624*	629*	634*	645*	646*	647*	652*	657*	683*	688

\$LF	001166	61#	918	936	951													
\$LFLG	013733	940#*																
\$LPADR	001106	61#	219*	442*	444*	446*	617*	919*	1051									
\$LPERR	001110	61#	219*	577*	919*	936												
\$MADR1	001222	61#																
\$MADR2	001226	61#																
\$MADR3	001232	61#																
\$MADR4	001236	61#																
\$MAIL	001170	60	61#	219	227	919	936	951										
\$MAMS1	001220	61#																
\$MAMS2	001224	61#																
\$MAMS3	001230	61#																
\$MAMS4	001234	61#																
\$MBADR	001002	60#																
\$MFLG	013732	940#*																
\$MNEW	012106	918#																
\$MSGAD	001204	61#	940*															
\$MSGLG	001206	61#	940*															
\$MSGTY	001170	61#	940*															
\$MSWR	012075	918#																
\$MTYP1	001221	61#																
\$MTYP2	001225	61#																
\$MTYP3	001231	61#																
\$MTYP4	001235	61#																
\$MXCNT	012402	919#																
\$NULL	001154	61#	951															
\$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#	693#	706#	719#	734#	754#	775#				
		818#	845#															
\$OCNT	013240	938#*																
\$OMODE	013242	938#*																
\$OVER	012366	919#																
\$PASS	001176	61#	219*	243*	871*	919	1027											
\$PASTM	001006	60#																
\$PWRAD	014076	943#																
\$PWDRN	013736	219	943#															
\$PWRMG	014072	943#																
\$PWRUP	014010	943#																
\$QUES	001164	61#	274	918	936	951												
\$RDCHR	011466	918#	955															
\$RDDEC-	***** U	955																
\$RDLIN	011556	918#	955															
\$RDOCT	014156	947#	955															
\$RDSZ -	000040	918#																
\$RTNAD	010246	871#																
\$R2A -	***** U	955																
\$SAVRE=	***** U	955																
\$SAVR6	014106	943#*																
\$SCOPE	012120	219	919#															
\$SETUP=	000137	211#	219	227	871	918	919	936										
\$STUP -	177777	211#																
\$SVLAD	012332	919#																
\$SVPC	000106	58#																
\$SWR -	165400	8#	15	22	61	219	420	430	442	444	445	446	448	449				
		450	452	453	454	456	457	458	460	461	462	471	481	490				

CVMNB-B MNCDI		DIAGNOSTIC		MACY11 30G(1063) 08-AUG-79 10:48 PAGE 36												SEQ 0077	
CVMNB8.P11		08-AUG-79 10:38		CROSS REFERENCE TABLE -- MACRO NAMES													
COMMEN	20#																
ENDCOM	20#																
ERROR	20#	299	320	429	436	442	444	445	446	448	449	450	452	453	454		
	456	457	458	460	461	469	478	488	497	507	518	528	539	551	559		
	564	567	574	589	602	613	630	653	691	704	717	732	748	768	797		
	813	831	841	994													
ESCAPE	20#																
FIXUNT	922#	936															
FLOATO	118#	444															
FLOAT1	106#	442															
GETPRI	20#																
GETSWR	12#	20#	227#														
HIBYTE	86#	446															
MULT	20#																
NEWTST	20#	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	693	706	719	734	754	775	818	845					
POP	20#	939	940	943	947												
PUSH	20#	939	940	943	947												
READRW	142#	448	449	450	452	453	454	456	457	458	460						
REPORT	20#																
RFSETO	132#	445	461														
SCOPE	20#	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	693	706	719	734	754	775	818	845	871				
SETPRI	20#	918															
SETTRA	955#																
SETUP	20#	219															
SKIP	20#	422	428	435	442	444	445	446	448	449	450	452	453	454	456		
	457	458	460	461	468	477	487	496	506	517	527	538	550	558	566		
	573	588	601	612	631	654	690	703	716	731	747	767	796	812	830		
	840	848	1028														
SLASH	20#																
SPACE	20#																
STARS	20#	58	60	61	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	693	706	719	734	754	775	818	845		
	871	918	919	936	937	938	939	940	943	947	951	952	955				
SWRSU	20#	219#															
TRMTRP	955#																
TYPBIN	20#																
TYPDEC	20#	871	937														
TYPNAM	20#	227															
TYPNUM	20#																
TYPOCS	20#	1035	1037														
TYPOCT	20#	918	937														
TYPTXT	20#																
\$\$CMRE	61#																
\$\$CMTM	61#																
\$\$ESCA	20#																
\$\$NEWT	20#	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	693	706	719	734	754	775	818	845					
\$\$SET	955#																
\$\$SETM	219#																
\$\$SKIP	20#	422	428	445	448	449	450	452	453	454	456	457	458	460	461		

	468	477	487	496	506	517	527	538	550	573	601	612	690	703	716
.EQUAT	10#	20													
.HEADE	10#	15													
.SETUP	11#	211													
.SWRHI	12#	22													
.SWRLO	22#														
.SACT1	13#	58													
.SAPT8	13#	61#													
.SAPTH	13#	60													
.SAPTY	13#	940													
.SCATC	10#														
.SCMTA	10#	61													
.SE >	10#	871													
.SERRO	10#	936													
.SERRT	12#	937													
.SPARM	11#														
.SPOWE	11#	943													
.SRDOC	12#	947													
.SREAD	11#	918													
.SSAVE	11#														
.SSCOP	11#	919													
.SSPAC	11#														
.SSWDO	11#														
.STRAP	11#	955													
.STYPB	12#	952													
.STYPD	12#	939													
.STYPE	10#	11#	951												
.STYPO	10#	938													

. ABS. 021021 000 CON RO ABS LCL I

ERRORS DETECTED: 0
 CVMNBB,CVMNBB/CRF=CVMNBB
 RUN-TIME: 26 13 1 SECONDS
 RUN-TIME RATIO: 166/41=3.9
 CORE USED: 26K (51 PAGES)