

MINC-11

MNCD DIAGNOSTIC
CVMNEA0

AH-B098A-MC

COPYRIGHT © 1978

FICHE 1 OF 1

DEC 1978

digital

MADE IN USA

This microfiche card contains a grid of frames, each displaying diagnostic data for the MINC-11 system. The data is organized into columns and rows, with some frames containing headers such as 'TEST RESULTS', 'STATUS', and 'ERRORS'. The text is small and dense, typical of microfiche storage. The card is oriented vertically, with the data frames on the left side and a large blank area on the right.

IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-B097A-MC
PRODUCT NAME: CVMNEAO MNCDO (DIGITAL OUT) DIAGNOSTIC TEST
DATE: AUGUST 1978
MAINTAINER: DIAGNOSTIC ENGINEERING

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

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

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

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

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
4.1	PROGRAM START
5.0	SOFTWARE SWITCH REGISTER
5.1	OPTIONS
5.2	CONTROL
6.0	ERROR REPORTING
6.1	ERROR COMMENT
6.2	ERROR DATA
7.0	MISCELLANEOUS
7.1	MNCDO BUS ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	MULTIPLE MNCDO INTERFACE TESTING
7.5	RESTRICTIONS
8.0	EXECUTION TIME
9.0	PROGRAM TEST DESCRIPTIONS
9.1	LOGIC TEST WITH NO TEST MODULE CONNECTED
9.2	LOGIC TEST WITH TEST MODULE CONNECTED
9.3	OUTPUT TEST MODULE L.E.D. LOOP
9.4	PULSE OUTPUT VERIFY LOOP
9.5	LOGIC TEST WITH TESTER SUPPORT
10.0	LISTING

1.0 ABSTRACT

THE MNCDO DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE. ADDITIONAL ROUTINES ARE PROVIDED TO VERIFY NON-DIAGNOSIBLE 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. MNCDO (DIGITAL OUT) 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 TYPEING '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 VMNE??' 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 THEA '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 MNCDO'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 COMMANDS.
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	MNCDO BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

*ALWAYS REPORTED

7.0 MISCELLANEOUS

7.1 MNCDO BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' (LOC. 1244) IF BASE BUS ADDRESS IS NOT 171260.
MODIFY LOCATIN '\$VECT1' (LOC. 1240) IF THE INTERRUPT VECTOR IS NOT 340.

*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 MNCDO INTERFACE TESTING

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

7.5 RESTRICTIONS

ALL USER CONNECTIONS MUST BE REMOVED.

8.0 EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS
TO ABOUT 20 SECONDS WITH ITERATIONS ENABLED WITH ONE MNCDO 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 TEST WITH NO TEST MODULE CONNECTED

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE MNCDO OUTPUT 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 MNCDO'S TO BE TESTED.

9.2 D = LOGIC TEST WITH TEST MODULE CONNECTED

WHEN THE MNCDO TEST MODULE IS CONNECTED, AN ADDITIONAL TEST OF THE CONTROL SIGNALS CAN BE MADE. BY STARTING THIS SECTION, THE OPERATOR INFORMS THE DIAGNOSTIC THAT THE TEST MODULE IS CONNECTED AND TO PERFORM THE ADDITIONAL TEST AT THE END OF THE NORMAL LOGIC TEST.

9.3 O = OUTPUT TEST MODULE L.E.D. LOOP

THE LOOP ENABLES VERIFICATION OF THE LOGIC NOT TESTABLE BY THE DIAGNOSTIC. A FLOATING LIGHT PATTERN IS PRODUCED FOR EASY VERIFICATION OF THE OUTPUT CIRCUITS.

9.4 P = PULSE OUTPUT VERIFY LOOP

THE LOOP ENABLES VERIFICATION OF THE TWO OUTPUT CONTROL SIGNALS. THE OPERATOR MUST USE A SCOPE TO OBSERVE THE OUTPUT PULSES.

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

THE IN-HOUSE TESTER CONSISTS OF A DIGITAL INPUT/OUTPUT DEVICE TO STIMULATE THE INPUT CONTROLS SIGNAL AND SENSE THE OUTPUT DATA AND CONTROL SIGNALS. AN EXPANDED LOGIC AND WRAP-AROUND TESTS ARE EXECUTED TO FULLY VERIFY THE COMPLETE MNCDO OPTION.

10.0 LISTING

19	BASIC DEFINITIONS
21	OPERATIONAL SWITCH SETTINGS
23	TRAP CATCHER
57	ACT11 HOOKS
59	APT PARAMETER BLOCK
60	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
197	INITIALIZE THE COMMON TAGS
204	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
213	INFORM THE OPER. THE TESTS AVAILABLE
252	DETERMINE THE NUMBER OF MNCDO'S ON THE SYSTEM
351	MNCDO TESTS
354	T1 VERIFY CORRECT I.D. CODE FOR MNCDO (IN-HOUSE TESTER ONLY)
364	T2 VERIFY A MNCDO ADDRESS RESPONSE
377	T3 FLOAT A 1 ACROSS THE MNCDO DATA REGISTER
379	T4 FLOAT A 0 ACROSS THE MNCDO DATA REGISTER
380	T5 ENSURE THAT 'RESET' CLEARS THE MNCDO DATA REGISTER
381	T6 VERIFY BYTE OPERATION ON THE MNCDO DATA REGISTER
383	T7 TEST THAT BIT6 OF MNCDO STATUS REGISTER IS READ-WRITE
384	T10 TEST THAT BIT4 OF MNCDO STATUS REGISTER IS READ-WRITE
385	T11 TEST THAT BIT3 OF MNCDO STATUS REGISTER IS READ-WRITE
387	T12 ENSURE THAT 'RESET' CLEARS THE MNCDO STATUS REGISTER
389	T13 VERIFY THAT MNCDO DONE FLAG SETS
398	T14 VERIFY THAT MNCDO DONE FLAG CLEARS WHEN WRITTEN TO A 0
407	T15 VERIFY THAT MNCDO DONE FLAG CLEARS WHEN OUTPUT DATA REGISTER IS WRITTEN
417	T16 INTERRUPT TEST -- VERIFY MNCDO DOES INTERRUPT
457	T17 INTERRUPT TEST -- VERIFY THAT REMOVING OUTPUT IRQ ENABLE WILL REMOVE INTERRUPT REQUEST
494	T20 VERIFY OUTPUT STROBE GENERATES REPLY - MNCDO FIELD TEST MODULE MODE
503	T21 FLOAT A 1 ACROSS THE OUTPUT DATA REGISTER (FIELD DRARF)
516	T22 VERIFY OUTPUT STROBE IS GENERATED (IN-HOUSE TESTER TEST)
547	T23 VERIFY OUTPUT DATA REGISTER (IN-HOUSE TESTER TEST)
566	T24 DETERMINE IF MORE MNCDO'S REMAIN TO BE TESTED
586	END OF PASS ROUTINE
604	MISCELLANEOUS TEST LOOPS
605	MNCDO BYTE STROBE LOOP
619	MNCDO TEST MODULE LAMP LOOP
635	CONTROL C/G HANDLER
651	TTY INPUT ROUTINE
652	SCOPE HANDLER ROUTINE
669	ERROR HANDLER ROUTINE
670	ERROR MESSAGE TIMEOUT ROUTINE
671	BINARY TO OCTAL (ASCII) AND TYPE
672	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
674	APT COMMUNICATIONS ROUTINE
675	POWER DOWN AND UP ROUTINES
679	READ AN OCTAL NUMBER FROM THE TTY
681	TYPE ROUTINE
682	BINARY TO ASCII AND TYPE ROUTINE
685	TRAP DECODER
(3)	TRAP TABLE
780	ASCII MESSAGES


```

15      .TITLE  CVMNE-A MNCDO  DIAGNOSTIC
(1)    :*COPYRIGHT (C) 1978
(1)    :*DIGITAL EQUIPMENT CORP.
(1)    :*MAYNARD, MASS. 01754
(1)    :*
(1)    :*PROGRAM BY SHOOP
(1)    :*
(1)    :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)    :*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)    :*
16
17      171260      ABASE=171260      ;DEFAULT DIGITAL OUTPUT BUS ADDRESS
18      000340      AVECT1=340      ;DEFAULT DIGITAL OUTPUT VECTOR ADDRESS
19      .SBTTL  BASIC DEFINITIONS
(1)
(1)    :*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)    001100      STACK= 1100
(1)    .EQUIV  EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
(1)    .EQUIV  IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)    :*MISCELLANEOUS DEFINITIONS
(1)    000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
(1)    000012      LF= 12      ;;CODE FOR LINE FEED
(1)    000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
(1)    000200      CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)    177776      PS= 177776   ;;PROCESSOR STATUS WORD
(1)    .EQUIV  PS,PSW
(1)    177774      STKLMT= 177774 ;;STACK LIMIT REGISTER
(1)    177772      PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)    177570      DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1)    177570      DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)
(1)    :*GENERAL PURPOSE REGISTER DEFINITIONS
(1)    000000      R0= %0      ;;GENERAL REGISTER
(1)    000001      R1= %1      ;;GENERAL REGISTER
(1)    000002      R2= %2      ;;GENERAL REGISTER
(1)    000003      R3= %3      ;;GENERAL REGISTER
(1)    000004      R4= %4      ;;GENERAL REGISTER
(1)    000005      R5= %5      ;;GENERAL REGISTER
(1)    000006      R6= %6      ;;GENERAL REGISTER
(1)    000007      R7= %7      ;;GENERAL REGISTER
(1)    000006      SP= %6      ;;STACK POINTER
(1)    000007      PC= %7      ;;PROGRAM COUNTER
(1)
(1)    :*PRIORITY LEVEL DEFINITIONS
(1)    000000      PR0= 0      ;;PRIORITY LEVEL 0
(1)    000040      PR1= 40     ;;PRIORITY LEVEL 1
(1)    000100      PR2= 100    ;;PRIORITY LEVEL 2
(1)    000140      PR3= 140    ;;PRIORITY LEVEL 3
(1)    000200      PR4= 200    ;;PRIORITY LEVEL 4
(1)    000240      PR5= 240    ;;PRIORITY LEVEL 5
(1)    000300      PR6= 300    ;;PRIORITY LEVEL 6
(1)    000340      PR7= 340    ;;PRIORITY LEVEL 7

```

```
(1)                                     ;*'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)                                     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)      100000                        BIT15= 100000
(1)      040000                        BIT14= 40000
(1)      020000                        BIT13= 20000
(1)      010000                        BIT12= 10000
(1)      004000                        BIT11= 4000
(1)      002000                        BIT10= 2000
(1)      001000                        BIT09= 1000
(1)      000400                        BIT08= 400
(1)      000200                        BIT07= 200
(1)      000100                        BIT06= 100
(1)      000040                        BIT05= 40
(1)      000020                        BIT04= 20
(1)      000010                        BIT03= 10
(1)      000004                        BIT02= 4
(1)      000002                        BIT01= 2
(1)      000001                        BIT00= 1
(1)                                     .EQUIV BIT09,BIT9
(1)                                     .EQUIV BIT08,BIT8
(1)                                     .EQUIV BIT07,BIT7
(1)                                     .EQUIV BIT06,BIT6
(1)                                     .EQUIV BIT05,BIT5
(1)                                     .EQUIV BIT04,BIT4
(1)                                     .EQUIV BIT03,BIT3
(1)                                     .EQUIV BIT02,BIT2
(1)                                     .EQUIV BIT01,BIT1
```

```
(1) .EQUIV BIT00,BIT0
(1)
(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;:'T' BIT
(1) 000014 TRTVEC= 14 ;:TRACE TRAP
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;:POWER FAIL
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;:'TRAP' TRAP
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
```

```

21      .SBTTL OPERATIONAL SWITCH SETTINGS
(1)      : *
(1)      : *
(1)      : *
(1)      : *
(1)      : *
(1)      : *
(1)      : *
(1)      : *
(1)      : *
(1)      : *
(1)      : *
22      : *
23      : *
24      : *
25      : *
26      : *
27      : *
28      : *
29      : *
30      : *
40      000004
41      000004 012150 000200
42      000174 000174
43      000174 000000
44      000176 000000
45
46      000200
47      000200 000137 001450
48      000204 000137 001476
49      000210 000137 001462
50
51
52      000100
53
54      000100 000104 000200 000002

```

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT SIZING THE # OF MNCDO'S
11	INHIBIT ITERATIONS
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<7:0>

```

.SBTTL TRAP CATCHER
.=0
;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ".+2"
;*AND "JSR PC,R0" SEQUENCE TO CATCH ILLEGAL INTERRUPTS.
;*AND INTERRUPTS TO THE WRONG VECTOR.
;*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
;*VECTORS.
.=4
.WORD IOTRD,200 ;HANDLE BUSS ERROR.
.=174
DISPREG: .WORD 0 ;SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;SOFTWARE SWITCH REGISTER
.=200
JMP BEGIN ;RESTART ADDRESS
JMP RESTRT ;INDICATE MNCDO IN-HOUSE TESTER MODE
JMP TESTER
.=100
104,200,RTI ;LSI-11 'B EVENT' PROTECTION

```

```

56
57      .SBTTL  ACT11 HOOKS
(1)
(2)      ::*****
(1)      ;HOOKS REQUIRED BY ACT11
(1)      000106      $SVPC=.          ;SAVE PC
(1)      000046      .=46
(1) 000046 005514      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)      000052      .=52
(1) 000052 000000      .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)      000106      .=$SVPC          ;; RESTORE PC
58      001000      .=1000
59      .SBTTL  APT PARAMETER BLOCK
(1)
(2)      ::*****
(1)      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      ::*****
(1)      001000      .$X=.          ;;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      $APTHDR      ;;POINT TO APT HEADER BLOCK
(1)      001000      .=$X          ;;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) 0010i2 000032      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

60

(1)

(2)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(2)

(2)

(2)

(3)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

.SBTTL COMMON TAGS

::*****

::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS

::*USED IN THE PROGRAM.

.=1100

\$CMTAG: ::START OF COMMON TAGS

.WORD 0

\$TSTNM: .BYTE 0 ::CONTAINS THE TEST NUMBER

\$ERFLG: .BYTE 0 ::CONTAINS ERROR FLAG

\$ICNT: .WORD 0 ::CONTAINS SUBTEST ITERATION COUNT

\$LPADR: .WORD 0 ::CONTAINS SCOPE LOOP ADDRESS

\$LPERR: .WORD 0 ::CONTAINS SCOPE RETURN FOR ERRORS

\$ERTTL: .WORD 0 ::CONTAINS TOTAL ERRORS DETECTED

\$ITEMB: .BYTE 0 ::CONTAINS ITEM CONTROL BYTE

\$ERMAX: .BYTE 1 ::CONTAINS MAX. ERRORS PER TEST

\$ERRPC: .WORD 0 ::CONTAINS PC OF LAST ERROR INSTRUCTION

\$GDADR: .WORD 0 ::CONTAINS ADDRESS OF 'GOOD' DATA

\$BDADR: .WORD 0 ::CONTAINS ADDRESS OF 'BAD' DATA

\$GDDAT: .WORD 0 ::CONTAINS 'GOOD' DATA

\$BDDAT: .WORD 0 ::CONTAINS 'BAD' DATA

.WORD 0 ::RESERVED--NOT TO BE USED

.WORD 0

\$AUTOB: .BYTE 0 ::AUTOMATIC MODE INDICATOR

\$INTAG: .BYTE 0 ::INTERRUPT MODE INDICATOR

.WORD 0

\$SWR: .WORD DSWR ::ADDRESS OF SWITCH REGISTER

\$DISPLAY: .WORD DDISP ::ADDRESS OF DISPLAY REGISTER

\$TKS: 177560 ::TTY KBD STATUS

\$TKB: 177562 ::TTY KBD BUFFER

\$TPS: 177564 ::TTY PRINTER STATUS REG. ADDRESS

\$TPB: 177566 ::TTY PRINTER BUFFER REG. ADDRESS

\$NULL: .BYTE 0 ::CONTAINS NULL CHARACTER FOR FILLS

\$FILLS: .BYTE 2 ::CONTAINS # OF FILLER CHARACTERS REQUIRED

\$FILLC: .BYTE 12 ::INSERT FILL CHARS. AFTER A 'LINE FEED'

\$TPFLG: .BYTE 0 ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)

\$TIMES: 0 ::MAX. NUMBER OF ITERATIONS

\$ESCAPE: 0 ::ESCAPE ON ERROR ADDRESS

\$QUES: .ASCII /?/ ::QUESTION MARK

\$CRLF: .ASCII <15> ::CARRIAGE RETURN

\$LF: .ASCIZ <12> ::LINE FEED

::*****

.SBTTL APT MAILBOX-ETABLE

::*****

.EVEN

\$MAIL: ::APT MAILBOX

\$MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE

\$FATAL: .WORD AFATAL ::FATAL ERROR NUMBER

\$TESTN: .WORD ATESTN ::TEST NUMBER

\$PASS: .WORD APASS ::PASS COUNT

\$DEVCT: .WORD ADEVCT ::DEVICE COUNT

\$UNIT: .WORD AUNIT ::I/O UNIT NUMBER

```

(2) 001204 000000 $MSGAD: .WORD  AMMSGAD  ::MESSAGE ADDRESS
(2) 001206 000000 $MSGLG: .WORD  AMMSGLG  ::MESSAGE LENGTH
(2) 001210          $ETABLE:          ::APT ENVIRONMENT TABLE
(2) 001210          000 $ENV: .BYTE   AENV      ::ENVIRONMENT BYTE
(2) 001211          000 $ENVM: .BYTE  AENVM     ::ENVIRONMENT MODE BITS
(2) 001212 000000 $SWREG: .WORD  ASWREG   ::APT SWITCH REGISTER
(2) 001214 000000 $USWR: .WORD  AUSWR    ::USER SWITCHES
(2) 001216 000000 $CPUOP: .WORD  ACPUOP   ::CPU TYPE,OPTIONS
(2)          :*          BITS 15-11=CPU TYPE
(2)          :*          11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)          :*          11/70=06,PDQ=07,Q=10
(2)          :*          BIT 10=REAL TIME CLOCK
(2)          :*          BIT 9=FLOATING POINT PROCESSOR
(2)          :*          BIT 8=MEMORY MANAGEMENT
(2) 001220          000 $MAMS1: .BYTE  AMAMS1   ::HIGH ADDRESS,M.S. BYTE
(2) 001221          000 $MTYP1: .BYTE  AMTYP1   ::MEM. TYPE,BLK#1
(2)          :*          MEM.TYPE BYTE  -- (HIGH BYTE)
(2)          :*          900 NSEC CORE=001
(2)          :*          300 NSEC BIPOLAR=002
(2)          :*          500 NSEC MOS=003
(2) 001222 000000 $MADR1: .WORD  AMADR1   ::HIGH ADDRESS,BLK#1
(2)          :*          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2) 001224          000 $MAMS2: .BYTE  AMAMS2   ::HIGH ADDRESS,M.S. BYTE
(2) 001225          000 $MTYP2: .BYTE  AMTYP2   ::MEM. TYPE,BLK#2
(2) 001226 000000 $MADR2: .WORD  AMADR2   ::MEM.LAST ADDRESS,BLK#2
(2) 001230          000 $MAMS3: .BYTE  AMAMS3   ::HIGH ADDRESS,M.S.BYTE
(2) 001231          000 $MTYP3: .BYTE  AMTYP3   ::MEM. TYPE,BLK#3
(2) 001232 000000 $MADR3: .WORD  AMADR3   ::MEM.LAST ADDRESS,BLK#3
(2) 001234          000 $MAMS4: .BYTE  AMAMS4   ::HIGH ADDRESS,M.S.BYTE
(2) 001235          000 $MTYP4: .BYTE  AMTYP4   ::MEM. TYPE,BLK#4
(2) 001236 000000 $MADR4: .WORD  AMADR4   ::MEM.LAST ADDRESS,BLK#4
(2) 001240 000340 $VECT1: .WORD  AVECT1   ::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001242 000000 $VECT2: .WORD  AVECT2   ::INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001244 171260 $BASE: .WORD  ABASE     ::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001246 000000 $DEVN: .WORD  ADEVN     ::DEVICE MAP
(2) 001250 000000 $CDW1: .WORD  ACDW1     ::CONTROLLER DESCRIPTION WORD#1
(2) 001252 000000 $CDW2: .WORD  ACDW2     ::CONTROLLER DESCRIPTION WORD#2
(2) 001254          $ETEND:
(2)          .MEXIT

```

```
(1) .SBTTL ERROR POINTER TABLE
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;* EM ;:POINTS TO THE ERROR MESSAGE
(1) ;* DH ;:POINTS TO THE DATA HEADER
(1) ;* DT ;:POINTS TO THE DATA
(1) ;* DF ;:POINTS TO THE DATA FORMAT
```

```
(1) 001254 $ERRTB:
61
62
63 001254 013074 014116 014666 ;ITEM 1 EM1,DH1,DT1,DF0 ;MNCDO BUS ERROR
001262 014764
64
65 001264 013203 014147 014700 ;ITEM 2 EM2,DH2,DT2,DF0 ;MNCDO DATA REGISTER ERROR
001272 014764
66
67 001274 013253 014203 014714 ;ITEM 3 EM3,DH3,DT3,DF0 ;MNCDO STATUS REGISTER ERROR
001302 014764
68
69 001304 013325 014237 014730 ;ITEM 4 EM4,DH4,DT4,DF0 ;MNCDO INTERRUPT ERROR
001312 014764
70
71 001314 013526 014203 014714 ;ITEM 5 EM14,DH3,DT3,DF0 ;MNCDO INCORRECT I.D. VALUE
001322 014764
72
73 001324 013371 014261 014740 ;ITEM 6 EM6,DH6,DT6,DF0 ;MNCDO ILLEGAL INTR. OR TRAP
001332 014764
74
75 001334 013442 014315 014752 ;ITEM 7 EM7,DH7,DT7,DF0 ;EXISTING MNCDO FAILED TO RESPOND
001342 014764
```



```

148 001344 000000      OCSR: 0
149 001346 000000      OCSR1: 0          ;HIGH BYTE ADDRESS
150
151 001350 000000      DOR: 0
152 001352 000000      DOR1: 0          ;HIGH BYTE ADDRESS
153
154 001354 000000      DODINV: 0
155 001356 000000      DODINS: 0
156 001360 000000      DWARF: 0          ;0= NO DWARF'S =1 MNCDO IN-HOUSE TESTER = 2 FIELD DWARF
157 001362 000000      TEMP: 0
158 001364 000000      TEMP1: 0
159 001366 167770      TSTR0: 167770    ;IN-HOUSE TESTER ADDRESS
160 001370 167772      TSTR2: 167772
161 001372 167774      TSTR4: 167774
162 001374 167776      TSTR6: 167776
163 001376 000000      MASKNM: 0
164 001400 000004      VADDR: 4
165 001402 000340      VECLST: AVECT1    ;DEVICE MAP
166 001404 000344      AVECT1+4         ;MULTIPLE UNITS, ADDRESS DIFFERENCE
167 001406 000350      AVECT1+10        ;VECTOR OF MNCDO #0
168 001410 000354      AVECT1+14        ;#1
169 001412 000530      AVECT1+170       ;#2
170 001414 000520      AVECT1+160       ;#3
171 001416 000510      AVECT1+150       ;#4
172 001420 000500      AVECT1+140       ;#5
173 001422 000000      VECOFF: 0        ;#6
174 001424 000004      4                ;OFFSET FROM $VECT1 TO UNIT #0 VECTOR
175 001426 000010      10               ;#1
176 001430 000014      14               ;#2
177 001432 000130      130              ;#3
178 001434 000120      120              ;#4
179 001436 000110      110              ;#5
180 001440 000100      100              ;#6
181 001442 000000      EVER: 0
182 001444 000000      BADUNT: 0        ;BAD UNIT INDICATOR
183 001446 000000      UNITBD: 0
184
185          010000      BITDAT=BIT12    ;MAINT INPUT INHIBIT
186          004000      BITEXT=BIT11   ;INPUT MAINT STROBE
187
189 001450 005037 001364  BEGIN: CLR      TEMP1    ;CLEAR RESTART INDICATOR
190 001454 005037 001360      CLR      DWARF        ;INDICATE NO DWARF
191 001460 000410          BR      RBEGO
192 001462 012737 000001 001360  TESTER: MOV     #1, DWARF ;INDICATE TESTER
193 001470 005037 001364      CLR      TEMP1        ;CLEAR RESTART INDICATOR
194 001474 000402          BR      RBEGO
195 001476 005237 001364  RESTRT: INC     TEMP1    ;INDICATE RESTART
196
197 001502          RBEGO:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001502 012706 001100      MOV     #$CMTAG, R6    ;;FIRST LOCATION TO BE CLEARED
(1) 001506 005026          CLR     (R6)+          ;;CLEAR MEMORY LOCATION
(1) 001510 022706 001140      CMP     #SWR, R6      ;;DONE?

```

```

(1) 001514 001374          BNE      .-6          ;;LOOP BACK IF NO
(1) 001516 012706 001100  MOV      #STACK,SP    ;;SETUP THE STACK POINTER
(1)                                ;;INITIALIZE A FEW VECTORS
(1) 001522 012737 007366 000020  MOV      #$$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001530 012737 000340 000022  MOV      #340,@#IOTVEC+2 ;;LEVEL 7
(1) 001536 012737 007706 000030  MOV      #$$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001544 012737 000340 000032  MOV      #340,@#EMTVEC+2 ;;LEVEL 7
(1) 001552 012737 012064 000034  MOV      #$$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001560 012737 000340 000036  MOV      #340,@#TRAPVEC+2;LEVEL 7
(1) 001566 012737 011204 000024  MOV      #$$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 001574 012737 000340 000026  MOV      #340,@#PWRVEC+2 ;;LEVEL 7
(1) 001602 013737 005462 005454  MOV      $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
(1) 001610 005037 001160          CLR      $TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001614 005037 001162          CLR      $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001620 112737 000001 001115  MOV      #1,$ERMAX      ;;ALLOW ONE ERROR PER TEST
(1) 001626 012737 001626 001106  MOV      #,$$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001634 012737 001634 001110  MOV      #,$$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) 001642 013746 000004          MOV      @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
(2) 001646 012737 001702 000004  MOV      #64,$@#ERRVEC  ;;SET UP ERROR VECTOR
(2) 001654 012737 177570 001140  MOV      #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001662 012737 177570 001142  MOV      #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001670 022777 177777 177242  CMP      #-1,@SWR       ;;TRY TO REFERENCE HARDWARE SWR
(2) 001676 001012          BNE      66$           ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                                ;;AND THE HARDWARE SWR IS NOT = -1
(2) 001700 000403          BR       65$           ;;BRANCH IF NO TIMEOUT
(2) 001702 012716 001710 64$:    MOV      #65$,(SP)      ;;SET UP FOR TRAP RETURN
(2) 001706 000002          RTI
(2) 001710 012737 000176 001140 65$:    MOV      #SWREG,SWR     ;;POINT TO SOFTWARE SWR
(2) 001716 012737 000174 001142  MOV      #DISPREG,DISPLAY
(2) 001724 012637 000004 66$:    MOV      (SP)+,@#ERRVEC ;;PESTORE ERROR VECTOR
(1)
(2) 001730 005037 001176          CLR      $PASS         ;;CLEAR PASS COUNT
(2) 001734 132737 000200 001211  BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
(2) 001742 001403          BEQ     67$           ;;YES,USE NON-APT SWITCH
(2) 001744 012737 001212 001140  MOV      #$$SWREG,SWR   ;;NO,USE APT SWITCH REGISTER
(2) 001752          67$:
198 ;ROUTINE TO OVERLAY THE 1ST 3 LOC. OF THE $TYPE ROUTINE
199 001752 012737 005046 011526  MOV      #5046,$TYPE    ;;OVERLAY TYPE ROUTINE
200 001760 012737 012746 011530  MOV      #12746,$TYPE+2 ;;CLR -(SP)
201 001766 012737 011540 011532  MOV      #$$TYPE+12,$TYPE+4 ;MOV      #$$TYPE+12,-(SP)
202 001774 012737 000002 011534  MOV      #RTI,$TYPE+6   ;;RTI
203 002002 004737 006104          JSR     PC,$TKINT       ;;ENABLE KRB INTR.
204 ;SBTTL TYPE PROGRAM NAME
(1) ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002006 005227 177777          INC     #-1            ;;FIRST TIME?
(1) 002012 001053          BNE     68$           ;;BRANCH IF NO
(1) 002014 022737 005514 000042  CMP     #$$ENDAD,@#42  ;;ACT-11?
(1) 002022 001447          BEQ     68$           ;;BRANCH IF YES
(1) 002024 104401 002072          TYPE   ,69$          ;;TYPE ASCIZ STRING
(2) ;SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002030 005737 000042          TST    @#42           ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002034 001012          BNE     70$           ;;BRANCH IF YES

```

```

(2) 002036 123727 001210 000001      CMPB  $ENV,#1      ;;ARE WE RUNNING UNDER APT?
(2) 002044 001406                    BEQ   70$          ;;BRANCH IF YES
(2) 002046 023727 001140 000176      CMP   SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
(2) 002054 001005                    BNE   71$          ;;BRANCH IF NO
(2) 002056 104407                    GTSWR              ;;GET SOFT-SWR SETTINGS
(2) 002060 000403                    BR    71$
(2) 002062 112737 000001 001134 70$:  MOVB  #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
(2) 002070 71$:
(1) 002070 000424                    BR    68$          ;;GET OVER THE ASCIZ
(1) 69$:  .ASCIZ <CRLF>#CVMNE-A MNCDO (DIGIAL OUT) DIAGNOSTIC#<CRLF>
(1) 68$:
205 002142 105737 001134              TSTB  $AUTOB      ;TEST IF ACT/XXDP/APT RUN MODE
206 002146 001402                    BEQ   3$          ;BR IF NONE
207 002150 000137 002424              JMP   LOGIC       ;RUN LOGIC TEST
208 002154 005737 001364 3$:      TST   TEMP1      ;TEST IF RESTART
209 002160 001007                    BNE   MTEST1     ;BR IF YES
210 002162 005737 001360              TST   DWARF      ;TEST IF NO DWARF
211 002166 001402                    BEQ   MTEST      ;BR IF NONE
212 002170 104401 014603              TYPE, MSGCAB     ;TELL OPERATOR ABOUT TESTER CABLE
213 .SBTTL INFORM THE OPER. THE TESTS AVAILABLE
214 002174 104401 012362  MTEST: TYPE, PRIME0 ;INFORM THEM OF THE DIFFERENT TESTS
215 002200 000005  MTEST1: RESET      ;HIT IT
216 002202 052777 000100 176734  BIS   #BIT6,@$TKS ;ENABLE TKS INTR.
217 002210 005037 001176              CLR   $PASS      ;INIT THE PASS COUNTER
218 002214 005037 001112              CLR   $ERTTL     ;INIT THE TOTAL ERROR COUNT
219 002220 005037 001442              CLR   EVER       ;INIT THE UNIT TIMEOUT FLAG
220 002224 004737 002712              JSR   PC,FIXADR  ;ENSURE CORRECT BASE/VECTOR ADDRESS IS LOADED
221 002230 104401 013004              TYPE, DOT        ;INDICATE THE REST POINT
222 002234 104412                    RDLIN           ;GET OPER. INPUT
223 002236 013637 002416              MOV   @(SP)+,RUNIT ;GET 1ST CHARACTER
224 002242 142737 000040 002416      BICB  #40,RUNIT  ;ENSURE UPPER CASE
225 002250 122737 000102 002416      CMPB  #'B,RUNIT  ;TEST FOR 'B'
226 002256 001002                    BNE   1$          ;BR IF NOT
227 002260 000137 003070              JMP   BASEXC     ;CHANGE THE BASE ADDRESS
228 002264 122737 000104 002416 1$:  CMPB  #'D,RUNIT  ;TEST FOR 'D'
229 002272 001005                    BNE   2$          ;BR IF NOT
230 002274 012737 000002 001360      MOV   #2,DWARF   ;INDICATE FIELD DWARF IS CONNECTED
231 002302 000137 002420              JMP   LOGICO     ;RUN LOGIC TEST WITH DWARF
232 002306 122737 000110 002416 2$:  CMPB  #'H,RUNIT  ;TEST FOR 'H'
233 002314 001727                    BEQ   MTEST      ;BR AND RETYPE THE LIST OF TEST
234 002316 122737 000114 002416 3$:  CMPB  #'L,RUNIT  ;TEST FOR 'L'
235 002324 001005                    BNE   4$          ;BR IF NOT
236 002326 042737 000002 001360      BIC   #2,DWARF   ;REMOVE DWARF INDICATOR
237 002334 000137 002424              JMP   LOGIC     ;RUN LOGIC TEST WITHOUT DWARF
238 002340 122737 000117 002416 4$:  CMPB  #'O,RUNIT  ;TEST FOR 'O'
239 002346 001002                    BNE   5$          ;BR IF NOT
240 002350 000137 005702              JMP   DOLAMP     ;RUN DWARF OUTPUT LAMP LOOP
241 002354 122737 000120 002416 5$:  CMPB  #'P,RUNIT  ;TEST FOR 'P'
242 002362 001002                    BNE   6$          ;BR IF NOT
243 002364 000137 005634              JMP   HIGHLO     ;RUN PULSE OUTPUT
244 002370 122737 000107 002416 6$:  CMPB  #'G,RUNIT  ;TEST FOR 'G'
245 002376 001003                    BNE   77$        ;BR IF NOT
246 002400 104407                    GTSWR
247 002402 000137 002200              JMP   MTEST1     ;RETYPE DOT

```

248	002406	104401	001164		77\$:	TYPE,	\$QUES		:TYPE '?'
249	002412	000137	002200			JMP	MTEST1		:RETYPE DOT
250	002416	000000			RUNIT:	0			
251									
252						.SBTTL	DETERMINE THE NUMBER OF MNCDO'S ON THE SYSTEM		
253	002420	104401	014471		LOGICO:	TYPE,	MSGDWF		:TELL OPER. ABOUT CONNECTING DWARF
254	002424	013737	001244	001126	LOGIC:	MOV	\$BASE,\$BDDAT		:GET BASE ADDRESS
255	002432	005037	001376			CLR	MASKNM		
256	002436	005037	001202			CLR	\$UNIT		:CLEAR UNIT NUMBER
257	002442	012737	002516	000004		MOV	#2\$,ERRVEC		:LOAD RETURN ADDRESS
258	002450	005777	176452		1\$:	TST	@\$BDDAT		:TEST IF ADDRESS EXISTS
259	002454	063737	001400	001126		ADD	VADDR,\$BDDAT		:UPDATE BUS ADDRESS
260	002462	005237	001202			INC	\$UNIT		:UPDATE UNIT COUNT
261	002466	005737	001210			TST	\$ENV		:TEST IF 'DO NOT SIZE'
262	002472	100423				BMI	3\$:BR IF NO SIZEING
263	002474	032777	010000	176436		BIT	#SW12,@SWR		:TEST IF SW 12 IS SET
264	002502	001017				BNE	3\$:BR IF INHIBIT SIZING SWITCH IS SET
265	002504	022737	000010	001202		CMP	#8.,\$UNIT		:TEST IF MAX NUMBER
266	002512	001356				BNE	1\$:BR IF NOT
267	002514	000412				BR	3\$:BR IF MAX
268	002516	022626			2\$:	CMP	(SP)+,(SP)+		:RESTORE STACK
269	002520	005737	001202			TST	\$UNIT		:TEST IF ANY EXIST
270	002524	001006				BNE	3\$:BR IF ANY ARE THERE
271	002526	005737	000042			TST	@#42		:TEST IF XXDP CHAIN MODE
272	002532	001003				BNE	3\$:BR IF YES
273	002534	104001				ERROR	1		:BASE ADDRESS CAUSED A BUS TRAP
274	002536	000137	005426			JMP	\$EOP		
275	002542	012737	012150	000004	3\$:	MOV	#IOTRD,ERRVEC		:RESTORE ERROR VECTOR
276	002550	012737	000200	000006		MOV	#200,ERRVEC+2		
277	002556	005737	001442			TST	EVER		:TEST IF # HAS BEEN REPORTED
278	002562	100426				BMI	4\$:BR IF IT HAS
279	002564	005737	001360			TST	DWARF		:TEST IF TESTER MODE
280	002570	001014				BNE	6\$:BR IF TESTER
281	002572	104401	013573			TYPE	,FOUND1		:TELL OPERATOR # OF MNCDO FOUND
282	002576	013746	001202			MOV	\$UNIT,-(SP)		
283	002602	104405				TYPDS			
284	002604	104401	013616			TYPE	,FOUND2		
285	002610	005737	001202			TST	\$UNIT		:TEST IF ANY UNITS
286	002614	001002				BNE	6\$:BR IF SOME
287	002616	000137	005426			JMP	\$EOP		:REPORT EOP
288	002622	013737	001202	001442	6\$:	MOV	\$UNIT,EVER		:SAVE THE # OF MNCDO'S FOR LATER
289	002630	052737	100000	001442		BIS	#BIT15,EVER		:SET 'REPORTED NUMBER FLAG'
290	002636	000410				BR	5\$		
291	002640	123737	001442	001202	4\$:	CMPB	EVER,\$UNIT		:TEST IF ANY HAVE GONE AWAY
292	002646	001404				BEQ	5\$:BR IF ALL ARE STILL THERE
293	002650	113737	001442	001362		MOVB	EVER,TEMP		:SAVE FOR ERROR REPORT
294	002656	104007				ERROR	7		:EXISTING UNIT FAILED TO RESPOND
295	002660	005037	001202		5\$:	CLR	\$UNIT		:RESET UNIT POINTER
296	002664	004737	002712			JSR	PC,FIXADR		:LOAD BUS ADDRESSES
297	002670	012737	000001	001376		MOV	#BIT0,MASKNM		:LOAD DEVICE MASK
298	002676	005037	001444			CLR	BADUNT		:RESET BAD UNIT INDICATOR
299	002702	005046				CLR	-(SP)		
300	002704	012746	003164			MOV	#TST1,-(SP)		
301	002710	000002				RTI			:LOWER CPU PRIORITY

```

302                                     ;SUBROUTINE TO FIX DEVICE ADDRESS AND VECTORS
303 002712 012700 001344     FIXADR: MOV    #OCSR,R0      ;LOAD ADDRESS POINTER
304 002716 013701 001244     MOV    $BASE,R1      ;LOAD INITIAL BUS ADDRESS
305 002722 010120             1$:  MOV    R1,(R0)+      ;LOAD DEVICE ADDRESS
306 002724 005201             INC    R1          ;UPDATE BUS ADDRESS VALUE
307 002726 020027 001354     CMP    R0,#DOR1+2    ;TEST IF DONE WITH BUS ADDRESSES
308 002732 001373             BNE    1$          ;BR IF NOT
309 002734 013701 001240     MOV    $VECT1,R1     ;LOAD VECTOR POINTER
310 002740 010120             MOV    R1,(R0)+      ;LOAD DEVICE VECTOR ADDRESS
311 002742 005721             TST    (R1)+         ;INC FOR STATUS WORD
312 002744 010110             MOV    R1,(R0)       ;LOAD ADDRESS OF STATUS WORD
313 002746 012700 001402     MOV    #VECLST,R0    ;GET POINTER TO ACTUAL VECTOR AREA
314 002752 012701 001422     MOV    #VECOFF,R1    ;GET POINTER TO VECTOR OFFSET VALUES
318 002756 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 002762 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
(1) 002764 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 002770 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
(1) 002772 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 002776 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
(1) 003000 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 003004 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
(1) 003006 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 003012 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
(1) 003014 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 003020 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
(1) 003022 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 003026 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
(1) 003030 013710 001240     MOV    $VECT1,(R0)   ;LOAD BASE VECTOR
(1) 003034 062120             ADD    (R1)+,(R0)+   ;ADD VECTOR OFFSET
319
320                                     ;ALSO TO LOAD INTELLIGENT TRAP CATCHER
321 003036 012700 000250     MOV    #250,R0       ;LOAD R0
322 003042 012701 000252     MOV    #252,R1       ;LOAD R1
323 003046 012702 004700     MOV    #4700,R2      ;LOAD ILLEGAL INSTR. INTO R2
324 003052 010120             4$:  MOV    R1,(R0)+      ;LOAD .+2
325 003054 010220             MOV    R2,(R0)+      ;LOAD ILLEGAL INSTR.
326 003056 022121             CMP    (R1)+,(R1)+   ;BUMP R1
327 003060 020027 001000     CMP    R0,#1000      ;DONE?
328 003064 001372             BNE    4$            ;NO-BR
329 003066 000207             RTS    PC            ;RETURN
330
331                                     ;ROUTINE TO ASK THE OPERATOR FOR ADRS. & VECTOR INFORMATION
332 003070 104401 014350     BASEXC: TYPE    ,ADROUT ;ASK FOR OUTPUT ADDR.
333 003074 013746 001244     MOV    $BASE,-(SP)    ;LOAD DEFAULT ONTO STACK
334 003100 104402             TYPOC
335 003102 104401 014464     TYPE    ,ENDOUT      ;CONTINUE MESSAGE
336 003106 104413             RDOCT                ;AND WAIT FOR INPUT
337 003110 005726             TST    (SP)+         ;DOES THE OPERATOR WANT THE DEFAULT
338 003112 001403             BEQ    3$            ;YES-BRANCH
339 003114 016637 177776 001244 3$:  MOV    -2(SP),$BASE   ;SAVE ANSWER
340 003122 104401 014415     TYPE    ,VECOUT      ;ASK FOR OUTPUT VECTOR
341 003126 013746 001240     MOV    $VECT1,-(SP)  ;LOAD DEFAULT ONTO STACK
342 003132 104402             TYPOC
343 003134 104401 014464     TYPE    ,ENDOUT      ;CONTINUE MESSAGE

```

CVMNE-A MNCDO
CVMNEA.P11

DIAGNOSTIC
DETERMINE THE NUMBER OF MNCDO'S ON THE SYSTEM

MACY11 27(654) 19-SEP-78 09:02 PAGE 3-5

1 2

SEQ 0021

344	003140	104413				RDOCT		:AND WAIT FOR INPUT
345	003142	005726				TST	(SP)+	:DOES THE OPERATOR WANT THE DEFAULT?
346	003144	001403				BEQ	4\$:YES-BRANCH
347	003146	016637	177776	001240		MOV	-2(SP), \$VECT1	:NO-SAVE ANSWER
348	003154	004737	002712		4\$:	JSR	PC, FIXADR	:LOAD THE NEW ADDR:
349	003160	000137	002200			JMP	MTEST1	:RESTART

```
354 (3) :*****  
(3) :*TEST 1 VERIFY CORRECT I.D. CODE FOR MNCDO (IN-HOUSE TESTER ONLY)  
(2) 003164 000004 :*****  
355 003166 022737 000001 001360 TST1: SCOPE  
356 003174 001020 CMP #1,DWARF ;TEST IF "IN-HOUSE TESTER" MODE  
357 003176 005077 176166 BNE TST2 ;:BR IF NOT  
358 003202 017737 176164 001126 CLR @TSTR2 ;ENSURE TESTER MODE  
359 003210 042737 177417 001126 MOV @TSTR4,$BDDAT ;READ I.D. VALUE  
360 003216 012737 000160 001124 BIC #177417,$BDDAT ;MASK TO OTHER BITS  
361 003224 023737 001124 001126 MOV #160,$GDDAT ;LOAD EXPECTED I.D. VALUE  
362 003232 001401 CMP $GDDAT,$BDDAT ;COMPARE THEM  
363 003234 104005 BEQ TST2 ;:BR IF SAME  
364 (3) ERROR 5 ;MNCDO INCORRECT I.D. VALUE READ  
(3) :*****  
(3) :*TEST 2 VERIFY A MNCDO ADDRESS RESPONSE  
(2) 003236 000004 :*****  
365 003240 012737 003260 000004 TST2: SCOPE  
366 003246 005777 176072 MOV #1$,ERRVEC ;LOAD BUS TRAP RETURN  
367 003252 005777 176072 TST @OCSR ;TEST OUTPUT STATUS REGISTER  
368 003256 000412 TST @DOR ;TEST OUTPUT DATA REGISTER  
369 003260 005726 BR 2$ ;:BR IF NO TIMEOUT  
370 003262 104001 1$: TST (SP)+ ;CLEAN THE STACK  
371 003264 012737 012150 000004 ERROR 1 ;BUS TIMEOUT WHEN ADDRESSING MNCDO  
372 003272 012737 000200 000006 MOV #IOTRD,ERRVEC ;RESTORE TRAP VECTOR  
373 003300 000137 005300 MOV #200,ERRVEC+2  
374 JMP REMAIN ;CHECK FOR MORE  
375 003304 012737 012150 000004 2$: MOV #IOTRD,ERRVEC ;UNITS  
376 003312 012737 000200 000006 MOV #200,ERRVEC+2 ;RESTORE TRAP VECTOR  
377 (4) :*****  
(4) :*TEST 3 FLOAT A 1 ACROSS THE MNCDO DATA REGISTER  
(3) 003320 000004 :*****  
(1) 003322 012737 000001 001124 TST3: SCOPE  
(1) 003330 012737 003336 001106 MOV #BIT0,$GDDAT ;LOAD EXPECTED BIT  
(1) 003336 013777 001124 176004 1$: MOV #1$,SLPADR ;LOAD LOOP ADDRESS  
(1) 003344 017737 176000 001126 MOV $GDDAT,@DOR ;LOAD MNCDO DATA REGISTER  
(1) 003352 023737 001124 001126 MOV @DOR,$BDDAT ;READ MNCDO DATA REGISTER  
(2) 003360 001401 CMP $GDDAT,$BDDAT ;COMPARE  
(1) 003362 104002 BEQ 2$ ;:BR IF EXPECTED  
(1) 003364 006337 001124 2$: ERROR 2 ; MNCDO DATA REGISTER FAILED TO HOLD A FLOATING 1  
(1) 003370 001362 ASL $GDDAT ;CHANGE THE DATA  
BNE 1$ ;BR IF MORE DATA
```

379
(4)
(4)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(1)
(1)
(1)
380
(4)
(4)
(3)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(3)
(1)
381
(4)
(4)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(1)
(1)

003372 000004
003374 012737 000001 001362
003402 012737 003410 001106
003410 013737 001362 001124
003416 005137 001124
003422 013777 001124 175720
003430 017737 175714 001126
003436 023737 001124 001126
003444 001401
003446 104002
003450 006337 001362
003454 001355

003456 000004
003460 012737 000040 001160
003466 012777 177777 175654
003474 005037 001124
003500 000005
003502 052777 000100 175434
003510 017737 175634 001126
003516 001401
003520 104002

003522 000004
003524 012737 003532 001106
003532 012777 177777 175610
003540 012737 000377 001124
003546 105077 175600
003552 017737 175572 001126
003560 023737 001124 001126
003566 001404
003570 104002
003572 012737 003600 001106
003600 012777 177777 175542
003606 012737 177400 001124
003614 105077 175530
003620 017737 175524 001126
003626 023737 001124 001126
003634 001401
003636 104002
003640

```
*****  
*TEST 4      FLOAT A 0 ACROSS THE MNCDO DATA 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      $GDDAT,@DOR     ;LOAD MNCDO DATA REGISTER  
      MOV      @DOR,$BDDAT    ;READ MNCDO DATA REGISTER  
      CMP      $GDDAT,$BDDAT  ;COMPARE  
      BEQ      2$             ;:BR IF EXPECTED  
      ERROR   2               ;MNCDO DATA REGISTER FAILED TO HOLD A FLOATING 0  
2$:    ASL      TEMP          ;CHANGE THE DATA  
      BNE     1$             ;BR IF MORE DATA  
*****  
*TEST 5      ENSURE THAT 'RESET' CLEARS THE MNCDO DATA REGISTER  
*****  
TST5:  SCOPE  
      MOV      #40,$TIMES     ;;DO 40 ITERATIONS  
      MOV      #-1,@DOR       ;LOAD BITS TO BE RESET  
      CLR      $GDDAT         ;CLEAR EXPECTED  
      RESET    ;CLEAR THE DEVICE  
      BIS      #BIT6,@$TKS    ;ENABLE TKB INTR.  
      MOV      @DOR,$BDDAT    ;READ MNCDO DATA REGISTER  
      BEQ      TST6          ;:BR IF CLEARED  
      ERROR   2               ;MNCDO DATA REGISTER FAILED TO CLEAR WITH 'RESET'  
*****  
*TEST 6      VERIFY BYTE OPERATION ON THE MNCDO DATA REGISTER  
*****  
TST6:  SCOPE  
      MOV      #1$, $LPADR     ;LOAD RETURN ADDRESS  
1$:    MOV      #-1,@DOR       ;LOAD MNCDO DATA REGISTER  
      MOV      #377,$GDDAT    ;LOAD EXPECTED  
      CLRB    @DOR1          ;CLEAR HIGH BYTE  
      MOV      @DOR,$BDDAT    ;READ MNCDO DATA REGISTER  
      CMP      $GDDAT,$BDDAT  ;COMPARE  
      BEQ      2$             ;:BR IF SAME  
      ERROR   2               ;CLEARING HIGH BYTE CHANGED LOW BYTE  
2$:    MOV      #2$, $LPADR     ;LOAD LOOP RETURN  
      MOV      #-1,@DOR       ;LOAD MNCDO DATA REGISTER  
      MOV      #177400,$GDDAT ;LOAD EXPECTED  
      CLRB    @DOR          ;CLEAR LOW BYTE  
      MOV      @DOR,$BDDAT    ;READ MNCDO DATA REGISTER  
      CMP      $GDDAT,$BDDAT  ;COMPARE  
      BEQ      3$             ;:BR IF SAME  
      ERROR   2               ;CLEARING LOW BYTE CHANGED HIGH BYTE  
3$:
```



```
383
(4)
(4)
(3) 003640 000004
(1) 003642 012737 000100 001124
(1) 003650 013777 001124 175466
(1) 003656 017737 175462 001126
(1) 003664 023737 001124 001126
(2) 003672 001401
(1) 003674 104003
(1)
(1) 003676 043777 001124 175440
(1) 003704 017737 175434 001126
(1) 003712 023737 001124 001126
(3) 003720 001001
(1) 003722 104003

::*****
:*TEST 7 TEST THAT BIT6 OF MNCDO STATUS REGISTER IS READ-WRITE
::*****
TST7: SCOPE
MOV #BIT6,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@OCSR ;LOAD BIT6 INTO MNCDO STATUS REGISTER
MOV @OCSR,$BDDAT ;READ MNCDO STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 3 ;BIT6 OF MNCDO STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@OCSR ;CLEAR THAT BIT
MOV @OCSR,$BDDAT ;READ MNCDO STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST10 ;;BR IF CLEARED
ERROR 3 ;BIT6 OF MNCDO STATUS REGISTER FAILED TO CLEAR

384
(4)
(4)
(3) 003724 000004
(1) 003726 012737 000020 001124
(1) 003734 013777 001124 175402
(1) 003742 017737 175376 001126
(1) 003750 023737 001124 001126
(2) 003756 001401
(1) 003760 104003
(1)
(1) 003762 043777 001124 175354
(1) 003770 017737 175350 001126
(1) 003776 023737 001124 001126
(3) 004004 001001
(1) 004006 104003

::*****
:*TEST 10 TEST THAT BIT4 OF MNCDO STATUS REGISTER IS READ-WRITE
::*****
TST10: SCOPE
MOV #BIT4,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@OCSR ;LOAD BIT4 INTO MNCDO STATUS REGISTER
MOV @OCSR,$BDDAT ;READ MNCDO STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 3 ;BIT4 OF MNCDO STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@OCSR ;CLEAR THAT BIT
MOV @OCSR,$BDDAT ;READ MNCDO STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST11 ;;BR IF CLEARED
ERROR 3 ;BIT4 OF MNCDO STATUS REGISTER FAILED TO CLEAR

385
(4)
(4)
(3) 004010 000004
(1) 004012 012737 000010 001124
(1) 004020 013777 001124 175316
(1) 004026 017737 175312 001126
(1) 004034 023737 001124 001126
(2) 004042 001401
(1) 004044 104003
(1)
(1) 004046 043777 001124 175270
(1) 004054 017737 175264 001126
(1) 004062 023737 001124 001126
(3) 004070 001001
(1) 004072 104003

::*****
:*TEST 11 TEST THAT BIT3 OF MNCDO STATUS REGISTER IS READ-WRITE
::*****
TST11: SCOPE
MOV #BIT3,$GDDAT ;LOAD EXPECTED
MOV $GDDAT,@OCSR ;LOAD BIT3 INTO MNCDO STATUS REGISTER
MOV @OCSR,$BDDAT ;READ MNCDO STATUS REGISTER
CMP $GDDAT,$BDDAT ;TEST THAT IT SET
BEQ 1$ ;;BR IF SET
ERROR 3 ;BIT3 OF MNCDO STATUS REGISTER FAILED TO SET

1$: BIC $GDDAT,@OCSR ;CLEAR THAT BIT
MOV @OCSR,$BDDAT ;READ MNCDO STATUS REGISTER AGAIN
CMP $GDDAT,$BDDAT ;TEST THE BIT
BNE TST12 ;;BR IF CLEARED
ERROR 3 ;BIT3 OF MNCDO STATUS REGISTER FAILED TO CLEAR
```

```

387
(4)
(4)
(3) 004074 000004
(2) 004076 012737 000040 001160
(1) 004104 012777 000130 175232
(1) 004112 005037 001124
(1) 004116 000005
(1) 004120 052777 000100 175016
(1) 004126 017737 175212 001126
(3) 004134 001401
(1) 004136 104003
388
389
(3)
(3)
(2) 004140 000004
390 004142 005077 175176
391 004146 012737 000200 001124
392 004154 105077 175170
393 004160 112777 000001 175160
394 004166 017737 175152 001126
395 004174 023737 001124 001126
396 004202 001401
397 004204 104003
398
(3)
(3)
(2) 004206 000004
399 004210 105077 175134
400 004214 112777 000001 175124
401 004222 005037 001124
402 004226 005077 175112
403 004232 017737 175106 001126
404 004240 001401
405 004242 104003
406
407
(3)
(3)
(2) 004244 000004
408 004246 105077 175076
409 004252 112777 000001 175066
410 004260 005037 001124
411 004264 105077 175060
412 004270 017737 175050 001126
413 004276 001401
414 004300 104003
415

```

```

:*****
:*TEST 12 ENSURE THAT 'RESET' CLEARS THE MNCDO STATUS REGISTER
:*****
TST12: SCOPE
MOV #40,$TIMES ;;DO 40 ITERATIONS
MOV #130,@OCSR ;LOAD BITS TO BE RESET
CLR $GDDAT ;CLEAR EXPECTED
RESET ;CLEAR THE DEVICE
BIS #BIT6,@$TKS ;ENABLE TKB INTR.
MOV @OCSR,$BDDAT ;READ MNCDO STATUS REGISTER
BEQ TST13 ;;BR IF CLEARED
ERROR 3 ;MNCDO STATUS REGISTER FAILED TO CLEAR WITH 'RESET'

:*****
:*TEST 13 VERIFY THAT MNCDO DONE FLAG SETS
:*****
TST13: SCOPE
CLR @OCSR ;CLEAR CLEARED FLAG
MOV #BIT7,$GDDAT ;LOAD EXPECTED
CLRB @DOR ;ENABLE
MOVB #BIT0,@OCSR1 ;GENERATE MAINT. REPLY
MOV @OCSR,$BDDAT ;READ OUTPUT STATUS REGISTER
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST14 ;;BR IF EXPECTED
ERROR 3 ;OUTPUT DONE FLAG FAILED TO SET

:*****
:*TEST 14 VERIFY THAT MNCDO DONE FLAG CLEARS WHEN WRITTEN TO A 0
:*****
TST14: SCOPE
CLRB @DOR ;ENABLE
MOVB #BIT0,@OCSR1 ;GENERATE MAINT. REPLY
CLR $GDDAT ;CLEAR EXPECTED
CLR @OCSR ;CLEAR OUTPUT DONE FLAG
MOV @OCSR,$BDDAT ;READ STATUS
BEQ TST15 ;;BR IF CLEARED
ERROR 3 ;WRITING OUTPUT FLAG TO A ZERO FAILED TO CLEAR 0

:*****
:*TEST 15 VERIFY THAT MNCDO DONE FLAG CLEARS WHEN OUTPUT DATA REGISTER IS WRITTEN
:*****
TST15: SCOPE
CLRB @DOR ;ENABLE
MOVB #BIT0,@OCSR1 ;GENERATE MAINT. REPLY
CLR $GDDAT ;CLEAR EXPECTED
CLRB @DOR ;WRITE THE OUTPUT DATA REGISTER
MOV @OCSR,$BDDAT ;READ OUTPUT STATUS REGISTER
BEQ TST16 ;;BR IF CLEARED
ERROR 3 ;OUTPUT DONE FLAG FAILED TO CLEAR
;WHEN OUTPUT DATA REGISTER WAS WRITTEN

```

```

417      ::*****
(3)      :*TEST 16      INTERRUPT TEST -- VERIFY MNCDO DOES INTERRUPT
(3)      :*****
(2) 004302 000004 TST16: SCOPE
418 004304 012737 004312 001106 MOV #64$, $LPADR
419 004312 012777 004376 175034 64$: MOV #1$, @DODINV ;LOAD RETURN VECTOR
420 004320 012777 000200 175030 MOV #200, @DODINS ;LOAD RETURN LEVEL
421 004326 105077 175016 CLR @DOR ;ENABLE
422 004332 112777 000001 175006 MOVB #BIT0, @OCSR1 ;GENERATE MAINT. REPLY
423 004340 005046 CLR -(SP)
424 004342 012746 004350 MOV #10$, -(SP)
425 004346 000002 RTI ;LSI-11 LOWER PRIORITY
426 004350 000240 10$: NOP
427 004352 052777 000100 174764 BIS #BIT6, @OCSR ;ENABLE INTR.
428 004360 000240 NOP
429 004362 000240 NOP
430 004364 000240 NOP
431 004366 005077 174752 CLR @OCSR ;DISABLE INTR.
432 004372 104004 ERROR 4 ;MNCDO FAILED TO INTERRUPT
433 004374 000425 BR 3$ ;;
434
435 004376 022626 1$: CMP (SP)+, (SP)+ ;CLEAR THE STACK
436 004400 012777 004440 174746 MOV #2$, @DODINV ;LOAD 2ND RETURN VECOR
437 004406 005077 174744 CLR @DODINS
438 004412 005046 CLR -(SP)
439 004414 012746 004422 MOV #11$, -(SP)
440 004420 000002 RTI ;LSI-11 LOWER PRIORITY
441 004422 000240 11$: NOP
442 004424 000240 NOP
443 004426 000240 NOP
444 004430 000240 NOP
445 004432 005077 174706 CLR @OCSR ;DISABLE INTR
446 004436 000404 BR 3$ ;;
447
448 004440 022626 2$: CMP (SP)+, (SP)+ ;CLEAN THE STACK
449 004442 005077 174676 CLR @OCSR ;DISABLE INTR
450 004446 104004 ERROR 4 ;CLEARING INTR. ENABLE FAILED TO REMOVE INTR. REQUEST
451 004450 005077 174670 3$: CLR @OCSR
452 004454 013777 001356 174672 MOV @DODINS, @DODINV ;RESET VECTORS
453 004462 012777 004700 174666 MOV #4700, @DODINS
454 004470 005077 174650 CLR @OCSR
455

```

```
457 (3) (3) (2) 004474 000004
458 004476 012746 000200
459 004502 012746 004510
460 004506 000002
461 004510 000240
462 004512 012777 004610 174634
463 004520 012777 000200 174630
464 004526 012777 000100 174610
465 004534 105077 174610
466 004540 152777 000001 174600
467 004546 000240
468 004550 000240
469 004552 042777 000100 174564
470 004560 000240
471 004562 000240
472 004564 005046
473 004566 012746 004574
474 004572 000002
475 004574 000240
476 004576 000240
477 004600 000240
478 004602 005077 174536
479 004606 000404
480
481 004610 022626
482 004612 005077 174526
483 004616 104004
484 004620 005077 174520
485 004624 013777 001356 174522
486 004632 012777 004700 174516
487 004640 005077 174500
488 004644 005046
489 004646 012746 004654
490 004652 000002
491 004654 000240

::*****
:*TEST 17 INTERRUPT TEST -- VERIFY THAT REMOVING OUTPUT IRQ ENABLE WILL REMOVE INT
:*****
TST17: SCOPE
MOV #200,-(SP)
MOV #10$,-(SP)
RTI ;LSI-11 RAISE PRIORITY
10$: NOP
MOV #1$,@DODINV ;LOAD INTR. VECTOR
MOV #200,@DODINS
MOV #BIT6,@OCSR ;ENABLE INTERRUPT
CLRB @DOR ;ENABLE
BISB #BIT0,@OCSR1 ;GENERATE MAINT. REPLY
NOP
NOP
BIC #BIT6,@OCSR ;DISABLE INTERRUPT
NOP
NOP
CLR -(SP)
MOV #11$,-(SP)
RTI ;LSI-11 LOWER PRIORITY
11$: NOP
NOP
NOP
CLR @OCSR ;CLEAR STATUS REGISTER
BR 2$ ::
1$: CMP (SP)+,(SP)+ ;CLEAN THE STACK
CLR @OCSR ;CLEAR OUTPUT STATUS
ERROR 4 ;CLEARING INTERRUPT ENABLE FAILED TO REMOVE INTERRUPT RE
2$: CLR @OCSR
MOV DODINS,@DODINV ;RESET VECTORS
MOV #4700,@DODINS
CLR @OCSR
CLR -(SP)
MOV #3$,-(SP) ;LOWER PRIORITY
3$: RTI
NOP
```

493
 494
 (3)
 (3)
 (2) 004656 000004
 495 004660 022737 000002 001360
 496 004666 001015
 497 004670 012737 000200 001124
 498 004676 005077 174446
 499 004702 017737 174436 001126
 500 004710 023737 001124 001126
 501 004716 001401
 502 004720 104003
 503
 (3)
 (3)
 (2) 004722 000004
 (1) 004724 012737 000001 001160
 504 004732 022737 000002 001360
 505 004740 001022
 506 004742 005737 001176
 507 004746 001417
 508 004750 012737 000001 001124
 509 004756 012737 100000 001362
 510 004764 013777 001124 174356
 511 004772 005337 001362
 512 004776 001375
 513 005000 006337 001124
 514 005004 001364

```

*****
*TEST 20      VERIFY OUTPUT STROBE GENERATES REPLY      -      MNCDO FIELD TEST MODULE
*****
TST20:  SCOPE
        CMP      #2,DWARF          ;;TEST IF MNCDO FIELD DWARF MODE
        BNE      TST21            ;;BR IF NOT
        MOV      #BIT7,$GDDAT      ;:LOAD EXPECTED
        CLR      @DOR              ;:CLEAR OUTPUT AND GENERATE STROBE
        MOV      @OCSR,$BDDAT      ;:READ OUTPUT STATUS REG.
        CMP      $GDDAT,$BDDAT     ;:COMPARE
        BEQ      TST21            ;;BR IF SAME
        ERROR    3                 ;:MNCDO STROBE FAILED TO GENERATE REPLY
*****
*TEST 21      FLOAT A 1 ACROSS THE OUTPUT DATA REGISTER (FIELD DRARF)
*****
TST21:  SCOPE
        MOV      #1,$TIMES         ;;DO 1 ITERATION
        CMP      #2,DWARF          ;:TEST IF FIELD MODE
        BNE      TST22            ;;BR IF NOT FIELD
        TST      $PASS             ;:TEST IF FIRST PASS
        BEQ      TST22            ;;BR IF YES
        MOV      #BIT0,$GDDAT      ;:LOAD VALUE
        1$:     MOV      #BIT15,TEMP ;:LOAD COUNTER
        MOV      $GDDAT,@DOR       ;:LOAD OUTPUT REGISTER
        2$:     DEC      TEMP        ;:DELAY MORE
        BNE      2$
        ASL      $GDDAT           ;:CHANGE THE DATA
        BNE      1$              ;:BR IF NOT DONE

```

```

516 (3) *****
(3) *TEST 22 VERIFY OUTPUT STROBE IS GENERATED (IN-HOUSE TESTER TEST)
(2) *****
(2) 005006 000004 TST22: SCOPE
517 005010 022737 000001 001360 CMP #1,DWARF ;TEST IF "IN-HOUSE DWARF" MODE
518 005016 001071 BNE TST23 ;:BR IF NOT
519 005020 012777 000004 174342 MOV #BIT2,@TSTR2 ;ENABLE TESTER
520 005026 005077 174334 CLR @TSTR0 ;CLEAR TESTER STATUS
521 005032 005077 174312 CLR @DOR ;CLEAR OUTPUT DATA <GEN. OUTPUT STROBE>
522 005036 005077 174302 CLR @OCSR ;CLEAR OUTPUT STATUS
523 005042 017737 174320 001126 MOV @TSTR0,$BDDAT ;READ TESTER STATUS
524 005050 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD EXPECTED VALUE
525 005056 123737 001124 001126 CMPB $GDDAT,$BDDAT ;COMPARE THEM
526 005064 001401 BEQ 1$ ;:BR IF SAME
527 005066 104003 ERROR 3 ;NO MNCDO STROBE FLAG
528 005070 017737 174250 001126 1$: MOV @OCSR,$BDDAT ;READ OUTPUT STATUS
529 005076 012737 000000 001124 MOV #0,$GDDAT ;LOAD EXPECTED
530 005104 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMAPRE THE STATUS
531 005112 001401 BEQ 2$ ;:BR IF SAME
532 005114 104003 ERROR 3 ;UNEXPECTED OUTPUT STATUS FLAG
533 005116 017700 174250 2$: MOV @TSTR4,R0 ;READ TESTER INPUT DATA
534 ;CAUSING A REPLY TO BE SENT TO OUTPUT
535 005122 017737 174216 001126 MOV @OCSR,$BDDAT ;READ STATUS
536 005130 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD EXPECTED OUTPUT STATUS
537 005136 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMAPRE THEM
538 005144 001401 BEQ 3$ ;:BR IF SAME
539 005146 104003 ERROR 3 ;REPLY FROM TESTER FAILED TO SET OUTPUT READY FLOP
540 005150 005077 174170 3$: CLR @OCSR ;CLEAR OUTPUT STATUS
541 005154 012737 000000 001124 MOV #0,$GDDAT ;LOAD EXPECTED
542 005162 017737 174156 001126 MOV @OCSR,$BDDAT ;READ ACTUAL OUTPUT STATUS
543 005170 023737 001124 001126 CMP $GDDAT,$BDDAT
544 005176 001401 BEQ TST23 ;:BR IF SAME
545 005200 104003 ERROR 3 ;OUTPUT STATUS FAILED TO CLEAR

```

```

546 (3) *****
547 (3) *TEST 23 VERIFY OUTPUT DATA REGISTER (IN-HOUSE TESTER TEST)
(2) *****
(2) 005202 000004 TST23: SCOPE
548 005204 022737 000001 001360 CMP #1,DWARF ;TEST IF "IN-HOUSE DWARF MODE"
549 005212 001031 BNE 3$ ;:BR IF NOT
550 005214 012777 000004 174146 MOV #BIT2,@TSTR2 ;ENSURE TESTER MODE
551 005222 012737 005236 001106 MOV #1$,$LPADR ;SET RETURN ADDRESS
552 005230 012737 000001 001124 MOV #BIT0,$GDDAT ;LOAD EXPECTED
553 005236 013777 001124 174104 1$: MOV $GDDAT,@DOR ;LOAD THE OUTPUT DATA REGISTER
554 005244 017737 174122 001126 MOV @TSTR4,$BDDAT ;READ THE TESTER INPUT REGISTER
555 005252 000240 NOP
556 005254 000240 NOP
557 005256 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE DATA
558 005264 001401 BEQ 2$ ;:BR IF SAME
559 005266 104002 ERROR 2 ;OUTPUT DATA REGISTER ERROR
560 005270 006337 001124 2$: ASL $GDDAT ;CHANGE THE EXPECTED DATA
561 005274 001360 BNE 1$
562 005276 000240 3$: NOP
563

```

CVMNE-A MNCDO
CVMNEA.P11

DIAGNOSTIC
T23

MACY11 27(654) 19-SEP-78 09:02 PAGE 12
VERIFY OUTPUT DATA REGISTER (IN-HOUSE TESTER TEST)

SEQ 0030

565 005300
566
(3)
(3)
(2) 005300 000004
(1) 005302 012737 000001 001160
567 005310 005237 001202
568 005314 123737 001202 001442
569 005322 001440
570 005324 063737 001400 001344
571 005332 063737 001400 001346
572 005340 063737 001400 001350
573 005346 063737 001400 001352
574 005354 006337 001376
575 005360 004737 007652
576 005364 013700 001446
577 005370 006300
578 005372 016037 001402 001354
579 005400 013737 001354 001356
580 005406 062737 000002 001356
581 005414 005037 001102
582 005420 000137 003164
583 005424 000240
584

REMAIN:

::*****
:*TEST 24 DETERMINE IF MORE MNCDO'S REMAIN TO BE TESTED
:*****

TST24: SCOPE
MOV #1,\$TIMES ;:DO 1 ITERATION
INC \$UNIT ;:UPDATE UNIT #
CMPB \$UNIT,EVER ;:TEST IF MORE
BEQ 1\$;:BR IF NOT
ADD VADDR,OCSR
ADD VADDR,OCSR1
ADD VADDR,DOR
ADD VADDR,DOR1
ASL MASKNM ;:CHANGE ERROR FLAG BIT
JSR PC,WHICHU ;:DETERMINE UNIT #
MOV UNITBD,RO ;:GET IT
ASL RO ;:MAKE WORD VALUE
MOV VECLST(RO),DODINV ;:GET THIS UNITS VECTOR VALUE
MOV DODINV,DODINS ;:COPY IT AND
ADD #2,DODINS ;:MAKE NEXT ADDRESS
CLR \$TSTNM ;:CLR TST NUMBER
JMP TST1 ;:TEST NEXT UNIT
1\$: NOP

```

586          .SBTTL  END OF PASS ROUTINE
(1)
(2)          ::*****
(1)          :*INCREMENT THE PASS NUMBER ($PASS)
(1)          :*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
(1)          :*IF THERES A MONITOR GO TO IT
(1)          :*IF THERE ISN'T JUMP TO EXTMSG
(1)
(1) 005426   $EOP:
(1) 005426   000004   SCOPE
(1) 005430   005037   001102   CLR   $STNM   ;;ZERO THE TEST NUMBER
(1) 005434   005037   001160   CLR   $TIMES  ;;ZERO THE NUMBER OF ITERATIONS
(1) 005440   005237   001176   INC   $PASS   ;;INCREMENT THE PASS NUMBER
(1) 005444   042737   100000   001176   BIC   #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
(1) 005452   005327   DEC   (PC)+  ;;LOOP?
(1) 005454   000001   $EOPCT: .WORD 1
(1) 005456   003022   BGT   $DOAGN  ;;YES
(1) 005460   012737   MOV   (PC)+,@(PC)+ ;;RESTORE COUNTER
(1) 005462   000001   $ENDCT: .WORD 1
(1) 005464   005454   $EOPCT
(1) 005466   104401   005533   TYPE  , $ENDMG  ;;TYPE 'END PASS #'
(2) 005472   013746   001176   MOV   $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
(2) 005476   104405   TYPDS  ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 005500   104401   005530   TYPE  , $ENULL  ;;TYPE A NULL CHARACTER
(1) 005504   013700   000042   $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
(1) 005510   001405   BEQ   $DOAGN  ;;BRANCH IF NO MONITOR
(1) 005512   000005   RESET  ;;CLEAR THE WORLD
(1) 005514   004710   $ENDAD: JSR   PC,(R0) ;;GO TO MONITOR
(1) 005516   000240   NOP    ;;SAVE ROOM
(1) 005520   000240   NOP    ;;FOR
(1) 005522   000240   NOP    ;;ACT11
(1) 005524   $DOAGN:
(1) 005524   000137   JMP   @(PC)+  ;;RETURN
(1) 005526   005550   $RTNAD: .WORD  EXTMSG
(1) 005530   377      377      000   $ENULL: .BYTE  -1,-1,0  ;;NULL CHARACTER STRING
(1) 005533   015      042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
(1) 005540   050040  051501  020123
(1) 005546   000043
587
588 005550   052777  000100  173366  EXTMSG: BIS   #BIT6,@$TKS  ;;ENABLE TKB INTR.
589 005556   005737  001112   TST   $ERTTL  ;;ANY ERRORS
590 005562   001416   BEQ   1$     ;;BR IF NONE
591 005564   104401  013650   TYPE  ,ERRTOT  ;;TYPE TOTAL ERROR MESSAGE
592 005570   013746  001112   MOV   $ERTTL,-(SP) ;;TOTAL ERRORS
593 005574   104405   TYPDS
594 005576   022737  000001  001376   CMP   #1,MASKNM ;;TEST IF MORE UNITS
595 005604   001405   BEQ   1$     ;;BR IF NONE
596 005606   104401  013677   TYPE  ,MESGD  ;;TYPE BAD UNITS MESSAGE
597 005612   013746  001444   MOV   BADUNT,-(SP) ;;GET BAD UNITS
598 005616   104406   TYPBN  ;;TYPE IT IN BINARY
599 005620   104401  005530   1$:   TYPE  , $ENULL  ;;ENSURE ALL DATA GETS OUTPUTED
600 005624   004737  005760   JSR   PC,CTRLCG ;;TEST FOR CONTROL C/G
601 005630   000137  002424   JMP   LOGIC   ;;TEST AGAIN

```



```

607 005634 012706 001100          HIGHLO: MOV      #STACK,SP          ;LOAD STACK POINTER
608 005640 004737 002712          JSR      PC,FIXADR          ;FIX ALL OTHER ADRS.
609 005644 012700 000400          1$:  MOV      #BIT8,RO          ;LOAD COUNTER
610 005650 105077 173476          2$:  CLRB    @DOR1          ;CLEAR HIGH BYTE
611 005654 005300                   DEC      RO                  ;DELAY
612 005656 001374                   BNE     2$
613 005660 012700 000400          3$:  MOV      #BIT8,RO          ;LOAD COUNTER
614 005664 105077 173460          CLRB    @DOR                ;CLEAR LOW BYTE
615 005670 005300                   DEC      RO                  ;DELAY
616 005672 001374                   BNE     3$
617 005674 004737 005760          JSR      PC,CTRLCG          ;TEST FOR CTRL C/G
618 005700 000761                   BR      1$
619                                     .SBTTL MNCDO TEST MODULE LAMP LOOP
620 005702 012706 001100          DOLAMP: MOV     #STACK,SP          ;LOAD STACK POINTER
621 005706 004737 002712          JSR      PC,FIXADR          ;FIX ALL OTHER ADRS.
622 005712 012737 000001 001362  1$:  MOV      #BIT0,TEMP          ;LOAD INITIAL VALUE
623 005720 012700 000002          2$:  MOV      #2,RO            ;LOAD DELAY LOOP
624 005724 005001                   CLR     R1
625 005726 013777 001362 173414  MOV     TEMP,@DOR            ;LOAD OUTPUT REGISTER LAMPS
626 005734 005301                   3$:  DEC     R1                ;DELAY
627 005736 001376                   BNE     3$
628 005740 005300                   DEC     RO
629 005742 100374                   BPL     3$
630 005744 004737 005760          JSR      PC,CTRLCG
631 005750 006337 001362          ASL     TEMP                ;CHANGE THE OUTPUT DATA
632 005754 001361                   BNE     2$
633 005756 000755                   BR      1$
634                                     ;
635                                     .SBTTL CONTROL C/G HANDLER
636
637 005760 105777 173160          CTRLCG: TSTB   @$TKS          ;TEST FOR FLAG
638 005764 100022                   BPL     2$
639 005766 017737 173154 006034  MOV     @$TKB,CTRCHA        ;GET CHAR
640 005774 042737 177640 006034  BIC     #177640,CTRCHA      ;MASK OFF BITS
641 006002 022737 000003 006034  CMP     #3,CTRCHA           ;TEST FOR CTRL C
642 006010 001003                   BNE     1$                   ;BR IF NOT
643 006012 005726                   TST    (SP)+                ;CLEAN STACK
644 006014 000137 002200                   JMP     MTEST1              ;AND REYTP E DOT
645 006020 022737 000007 006034  1$:  CMP     #7,CTRCHA           ;TEST FOR CTRL G
646 006026 001001                   BNE     2$                   ;BR IF NOT
647 006030 104407                   GTSWR                    ;GET SWITCHES
648 006032 000207                   2$:  RTS     PC                ;EXIT
649 006034 000000          CTRLCHA: 0          ;CHAR. THE OPER TYPED DURING RUNNING

```

```

651          .SBTTL  TTY INPUT ROUTINE
(1)
(2)          ::*****
(1)          .ENABL  LSB
(1) 006036 000000 $TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
(1) 006040 000000 $TKQIN: .WORD 0          ;;INPUT POINTER
(1) 006042 000000 $TKQOUT: .WORD 0         ;;OUTPUT POINTER
(1) 006044 000040 $TKQSRV: .BLKB 32.        ;;TTY KEYBOARD QUEUE
(1)          $TKQEND=.
(1)
(1)          ;*TK INITIALIZE ROUTINE
(1)          ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
(1)          ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
(1)          ;
(1)          ;*CALL:
(1)          ;*      JSR      PC,$TKINT
(1)          ;*      RETURN
(1)          ;
(1) 006104 005037 006036 $TKINT: CLR $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
(1) 006110 012737 006044 006040 MOV # $TKQSRV,$TKQIN      ;;MOVE THE STARTING ADDRESS OF THE
(1) 006116 013737 006040 006042 MOV $TKQIN,$TKQOUT      ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
(1) 006124 012737 006154 000060 MOV # $TKSRV,@TKVEC    ;;INITIALIZE THE KEYBOARD VECTOR
(1) 006132 012737 000200 000062 MOV #200,@TKVEC+2      ;;'BR' LEVEL 4
(1) 006140 005777 173002 TST @TKB          ;;CLEAR DONE FLAG
(1) 006144 012777 000100 172772 MOV #100,@TKS          ;;ENABLE TTY KEYBOARD INTERRUPT
(1) 006152 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) 006154 117746 172766 $TKSRV: MOVB @TKB,-(SP)      ;;PICKUP THE CHARACTER
(1) 006160 042716 177600 BIC #^C177,(SP)      ;;STRIP THE JUNK
(1) 006164 021627 000003 CMP (SP),#3          ;;IS IT A CONTROL C?
(1) 006170 001007 BNE 1$          ;;BRANCH IF NO
(1) 006172 104401 007324 TYPE ,SCNTLC          ;;TYPE A CONTROL-C (^C)
(1) 006176 004737 006104 JSR PC,$TKINT          ;;INIT THE KEYBOARD
(1) 006202 005726 TST (SP)+          ;;CLEAN UP STACK
(1) 006204 000137 002200 JMP MTEST1          ;;CONTROL C RESTART
(1) 006210 021627 000007 1$: CMP (SP),#7          ;;IS IT A CONTROL G?
(1) 006214 001004 BNE 2$          ;;BRANCH IF NO
(1) 006216 022737 000176 001140 CMP #SWREG,SWR          ;;IS SOFT-SWR SELECTED?
(1) 006224 001500 BEQ 6$          ;;GO TO SWR CHANGE
(1)
(1)          2$:
(1) 006226 022737 000040 006036 CMP #32.,$TKCNT      ;;IS THE QUEUE FULL?
(1) 006234 001004 BNE 3$          ;;BRANCH IF NO
(1) 006236 104401 007320 TYPE ,SBELL          ;;RING THE TTY BELL
(1) 006242 005726 TST (SP)+          ;;CLEAN CHARACTER OFF OF STACK
(1) 006244 000451 BR 5$          ;;EXIT
(1) 006246 021627 000023 3$: CMP (SP),#23          ;;IS IT A CONTROL-S?

```

```

(1) 006252 001021 BNE 32$ ;:BRANCH IF NO
(1) 006254 005077 172664 CLR @STKS ;:DISABLE TTY KEYBOARD INTERRUPTS
(1) 006260 005726 TST (SP)+ ;:CLEAN CHAR OFF STACK
(1) 006262 105777 172656 31$: TSTB @STKS ;:WAIT FOR A CHAR
(1) 006266 100375 BPL 31$ ;:LOOP UNTIL ITS THERE
(1) 006270 117746 172652 MOVB @STKB, -(SP) ;:GET THE CHARACTER
(1) 006274 042716 177600 BIC #^C177, (SP) ;:MAKE IT 7-BIT ASCII
(1) 006300 022627 000021 CMP (SP)+, #21 ;:IS IT A CONTROL-Q?
(1) 006304 001366 BNE 31$ ;:BRANCH IF NO
(1) 006306 012777 000100 172630 MOV #100, @STKS ;:REENABLE TTY KEYBOARD INTERRUPTS
(1) 006314 000002 RTI ;:RETURN
(1) 006316 005237 006036 32$: INC $TKCNT ;:COUNT THIS CHARACTER
(1) 006322 021627 000140 CMP (SP), #140 ;:IS IT UPPER CASE?
(1) 006326 002405 BLT 4$ ;:BRANCH IF YES
(1) 006330 021627 000175 CMP (SP), #175 ;:IS IT A SPECIAL CHAR?
(1) 006334 003002 BGT 4$ ;:BRANCH IF YES
(1) 006336 042716 000040 BIC #40, (SP) ;:MAKE IT UPPER CASE
(1) 006342 112677 177472 4$: MOVB (SP)+, @STKQIN ;:AND PUT IT IN QUEUE
(1) 006346 005237 006040 INC $TKQIN ;:UPDATE THE POINTER
(1) 006352 023727 006040 006104 CMP $TKQIN, # $TKQEND ;:GO OFF THE END?
(1) 006360 001003 BNE 5$ ;:BRANCH IF NO
(1) 006362 012737 006044 006040 MOV # $TKQSR, $TKQIN ;:RESET THE POINTER
(1) 006370 000002 5$: RTI ;:RETURN
(2)
(1) ;:*****
(1) ;:*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ;:*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ;:*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
(1) ;:*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
(1) 006372 022737 000176 001140 $CKSWR: CMP # $SWREG, SWR ;:IS THE SOFT-SWR SELECTED
(1) 006400 001124 BNE 15$ ;:EXIT IF NOT
(1) 006402 105777 172536 TSTB @STKS ;:IS A CHAR WAITING?
(1) 006406 100121 BPL 15$ ;:IF NOT, EXIT
(1) 006410 117746 172532 MOVB @STKB, -(SP) ;:YES
(1) 006414 042716 177600 BIC #^C177, (SP) ;:MAKE IT 7-BIT ASCII
(1) 006420 021627 000007 CMP (SP), #7 ;:IS IT A CONTROL-G?
(1) 006424 001300 BNE 2$ ;:IF NOT, PUT IT IN THE TTY QUEUE
(1) ;:AND EXIT
(2)
(1) ;:*****
(1) ;:*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
(1) ;:*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
(1) ;:*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
(1) 006426 123727 001134 000001 6$: CMPB $AUTOB, #1 ;:ARE WE RUNNING IN AUTO-MODE?
(1) 006434 001674 BEQ 2$ ;:BRANCH IF YES
(1) 006436 005726 TST (SP)+ ;:CLEAR CONTROL-G OFF STACK
(1) 006440 004737 006104 JSR PC, $TKINT ;:FLUSH THE TTY INPUT QUEUE
(1) 006444 005077 172474 CLR @STKS ;:DISABLE TTY KEYBOARD INTERRUPTS
(1) 006450 112737 000001 001135 MOVB #1, $INTAG ;:SET INTERRUPT MODE INDICATOR
(1)
(1) 006456 104401 007336 $GTSWR: TYPE , $CNTLG ;:ECHO THE CONTROL-G (^G)
(1) 006462 104401 007343 TYPE , $MSWR ;:TYPE CURRENT CONTENTS
(2) 006466 013746 000176 MOV $WREG, -(SP) ;:SAVE SWREG FOR TYPEOUT
(2) 006472 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)

```

(1)	006474	104401	007354		TYPE	,\$MNEW	::PROMPT FOR NEW SWR
(1)	006500	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
(1)	006502	005046			CLR	-(SP)	::THE NEW SWR
(1)	006504	105777	172434	7\$:	TSTB	@\$TKS	::CHAR THERE?
(1)	006510	100375			BPL	7\$::IF NOT TRY AGAIN
(1)	006512	117746	172430		MOVB	@\$TKB, -(SP)	::PICK UP CHAR
(1)	006516	042716	177600		BIC	#^C177, (SP)	::MAKE IT 7-BIT ASCII
(1)	006522	021627	000003		CMP	(SP), #3	::IS IT A CONTROL-C?
(1)	006526	001015			BNE	9\$::BRANCH IF NOT
(1)	006530	104401	007324		TYPE	,\$CNTLC	::YES, ECHO CONTROL-C (^C)
(1)	006534	062706	000006		ADD	#6, SP	::CLEAN UP STACK
(1)	006540	123727	001135	000001	CMPB	\$INTAG, #1	::REENABLE TTY KEYBOARD INTERRUPTS?
(1)	006546	001003			BNE	8\$::BRANCH IF NO
(1)	006550	012777	000100	172366	MOV	#100, @\$TKS	::ALLOW TTY KEYBOARD INTERRUPTS
(1)	006556	000137	002200	8\$:	JMP	MTEST1	::CONTROL-C RESTART
(1)	006562	021627	000025	9\$:	CMP	(SP), #25	::IS IT A CONTROL-U?
(1)	006566	001005			BNE	10\$::BRANCH IF NOT
(1)	006570	104401	007331		TYPE	,\$CNTLU	::YES, ECHO CONTROL-U (^U)
(1)	006574	062706	000006	20\$:	ADD	#6, SP	::IGNORE PREVIOUS INPUT
(1)	006600	000737			BR	19\$::LET'S TRY IT AGAIN
(1)	006602	021627	000015	10\$:	CMP	(SP), #15	::IS IT A <CR>?
(1)	006606	001022			BNE	16\$::BRANCH IF NO
(1)	006610	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
(1)	006614	001403			BEQ	11\$::BRANCH IF YES
(1)	006616	016677	000002	172314	MOV	2(SP), @SWR	::SAVE NEW SWR
(1)	006624	062706	000006	11\$:	ADD	#6, SP	::CLEAR UP STACK
(1)	006630	104401	001165	14\$:	TYPE	,\$CRLF	::ECHO <CR> AND <LF>
(1)	006634	123727	001135	000001	CMPB	\$INTAG, #1	::RE-ENABLE TTY KBD INTERRUPTS?
(1)	006642	001003			BNE	15\$::BRANCH IF NOT
(1)	006644	012777	000100	172272	MOV	#100, @\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
(1)	006652	000002		15\$:	RTI		::RETURN
(1)	006654	004737	011740	16\$:	JSR	PC, \$TYPEC	::ECHO CHAR
(1)	006660	021627	000060		CMP	(SP), #60	::CHAR < 0?
(1)	006664	002420			BLT	18\$::BRANCH IF YES
(1)	006666	021627	000067		CMP	(SP), #67	::CHAR > 7?
(1)	006672	003015			BGT	18\$::BRANCH IF YES
(1)	006674	042726	000060		BIC	#60, (SP)+	::STRIP-OFF ASCII
(1)	006700	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
(1)	006704	001403			BEQ	17\$::BRANCH IF YES
(1)	006706	006316			ASL	(SP)	::NO, SHIFT PRESENT
(1)	006710	006316			ASL	(SP)	::CHAR OVER TO MAKE
(1)	006712	006316			ASL	(SP)	::ROOM FOR NEW ONE.
(1)	006714	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
(1)	006720	056616	177776		BIS	-2(SP), (SP)	::SET IN NEW CHAR
(1)	006724	000667			BR	7\$::GET THE NEXT ONE
(1)	006726	104401	001164	18\$:	TYPE	,\$QUES	::TYPE ?<CR><LF>
(1)	006732	000720			BR	20\$::SIMULATE CONTROL-U
(1)					.DSABL	LSB	


```

(1) 007122 001406          BEQ      7$          ;;BR IF NO
(1) 007124 112737 000134 007256  MOVB    #'\.9$      ;;TYPE A BACK SLASH
(1) 007132 104401 007256          TYPE    ,9$
(1) 007136 005016          CLR      (SP)        ;;CLEAR THE RUBOUT KEY
(1) 007140 122713 000025 7$:    CMPB    #25,(R3)    ;;IS CHARACTER A CTRL U?
(1) 007144 001003          BNE     8$          ;;BR IF NO
(1) 007146 104401 007331          TYPE    ,SCNTLU     ;;TYPE A CONTROL 'U'
(1) 007152 000726          BR      1$          ;;GO START OVER
(1) 007154 122713 000022 8$:    CMPB    #22,(R3)    ;;IS CHARACTER A '^R'?
(1) 007160 001011          BNE     3$          ;;BRANCH IF NO
(1) 007162 105013          CLRB   (R3)         ;;CLEAR THE CHARACTER
(1) 007164 104401 001165          TYPE    ,SCRLF     ;;TYPE A 'CR' & 'LF'
(1) 007170 104401 007260          TYPE    ,STTYIN    ;;TYPE THE INPUT STRING
(1) 007174 000717          BR      2$          ;;GO PICKUP ANOTHER CHACTER
(1) 007176 104401 001164 4$:    TYPE    ,SQUES     ;;TYPE A '?'
(1) 007202 000712          BR      1$          ;;CLEAR THE BUFFER AND LOOP
(1) 007204 111337 007256 3$:    MOVB    (R3),9$     ;;ECHO THE CHARACTER
(1) 007210 104401 007256          TYPE    ,9$
(1) 007214 122723 000015          CMPB    #15,(R3)+  ;;CHECK FOR RETURN
(1) 007220 001305          BNE     2$          ;;LOOP IF NOT RETURN
(1) 007222 105063 177777          CLRB   -1(R3)      ;;CLEAR RETURN (THE 15)
(1) 007226 104401 001166          TYPE    ,SLF       ;;TYPE A LINE FEED
(1) 007232 005726          TST    (SP)+       ;;CLEAN RUBOUT KEY FROM THE STACK
(1) 007234 012603          MOV     (SP)+,R3   ;;RESTORE R3
(1) 007236 011646          MOV     (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 007240 016666 000004 000002  MOV     4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 007246 012766 007260 000004  MOV     #$TTYIN,4(SP)
(1) 007254 000002          RTI              ;;RETURN
(1) 007256 000          9$:    .BYTE   0          ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 007257 000          .BYTE   0          ;;TERMINATOR
(1) 007260 000040          $TTYIN: .BLKB  32.  ;;RESERVE 32. BYTES FOR TTY INPUT
(1) 007320 177607 000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
(1) 007324 041536 005015 000          $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL 'C'
(1) 007331 136 006525 000012          $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
(1) 007336 043536 005015 000          $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
(1) 007343 015 051412 051127          $MSWR:  .ASCIZ  <15><12>/SWR = /
(1) 007350 036440 000040          $MNEW:  .ASCIZ  / NEW = /
(1) 007354 020040 042516 020127          $MNEW:  .ASCIZ  / NEW = /
(1) 007362 020075 000          .EVEN
007366          .SBTTL SCOPE HANDLER ROUTINE

```

```

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

```

```

(1) 007366          $SCOPE:
(1) 007366 104410          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(2) 007370 004737 005760          JSR          PC,CTRLCG
(1) 007374 032777 040000 171536 1$: BIT          #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
(1) 007402 001114          BNE          $OVER          ;;YES IF SW14=1
(1) 007404 000416          :#####START OF CODE FOR THE XOR TESTER#####
(1) 007404 000416          $XTSTR: BR          6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1) 007406 013746 000004          MOV          @WERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 007412 012737 007432 000004          MOV          #5$,@WERRVEC          ;;SET FOR TIMEOUT
(1) 007420 005737 177060          TST          @#177060          ;;TIME OUT ON XOR?
(1) 007424 012637 000004          MOV          (SP)+,@WERRVEC          ;;RESTORE THE ERROR VECTOR
(1) 007430 000463          BR          $SVLAD          ;;GO TO THE NEXT TEST
(1) 007432 022626          5$: CMP          (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
(1) 007434 012637 000004          MOV          (SP)+,@WERRVEC          ;;RESTORE THE ERROR VECTOR
(1) 007440 000423          BR          7$          ;;LOOP ON THE PRESENT TEST
(1) 007442          6$:;#####END OF CODE FOR THE XOR TESTER#####
(1) 007442 032777 000400 171470          BIT          #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
(1) 007450 001404          BEQ          2$          ;;BR IF NO
(1) 007452 127737 171462 001102          CMPB         @SWR,$STSTM          ;;ON THE RIGHT TEST? SWR<7:0>
(1) 007460 001465          BEQ          $OVER          ;;BR IF YES
(1) 007462 105737 001103          2$: TSTB         $ERFLG          ;;HAS AN ERROR OCCURRED?
(1) 007466 001421          BEQ          3$          ;;BR IF NO
(1) 007470 123737 001115 001103          CMPB         $ERMAX,$ERFLG          ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 007476 101015          BHI          3$          ;;BR IF NO
(1) 007500 032777 001000 171432          BIT          #BIT09,@SWR          ;;LOOP ON ERROR?
(1) 007506 001404          BEQ          4$          ;;BR IF NO
(1) 007510 013737 001110 001106          7$: MOV          $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 007516 000446          BR          $OVER
(1) 007520 105037 001103          4$: CLRB         $ERFLG          ;;ZERO THE ERROR FLAG
(1) 007524 005037 001160          CLR          $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 007530 000415          BR          1$          ;;ESCAPE TO THE NEXT TEST
(1) 007532 032777 004000 171400          3$: BIT          #BIT11,@SWR          ;;INHIBIT ITERATIONS?
(1) 007540 001011          BNE          1$          ;;BR IF YES
(1) 007542 005737 001176          TST          $PASS          ;;IF FIRST PASS OF PROGRAM
(1) 007546 001406          BEQ          1$          ;;INHIBIT ITERATIONS
(1) 007550 005237 001104          INC          $ICNT          ;;INCREMENT ITERATION COUNT
(1) 007554 023737 001160 001104          CMP          $TIMES,$ICNT          ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 007562 002024          BGE          $OVER          ;;BR IF MORE ITERATION REQUIRED
(1) 007564 012737 000001 001104          1$: MOV          #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
(1) 007572 013737 007650 001160          MOV          $MXCNT,$TIMES          ;;SET NUMBER OF ITERATIONS TO DO
(1) 007600 105237 001102          $SVLAD: INCB         $STSTM          ;;COUNT TEST NUMBERS
(1) 007604 113737 001102 001174          MOVB        $STSTM,$TESTN          ;;SET TEST NUMBER IN APT MAILBOX
(1) 007612 011637 001106          MOV          (SP),$LPADR          ;;SAVE SCOPE LOOP ADDRESS
(1) 007616 011637 001110          MOV          (SP),$LPERR          ;;SAVE ERROR LOOP ADDRESS
(1) 007622 005037 001162          CLR          $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 007626 112737 000001 001115          MOVB        #1,$ERMAX          ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 007634 013777 001102 171300          $OVER: MOV          $STNM,@DISPLAY          ;;DISPLAY TEST NUMBER
(1) 007642 013716 001106          MOV          $LPADR,(SP)          ;;FUDGE RETURN ADDRESS
(1) 007646 000002          RTI          ;;FIXES PS
(1) 007650 003720          $MXCNT: 2000.          ;;MAX. NUMBER OF ITERATIONS

```

```

654
660 ;SUBROUTINE TO CONVERT THE UNIT MASK INTO UNIT #
661 007652 013737 001376 007702 WHICHU: MOV MASKNM,11$ ;GET CURRNET UNIT
662 007660 012737 000000 001446 MOV #0,UNITBD ;PRIME THE UNIT
663 007666 006237 007702 10$: ASR 11$ ;SHIFT
664 007672 001404 BEQ 12$ ;BR IF DONE
665 007674 005237 001446 INC UNITBD ;UPDATE BAD TYPEOUT VALUE
666 007700 000772 BR 10$
667 007702 000000 11$: 0
668 007704 000207 12$: RTS PC
669 .SBTTL ERROR HANDLER ROUTINE
(1)
(2)
(1) ::*****
(1) ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1) ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) ;*AND GO TO $ERRTYP ON ERROR
(1) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) ;*SW15=1 HALT ON ERROR
(1) ;*SW13=1 INHIBIT ERROR TYPEOUTS
(1) ;*SW09=1 LOOP ON ERROR
(1) ;*CALL
(1) ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1) $ERROR:
(1) 007706 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 007706 104410 JSR PC,CTRLCG ;;TEST FOR CTRL C/G
(3) 007710 004737 005760 BIS MASKNM,BADUNT
(3) 007714 053737 001376 001444 JSR PC,WHICHU ;DETERMINE UNIT #
(3) 007722 004737 007652 7$: INCB $ERFLG ;;SET THE ERROR FLAG
(1) 007726 105237 001103 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
(1) 007732 001775 MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 007734 013777 001102 171200 INC $ERTTL ;;INC THE ERROR COUNT
(1) 007742 005237 001112 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 007746 011637 001116 SUB #2,$ERRPC
(1) 007752 162737 000002 001116 MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 007760 117737 171132 001114 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
(1) 007766 032777 020000 171144 BNE 20$ ;;SKIP TYPEOUTS
(1) 007774 001004 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
(1) 007776 004737 010110 TYPE , $CRLF
(1) 010002 104401 001165 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 010006 122737 000001 001210 BNE 2$ ;;NO,SKIP APT ERROR REPORT
(1) 010014 001007 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 010016 113737 001114 010030 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
(1) 010024 004737 010754 21$: .BYTE 0
(1) 010030 000 .BYTE 0
(1) 010031 000 22$: BR 22$ ;;APT ERROR LOOP
(1) 010032 000777 171100 2$: TST @SWR ;;HALT ON ERROR
(1) 010034 005777 BPL 3$ ;;SKIP IF CONTINUE
(1) 010040 100002 HALT ;;HALT ON ERROR!
(1) 010042 000000 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 010044 104410 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(1) 010046 032777 001000 171064 BEQ 4$ ;;BR IF NO
(1) 010054 001402 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 010056 013716 001110

```


CVMNE-A MNCDO
CVMNEA.P11

DIAGNOSTIC
ERROR HANDLER ROUTINE

B 4

```

(1) 010062 005737 001162 4$: TST $ESCAPE ;:CHECK FOR AN ESCAPE ADDRESS
(1) 010066 001402 BEQ 5$ ;:BR IF NONE
(1) 010070 013716 001162 MOV $ESCAPE,(SP) ;:FUDGE RETURN ADDRESS FOR ESCAPE
(1) 010074 022737 005514 000042 5$: CMP #SENDAD,@#42 ;:ACT-11 AUTO-ACCEPT?
(1) 010102 001001 BNE 6$ ;:BRANCH IF NO
(1) 010104 000000 HALT ;:YES
(1) 010106 000002 6$: RTI ;:RETURN

```

670 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

(1) ;:*****
(2) ;:*THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
(1) ;:*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
(1) ;:*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

```

(1) 010110 $ERRTYP:
(1) 010110 104401 001165 TYPE , $CRLF ;:'CARRIAGE RETURN' & 'LINE FEED'
(1) 010114 010046 MOV RO, -(SP) ;:SAVE RO
(1) 010116 005000 CLR RO ;:PICKUP THE ITEM INDEX
(1) 010120 153700 001114 BISB @#$ITEMB,RO
(1) 010124 001004 BNE 1$ ;:IF ITEM NUMBER IS ZERO, JUST
(1) ;:TYPE THE PC OF THE ERROR
(2) 010126 013746 001116 MOV $ERRPC, -(SP) ;:SAVE $ERRPC FOR TYPEOUT
(2) ;:ERROR ADDRESS
(2) 010132 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 010134 000445 BR 10$ ;:GET OUT
(1) 010136 005300 1$: DEC RO ;:ADJUST THE INDEX SO THAT IT WILL
(1) 010140 006300 ASL RO ;: WORK FOR THE ERROR TABLE
(1) 010142 006300 ASL RO
(1) 010144 006300 ASL RO
(1) 010146 062700 001254 ADD # $ERRTB, RO ;:FORM TABLE POINTER
(1) 010152 012037 010162 MOV (RO)+, 2$ ;:PICKUP 'ERROR MESSAGE' POINTER
(1) 010156 001404 BEQ 3$ ;:SKIP TYPEOUT IF NO POINTER
(1) 010160 104401 TYPE ;:TYPE THE 'ERROR MESSAGE'
(1) 010162 000000 2$: .WORD 0 ;:'ERROR MESSAGE' POINTER GOES HERE
(1) 010164 104401 001165 TYPE , $CRLF ;:'CARRIAGE RETURN' & 'LINE FEED'
(1) 010170 012037 010200 3$: MOV (RO)+, 4$ ;:PICKUP 'DATA HEADER' POINTER
(1) 010174 001404 BEQ 5$ ;:SKIP TYPEOUT IF 0
(1) 010176 104401 TYPE ;:TYPE THE 'DATA HEADER'
(1) 010200 000000 4$: .WORD 0 ;:'DATA HEADER' POINTER GOES HERE
(1) 010202 104401 001165 TYPE , $CRLF ;:'CARRIAGE RETURN' & 'LINE FEED'
(1) 010206 010146 5$: MOV R1, -(SP) ;:SAVE R1
(1) 010210 012001 MOV (R0)+, R1 ;:PICKUP 'DATA TABLE' POINTER
(1) 010212 001415 BEQ 9$ ;:BR IF NO DATA TO BE TYPED
(1) 010214 012000 MOV (R0)+, R0 ;:PICKUP 'DATA FORMAT' POINTER
(1) 010216 105720 6$: TSTB (R0)+ ;:'OCTAL' OR 'DECIMAL'
(1) 010220 001003 BNE 7$ ;:BR IF DECIMAL
(2) 010222 013146 MOV @ (R1)+, -(SP) ;:SAVE @ (R1)+ FOR TYPEOUT
(2) 010224 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 010226 000402 BR 8$
(2) 010230 7$:
(2) 010230 013146 MOV @ (R1)+, -(SP) ;:SAVE @ (R1)+ FOR TYPEOUT
(2) 010232 104405 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN

```

```
(1) 010234 005711      8$: TST (R1)      ;; IS THERE ANOTHER NUMBER?
(1) 010236 001403      BEQ 9$          ;; BR IF NO
(1) 010240 104401 010260 TYPE ,11$       ;; TYPE TWO(2) SPACES
(1) 010244 000764      BR 6$          ;; LOOP
(1)
(1) 010246 012601      9$: MOV (SP)+,R1   ;; RESTORE R1
(1) 010250 012600      10$: MOV (SP)+,R0  ;; RESTORE R0
(1) 010252 104401 001165 TYPE ,SCRLF     ;; 'CARRIAGE RETURN' & 'LINE FEED'
(1) 010256 000207      RTS PC          ;; RETURN
(1) 010260 020040 000 11$: .ASCIZ / /      ;; TWO(2) SPACES
(1) 010264
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

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

*\$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*\$TYPOS OR \$TYPOC

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

*\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

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

```
(1) 010264 017646 000000 010507 $TYPOS: MOV @ (SP),-(SP)  ;; PICKUP THE MODE
(1) 010270 116637 000001 010507 MOV B 1 (SP), $OFILL ;; LOAD ZERO FILL SWITCH
(1) 010276 112637 010511 010507 MOV B (SP)+, $OMODE+1 ;; NUMBER OF DIGITS TO TYPE
(1) 010302 062716 000002 010507 ADD #2, (SP)        ;; ADJUST RETURN ADDRESS
(1) 010306 000406 010507 BR $TYPON
(1) 010310 112737 000001 010507 $TYPOC: MOV B #1, $OFILL   ;; SET THE ZERO FILL SWITCH
(1) 010316 112737 000006 010511 MOV B #6, $OMODE+1  ;; SET FOR SIX(6) DIGITS
(1) 010324 112737 000005 010506 $TYPON: MOV B #5, $OCNT  ;; SET THE ITERATION COUNT
(1) 010332 010346 010511 MOV R3, -(SP)      ;; SAVE R3
(1) 010334 010446 010511 MOV R4, -(SP)      ;; SAVE R4
(1) 010336 010546 010511 MOV R5, -(SP)      ;; SAVE R5
(1) 010340 113704 010511 MOV B $OMODE+1, R4  ;; GET THE NUMBER OF DIGITS TO TYPE
(1) 010344 005404 010511 NEG R4
(1) 010346 062704 000006 010511 ADD #6, R4        ;; SUBTRACT IT FOR MAX. ALLOWED
(1) 010352 110437 010510 010511 MOV B R4, $OMODE  ;; SAVE IT FOR USE
(1) 010356 113704 010507 010511 MOV B $OFILL, R4  ;; GET THE ZERO FILL SWITCH
(1) 010362 016605 000012 010511 MOV 12 (SP), R5   ;; PICKUP THE INPUT NUMBER
(1) 010366 005003 010511 CLR R3              ;; CLEAR THE OUTPUT WORD
```

```

(1) 010370 006105        1$:  ROL    R5            ;;ROTATE MSB INTO 'C'
(1) 010372 000404        BR      3$            ;;GO DO MSB
(1) 010374 006105        2$:  ROL    R5            ;;FORM THIS DIGIT
(1) 010376 006105        ROL    R5
(1) 010400 006105        ROL    R5
(1) 010402 010503        MOV    R5,R3
(1) 010404 006103        3$:  ROL    R3            ;;GET LSB OF THIS DIGIT
(1) 010406 105337 010510 DECB   $OMODE        ;;TYPE THIS DIGIT?
(1) 010412 100016        BPL    7$            ;;BR IF NO
(1) 010414 042703 177770 BIC    #177770,R3    ;;GET RID OF JUNK
(1) 010420 001002        BNE    4$            ;;TEST FOR 0
(1) 010422 005704        TST    R4            ;;SUPPRESS THIS 0?
(1) 010424 001403        BEQ    5$            ;;BR IF YES
(1) 010426 005204        4$:  INC    R4            ;;DON'T SUPPRESS ANYMORE 0'S
(1) 010430 052703 000060 BIS    #'0,R3        ;;MAKE THIS DIGIT ASCII
(1) 010434 052703 000040 5$:  BIS    #' ,R3    ;;MAKE ASCII IF NOT ALREADY
(1) 010440 110337 010504 MOV    R3,8$        ;;SAVE FOR TYPING
(1) 010444 104401 010504 TYPE   ,8$            ;;GO TYPE THIS DIGIT
(1) 010450 105337 010506 7$:  DECB   $OCNT        ;;COUNT BY 1
(1) 010454 003347        BGT    2$            ;;BR IF MORE TO DO
(1) 010456 002402        BLT    6$            ;;BR IF DONE
(1) 010460 005204        INC    R4            ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 010462 000744        BR      2$            ;;GO DO THE LAST DIGIT
(1) 010464 012605        6$:  MOV    (SP)+,R5    ;;RESTORE R5
(1) 010466 012604        MOV    (SP)+,R4    ;;RESTORE R4
(1) 010470 012603        MOV    (SP)+,R3    ;;RESTORE R3
(1) 010472 016666 000002 000004 MOV    2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
(1) 010500 012616        MOV    (SP)+,(SP)
(1) 010502 000002        RTI                    ;;RETURN
(1) 010504    000        8$:  .BYTE  0            ;;STORAGE FOR ASCII DIGIT
(1) 010505    000        .BYTE  0            ;;TERMINATOR FOR TYPE ROUTINE
(1) 010506    000        $OCNT: .BYTE  0        ;;OCTAL DIGIT COUNTER
(1) 010507    000        $OFILL: .BYTE  0      ;;ZERO FILL SWITCH
(1) 010510 000000        $OMODE: .WORD   0     ;;NUMBER OF DIGITS TO TYPE
672 .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)
(1)
(1) ;*****
(1) ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1) ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1) ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1) ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1) ;*REPLACED WITH SPACES.
(1) ;*CALL:
(1) ;*      MOV    NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
(1) ;*      TYPDS            ;;GO TO THE ROUTINE
(1)
(2) $TYPDS:
(3) 010512 010046        MOV    R0,-(SP)      ;;PUSH R0 ON STACK
(3) 010514 010146        MOV    R1,-(SP)      ;;PUSH R1 ON STACK
(3) 010516 010246        MOV    R2,-(SP)      ;;PUSH R2 ON STACK
(3) 010520 010346        MOV    R3,-(SP)      ;;PUSH R3 ON STACK
(3) 010522 010546        MOV    R5,-(SP)      ;;PUSH R5 ON STACK
(1) 010524 012746 020200 MOV    #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
(1) 010530 016605 000020 MOV    20(SP),R5     ;;GET THE INPUT NUMBER

```

```

(1) 010534 100004          BPL      1$          ;;BR IF INPUT IS POS.
(1) 010536 005405          NEG      R5          ;;MAKE THE BINARY NUMBER POS.
(1) 010540 112766 000055 000001  MOVB    #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
(1) 010546 005000          CLR      R0          ;;ZERO THE CONSTANTS INDEX
(1) 010550 012703 010726    MOV      #$DBLK,R3   ;;SETUP THE OUTPUT POINTER
(1) 010554 112723 000040    MOVB    #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
(1) 010560 005002          CLR      R2          ;;CLEAR THE BCD NUMBER
(1) 010562 016001 010716    MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
(1) 010566 160105          SUB      R1,R5       ;;FORM THIS BCD DIGIT
(1) 010570 002402          BLT     4$          ;;BR IF DONE
(1) 010572 005202          INC     R2          ;;INCREASE THE BCD DIGIT BY 1
(1) 010574 000774          BR      3$
(1) 010576 060105          4$:    ADD     R1,R5     ;;ADD BACK THE CONSTANT
(1) 010600 005702          TST     R2          ;;CHECK IF BCD DIGIT=0
(1) 010602 001002          BNE     5$          ;;FALL THROUGH IF 0
(1) 010604 105716          TSTB   (SP)         ;;STILL DOING LEADING 0'S?
(1) 010606 100407          BMI     7$          ;;BR IF YES
(1) 010610 106316          5$:    ASLB   (SP)         ;;MSD?
(1) 010612 103003          BCC     6$          ;;BR IF NO
(1) 010614 116663 000001 177777  MOVB    1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 010622 052702 000060    BIS     #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1) 010626 052702 000040    7$:    BIS     #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 010632 110223          MOVB    R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 010634 005720          TST     (R0)+      ;;JUST INCREMENTING
(1) 010636 020027 000010    CMP     R0,#10     ;;CHECK THE TABLE INDEX
(1) 010642 002746          BLT     2$          ;;GO DO THE NEXT DIGIT
(1) 010644 003002          BGT     8$          ;;GO TO EXIT
(1) 010646 010502          MOV     R5,R2      ;;GET THE LSD
(1) 010650 000764          BR      6$          ;;GO CHANGE TO ASCII
(1) 010652 105726          8$:    TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 010654 100003          BPL     9$          ;;BR IF NO
(1) 010656 116663 177777 177776  MOVB    -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 010664 105013          9$:    CLRB   (R3)         ;;SET THE TERMINATOR
(3) 010666 012605          MOV     (SP)+,R5    ;;POP STACK INTO R5
(3) 010670 012603          MOV     (SP)+,R3    ;;POP STACK INTO R3
(3) 010672 012602          MOV     (SP)+,R2    ;;POP STACK INTO R2
(3) 010674 012601          MOV     (SP)+,R1    ;;POP STACK INTO R1
(3) 010676 012600          MOV     (SP)+,R0    ;;POP STACK INTO R0
(1) 010700 104401 010726    TYPE   , $DBLK      ;;NOW TYPE THE NUMBER
(1) 010704 016666 000002 000004  MOV     2(SP),4(SP) ;;ADJUST THE STACK
(1) 010712 012616          MOV     (SP)+,(SP)
(1) 010714 000002          RTI
(1) 010716 023420          $DTBL: 10000.
(1) 010720 001750          1000.
(1) 010722 000144          100.
(1) 010724 000012          10.
(1) 010726 000004          $DBLK: .BLKW 4

```

```

674          .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)
(1) 010736 112737 000001 011202 $ATY1:  MOVB  #1,$FFLG      ::TO REPORT FATAL ERROR
(1) 010744 112737 000001 011200 $ATY3:  MOVB  #1,$MFLG      ::TO TYPE A MESSAGE
(1) 010752 000403                BR      $ATYC
(1) 010754 112737 000001 011202 $ATY4:  MOVB  #1,$FFLG      ::TO ONLY REPORT FATAL ERROR
(2) 010762 $ATYC:
(3) 010762 010046                MOV    R0,-(SP)          ::PUSH R0 ON STACK
(3) 010764 010146                MOV    R1,-(SP)          ::PUSH R1 ON STACK
(1) 010766 105737 011200                TSTB  $MFLG              ::SHOULD TYPE A MESSAGE?
(1) 010772 001450                BEQ   5$                  ::IF NOT: BR
(1) 010774 122737 000001 001210        CMPB  #APTENV,$ENV       ::OPERATING UNDER APT?
(1) 011002 001031                BNE   3$                  ::IF NOT: BR
(1) 011004 132737 000100 001211        BITB  #APTSPOOL,$ENVM   ::SHOULD SPOOL MESSAGES?
(1) 011012 001425                BEQ   3$                  ::IF NOT: BR
(1) 011014 017600 000004                MOV    @4(SP),R0         ::GET MESSAGE ADDR.
(1) 011020 062766 000002 000004        ADD    #2,4(SP)          ::BUMP RETURN ADDR.
(1) 011026 005737 001170                1$:  TST  $MSGTYPE         ::SEE IF DONE W/ LAST XMISSION?
(1) 011032 001375                BNE   1$                  ::IF NOT: WAIT
(1) 011034 010037 001204                MOV    R0,$MSGAD        ::PUT ADDR IN MAILBOX
(1) 011040 105720                2$:  TSTB (R0)+           ::FIND END OF MESSAGE
(1) 011042 001376                BNE   2$
(1) 011044 163700 001204                SUB    $MSGAD,R0         ::SUB START OF MESSAGE
(1) 011050 006200                ASR   R0                  ::GET MESSAGE LNGTH IN WORDS
(1) 011052 010037 001206                MOV    R0,$MSGGLT       ::PUT LENGTH IN MAILBOX
(1) 011056 012737 000004 001170        MOV    #4,$MSGTYPE      ::TELL APT TO TAKE MSG.
(1) 011064 000413                BR    5$
(1) 011066 017637 000004 011112 3$:  MOV    @4(SP),4$         ::PUT MSG ADDR IN JSR LINKAGE
(1) 011074 062766 000002 000004        ADD    #2,4(SP)          ::BUMP RETURN ADDRESS
(3) 011102 013746 177776                MOV    177776,-(SP)     ::PUSH 177776 ON STACK
(1) 011106 004737 011526                JSR   PC,$TYPE          ::CALL TYPE MACRO
(1) 011112 000000                4$:  .WORD 0
(1) 011114                5$:
(1) 011114 105737 011202                10$: TSTB  $FFLG             ::SHOULD REPORT FATAL ERROR?
(1) 011120 001416                BEQ   12$                 ::IF NOT: BR
(1) 011122 005737 001210                TST  $ENV                 ::RUNNING UNDER APT?
(1) 011126 001413                BEQ   12$                 ::IF NOT: BR
(1) 011130 005737 001170                11$: TST  $MSGTYPE           ::FINISHED LAST MESSAGE?
(1) 011134 001375                BNE   11$                 ::IF NOT: WAIT
(1) 011136 017637 000004 001172        MOV    @4(SP),$FATAL     ::GET ERROR #
(1) 011144 062766 000002 000004        ADD    #2,4(SP)          ::BUMP RETURN ADDR.
(1) 011152 005237 001170                INC  $MSGTYPE            ::TELL APT TO TAKE ERROR
(1) 011156 105037 011202                12$: CLRB  $FFLG           ::CLEAR FATAL FLAG
(1) 011162 105037 011201                CLRB  $LFLG              ::CLEAR LOG FLAG
(1) 011166 105037 011200                CLRB  $MFLG              ::CLEAR MESSAGE FLAG
(3) 011172 012601                MOV    (SP)+,R1          ::POP STACK INTO R1
(3) 011174 012600                MOV    (SP)+,R0          ::POP STACK INTO R0
(1) 011176 000207                RTS   PC                 ::RETURN
(1) 011200                000 $MFLG: .BYTE 0       ::MESSG. FLAG
(1) 011201                000 $LFLG: .BYTE 0       ::LOG FLAG
(1) 011202                000 $FFLG: .BYTE 0       ::FATAL FLAG
(1)                011204 .EVEN
(1)                000200 APTSIZE=200

```

```

(1) 000001 APTENV=001
(1) 000100 APTSPOOL=100
(1) 000040 APTCSUP=040
675 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)
(1)
(1) 011204 012737 011350 000024 $PWRDN: MOV #IILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 011212 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
(3) 011220 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 011222 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 011224 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 011226 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 011230 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
(3) 011232 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(3) 011234 017746 167700 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
(1) 011240 010637 011354 MOV SP,$SAVR6 ;;SAVE SP
(1) 011244 012737 011256 000024 MOV #PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 011252 000000 HALT
(1) 011254 000776 BR -2 ;;HANG UP
(1)
(2)
(1)
(1) 011256 012737 011350 000024 $PWRUP: MOV #IILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1) 011264 013706 011354 MOV $SAVR6,SP ;;GET SP
(1) 011270 005037 011354 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
(1) 011274 005237 011354 1$: INC $SAVR6 ;;WAIT FOR THE INC
(1) 011300 001375 BNE 1$ ;;OF WORD
(3) 011302 012677 167632 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
(3) 011306 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
(3) 011310 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
(3) 011312 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 011314 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 011316 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 011320 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 011322 012737 011204 000024 MOV #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 011330 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
(1) 011336 104401 TYPE ;;REPORT THE POWER FAILURE
(1) 011340 011356 $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
(1) 011342 012716 MOV (PC)+,(SP) ;;RESTART AT BEGIN
(1) 011344 001450 $PWRAD: .WORD BEGIN ;;RESTART ADDRESS
(1) 011346 000002 RTI
(1) 011350 000000 $IILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
(1) 011352 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 011354 000000 $SAVR6: 0 ;;PUT THE SP HERE
676 011356 005015 042522 052123 PWRMSG: .ASCIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
011364 051101 044524 043516
011372 040440 052106 051105
011400 040440 050040 053517
011406 051105 043040 044501
011414 052514 042522 005015
011422 000
677 011424 .EVEN

```

```

679          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)          ::*****
(1)          ::*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)          ::*CHANGE IT TO BINARY.
(1)          ::*CALL:
(1)          ::*      RDOCT          ::READ AN OCTAL NUMBER
(1)          ::*      RETURN HERE    ::LOW ORDER BITS ARE ON TOP OF THE STACK
(1)          ::*          ::HIGH ORDER BITS ARE IN $HIOCT
(1)
(1) 011424 011646          $RDOCT: MOV      (SP),-(SP)      ::PROVIDE SPACE FOR THE
(1) 011426 016666          MOV      4(SP),2(SP)      ::INPUT NUMBER
(3) 011434 010046          MOV      R0,-(SP)      ::PUSH R0 ON STACK
(3) 011436 010146          MOV      R1,-(SP)      ::PUSH R1 ON STACK
(3) 011440 010246          MOV      R2,-(SP)      ::PUSH R2 ON STACK
(1) 011442 104412          1$:  RDLIN      ::READ AN ASCII LINE
(1) 011444 012600          MOV      (SP)+,R0      ::GET ADDRESS OF 1ST CHARACTER
(1) 011446 005001          CLR      R1          ::CLEAR DATA WORD
(1) 011450 005002          CLR      R2
(1) 011452 112046          2$:  MOVB     (R0)+,-(SP)  ::PICKUP THIS CHARACTER
(1) 011454 001412          BEQ     3$          ::IF ZERO GET OUT
(1) 011456 006301          ASL     R1          ::*2
(1) 011460 006102          ROL     R2
(1) 011462 006301          ASL     R1          ::*4
(1) 011464 006102          ROL     R2
(1) 011466 006301          ASL     R1          ::*8
(1) 011470 006102          ROL     R2
(1) 011472 042716          BIC     #'C7,(SP)   ::STRIP THE ASCII JUNK
(1) 011476 062601          ADD     (SP)+,R1   ::ADD IN THIS DIGIT
(1) 011500 000764          BR      2$          ::LOOP
(1) 011502 005726          3$:  TST     (SP)+    ::CLEAN TERMINATOR FROM STACK
(1) 011504 010166          MOV     R1,12(SP)  ::SAVE THE RESULT
(1) 011510 010237          MOV     R2,$HIOCT
(3) 011514 012602          MOV     (SP)+,R2   ::POP STACK INTO R2
(3) 011516 012601          MOV     (SP)+,R1   ::POP STACK INTO R1
(3) 011520 012600          MOV     (SP)+,R0   ::POP STACK INTO R0
(1) 011522 000002          RTI          ::RETURN
(1) 011524 000000          $HIOCT: .WORD 0    ::HIGH ORDER BITS GO HERE
680          ;CAUTION  THE FIRST 4 LOC. ARE OVERLAYED TO LOWER CPU LEVEL
681          .SBTTL  TYPE ROUTINE
(1)
(2)          ::*****
(1)          ::*ROUTINE TO TYPE ASCII 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 ASCII STRING
(1)          ::*OR
(1)          ::*      TYPE
(1)          ::*      MESADR

```

```
(1) ;*
(1)
(1) 011526 105737 001157 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
(1) 011532 100002 BPL 1$ ;; BR IF YES
(1) 011534 000000 HALT ;; HALT HERE IF NO TERMINAL
(1) 011536 000430 BR 3$ ;; LEAVE
(1) 011540 010046 1$: MOV R0,-(SP) ;; SAVE R0
(1) 011542 017600 000002 MOV @2(SP),R0 ;; GET ADDRESS OF ASCIZ STRING
(1) 011546 122737 000001 001210 CMPB #APTENV,$ENV ;; RUNNING IN APT MODE
(1) 011554 001011 BNE 62$ ;; NO,GO CHECK FOR APT CONSOLE
(1) 011556 132737 000100 001211 BITB #APTSPOOL,$ENVM ;; SPOOL MESSAGE TO APT
(1) 011564 001405 BEQ 62$ ;; NO,GO CHECK FOR CONSOLE
(1) 011566 010037 011576 MOV R0,61$ ;; SETUP MESSAGE ADDRESS FOR APT
(1) 011572 004737 010744 JSR PC,$ATY3 ;; SPOOL MESSAGE TO APT
(1) 011576 000000 61$: .WORD 0 ;; MESSAGE ADDRESS
(1) 011600 132737 000040 001211 62$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
(1) 011606 001003 BNE 60$ ;; YES,SKIP TYPE OUT
(1) 011610 112046 2$: MOVB (R0)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 011612 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
(1) 011614 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
(1) 011616 012600 60$: MOV (SP)+,R0 ;; RESTORE R0
(1) 011620 062716 000002 3$: ADD #2,(SP) ;; ADJUST RETURN PC
(1) 011624 000002 RTI ;; RETURN
(1) 011626 122716 000011 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
(1) 011632 001430 BEQ 8$
(1) 011634 122716 000200 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
(1) 011640 001006 BNE 5$
(1) 011642 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
(1) 011644 104401 TYPE ;; TYPE A CR AND LF
(1) 011646 001165 $CRLF
(1) 011650 105037 012004 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
(1) 011654 000755 BR 2$ ;; GET NEXT CHARACTER
(1) 011656 004737 011740 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
(1) 011662 123726 001156 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
(1) 011666 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
(1) 011670 013746 001154 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
(1) ;; AND THE NULL CHAR.
(1) 011674 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
(1) 011700 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 011702 004737 011740 JSR PC,$TYPEC ;; GO TYPE A NULL
(1) 011706 105337 012004 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
(1) 011712 000770 BR 7$ ;; LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 011714 112716 000040 8$: MOVB #' ,(SP) ;; REPLACE TAB WITH SPACE
(1) 011720 004737 011740 9$: JSR PC,$TYPEC ;; TYPE A SPACE
(1) 011724 132737 000007 012004 BITB #7,$CHARCNT ;; BRANCH IF NOT AT
(1) 011732 001372 BNE 9$ ;; TAB STOP
(1) 011734 005726 TST (SP)+ ;; POP SPACE OFF STACK
(1) 011736 000724 BR 2$ ;; GET NEXT CHARACTER
(1) 011740 105777 167204 $TYPEC: TSTB @2$TPS ;; WAIT UNTIL PRINTER IS READY
(1) 011744 100375 BPL $TYPEC
(1) 011746 116677 000002 167176 MOVB 2(SP),@2$TPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
```



```

(1) 011754 122766 000015 000002      CMPB   #CR,2(SP)      ;; IS CHARACTER A CARRIAGE RETURN?
(1) 011762 001003                    BNE    1$            ;; BRANCH IF NO
(1) 011764 105037 012004              CLRB   $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
(1) 011770 000406                    BR     $TYPEX        ;; EXIT
(1) 011772 122766 000012 000002 1$:  CMPB   #LF,2(SP)      ;; IS CHARACTER A LINE FEED?
(1) 012000 001402                    BEQ    $TYPEX        ;; BRANCH IF YES
(1) 012002 105227                    INCB   (PC)+         ;; COUNT THE CHARACTER
(1) 012004 000000                    $CHARCNT: .WORD    0 ;; CHARACTER COUNT STORAGE
(1) 012006 000207                    $TYPEX: RTS         PC

(1)
682      .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
(1)
(2)      ;*****
(1)      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
(1)      ;*BINARY-ASCII NUMBER AND TYPE IT.
(1)      ;*CALL:
(1)      ;*      MOV     NUMBER,-(SP)      ;;NUMBER TO BE TYPED
(1)      ;*      TYPBN                    ;;TYPE IT
(1)
(1) 012010 010146                    $TYPBN: MOV    R1,-(SP)      ;;SAVE R1 ON THE STACK
(1) 012012 016601 000006              MOV    6(SP),R1        ;;GET THE INPUT NUMBER
(1) 012016 000261                    SEC                      ;;SET 'C' SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1) 012020 112737 000060 012062 1$:  MOVB   #'0,$BIN        ;;SET CHARACTER TO AN ASCII '0'.
(1) 012026 006101                    ROL    R1              ;;GET THIS BIT
(1) 012030 001406                    BEQ    2$              ;;DONE?
(1) 012032 105537 012062              ADCB   $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1) 012036 104401 012062              TYPE   , $BIN         ;;GO TYPE THIS BIT
(1) 012042 000241                    CLC                      ;;CLEAR 'C' SO CAN KEEP TRACK OF BITS
(1) 012044 000765                    BR     1$              ;;GO DO THE NEXT BIT
(1) 012046 012601                    MOV    (SP)+,R1        ;;POP THE STACK INTO R1
(1) 012050 016666 000002 000004      MOV    2(SP),4(SP)    ;;ADJUST THE STACK
(1) 012056 012616                    MOV    (SP)+,(SP)
(1) 012060 000002                    RTI                      ;;RETURN TO USER
(1) 012062 000      000      $BIN:  .BYTE  0,0      ;;STORAGE FOR ASCII CHAR. AND TERMINATOR

```

```
684
685      .SBTTL  TRAP DECODER
(1)
(2)      ;:*****
(1)      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)      ;*GO TO THAT ROUTINE.
(1)
(1) 012064 010046      $TRAP:  MOV   R0,-(SP)          ;:SAVE R0
(1) 012066 016600 000002      MOV   2(SP),R0          ;:GET TRAP ADDRESS
(1) 012072 005740          TST   -(R0)            ;:BACKUP BY 2
(1) 012074 111000          MOVB  (R0),R0          ;:GET RIGHT BYTE OF TRAP
(1) 012076 006300          ASL   R0                ;:POSITION FOR INDEXING
(1) 012100 016000 012120      MOV   $TRPAD(R0),R0    ;:INDEX TO TABLE
(1) 012104 000200          RTS   R0                ;:GO TO ROUTINE
(1)
(1)
(1)      ;:THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1) 012106 011646      $TRAP2: MOV (SP),-(SP)      ;:MOVE THE PC DOWN
(1) 012110 016666 000004 000002  MOV  4(SP),2(SP)        ;:MOVE THE PSW DOWN
(1) 012116 000002          RTI                    ;:RESTORE THE PSW
(1)
(3)      .SBTTL  TRAP TABLE
(3)
(3)      ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)      ;*BY THE "TRAP" INSTRUCTION.
(3)
(3)      : ROUTINE
(3)      : -----
(3) $TRPAD: .WORD  $TRAP2
(3) 012120 012106      $TYPE   ;:CALL=TYPE          TRAP+1(104401) TTY TYPEOUT ROUTINE
(3) 012122 011526      $TYPOC  ;:CALL=TYPOC         TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 012124 010310      $TYPOS  ;:CALL=TYPOS         TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 012126 010264      $TYPON  ;:CALL=TYPON         TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 012130 010324      $TYPDS  ;:CALL=TYPDS         TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(3) 012132 010512      $TYPBN  ;:CALL=TYPBN         TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
(1)
(1) 012136 006462      $GTSWR  ;:CALL=GTSWR         TRAP+7(104407) GET SOFT-SWR SETTING
(1)
(3) 012140 006372      $CKSWR  ;:CALL=CKSWR         TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
(3) 012142 006734      $RDCHR  ;:CALL=RDCHR         TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
(3) 012144 007024      $RDLIN  ;:CALL=RDLIN         TRAP+12(104412) TTY TYPEIN STRING ROUTINE
(3) 012146 011424      $RDOCT  ;:CALL=RDOCT         TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
```


741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
(1)
(1)
(1)
767
768
(1)
(1)
(1)
769
770
771
772
773
774
775
776
777
778

012222 000002
012224 022626
012226 005737 001176
012232 001025
012234 104401 013714
012240 013746 001202
012244 104405
012246 000240
012250 000240
012252 104401 013750
012256 013746 001354
012262 104403
012264 003
012265 001
012266 104401 014000
012272 013746 012356
012276 104403
012300 003
012301 001
012302 104401 014030
012306 013777 001356 167040
012314 012777 004700 167034
012322 013737 012356 001354
012330 042737 000003 001354
012336 013737 001354 001356
012344 062737 000002 001356
012352 000177 166530
012356 000000
012360 000000

000002
022626
005737 001176
001025
104401 013714
013746 001202
104405
000240
000240
104401 013750
013746 001354
104403
003
001
104401 014000
013746 012356
104403
003
001
104401 014030
013777 001356 167040
012777 004700 167034
013737 012356 001354
042737 000003 001354
013737 001354 001356
062737 000002 001356
000177 166530
000000
000000

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

```
RTI  
3$: CMP (SP)+, (SP)+  
TST $PASS  
BNE 4$  
TYPE, VTMSG  
MOV $UNIT, -(SP)  
TYPDS  
NOP  
NOP  
TYPE ,VTMSG3  
MOV DODINV, -(SP)  
TYPOS  
.BYTE 3  
.BYTE 1  
TYPE ,VTMSG1  
MOV TRTO, -(SP)  
TYPOS  
.BYTE 3  
.BYTE 1  
TYPE ,VTMSG2  
4$: MOV DODINS, @DODINV  
MOV #4700, @DODINS  
MOV TRTO, DODINV  
BIC #3, DODINV  
MOV DODINV, DODINS  
ADD #2, DODINS  
JMP @SLPADR  
TRTO: .WORD 0  
TRFRO: .WORD 0
```

```
:DEVICE PUT THE WRONG INTERRUPT  
:VECTOR ON THE BUS DURING THE INTERRUPT  
: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.  
  
:IS THIS THE FIRST PASS?  
:NO, DON'T REPORT  
:TELL OPER.  
  
:TYPE 'EXPECTED INTR. AT '  
:SAVE DODINV 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'  
:ADJUST VECTOR  
  
:START TEST OVER AGAIN.  
:CONTAINS ADDR. WE TRAPPED OR INTERRUPTED TO.  
:CONTAINS ADDR. WE TRAPPED OR INTR. FROM.
```

780					.SBTTL ASCII MESSAGES
781	012362	015	012		PRIMEO: .BYTE 15,12
782	012364	020114	020075	047514	.ASCII \L = LOGIC TEST WITH NO TEST MODULE CONNECTED\
	012372	044507	020103	042524	
	012400	052123	053440	052111	
	012406	020110	047516	052040	
	012414	051505	020124	047515	
	012422	052504	042514	041440	
	012430	047117	042516	052103	
	012436	042105			
783	012440	015	012		.BYTE 15,12
784	012442	020104	020075	047514	.ASCII \D = LOGIC TEST WITH TEST MODULE CONNECTED\
	012450	044507	020103	042524	
	012456	052123	053440	052111	
	012464	020110	042524	052123	
	012472	046440	042117	046125	
	012500	020105	047503	047116	
	012506	041505	042524	104	
785	012513	015	012		.BYTE 15,12
786	012515	117	036440	047440	.ASCII \O = OUTPUT TEST MODULE L.E.D. LOOP\
	012522	052125	052520	020124	
	012530	042524	052123	046440	
	012536	042117	046125	020105	
	012544	027114	027105	027104	
	012552	046040	047517	120	
787	012557	015	012		.BYTE 15,12
788	012561	120	036440	050040	.ASCII \P = PULSE OUTPUT VERIFY LOOP\
	012566	046125	042523	047440	
	012574	052125	052520	020124	
	012602	042526	044522	054506	
	012610	046040	047517	120	
789	012615	015	012		.BYTE 15,12
790	012617	107	036440	043440	.ASCII \G = GET NEW SWITCH REGISTER VALUE\
	012624	052105	047040	053505	
	012632	051440	044527	041524	
	012640	020110	042522	044507	
	012646	052123	051105	053040	
	012654	046101	042525		
791	012660	015	012		.BYTE 15,12
792	012662	020102	020075	040502	.ASCII \B = BASE OR VECTOR ADDRESS CHANGE\
	012670	042523	047440	020122	
	012676	042526	052103	051117	
	012704	040440	042104	042522	
	012712	051523	041440	040510	
	012720	043516	105		
793	012723	015	012		.BYTE 15,12
794	012725	110	036440	044040	.ASCII \H = HELP THE OPERATOR AND RETYPE THIS LIST \
	012732	046105	020120	044124	
	012740	020105	050117	051105	
	012746	052101	051117	040440	
	012754	042116	051040	052105	
	012762	050131	020105	044124	
	012770	051511	046040	051511	
	012776	020124	020040	000040	

795	013004	015	012		
796	013006	054524	042520	052040	DOT: .BYTE 15,12
	013014	042510	021040	042524	.ASCIZ /TYPE THE 'TEST CHARACTER' THEN DEPRESS 'RETURN KEY' /
	013022	052123	041440	040510	
	013030	040522	052103	051105	
	013036	020042	044124	047105	
	013044	042040	050105	042522	
	013052	051523	021040	042522	
	013060	052524	047122	045440	
	013066	054505	020042	000040	
797					
798	013074	047115	042103	020117	EM1: .ASCIZ VMNCDO (DIGITAL OUT) DOES NOT EXIST <BUS ERROR> CHECK ADDRESS SWITCH
	013102	042050	043511	052111	
	013110	046101	047440	052125	
	013116	004451	042040	042517	
	013124	020123	047516	020124	
	013132	054105	051511	020124	
	013140	041074	051525	042440	
	013146	051122	051117	020076	
	013154	044103	041505	020113	
	013162	042101	051104	051505	
	013170	020123	053523	052111	
	013176	044103	051505	000	
799	013203	115	041516	047504	EM2: .ASCIZ VMNCDO (DIGITAL OUT) DATA REGISTER ERROR\
	013210	024040	044504	044507	
	013216	040524	020114	052517	
	013224	024524	042040	052101	
	013232	020101	042522	044507	
	013240	052123	051105	042440	
	013246	051122	051117	000	
800	013253	115	041516	047504	EM3: .ASCIZ VMNCDO (DIGITAL OUT) STATUS REGISTER ERROR\
	013260	024040	044504	044507	
	013266	040524	020114	052517	
	013274	024524	051440	040524	
	013302	052524	020123	042522	
	013310	044507	052123	051105	
	013316	042440	051122	051117	
	013324	000			
801	013325	115	041516	047504	EM4: .ASCIZ VMNCDO (DIGITAL OUT) INTERRUPT ERROR\
	013332	024040	044504	044507	
	013340	040524	020114	052517	
	013346	024524	044440	052116	
	013354	051105	052522	052120	
	013362	042440	051122	051117	
	013370	000			
802	013371	115	041516	047504	EM6: .ASCIZ VMNCDO (DIGITAL OUT) UNEXPECTED INTERRUPT\
	013376	024040	044504	044507	
	013404	040524	020114	052517	
	013412	024524	052440	042516	
	013420	050130	041505	042524	
	013426	020104	047111	042524	
	013434	051122	050125	000124	
803	013442	047115	042103	020117	EM7: .ASCIZ VMNCDO (DIGITAL OUT) EXISTING UNIT FAILED TO RESPOND\
	013450	042050	043511	052111	

	013456	046101	047440	052125	
	013464	004451	054105	051511	
	013472	044524	043516	052440	
	013500	044516	020124	040506	
	013506	046111	042105	052040	
	013514	020117	042522	050123	
	013522	047117	000104		
804	013526	047115	042103	004517	EM14: .ASCIZ \MNCDO INCORRECT I.D. VALUE FOR MNCDO\
	013534	047111	047503	051122	
	013542	041505	020124	027111	
	013550	027104	053040	046101	
	013556	042525	043040	051117	
	013564	046440	041516	047504	
	013572	000			
805	013573	200	051120	043517	FOUND1: .ASCIZ <200> /PROGRAM DETECTED /
	013600	040522	020115	042504	
	013606	042524	052103	042105	
	013614	000040			
806	013616	046440	041516	047504	FOUND2: .ASCIZ \ MNCDO (DIGITAL OUT)'S \
	013624	024040	044504	044507	
	013632	040524	020114	052517	
	013640	024524	051447	020040	
	013646	000040			
807	013650	035440	047524	040524	ERRTOT: .ASCIZ / ;TOTAL ERROR COUNT = /
	013656	020114	051105	047522	
	013664	020122	047503	047125	
	013672	020124	020075	000	
808	013677	040	041073	042101	MESGD: .ASCIZ / ;BAD UNITS /
	013704	052440	044516	051524	
	013712	000040			
809	013714	046600	041516	047504	VTMSG: .ASCIZ <200>/MNCDO (DIGITAL OUT) UNIT #/
	013722	024040	044504	044507	
	013730	040524	020114	052517	
	013736	024524	052440	044516	
	013744	020124	000043		
810	013750	042600	050130	041505	VTMSG3: .ASCIZ <200>/EXPECTED INTERRUPT AT /
	013756	042524	020104	047111	
	013764	042524	051122	050125	
	013772	020124	052101	000040	
811	014000	051040	041505	044505	VTMSG1: .ASCIZ / RECEIVED INTERRUPT AT /
	014006	042526	020104	047111	
	014014	042524	051122	050125	
	014022	020124	052101	000040	
812	014030	050200	042514	051501	VTMSG2: .ASCII <200>/PLEASE CHECK VECTOR SWITCHES/<200>
	014036	020105	044103	041505	
	014044	020113	042526	052103	
	014052	051117	051440	044527	
	014060	041524	042510	100123	
813	014066	051011	051505	040524	.ASCIZ / RESTARTING LOGIC TEST/<200>
	014074	052122	047111	020107	
	014102	047514	044507	020103	
	014110	042524	052123	000200	
814	014116	047125	052111	042411	DH1: .ASCIZ /UNIT ERRPC OUTCSR OUTDAT/
	014124	051122	041520	047411	

CVMNE-A MNCDO
CVMNEA.P11

DIAGNOSTIC
ASCII MESSAGES

MACY11 27(654) 19-SEP-78 09:02 ^{D 5} PAGE 21-3

SEQ 0055

```

014132 052125 051503 004522
014140 052517 042124 052101
815 014146      000
014147      125 044516 004524 DH2: .ASCIZ /UNIT  ERRPC  OUTADR  GOOD  BAD/
014154 051105 050122 004503
014162 052517 040524 051104
014170 043411 047517 004504
816 014176 041040 042101      000
014203      125 044516 004524 DH3: .ASCIZ /UNIT  ERRPC  OUTADR  GOOD  BAD/
014210 051105 050122 004503
014216 052517 040524 051104
014224 043411 047517 004504
817 014232 041040 042101      000
014237      125 044516 004524 DH4: .ASCIZ /UNIT  ERRPC  OUTADR/
014244 051105 050122 004503
014252 052517 040524 051104
818 014260      000
014261      125 044516 004524 DH6: .ASCIZ /UNIT  ERRPC  TO  FROM ADRS /
014266 051105 050122 020103
014274 020040 052040 004517
014302 051106 046517 040440
819 014310 051104 020123      000
014315      125 044516 004524 DH7: .ASCIZ /UNIT  ERRPC  WERE  ARE /
014322 051105 050122 020103
014330 020040 042527 042522
014336 020040 020040 040440
820 014344 042522 000040
014350 005015 047115 042103 ADROUT: .ASCIZ <15><12>/MNCDO (DIGITAL OUT) BASE ADDRESS </
014356 020117 042050 043511
014364 052111 046101 047440
014372 052125 020051 040502
014400 042523 040440 042104
014406 042522 051523 036040
821 014414      000
014415      015 046412 041516 VECOUT: .ASCIZ <15><12>/MNCDO (DIGITAL OUT) VECTOR ADDRESS </
014422 047504 024040 044504
014430 044507 040524 020114
014436 052517 024524 053040
014444 041505 047524 020122
014452 042101 051104 051505
822 014460 020123 000074
014464 020076 020077      000
823 014471      015      012 ENDOUT: .ASCIZ /> ? /
824 014473      115 041516 047504 MSGDWF: .BYTE 15,12
014500 024040 044504 044507 .ASCII \MNCDO (DIGITAL OUT) TEST MODULE MUST BE CONNECTED TO THE MNCDO MODULE\
014506 040524 020114 052517
014514 024524 052040 051505
014522 020124 047515 052504
014530 042514 046440 051525
014536 020124 042502 041440
014544 047117 042516 052103
014552 042105 052040 020117
014560 044124 020105 047115
014566 042103 020117 047515
```


CVMNE-A MNCDO
CVMNEA.P11

DIAGNOSTIC
ASCII MESSAGES

MACY11 27(654) 19-SEP-78 09:02 ^{E 5} PAGE 21-4

SEQ 0056

825	014574	052504	042514						
	014600	015	012	000					
826	014603	015	012	007	MSGCAB:	.BYTE	15,12,0		
827	014606	047115	042103	020117		.BYTE	15,12,7		
	014614	052515	052123	041040		.ASCII	/MNCDO MUST BE CONNECTED TO THE TESTER/		
	014622	020105	047503	047116					
	014630	041505	042524	020104					
	014636	047524	052040	042510					
	014644	052040	051505	042524					
	014652	122							
828	014653	015	012	000	ADASH:	.BYTE	15,12,0		
829	014656	026440	000040			.ASCII	/ - /		
830	014662	015	012	000	TCRLF:	.BYTE	15,12,0		
831		014666				.EVEN			
832									
833	014666	001446	001116	001344	DT1:	UNITBD,\$ERRPC,OCSR,DOR,0			
	014674	001350	000000						
834	014700	001446	001116	001350	DT2:	UNITBD,\$ERRPC,DOR,\$GDDAT,\$BDDAT,0			
	014706	001124	001126	000000					
835	014714	001446	001116	001344	DT3:	UNITBD,\$ERRPC,OCSR,\$GDDAT,\$BDDAT,0			
	014722	001124	001126	000000					
836	014730	001446	001116	001344	DT4:	UNITBD,\$ERRPC,OCSR,0			
	014736	000000							
837	014740	001446	001116	012356	DT6:	UNITBD,\$ERRPC,TRTO,TRFRO,0			
	014746	012360	000000						
838	014752	001446	001116	001362	DT7:	UNITBD,\$ERRPC,TEMP,\$UNIT,0			
	014760	001202	000000						
839	014764	000	000	000	DF0:	.BYTE 0,0,0,0,0,0,0,0,0,0			
	014767	000	000	000					
	014772	000	000	000					
	014775	000							
840									
841		000001				.END			

.SWRLO	21#		
.SACT1	13#	57	
.SAPT8	13#	60#	
.SAPTH	13#	59	
.SAPTY	13#	674	
.SCATC	10#		
.SCMTA	10#	60	
.SEOP	10#	586	
.SERRO	10#	669	
.SERRT	12#	670	
.SPARM	11#		
.SPOWE	11#	675	
.SRDOC	12#	679	
.SREAD	11#	651	
.SSAVE	11#		
.SSCOP	11#	652	
.SSPAC	11#		
.SSWDO	11#		
.STRAP	11#	685	
.STYP8	12#	682	
.STYPD	12#	672	
.STYPE	10#	11#	681
.STYPO	10#	671	

F 6
PAGE 21-18

MACY11 27(654) 19-SEP-78 09:02

CVMNE-A MNCDO
CVMNEA.P11

SEQ 0070

	DIAGNOSTIC CROSS REFERENCE	MACY11	27(654)	19-SEP-78	09:02	654	655	656	657	658	659	660	661	662	663
.IFTF	204	651	652	669	679										
.IIF	15	21	60	197	204	586	651	652	669	670	681	685	766	768	
.IRP	188	354	364	377	379	380	381	383	384	385	387	389	398	407	417
	457	494	503	516	547	566	652	669	672	674	675	679			
.LIST	5	14	19	21	39	60	188	197	204	352	354	364	377	379	380
	381	383	384	385	387	389	398	407	417	457	494	503	516	547	566
	586	606	651	652	669	685									
.MACRO	21	60	77	97	109	123	133	197	655	685					
.MCALL	10	11	12	13	19	60	197	204							
.MEXIT	60														
.NLIST	1	6	19	21	31	60	188	197	204	350	354	364	377	379	380
	381	383	384	385	387	389	398	407	417	457	494	503	516	547	566
	586	603	651	652	669	685									
.PAGE	20	60													
.REPT	33	315													
.SBTTL	19	21	23	57	59	60	197	204	213	252	351	354	364	377	379
	380	381	383	384	385	387	389	398	407	417	457	494	503	516	547
	566	586	604	605	619	635	651	652	669	670	671	672	674	675	679
	681	682	685	780											
.TITLE	15														
.WORD	32	38	41	43	44	57	59	60	586	651	670	671	674	675	679
	681	685	777	778											

ERRORS DETECTED: 0

CVMNE-A MNCDO DIAGNOSTIC
CVMNEA.P11

MACY11 27(654) 19-SEP-78 09:02 ^{G 6} PAGE 21-19

SEQ 0071

*CVMNEA,CVMNEA/CRF=CVMNEA
RUN-TIME: 23 10 1 SECONDS
CORE USED: 25K