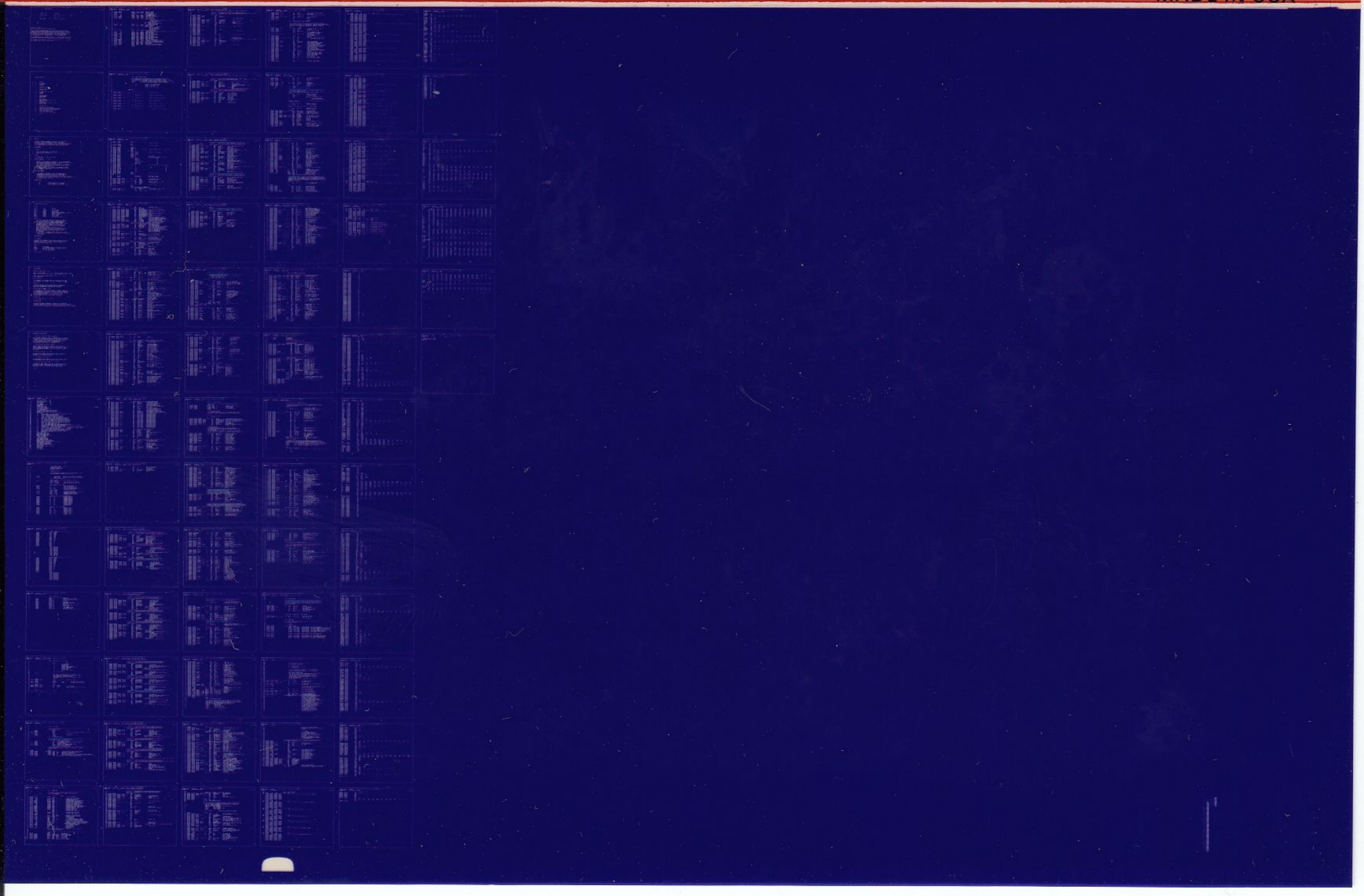
MINC-11

MNCD DIAGNOSTIC
CVMNEA0

AH-B098A-MC

FICHE 1 OF 1

DEC 1978
digital
MADE IN USA



IDENTIFICATION

PRODUCT CODE: AC-B097A-MC

PRODUCT NAME: CVMNEA0 MNCDO (DIGITAL OUT) DIAGNOSTIC TEST

DATE: AUGUST 1978

MAINTAINER: DIAGNOSTIC ENGINEERING

TABLE OF CONTENTS

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

SEQ 0004

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

SEQ 0005

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

SEQ 0006

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 CONTOLS 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

```
 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)
```

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

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

```
                                        .SBTTL  OPERATIONAL SWITCH SETTINGS
(1)                                     ;*
(1)                                     ;*      SWITCH                  USE
(1)                                     ;*      ------          --------------------
(1)                                     ;*        15            HALT ON ERROR
(1)                                     ;*        14            LOOP ON TEST
(1)                                     ;*        13            INHIBIT ERROR TYPEOUTS
(1)                                     ;*        12            INHIBIT SIZING THE # OF MNCDO'S
(1)                                     ;*        11            INHIBIT ITERATIONS
(1)                                     ;*         9            LOOP ON ERROR
(1)                                     ;*         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
000004  012150  000200                  .WORD   IOTRD,200       ;HANDLE BUSS ERROR.
                                        .=174
000174  000000                          DISPREG:        .WORD   0       ;SOFTWARE DISPLAY REGISTER
000176  000000                          SWREG:          .WORD   0       ;SOFTWARE SWITCH REGISTER

                                        .=200
000200  000137  001450                  JMP     BEGIN
000204  000137  001476                  JMP     RESTRT          ;RESTART ADDRESS
000210  000137  001462                  JMP     TESTER          ;INDICATE MNCDO IN-HOUSE TESTER MODE


                                        .=100
000100  000104  000200  000002          104,200,RTI             ;LSI-11 ''B EVENT'' PROTECTION
```

```
                                .SBTTL  ACT11 HOOKS
                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
000106                                  $SVPC=.                 ;SAVE PC
000046                                  .=46
000046  005514                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052                                  .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
000106                                  .=$SVPC                 ;; RESTORE PC
001000                                  .=1000
                                .SBTTL  APT PARAMETER BLOCK
                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
001000                                  .$X=.   ;;SAVE CURRENT LOCATION
000024                                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
000044                                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
001000                                  .=.$X   ;;RESET LOCATION COUNTER
                                ;;******************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001170                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000030                  $TSTM:  .WORD   30      ;;RUN TIM OF LONGEST TEST
001006  000030                  $PASTM: .WORD   30      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000030                  $UNITM: .WORD   30      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000032                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

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

```
(2)  001204  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(2)  001206  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001210                        $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001210     000                $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001211     000                $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001212  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001214  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001216  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                ;*                      BITS 15-11=CPU TYPE
(2)                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                                ;*                              11/70=06,PDQ=07,Q=10
(2)                                ;*                      BIT 10=REAL TIME CLOCK
(2)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001220     000                $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001221     000                $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                ;*                              900 NSEC CORE=001
(2)                                ;*                              300 NSEC BIPOLAR=002
(2)                                ;*                              500 NSEC MOS=003
(2)  001222  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE'' ABOVE
(2)  001224     000                $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001225     000                $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001226  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001230     000                $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001231     000                $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001232  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001234     000                $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001235     000                $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001236  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001240  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                $DEVM:  .WORD   ADEVM   ;;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)
(1)
(1)  001254                             $ERRTB:
 61
 62                                     ;ITEM   1
 63  001254  013074  014116  014666             EM1,DH1,DT1,DF0         ;MNCDO BUS ERROR
     001262  014764
 64                                     ;ITEM   2
 65  001264  013203  014147  014700             EM2,DH2,DT2,DF0         ;MNCDO DATA REGISTER ERROR
     001272  014764
 66                                     ;ITEM   3
 67  001274  013253  014203  014714             EM3,DH3,DT3,DF0         ;MNCDO STATUS REGISTER ERROR
     001302  014764
 68                                     ;ITEM   4
 69  001304  013325  014237  014730             EM4,DH4,DT4,DF0         ;MNCDO INTERRUPT ERROR
     001312  014764
 70                                     ;ITEM   5
 71  001314  013526  014203  014714             EM14,DH3,DT3,DF0        ;MNCDO INCORRECT I.D. VALUE
     001322  014764
 72                                     ;ITEM   6
 73  001324  013371  014261  014740             EM6,DH6,DT6,DF0         ;MNCDO ILLEGAL INTR. OR TRAP
     001332  014764
 74                                     ;ITEM   7
 75  001334  013442  014315  014752             EM7,DH7,DT7,DF0         ;EXISTING MNCD 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                       ;DEVICE MAP
   164  001400  000004                    VADDR:  4                       ;MULTIPLE UNITS,ADDRESS D'FFERENCE
   165  001402  000340                    VECLST: AVECT1                  ;VECTOR OF MNCDO #0
   166  001404  000344                            AVECT1+4                ;                #1
   167  001406  000350                            AVECT1+10               ;                #2
   168  001410  000354                            AVECT1+14               ;                #3
   169  001412  000530                            AVECT1+170              ;                #4
   170  001414  000520                            AVECT1+160              ;                #5
   171  001416  000510                            AVECT1+150              ;                #6
   172  001420  000500                            AVECT1+140              ;                #7
   173  001422  000000                    VECOFF: 0                       ;OFFSET FROM $VECT1 TO UNIT #0 VECTOR
   174  001424  000004                            4                       ;                              #1
   175  001426  000010                            10                      ;                              #2
   176  001430  000014                            14                      ;                              #3
   177  001432  000130                            130                     ;                              #4
   178  001434  000120                            120                     ;                              #5
   179  001436  000110                            110                     ;                              #6
   180  001440  000100                            100                     ;                              #7
   181  001442  000000                    EVER:   0
   182  001444  000000                    BADUNT: 0                       ;BAD UNIT INDICATOR
   183  001446  000000                    UNITBD: 0
   184
   185          010000                            BITDAT=BIT12            ;MAINT INPUT INHIBIT
   186          004000                            BITEXT=BIT11            ;INPUT MAINT STROBE
   187
   189  001450  005037  001364            BEGIN:  CLR     TEMP1           ;CLEAR RESTART INDICATOR
   190  001454  005037  001360                    CLR     DWARF           ;INDICATE NO DWARF
   191  001460  000410                            BR      RBEG0
   192  001462  012737  000001  001360    TESTER: MOV     #1,DWARF        ;INDICATE TESTER
   193  001470  005037  001364                    CLR     TEMP1           ;CLEAR RESTART INDICATOR
   194  001474  000402                            BR      RBEG0
   195  001476  005237  001364            RESTRT: INC     TEMP1           ;INDICATE RESTART
   196
   197  001502                            RBEG0:
   (1)                                    .SBTTL  INITIALIZE THE COMMON TAGS
   (1)                                    ;;CLEAR THE COMMON TAGS ($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     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1)  001620  112737  000001  001115           MOVB    #1,$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  ;;RESTORE 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$:     .ASCIZ  <CRLF>#CVMNE-A  MNCDO (DIGIAL OUT) DIAGNOSTIC#<CRLF>
    (1)  002142                        68$:
    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$:     TST     TEMP1            ;TEST IF RESTART
    209  002160  001007                          BNE     MTEST1           ;BR IF YES
    210  002162  005737  001360                  TST     DWARF            ;TEST IF NO DWARF
    211  002166  001402                          BEQ     MTEST            ;BR IF NONE
    212  002170  104401  014603                  TYPE,   MSGCAB           ;TELL OPERATOR ABOUT TESTER CABLE
    213                                .SBTTL  INFORM THE OPER. THE TESTS AVAILABLE
    214  002174  104401  012362          MTEST:  TYPE,   PRIMEO           ;INFORM THEM OF THE DIFFERENT TESTS
    215  002200  000005                  MTEST1: RESET                    ;HIT IT
    216  002202  052777  000100  176734          BIS     #BIT6,@$TKS      ;ENABLE TKS INTR.
    217  002210  005037  001176                  CLR     $PASS            ;INIT THE PASS COUNTER
    218  002214  005037  001112                  CLR     $ERTTL           ;INIT THE TOTAL ERROR COUNT
    219  002220  005037  001442                  CLR     EVER             ;INIT THE UNIT TYPEOUT FLAG
    220  002224  004737  002712                  JSR     PC,FIXADR        ;ENSURE CORRECT BASE/VECTOR ADDRESS IS LOADED
    221  002230  104401  013004                  TYPE,   DOT              ;INDICATE THE REST POINT
    222  002234  104412                          RDLIN                    ;GET OPER. INPUT
    223  002236  013637  002416                  MOV     @(SP)+,RUNIT     ;GET 1ST CHARACTER
    224  002242  142737  000040  002416          BICB    #40,RUNIT        ;ENSURE UPPER CASE
    225  002250  122737  000102  002416          CMPB    #'B,RUNIT        ;TEST FOR 'B'
    226  002256  001002                          BNE     1$               ;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     LOGIC0           ;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
```

```
                                                    ;TYPE ''?''
                                                    ;RETYPE DOT
                                                    0

                                                    .SBTTL  DETERMINE THE NUMBER OF MNCDO'S ON THE SYSTEM
                                                    ;TELL OPER. ABOUT CONNECTING DWARF
                                                    ;GET BASE ADDRESS
                                                    MASKNM
                                                    ;CLEAR UNIT NUMBER
                                                    ;LOAD RETURN ADDRESS
                                                    ;TEST IF ADDRESS EXISTS
                                                    ;UPDATE BUS ADDRESS
                                                    ;UPDATE UNIT COUNT
                                                    ;TEST IF ''DO NOT SIZE''
                                                    ;BR IF NO SIZEING
                                                    ;TEST IF SW 12 IS SET
                                                    ;BR IF INHIBIT SIZING SWITCH IS SET
                                                    ;TEST IF MAX NUMBER
                                                    ;BR IF NOT
                                                    ;BR IF MAX
                                                    ;RESTORE STACK
                                                    ;TEST IF ANY EXIST
                                                    ;BR IF ANY ARE THERE
                                                    ;TEST IF XXDP CHAIN MODE
                                                    ;BR IF YES
                                                    ;BASE ADDRESS CAUSED A BUS TRAP
                                                    $EOP
                                                    ;RESTORE ERROR VECTOR
                                                    #200,ERRVEC+2
                                                    ;TEST IF # HAS BEEN REPORTED
                                                    ;BR IF IT HAS
                                                    ;TEST IF TESTER MODE
                                                    ;BR IF TESTER
                                                    ;TELL OPERATOR # OF MNCDO FOUND
                                                    $UNIT,-(SP)
                                                    TYPDS
                                                    ,FOUND2
                                                    ;TEST IF ANY UNITS
                                                    ;BR IF SOME
                                                    ;REPORT EOP
                                                    ;SAVE THE # OF MNCDO'S FOR LATER
                                                    ;SET ''REPORTED NUMBER FLAG''
                                                    5$
                                                    ;TEST IF ANY HAVE GONE AWAY
                                                    ;BR IF ALL ARE STILL THERE
                                                    ;SAVE FOR ERROR REPORT
                                                    ;EXISTING UNIT FAILED TO RESPOND
                                                    ;RESET UNIT POINTER
                                                    ;LOAD BUS ADDRESSES
                                                    ;LOAD DEVICE MASK
                                                    ;RESET BAD UNIT INDICATOR
                                                    -(SP)
                                                    #TST1,-(SP)
                                                    ;LOWER CPU PRIORITY
```

```
                                                ;SUBROUTINE TO FIX DEVICE ADDRESS AND VECTORS
002712  012700  001344          FIXADR: MOV     #OCSR,R0        ;LOAD ADDRESS POINTER
002716  013701  001244                  MOV     $BASE,R1        ;LOAD INITIAL BUS ADDRESS
002722  010120                  1$:     MOV     R1,(R0)+        ;LOAD DEVICE ADDRESS
002724  005201                          INC     R1              ;UPDATE BUS ADDRESS VALUE
002726  020027  001354                  CMP     R0,#DOR1+2      ;TEST IF DONE WITH BUS ADDRESSES
002732  001373                          BNE     1$              ;BR IF NOT
002734  013701  001240                  MOV     $VECT1,R1       ;LOAD VECTOR POINTER
002740  010120                          MOV     R1,(R0)+        ;LOAD DEVICE VECTOR ADDRESS
002742  005721                          TST     (R1)+           ;INC FOR STATUS WORD
002744  010110                          MOV     R1,(R0)         ;LOAD ADDRESS OF STATUS WORD
002746  012700  001402                  MOV     #VECLST,R0      ;GET POINTER TO ACTUAL VECTOR AREA
002752  012701  001422                  MOV     #VECOFF,R1      ;GET POINTER TO VECTOR OFFSET VALUES
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

                                        ;ALSO TO LOAD INTELLIGENT TRAP CATCHER
003036  012700  000250                  MOV     #250,R0         ;LOAD R0
003042  012701  000252                  MOV     #252,R1         ;LOAD R1
003046  012702  004700                  MOV     #4700,R2        ;LOAD ILLEGAL INSTR. INTO R2
003052  010120                  4$:     MOV     R1,(R0)+        ;LOAD .+2
003054  010220                          MOV     R2,(R0)+        ;LOAD ILLEGAL INSTR.
003056  022121                          CMP     (R1)+,(R1)+     ;BUMP R1
003060  020027  001000                  CMP     R0,#1000        ;DONE?
003064  001372                          BNE     4$              ;NO-BR
003066  000207                          RTS     PC              ;RETURN

                                        ;ROUTINE TO ASK THE OPERATOR FOR ADRS. & VECTOR INFORMATION
003070  104401  014350          BASEXC: TYPE    ,ADROUT         ;ASK FOR OUTPUT ADDR.
003074  013746  001244                  MOV     $BASE,-(SP)     ;LOAD DEFAUT ONTO STACK
003100  104402                          TYPOC
003102  104401  014464                  TYPE    ,ENDOUT         ;CONTINUE MESSAGE
003106  104413                          RDOCT                   ;AND WAIT FOR INPUT
003110  005726                          TST     (SP)+           ;DOES THE OPERATOR WANT THE DEFAULT
003112  001403                          BEQ     3$              ;YES-BRANCH
003114  016637  177776  001244          MOV     -2(SP),$BASE    ;SAVE ANSWER
003122  104401  014415          3$:     TYPE    ,VECOUT         ;ASK FOR OUTPUT VECTOR
003126  013746  001240                  MOV     $VECT1,-(SP)    ;LOAD DEFAUT ONTO STACK
003132  104402                          TYPOC
003134  104401  014464                  TYPE    ,ENDOUT         ;CONTINUE MESSAGE
```

```
   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
```

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

```
379                                     ;;*****************************************************************
(4)                                     ;*TEST 4        FLOAT A 0 ACROSS THE MNCDO DATA REGISTER
(4)                                     ;;*****************************************************************
(3)  003372  000004                     TST4:   SCOPE
(1)  003374  012737  000001  001362             MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
(1)  003402  012737  003410  001106             MOV     #1$,$LPADR              ;LOAD LOOP ADDRESS
(1)  003410  013737  001362  001124  1$:        MOV     TEMP,$GDDAT             ;LOAD EXPECTED
(1)  003416  005137  001124                     COM     $GDDAT                  ;COMPLEMENT
(1)  003422  013777  001124  175720             MOV     $GDDAT,@DOR             ;LOAD MNCDO DATA REGISTER
(1)  003430  017737  175714  001126             MOV     @DOR,$BDDAT             ;READ MNCDO DATA REGISTER
(1)  003436  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
(2)  003444  001401                             BEQ     2$              ;;BR IF EXPECTED
(1)  003446  104002                             ERROR   2               ;MNCDO DATA REGISTER FAILED TO HOLD A FLOATING 0
(1)  003450  006337  001362             2$:     ASL     TEMP                    ;CHANGE THE DATA
(1)  003454  001355                             BNE     1$                      ;BR IF MORE DATA
380                                     ;;*****************************************************************
(4)                                     ;*TEST 5        ENSURE THAT 'RESET' CLEARS THE MNCDO DATA REGISTER
(4)                                     ;;*****************************************************************
(3)  003456  000004                     TST5:   SCOPE
(2)  003460  012737  000040  001160             MOV     #40,$TIMES      ;;DO 40 ITERATIONS
(1)  003466  012777  177777  175654             MOV     #-1,@DOR                ;LOAD BITS TO BE RESET
(1)  003474  005037  001124                     CLR     $GDDAT                  ;CLEAR EXPECTED
(1)  003500  000005                             RESET                           ;CLEAR THE DEVICE
(1)  003502  052777  000100  175434             BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
(1)  003510  017737  175634  001126             MOV     @DOR,$BDDAT             ;READ MNCDO DATA REGISTER
(3)  003516  001401                             BEQ     TST6            ;;BR IF CLEARED
(1)  003520  104002                             ERROR   2               ;MNCDO DATA REGISTER FAILED TO CLEAR WITH 'RESET'
381                                     ;;*****************************************************************
(4)                                     ;*TEST 6        VERIFY BYTE OPERATION ON THE MNCDO DATA REGISTER
(4)                                     ;;*****************************************************************
(3)  003522  000004                     TST6:   SCOPE
(1)  003524  012737  003532  001106             MOV     #1$,$LPADR              ;LOAD RETURN ADDRESS
(1)  003532  012777  177777  175610  1$:        MOV     #-1,@DOR                ;LOAD MNCDO DATA REGISTER
(1)  003540  012737  000377  001124             MOV     #377,$GDDAT             ;LOAD EXPECTED
(1)  003546  105077  175600                     CLRB    @DOR1                   ;CLEAR HIGH BYTE
(1)  003552  017737  175572  001126             MOV     @DOR,$BDDAT             ;READ MNCDO DATA REGISTER
(1)  003560  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
(2)  003566  001404                             BEQ     2$              ;;BR IF SAME
(1)  003570  104002                             ERROR   2               ;CLEARING HIGH BYTE CHANGED LOW BYTE
(1)  003572  012737  003600  001106             MOV     #2$,$LPADR      ;LOAD LOOP RETURN
(1)  003600  012777  177777  175542  2$:        MOV     #-1,@DOR                ;LOAD MNCDO DATA REGISTER
(1)  003606  012737  177400  001124             MOV     #177400,$GDDAT          ;LOAD EXPECTED
(1)  003614  105077  175530                     CLRB    @DOR                    ;CLEAR LOW BYTE
(1)  003620  017737  175524  001126             MOV     @DOR,$BDDAT             ;READ MNCDO DATA REGISTER
(1)  003626  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;COMPARE
(2)  003634  001401                             BEQ     3$              ;;BR IF SAME
(1)  003636  104002                             ERROR   2               ;CLEARING LOW BYTE CHANGED HIGH BYTE
(1)  003640                             3$:
```

```
383                                     ;;*****************************************************************
(4)                                     ;*TEST 7        TEST THAT BIT6 OF MNCDO STATUS REGISTER IS READ-WRITE
(4)                                     ;;*****************************************************************
(3)  003640  000004                     TST7:   SCOPE
(1)  003642  012737  000100  001124             MOV     #BIT6,$GDDAT            ;LOAD EXPECTED
(1)  003650  013777  001124  175466             MOV     $GDDAT,@OCSR            ;LOAD BIT6 INTO MNCDO STATUS REGISTER
(1)  003656  017737  175462  001126             MOV     @OCSR,$BDDAT            ;READ MNCDO STATUS REGISTER
(1)  003664  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  003672  001401                             BEQ     1$              ;;BR IF SET
(1)  003674  104003                             ERROR   3               ;BIT6 OF MNCDO STATUS REGISTER FAILED TO SET
(1)
(1)  003676  043777  001124  175440  1$:      BIC     $GDDAT,@OCSR            ;CLEAR THAT BIT
(1)  003704  017737  175434  001126             MOV     @OCSR,$BDDAT            ;READ MNCDO STATUS REGISTER AGAIN
(1)  003712  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  003720  001001                             BNE     TST10           ;;BR IF CLEARED
(1)  003722  104003                             ERROR   3               ;BIT6 OF MNCDO STATUS REGISTER FAILED TO CLEAR
384                                     ;;*****************************************************************
(4)                                     ;*TEST 10       TEST THAT BIT4 OF MNCDO STATUS REGISTER IS READ-WRITE
(4)                                     ;;*****************************************************************
(3)  003724  000004                     TST10:  SCOPE
(1)  003726  012737  000020  001124             MOV     #BIT4,$GDDAT            ;LOAD EXPECTED
(1)  003734  013777  001124  175402             MOV     $GDDAT,@OCSR            ;LOAD BIT4 INTO MNCDO STATUS REGISTER
(1)  003742  017737  175376  001126             MOV     @OCSR,$BDDAT            ;READ MNCDO STATUS REGISTER
(1)  003750  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  003756  001401                             BEQ     1$              ;;BR IF SET
(1)  003760  104003                             ERROR   3               ;BIT4 OF MNCDO STATUS REGISTER FAILED TO SET
(1)
(1)  003762  043777  001124  175354  1$:      BIC     $GDDAT,@OCSR            ;CLEAR THAT BIT
(1)  003770  017737  175350  001126             MOV     @OCSR,$BDDAT            ;READ MNCDO STATUS REGISTER AGAIN
(1)  003776  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004004  001001                             BNE     TST11           ;;BR IF CLEARED
(1)  004006  104003                             ERROR   3               ;BIT4 OF MNCDO STATUS REGISTER FAILED TO CLEAR
385                                     ;;*****************************************************************
(4)                                     ;*TEST 11       TEST THAT BIT3 OF MNCDO STATUS REGISTER IS READ-WRITE
(4)                                     ;;*****************************************************************
(3)  004010  000004                     TST11:  SCOPE
(1)  004012  012737  000010  001124             MOV     #BIT3,$GDDAT            ;LOAD EXPECTED
(1)  004020  013777  001124  175316             MOV     $GDDAT,@OCSR            ;LOAD BIT3 INTO MNCDO STATUS REGISTER
(1)  004026  017737  175312  001126             MOV     @OCSR,$BDDAT            ;READ MNCDO STATUS REGISTER
(1)  004034  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THAT IT SET
(2)  004042  001401                             BEQ     1$              ;;BR IF SET
(1)  004044  104003                             ERROR   3               ;BIT3 OF MNCDO STATUS REGISTER FAILED TO SET
(1)
(1)  004046  043777  001124  175270  1$:      BIC     $GDDAT,@OCSR            ;CLEAR THAT BIT
(1)  004054  017737  175264  001126             MOV     @OCSR,$BDDAT            ;READ MNCDO STATUS REGISTER AGAIN
(1)  004062  023737  001124  001126             CMP     $GDDAT,$BDDAT           ;TEST THE BIT
(3)  004070  001001                             BNE     TST12           ;;BR IF CLEARED
(1)  004072  104003                             ERROR   3               ;BIT3 OF MNCDO STATUS REGISTER FAILED TO CLEAR
```

```
                                        ;;*****************************************************************
                                        ;*TEST 12       ENSURE THAT 'RESET'' CLEARS THE MNCDO STATUS REGISTER
                                        ;;*****************************************************************
004074  000004                  TST12:  SCOPE
004076  012737  000040  001160          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
004104  012777  000130  175232          MOV     #130,@OCSR              ;LOAD BITS TO BE RESET
004112  005037  C01124                  CLR     $GDDAT                  ;CLEAR EXPECTED
004116  000005                          RESET                           ;CLEAR THE DEVICE
004120  052777  000100  175016          BIS     #BIT6,@$TKS             ;ENABLE TKB INTR.
004126  017737  175212  001126          MOV     @OCSR,$BDDAT            ;READ MNCDO STATUS REGISTER
004134  001401                          BEQ     TST13           ;;BR IF CLEARED
004136  104003                          ERROR   3               ;MNCDO STATUS REGISTER FAILED TO CLEAR WITH 'RESET''

                                        ;;*****************************************************************
                                        ;*TEST 13       VERIFY THAT MNCDO DONE FLAG SETS
                                        ;;*****************************************************************
004140  000004                  TST13:  SCOPE
004142  005077  175176                  CLR     @OCSR                   ;CLEAR CLEARED FLAG
004146  012737  000200  001124          MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
004154  105077  175170                  CLRB    @DOR                    ;ENABLE
004160  112777  000001  175160          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
004166  017737  175152  001126          MOV     @OCSR,$BDDAT            ;READ OUTPUT STATUS REGISTER
004174  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;COMMPARE
004202  001401                          BEQ     TST14           ;;BR IF EXPECTED
004204  104003                          ERROR   3                       ;OUTPUT DONE FLAG FAILED TO SET
                                        ;;*****************************************************************
                                        ;*TEST 14       VERIFY THAT MNCDO DONE FLAG CLEARS WHEN WRITTEN TO A 0
                                        ;;*****************************************************************
004206  000004                  TST14:  SCOPE
004210  105077  175134                  CLRB    @DOR                    ;ENABLE
004214  112777  000001  17>124          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
004222  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
004226  005077  175112                  CLR     @OCSR                   ;CLEAR OUTPUT DONE FLAG
004232  017737  175106  001126          MOV     @OCSR,$BDDAT            ;READ STATUS
004240  001401                          BEQ     TST15           ;;BR IF CLEARED
004242  104003                          ERROR   3                       ;WRITING OUTPUT FLAG TO A ZERO FAILED TO CLEAR O

                                        ;;*****************************************************************
                                        ;*TEST 15       VERIFY THAT MNCDO DONE FLAG CLEARS WHEN OUTPUT DATA REGISTER IS WRITTEN
                                        ;;*****************************************************************
004244  000004                  TST15:  SCOPE
004246  105077  175076                  CLRB    @DOR                    ;ENABLE
004252  112777  000001  175066          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
004260  005037  001124                  CLR     $GDDAT                  ;CLEAR EXPECTED
004264  105077  175060                  CLRB    @DOR                    ;WRITE THE OUTPUT DATA REGISTER
004270  017737  175050  001126          MOV     @OCSR,$BDDAT            ;READ OUTPUT STATUS REGISTER
004276  001401                          BEQ     TST16           ;;BR IF CLEARED
004300  104003                          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                    CLRB    @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 VECOOR
 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
```

```
                                ;;*****************************************************************
                                ;*TEST 17        INTERRUPT TEST -- VERIFY THAT REMOVING OUTPUT IRQ ENABLE WILL REMOVE INT
                                ;;*****************************************************************
004474  000004                TST17:  SCOPE
004476  012746  000200                MOV     #200,-(SP)
004502  012746  004510                MOV     #10$,-(SP)
004506  000002                        RTI                             ;LSI-11 RAISE PRIORITY
004510  000240                10$:    NOP
004512  012777  004610  174634        MOV     #1$,@DODINV             ;LOAD INTR. VECTOR
004520  012777  000200  174630        MOV     #200,@DODINS            ;
004526  012777  000100  174610        MOV     #BIT6,@OCSR             ;ENABLE INTERRUPT
004534  105077  174610                CLRB    @DOR                    ;ENABLE
004540  152777  000001  174600        BISB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
004546  000240                        NOP
004550  000240                        NOP
004552  042777  000100  174564        BIC     #BIT6,@OCSR             ;DISABLE INTERRUPT
004560  000240                        NOP
004562  000240                        NOP
004564  005046                        CLR     -(SP)
004566  012746  004574                MOV     #11$,-(SP)
004572  000002                        RTI                             ;LSI-11 LOWER PRIORITY
004574  000240                11$:    NOP
004576  000240                        NOP
004600  000240                        NOP
004602  005077  174536                CLR     @OCSR                   ;CLEAR STATUS REGISTER
004606  000404                        BR      2$              ;;

004610  022626                1$:     CMP     (SP)+,(SP)+             ;CLEAN THE STACK
004612  005077  174526                CLR     @OCSR                   ;CLEAR OUTPUT STATUS
004616  104004                        ERROR   4               ;CLEARING INTERRUPT ENABLE FAILED TO REMOVE INTERRUPT RE
004620  005077  174520        2$:     CLR     @OCSR
004624  013777  001356  174522        MOV     DODINS,@DODINV          ;RESET VECTORS
004632  012777  004700  174516        MOV     #4700,@DODINS
004640  005077  174500                CLR     @OCSR
004644  005046                        CLR     -(SP)
004646  012746  004654                MOV     #3$,-(SP)               ;LOWER PRIORITY
004652  000002                        RTI
004654  000240                3$:     NOP
```

```
                                        ;;********************* ************************************************
                                        ;*TEST 20        VERIFY OUTPUT STROBE GENERATES REPLY    -       MNCDO FIELD TEST MODULE
                                        ;;************************************************************************
                                        TST20:  SCOPE
004656  000004                                 CMP     #2,DWARF                ;TEST IF MNCDO FIELD DWARF MODE
004660  022737  000002  001360                 BNE     TST21           ;;BR IF NOT
004666  001015                                 MOV     #BIT7,$GDDAT            ;LOAD EXPECTED
004670  012737  000200  00112'                 CLR     @DOR                    ;CLEAR OUTPUT AND GENERATE STROBE
004676  005077  174446                         MOV     @OCSR,$BDDAT            ;READ OUTPUT STATUS REG.
004702  017737  174436  001126                 CMP     $GDDAT,$BDDAT           ;COMPARE
004710  023737  001124  001126                 BEQ     TST21           ;;BR IF SAME
004716  001401                                 ERROR   3                       ;MNCDO STROBE FAILED TO GENERATE REPLY
004720  104003
                                        ;;************************************************************************
                                        ;*TEST 21        FLOAT A 1 ACROSS THE OUTPUT DATA REGISTER (FIELD DRARF)
                                        ;;************************************************************************
                                        TST21:  SCOPE
004722  000004
004724  012737  000001  001160                 MOV     #1,$TIMES       ;;DO 1 ITERATION
004732  022737  000002  001360                 CMP     #2,DWARF        ;TEST IF FIELD MODE
004740  001022                                 BNE     TST22           ;;BR IF NOT FIELD
004742  005737  001176                         TST     $PASS           ;TEST IF FIRST PASS
004746  001417                                 BEQ     TST22           ;;BR IF YES
004750  012737  000001  001124                 MOV     #BIT0,$GDDAT    ;LOAD VALUE
004756  012737  100000  001362  1$:     MOV     #BIT15,TEMP     ;LOAD COUNTER
004764  013777  001124  174356                 MOV     $GDDAT,@DOR     ;LOAD OUTPUT REGISTER
004772  005337  001362          2$:     DEC     TEMP            ;DELAY MORE
004776  001375                                 BNE     2$
005000  006337  001124                         ASL     $GDDAT          ;CHANGE THE DATA
005004  001364                                 BNE     1$              ;BR IF NOT DONE
```

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

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

```
565  005300                                REMAIN:
566                                        ;;**********************  ***************************************************
(3)                                        ;*TEST 24        DETERMINE IF MORE MNCDO'S REMAIN TO BE TESTED
(3)                                        ;;**************************************************************************
(2)  005300  000004                        TST24:  SCOPE
(1)  005302  012737  000001  001160               MOV     #1,$TIMES       ;;DO 1 ITERATION
567  005310  005237  001202                      INC     $UNIT           ;UPDATE UNIT #
568  005314  123737  001202  001442              CMPB    $UNIT,EVER      ;TEST IF MORE
569  005322  001440                              BEQ     1$              ;;BR IF NOT
570  005324  063737  001400  001344              ADD     VADDR,OCSR
571  005332  063737  001400  001346              ADD     VADDR,OCSR1
572  005340  063737  001400  001350              ADD     VADDR,DOR
573  005346  063737  001400  001352              ADD     VADDR,DOR1
574  005354  006337  001376                      ASL     MASKNM          ;CHANGE ERROR FLAG BIT
575  005360  004737  007652                      JSR     PC,WHICHU       ;DETERMINE UNIT #
576  005364  013700  001446                      MOV     UNITBD,R0       ;GET IT
577  005370  006300                              ASL     R0              ;MAKE WORD VALUE
578  005372  016037  001402  001354              MOV     VECLST(R0),DODINV       ;GET THIS UNITS VECTOR VALUE
579  005400  013737  001354  001356              MOV     DODINV,DODINS   ;COPY IT AND
580  005406  062737  000002  001356              ADD     #2,DODINS       ;MAKE NEXT ADDRESS
581  005414  005037  001102                      CLR     $TSTNM          ;CLR TST NUMBER
582  005420  000137  003164                      JMP     TST1            ;TEST NEXT UNIT
583  005424  000240                      1$:     NOP
584
```

```
                        .SBTTL  END OF PASS ROUTINE
                        ;;*****************************************************************
                        ;*INCREMENT THE PASS NUMBER ($PASS)
                        ;*TYPE 'END PASS #XXXXX'' (WHERE XXXXX IS A DECIMAL NUMBER)
                        ;*IF THERES A MONITOR GO TO IT
                        ;*IF THERE ISN'T JUMP TO EXTMSG

005426                  $EOP:
005426  000004                    SCOPE
005430  005037  001102            CLR     $TSTNM          ;;ZERO THE TEST NUMBER
005434  005037  001160            CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
005440  005237  001176            INC     $PASS           ;;INCREMENT THE PASS NUMBER
005444  042737  100000  001176    BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
005452  005327                    DEC     (PC)+           ;;LOOP?
005454  000001          $EOPCT:   .WORD   1
005456  003022                    BGT     $DOAGN          ;;YES
005460  012737                    MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
005462  000001          $ENDCT:   .WORD   1
005464  005454                    $EOPCT
005466  104401  005533            TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
005472  013746  001176            MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
005476  104405                    TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
005500  104401  005530            TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
005504  013700  000042  $GET42:   MOV     @#42,R0         ;;GET MONITOR ADDRESS
005510  001405                    BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
005512  000005                    RESET                   ;;CLEAR THE WORLD
005514  004710          $ENDAD:   JSR     PC,(R0)         ;;GO TO MONITOR
005516  000240                    NOP                     ;;SAVE ROOM
005520  000240                    NOP                     ;;FOR
005522  000240                    NOP                     ;;ACT11
005524                  $DOAGN:
005524  000137                    JMP     @(PC)+          ;;RETURN
005526  005550          $RTNAD:   .WORD   EXTMSG
005530     377     377     000  $ENULL:   .BYTE   -1,-1,0      ;;NULL CHARACTER STRING
005533     015  042412  042116  $ENDMG:   .ASCIZ  <15><12>/END PASS #/
005540  050040  051501  020123
005546  000043

005550  052777  000100  173366  EXTMSG:  BIS     #BIT6,@$TKS     ;ENABLE TKB INTR.
005556  005737  001112            TST     $ERTTL          ;ANY ERRORS
005562  001416                    BEQ     1$              ;BR IF NONE
005564  104401  013650            TYPE    ,ERRTOT         ;TYPE TOTAL ERROR MESSAGE
005570  013746  001112            MOV     $ERTTL,-(SP)    ;TOTAL ERRORS
005574  104405                    TYPDS
005576  022737  000001  001376    CMP     #1,MASKNM       ;TEST IF MORE UNITS
005604  001405                    BEQ     1$              ;BR IF NONE
005606  104401  013677            TYPE,   MESGD           ;TYPE BAD UNITS MESSAGE
005612  013746  001444            MOV     BADUNT,-(SP)    ;GET BAD UNITS
005616  104406                    TYPBN                   ;TYPE IT IN BINARY
005620  104401  005530    1$:     TYPE,   $ENULL          ;ENSURE ALL DATA GETS OUTPUTED
005624  004737  005760            JSR     PC,CTRLCG       ;TEST FOR CONTROL C/G
005630  000137  002424            JMP     LOGIC           ;TEST AGAIN
```

```
                                HIGHLO: MOV     #STACK,SP               ;LOAD STACK POINTER
                                        JSR     PC,FIXADR               ;FIX ALL OTHER ADRS.
                                1$:     MOV     #BIT8,R0                ;LOAD COUNTER
                                2$:     CLRB    @DOR1                   ;CLEAR HIGH BYTE
                                        DEC     R0                      ;DELAY
                                        BNE     2$
                                        MOV     #BIT8,R0                ;LOAD COUNTER
                                3$:     CLRB    @DOR                    ;CLEAR LOW BYTE
                                        DEC     R0                      ;DELAY
                                        BNE     3$
                                        JSR     PC,CTRLCG               ;TEST FOR CTRL C/G
                                        BR      1$
                                .SBTTL  MNCDO TEST MODULE LAMP LOOP
                                DOLAMP: MOV     #STACK,SP               ;LOAD STACK POINTER
                                        JSR     PC,FIXADR               ;FIX ALL OTHER ADRS.
                                1$:     MOV     #BIT0,TEMP              ;LOAD INITIAL VALUE
                                2$:     MOV     #2,R0                   ;LOAD DELAY LOOP
                                        CLR     R1                      ;
                                        MOV     TEMP,@DOR               ;LOAD OUTPUT REGISTER LAMPS
                                3$:     DEC     R1                      ;DELAY
                                        BNE     3$
                                        DEC     R0                      ;DELAY
                                        BPL     3$
                                        JSR     PC,CTRLCG
                                        ASL     TEMP                    ;CHANGE THE OUTPUT DATA
                                        BNE     2$
                                        BR      1$                      ;

                                .SBTTL  CONTROL C/G HANDLER

                                CTRLCG: TSTB    @$TKS                   ;TEST FOR FLAG
                                        BPL     2$
                                        MOV     @$TKB,CTRCHA            ;GET CHAR
                                        BIC     #177640,CTRCHA          ;MASK OFF BITS
                                        CMP     #3,CTRCHA               ;TEST FOR CTRL C
                                        BNE     1$                      ;BR IF NOT
                                        TST     (SP)+                   ;CLEAN STACK
                                        JMP     MTEST1                  ;AND REYTPE DOT
                                1$:     CMP     #7,CTRCHA               ;TEST FOR CTRL G
                                        BNE     2$                      ;BR IF NOT
                                        GTSWR                           ;GET SWITCHES
                                2$:     RTS     PC                      ;EXIT
                                CTRCHA: 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                   $TKQSRT: .BLKB  32.             ;;TTY KEYBOARD QUEUE
  (1)          006104                   $TKQEND=.
  (1)
  (1)                                   ;*TK INITIALIZE ROUTINE
  (1)                                   ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
  (1)                                   ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
  (1)                                   ;
  (1)                                   ;*CALL:
  (1)                                   ;*      JSR     PC,$TKINT
  (1)                                   ;*      RETURN
  (1)                                   ;
  (1)  006104  005037  006036           $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
  (1)  006110  012737  006044  006040           MOV     #$TKQSRT,$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    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
  (1)  006176  004737  006104                   JSR     PC,$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)  006226                           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    ,$BELL          ;;RING THE TTY BELL
  (1)  006242  005726                           TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
  (1)  006244  000451                           BR      5$              ;;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     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  006260  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
(1)  006262  105777  172656          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
(1)  006266  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
(1)  006270  117746  172652                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
(1)  006274  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  006300  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  006304  001366                          BNE     31$             ;;BRANCH IF NO
(1)  006306  012777  000100  172630          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
(1)  006314  000002                          RTI                     ;;RETURN
(1)  006316  005237  006036          32$:    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$:     MOVB    (SP)+,@$TKQIN   ;;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     #$TKQSRT,$TKQIN ;;RESET THE POINTER
(1)  006370  000002                  5$:     RTI                     ;;RETURN
(1)
(2)                                  ;;**********************************************************************
(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    @$TKS           ;;IS A CHAR WAITING?
(1)  006406  100121                          BPL     15$             ;;IF NOT, EXIT
(1)  006410  117746  172532                  MOVB    @$TKB,-(SP)     ;;YES
(1)  006414  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
(1)  006420  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
(1)  006424  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
(1)                                                                  ;;AND EXIT
(1)
(2)                                  ;;**********************************************************************
(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,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
(1)  006444  005077  172474                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  006450  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
(1)
(1)  006456  104401  007336                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  006462  104401  007343          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  006466  013746  000176                  MOV     SWREG,-(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)
(1)  006512  117746  172430              MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  006516  042716  177600              BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)  006522  021627  000003              CMP     (SP),#3         ;;IS IT A CONTROL-C?
(1)  006526  001015                      BNE     9$              ;;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)
(1)
(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)
(1)
(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)
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
(1)                                     ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                     ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                     ;
(1)
(1)  006734  011646                     $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
(1)  006736  016666  000004  000002             MOV     4(SP),2(SP)     ;;THE PS
(1)  006744  005066  000004                     CLR     4(SP)           ;;GET READY FOR A CHARACTER
(2)  006750  005046                             CLR     -(SP)           ;;PUT NEW PS ON STACK
(2)  006752  012746  006760                     MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
(2)  006756  000002                             RTI                     ;;POP NEW PC AND PS
(2)  006760                             64$:
(1)  006760  005737  006036             1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
(1)  006764  001775                             BEQ     1$
(1)  006766  005337  006036                     DEC     $TKCNT          ;;DECREMENT THE COUNTER
(1)  006772  117766  177044  000004             MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
(1)  007000  005237  006042                     INC     $TKQOUT         ;;UPDATE THE POINTER
(1)  007004  023727  006042  006104             CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
(1)  007012  001003                             BNE     2$              ;;BRANCH IF NO
(1)  007014  012737  006044  006042             MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
(1)  007022  000002                     2$:     RTI                     ;;RETURN
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                     ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  007024  010346                     $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  007026  005046                             CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  007030  012703  007260             1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  007034  022703  007320             2$:     CMP     #$TTYIN+32.,R3  ;;BUFFER FULL?
(1)  007040  101456                             BLOS    4$              ;;BR IF YES
(1)  007042  104411                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  007044  112613                             MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  007046  122713  000177             10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  007052  001022                             BNE     5$              ;;BR IF NO
(1)  007054  005716                             TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  007056  001007                             BNE     6$              ;;BR IF NO
(1)  007060  112737  000134  007256             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  007066  104401  007256                     TYPE    ,9$
(1)  007072  012716  177777                     MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  007076  005303                     6$:     DEC     R3              ;;BACKUP BY ONE
(1)  007100  020327  007260                     CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  007104  103434                             BLO     4$              ;;BR IF YES
(1)  007106  111337  007256                     MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  007112  104401  007256                     TYPE    ,9$             ;;GO TYPE
(1)  007116  000746                             BR      2$              ;;GO READ ANOTHER CHAR.
(1)  007120  005716                     5$:     TST     (SP)            ;;RUBOUT KEY SET?
```

```
(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    ,$CNTLU         ;;TYPE A CONTROL 'U''
(1)  007152  000726                        BR      1$              ;;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    ,$CRLF          ;;TYPE A ''CR'' & ''LF''
(1)  007170  104401  007260                TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  007174  000717                        BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  007176  104401  001164          4$:   TYPE    ,$QUES          ;;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    ,$LF            ;;TYPE A LINE FEED
(1)  007232  005726                        TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  007234  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  007236  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  007240  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  007246  012766  007260  000004        MOV     #$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
(1)  007354  020040  042516  020127 $MNEW:  .ASCIZ  /  NEW = /
(1)  007362  020075     000
(1)          007366                .EVEN
652                                 .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)                                 ;;**********************************************************************
(1)                                 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)                                 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)                                 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                                 ;*SW14=1        LOOP ON TEST
(1)                                 ;*SW11=1        INHIBIT ITERATIONS
(1)                                 ;*SW09=1        LOOP ON ERROR
(1)                                 ;*SW08=1        LOOP ON TEST IN SWR<7:0>
(1)                                 ;*CALL
(1)                                 ;*      SCOPE           ;;SCOPE=IOT
(1)
```

```
(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)                                          ;#####START OF CODE FOR THE XOR TESTER#####
(1)  007404  000416                          $XTSTR: BR      6$                   ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
(1)                                                                               ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
(1)  007406  013746  000004                          MOV     @#ERRVEC,-(SP)       ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  007412  012737  007432  000004                  MOV     #5$,@#ERRVEC         ;;SET FOR TIMEOUT
(1)  007420  005737  177060                          TST     @#177060             ;;TIME OUT ON XOR?
(1)  007424  012637  000004                          MOV     (SP)+,@#ERRVEC       ;;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)+,@#ERRVEC       ;;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,$TSTNM          ;;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    $TSTNM               ;;COUNT TEST NUMBERS
(1)  007604  113737  001102  001174                  MOVB    $TSTNM,$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     $TSTNM,@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)                                          ;*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)
   (1)  007706                                  $ERROR:
   (1)  007706  104410                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
   (3)  007710  004737  005760                  JSR     PC,CTRLCG       ;TEST FOR CTRL C/G
   (3)  007714  053737  001376  001444          BIS     MASKNM,BADUNT
   (3)  007722  004737  007652                  JSR     PC,WHICHU       ;DETERMINE UNIT #
   (1)  007726  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
   (1)  007732  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
   (1)  007734  013777  001102  171200          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
   (1)  007742  005237  001112                  INC     $ERTTL          ;;INC THE ERROR COUNT
   (1)  007746  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
   (1)  007752  162737  000002  001116          SUB     #2,$ERRPC
   (1)  007760  117737  171132  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
   (1)  007766  032777  020000  171144          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
   (1)  007774  001004                          BNE     20$             ;;SKIP TYPEOUTS
   (1)  007776  004737  010110                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
   (1)  010002  104401  001165                  TYPE    ,$CRLF
   (1)  010006                          20$:
   (1)  010006  122737  000001  001210          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
   (1)  010014  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
   (1)  010016  113737  001114  010030          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
   (1)  010024  004737  010754                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
   (1)  010030     000                  21$:    .BYTE   0
   (1)  010031     000                          .BYTE   0
   (1)  010032  000777                  22$:    BR      22$             ;;APT ERROR LOOP
   (1)  010034  005777  171100          2$:     TST     @SWR            ;;HALT ON ERROR
   (1)  010040  100002                          BPL     3$              ;;SKIP IF CONTINUE
   (1)  010042  000000                          HALT                    ;;HALT ON ERROR!
   (1)  010044  104410                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
   (1)  010046  032777  001000  171064  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
   (1)  010054  001402                          BEQ     4$              ;;BR IF NO
   (1)  010056  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
```

```
(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     $ESCAPE,(SP)     ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  010074                          5$:
(1)  010074  022737  005514  000042          CMP     #$ENDAD,@#42     ;;ACT-11 AUTO-ACCEPT?
(1)  010102  001001                          BNE     6$               ;;BRANCH IF NO
(1)  010104  000000                          HALT                     ;;YES
(1)  010106                          6$:
(1)  010106  000002                          RTI                      ;;RETURN
670                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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)
(1)  010110                          $ERRTYP:
(1)  010110  104401  001165                  TYPE    ,$CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  010114  010046                          MOV     R0,-(SP)         ;;SAVE R0
(1)  010116  005000                          CLR     R0               ;;PICKUP THE ITEM INDEX
(1)  010120  153700  001114                  BISB    @#$ITEMB,R0
(1)  010124  001004                          BNE     1$               ;;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$:     DEC     R0               ;;ADJUST THE INDEX SO THAT IT WILL
(1)  010140  006300                          ASL     R0               ;;      WORK FOR THE ERROR TABLE
(1)  010142  006300                          ASL     R0
(1)  010144  006300                          ASL     R0
(1)  010146  062700  001254                  ADD     #$ERRTB,R0       ;;FORM TABLE POINTER
(1)  010152  012037  010162                  MOV     (R0)+,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$:     MOV     (R0)+,4$         ;;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    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
(1)  010256  000207                            RTS     PC              ;;RETURN
(1)  010260  020040     000            11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
(1)          010264                            .EVEN
671                                     .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)                                     ;;**************************************************************************
(1)                                     ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                                     ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                                     ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPOS                   ;;CALL FOR TYPEOUT
(1)                                     ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                                     ;*      .BYTE   M               ;;M=1 OR 0
(1)                                     ;*                                      ;;1=TYPE LEADING ZEROS
(1)                                     ;*                                      ;;0=SUPPRESS LEADING ZEROS
(1)                                     ;*
(1)                                     ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                                     ;*$TYPOS OR $TYPOC
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPON                   ;;CALL FOR TYPEOUT
(1)                                     ;*
(1)                                     ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)                                     ;*CALL:
(1)                                     ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                     ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)
(1)  010264  017646  000000            $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  010270  116637  000001  010507            MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  010276  112637  010511                    MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  010302  062716  000002                    ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  010306  000406                            BR      $TYPON
(1)  010310  112737  000001  010507    $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  010316  112737  000006  010511            MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  010324  112737  000005  010506    $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  010332  010346                            MOV     R3,-(SP)        ;;SAVE R3
(1)  010334  010446                            MOV     R4,-(SP)        ;;SAVE R4
(1)  010336  010546                            MOV     R5,-(SP)        ;;SAVE R5
(1)  010340  113704  010511                    MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  010344  005404                            NEG     R4
(1)  010346  062704  000006                    ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  010352  110437  010510                    MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  010356  113704  010507                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  010362  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  010366  005003                            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                 MOVB    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)                                  ;*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)  010512                          $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                 1$:    CLR     R0              ;;ZERO THE CONSTANTS INDEX
(1)  010550  012703  010726                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  010554  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  010560  005002                 2$:    CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  010562  016001  010716                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  010566  160105                 3$:    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         6$:    BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  010626  052702  000040         7$:    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                     ;;RETURN TO USER
(1)  010716  023420                 $DTBL: 10000.
(1)  010720  001750                        1000.
(1)  010722  000144                        100.
(1)  010724  000012                        10.
(1)  010726  000004                 $DBLK: .BLKW   4
```

```
                                        .SBTTL  APT COMMUNICATIONS ROUTINE

                                        ;;*****************************************************************************
(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     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  011050  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  011052  010037  001206                  MOV     R0,$MSGLGT      ;;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)                                             ;POWER DOWN ROUTINE
(1)  011204  012737  011350  000024  $PWRDN:    MOV     #$ILLUP,@#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)                                             ;POWER UP ROUTINE
(1)  011256  012737  011350  000024  $PWRUP:    MOV     #$ILLUP,@#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                     $ILLUP: 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                     $RDOCT: MOV     (SP),-(SP)              ;;PROVIDE SPACE FOR THE
(1)  011426  016666  000004  000002             MOV     4(SP),2(SP)             ;;INPUT NUMBER
(3)  011434  010046                             MOV     R0,-(SP)                ;;PUSH R0 ON STACK
(3)  011436  010146                             MOV     R1,-(SP)                ;;PUSH R1 ON STACK
(3)  011440  010246                             MOV     R2,-(SP)                ;;PUSH R2 ON STACK
(1)  011442  104412                     1$:     RDLIN                           ;;READ AN ASCIZ 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  177770                     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$:     TST     (SP)+                   ;;CLEAN TERMINATOR FROM STACK
(1)  011504  010166  000012                     MOV     R1,12(SP)               ;;SAVE THE RESULT
(1)  011510  010237  011524                     MOV     R2,$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 ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)                                     ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)                                     ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)                                     ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)                                     ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)                                     ;*
(1)                                     ;*CALL:
(1)                                     ;*1) USING A TRAP INSTRUCTION
(1)                                     ;*      TYPE    ,MESADR                 ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)                                     ;*OR
(1)                                     ;*      TYPE
(1)                                     ;*      MESADR
```

```
(1)                                     ;*
(1)
(1)  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         $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  011744  100375                        BPL     $TYPEC
(1)  011746  116677  000002  167176        MOVB    2(SP),@$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                  2$:     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
```

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

012064  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
012066  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
012072  005740                          TST     -(R0)           ;;BACKUP BY 2
012074  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
012076  006300                          ASL     R0              ;;POSITION FOR INDEXING
012100  016000  012120                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
012104  000200                          RTS     R0              ;;GO TO ROUTINE


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

012106  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
012110  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
012116  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
012120  012106                  $TRPAD: .WORD   $TRAP2
012122  011526                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
012124  010310                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
012126  010264                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
012130  010324                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
012132  010512                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
012134  012010                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER

012136  006462                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING

012140  006372                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
012142  006734                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
012144  007024                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
012146  011424                          $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
```

```
                                        ;*
                                        ;*THIS ROUTINE WILL PROTECT THE PROGRAM
                                        ;*FROM INTERUPTS (BAD ONES).
                                        ;*
                                        ;*THE TRAP CATCHER IS SET UP FOR
                                        ;*      .WORD   .+2
                                        ;*      JSR PC,R0
                                        ;*
                                        ;*ILLEGAL INTERRUPTS OR INTERRUPTS TO THE WRONG VECTOR
                                        ;*GOTO THE VECTOR AND PICK UP THE ''.+2'' AS AN ADDRESS

                                        ;*AND ''4700'' AS NEW STATUS.
                                        ;*THE .+2 AS A PC WILL CAUSE EXECUTION OF THE ''JSR PC,R0'' (AN ILLEGAL INSTR.).
                                        ;*AND TRAP TO LOCATION ''4''. IN LOCATION 4 WE HAVE A
                                        ;*POINTER HERE. IF THIS CONDITION CAUSES A TRAP TO LOC. 4.
                                        ;*WE WILL REPORT IT IN THE SAME MANNER THAT WE WOULD
                                        ;*REPORT ANY OTHER ERROR.
                                        ;*IF A BUSS ERROR TRAP DID OCCUR AND CAUSE A TRAP TO 4.
                                        ;*WE WILL HALT.

012150  011637  012356          IOTRD:  MOV     (6),TRTO        ;GET WHERE WE CAME TO.
012154  162737  000004  012356          SUB     #4,TRTO         ;FROM READ ADDR.

012162  023727  012356  001000          CMP     TRTO,#1000      ;DID TRAP FROM LESS THAN ADDR. 1000?
012170  003402                          BLE     2$              ;NO-CONTINUE.

012172  000000                  1$:     HALT                    ;A BUSS ERROR TIME OUT TRAP BROUGHT US HERE.
                                                                ;ADDRESS CONTAINED IN TRTO.

012174  000776                          BR      1$              ;DON'T ALLOW CONTINUE.

012176  016637  000004  012360  2$:     MOV     4(6),TRFRO      ;GET TRAPPED FROM ADDR.
012204  062706  000004                  ADD     #4,SP           ;ADJUST THE STACK POINTER

012210  122737  000016  001102          CMPB    #16,$TSTNM      ;LESS THAN INTERRUPT TESTS?
012216  003402                          BLE     3$              ;NO MUST BE WRONG VECTOR.
                                ;$$$$$$$$$$$$$$$$$$$$$$$$>>>ERROR<<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
012220  104006                          ERROR   6               ;ERROR! ILLEGAL INTERRUPT OR
                                                                ;INTERRUPT TO WRONG VECTOR.
                                                                ;IF TEST NO. IS LESS THAN 16,ITS
                                                                ;LIKELY(BUT NO EXCLUSIVELY)TO BE A
                                                                ;DEVICE OTHER THAN THE DEVICE UNDER TEST.
                                                                ;IF THE INTERRUPT OCCURED
                                                                ;DURING AN INTERRUPT TEST, I'D
                                                                ;SUSPECT A PROBLEM WITH THE DEVICE UNDER TEST.
                                                                ;IF THE ADDRESS THE INTERRUPT
                                                                ;VECTORED TO IS WITHIN THE RANGE OF
                                                                ;VECTORS ASSIGNED TO THE DEVICE,
                                                                ;THEN I'D SUSPECT THE DEVICE
                                                                ;INTERRUPTED ILLEGALLY.
                                                                ;IF THE ADDRESS THE INTERRUPT
                                                                ;VECTORED TO IS OUTSIDE OF THE
                                                                ;RANGE ASSIGNED TO THE DEVICE
                                                                ;I'D SUSPECT THAT THE
```

```
                                                        ;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.

                        ;$$$$$$$$$$$$$$$$$$$$$$$$>>>ERROR<<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
012222  000002                    RTI
012224  022626                3$: CMP     (SP)+,(SP)+
012226  005737  001176            TST     $PASS           ;IS THIS THE FIRST PASS?
012232  001025                    BNE     4$              ;;NO, DON'T REPORT
012234  104401  013714            TYPE,   VTMSG           ;TELL OPER.
012240  013746  001202            MOV     $UNIT,-(SP)
012244  104405                    TYPDS
012246  000240                    NOP
012250  000240                    NOP
012252  104401  013750            TYPE    ,VTMSG3         ;TYPE 'EXPECTED INTR. AT ''
012256  013746  001354            MOV     DODINV,-(SP)    ;;SAVE DODINV FOR TYPEOUT
(1) 012262  104403                TYPOS                   ;;GO TYPE--OCTAL ASCII
(1) 012264     003                .BYTE   3               ;;TYPE 3 DIGIT(S)
(1) 012265     001                .BYTE   1               ;;TYPE LEADING ZEROS
012266  104401  014000            TYPE    ,VTMSG1         ;TYPE '' RECEIVED INTR. AT ''
012272  013746  012356            MOV     TRTO,-(SP)      ;;SAVE TRTO FOR TYPEOUT
(1) 012276  104403                TYPOS                   ;;GO TYPE--OCTAL ASCII
(1) 012300     003                .BYTE   3               ;;TYPE 3 DIGIT(S)
(1) 012301     001                .BYTE   1               ;;TYPE LEADING ZEROS
012302  104401  014030            TYPE    ,VTMSG2         ;TYPE 'RESTARTING TEST''
012306  013777  001356  167040 4$: MOV     DODINS,@DODINV  ;ADJUST VECTOR
012314  012777  004700  167034    MOV     #4700,@DODINS
012322  013737  012356  001354    MOV     TRTO,DODINV
012330  042737  000003  001354    BIC     #3,DODINV
012336  013737  001354  001356    MOV     DODINV,DODINS
012344  062737  000002  001356    ADD     #2,DODINS
012352  000177  166530            JMP     @$LPADR         ;START TEST OVER AGAIN.
012356  000000            TRTO:   .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTERRUPTED TO.
012360  000000            TRFRO:  .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTR. FROM.
```

```
                                        .SBTTL  ASCII MESSAGES
012362    015     012                   PRIME0: .BYTE   15,12
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
012440    015     012                         .BYTE   15,12
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
012513    015     012                         .BYTE   15,12
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
012557    015     012                         .BYTE   15,12
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
012615    015     012                         .BYTE   15,12
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
012660    015     012                         .BYTE   15,12
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
012723    015     012                         .BYTE   15,12
012725    110  036440  044040                .ASCIZ  \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
```

```
                                                DOT:    .BYTE   15,12
 795  013004     015     012
 796  013006  054524  042520  052040                    .ASCIZ  /TYPE THE ''TEST CHARACTER'' THEN DEPRESS ''RETURN KEY'' /
      013014  042510  021040  042524
      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  \MNCDO (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  \MNCDO (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  \MNCDO (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  \MNCDO (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  \MNCDO (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  \MNCDO (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
```

```
        014132  052125  051503  004522
        014140  052517  042124  052101
        014146    000
  815   014147    125   044516  004524  DH2:    .ASCIZ  /UNIT   ERRPC   OUTADR  GOOD     BAD/
        014154  051105  050122  004503
        014162  052517  040524  051104
        014170  043411  047517  004504
        014176  041040  042101    000
  816   014203    125   044516  004524  DH3:    .ASCIZ  /UNIT   ERRPC   OUTADR  GOOD     BAD/
        014210  051105  050122  004503
        014216  052517  040524  051104
        014224  043411  047517  004504
        014232  041040  042101    000
  817   014237    125   044516  004524  DH4:    .ASCIZ  /UNIT   ERRPC   OUTADR/
        014244  051105  050122  004503
        014252  052517  040524  051104
        014260    000
  818   014261    125   044516  004524  DH6:    .ASCIZ  /UNIT   ERRPC    TO     FROM ADRS /
        014266  051105  050122  020103
        014274  020040  052040  004517
        014302  051106  046517  040440
        014310  051104  020123    000
  819   014315    125   044516  004524  DH7:    .ASCIZ  /UNIT   ERRPC   WERE     ARE /
        014322  051105  050122  020103
        014330  020040  042527  042522
        014336  020040  020040  040440
        014344  042522  000040
  820   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
        014414    000
  821   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
        014460  020123  000074
  822   014464  020076  020077    000  ENDOUT: .ASCIZ  /> ? /
  823   014471    015     012         MSGDWF: .BYTE   15,12
  824   014473    115   041516  047504        .ASCII  \MNCDO (DIGITAL OUT) TEST MODULE MUST BE CONNECTED TO THE MNCDO MODULE\
        014500  024040  044504  044507
        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
```

```
                014574  052504  042514
825             014600     015     012     000             .BYTE   15,12,0
826             014603     015     012     007     MSGCAB: .BYTE   15,12,7
827             014606  047115  042103  020117             .ASCII  /MNCDO MUST BE CONNECTED TO THE TESTER/
                014614  052515  052123  041040
                014622  020105  047503  047116
                014630  041505  042524  020104
                014636  047524  052040  042510
                014644  052040  051505  042524
                014652     122
828             014653     015     012     000             .BYTE   15,12,0
829             014656  026440  000040             ADASH:  .ASCIZ  / - /
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  DFO:       .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
```

```
ABASE = 171260              17#     60
ACDW1 = 000000              60
ACDW2 = 000000              60
ACPUOP= 000000              60
ADASH   014656             829#
ADDW0 = 000000              60
ADDW1 = 000000              60
ADDW10= 000000              60
ADDW11= 000000              60
ADDW12= 000000              60
ADDW13= 000000              60
ADDW14= 000000              60
ADDW15= 000000              60
ADDW2 = 000000              60
ADDW3 = 000000              60
ADDW4 = 000000              60
ADDW5 = 000000              60
ADDW6 = 000000              60
ADDW7 = 000000              60
ADDW8 = 000000              60
ADDW9 = 000000              60
ADEVCT= 000000              60
ADEVM = 000000              60
ADROUT  014350             332     820#
AENV  = 000000              60
AENVM = 000000              60
AFATAL= 000000              60
AMADR1= 000000              60
AMADR2= 000000              60
AMADR3= 000000              60
AMADR4= 000000              60
AMAMS1= 000000              60
AMAMS2= 000000              60
AMAMS3= 000000              60
AMAMS4= 000000              60
AMSGAD= 000000              60
AMSGLG= 000000              60
AMSGTY= 000000              60
AMTYP1= 000000              60
AMTYP2= 000000              60
AMTYP3= 000000              60
AMTYP4= 000000              60
APASS = 000000              60
APRIOR= 000000              60
APTCSU= 000040             674#    681
APTENV= 000001             669     674#    681
APTSIZ= 000200             197     674#
APTSPO= 000100             674#    681
ASWREG= 000000              60
ATESTN= 000000              60
AUNIT = 000000              60
AUSWR = 000000              60
AVECT1= 000340              18#     60      165     166     167     168     169     170     171     172
AVECT2= 000000              60
```

```
BADUNT  001444          182#    298*    597     669*
BASEXC  003070          227     332#
BEGIN   001450           47     189#    675
BITDAT= 010000          185#
BITEXT= 004000          186#
BIT0  = 000001           19#    297     377     379     393     400     409     422     466     508     552     622
BIT00 = 000001           19#
BIT01 = 000002           19#
BIT02 = 000004           19#
BIT03 = 000010           19#
BIT04 = 000020           19#
BIT05 = 000040           19#
BIT06 = 000100           19#
BIT07 = 000200           19#
BIT08 = 000400           19#    652
BIT09 = 001000           19#    652     669
BIT1  = 000002           19#
BIT10 = 002000           19#
BIT11 = 004000           19#    186     652
BIT12 = 010000           19#    185
BIT13 = 020000           19#    669
BIT14 = 040000           19#    652
BIT15 = 100000           19#    289     509
BIT2  = 000004           19#    519     550
BIT3  = 000010           19#    385
BIT4  = 000020           19#    384
BIT5  = 000040           19#
BIT6  = 000100           19#    216     380     383     387     427     464     469     588
BIT7  = 000200           19#    391     497     524     536
BIT8  = 000400           19#    609     613
BIT9  = 001000           19#
BPTVEC= 000014           19#
CKSWR = 104410          652     669     685#
CR    = 000015           19#    681
CRLF  = 000200           19#    204     681
CTRCHA  006034          639*    640*    641     645     649#
CTRLCG  005760          600     617     630     637#    652     669
DDISP = 177570           19#     60     197
DF0     014764           63      65      67      69      71      73      75     839#
DH1     014116           63     814#
DH2     014147           65     815#
DH3     014203           67      71     816#
DH4     014237           69     817#
DH6     014261           73     818#
DH7     014315           75     819#
DISPLA  001142           60#    197*    652*    669*
DISPRE  000174           43#    197
DODINS  001356          155#    420*    437*    452     453*    463*    485     486*    579*    580*    770     771*    774*
                        775*
DODINV  001354          154#    419*    436*    452*    462*    485*    578*    579     766     770*    772*    773*    774
DOLAMP  005702          240     620#
DOR     001350          151#    367     377*    379*    380*    381*    392*    399*    408*    411*    421*    465*    498*
                        510*    521*    553*    572*    614*    625*    833     834
DOR1    001352          152#    307     381*    573*    610*
```

```
DOT      013004              221     795#
DSWR   = 177570               19#     60      197
DT1      014666               63     833#
DT2      014700               65     834#
DT3      014714               67      71     835#
DT4      014730               69     836#
DT6      014740               73     837#
DT7      014752               75     838#
DWARF    001360              156#    190*    192*    210     230*    236*    279     355     495     504     517     548
EMTVEC=  000030               19#    197*
EM1      013074               63     798#
EM14     013526               71     804#
EM2      013203               65     799#
EM3      013253               67     800#
EM4      013325               69     801#
EM6      013371               73     802#
EM7      013442               75     803#
ENDOUT   014464              335     343     822#
ERRTOT   013650              591     807#
ERRVEC=  000004               19#    197*    257*    275*    276*    365*    371*    372*    375*    376*    652*
EVER     001442              181#    219*    277     288*    289*    291     293     568
EXTMSG   005550              586     588#
FIXADR   002712              220     296     303#    348     608     621
FOUND1   013573              281     805#
FOUND2   013616              284     806#
GNS    = ****** U            204     685
GTSWR  = 104407              204     246     647     685#
HIGHLO   005634              243     607#
HT     = 000011               19#    681
IOTRD    012150               41     275     371     375     707#
IOTVEC=  000020               19#    197*
LF     = 000012               19#    681
LOGIC    002424              207     237     254#    601
LOGICO   002420              231     253#
MASKNM   001376              163#    255*    297*    574*    594     661     669
MESGD    013677              596     808#
MSGCAB   014603              212     826#
MSGDWF   014471              253     823#
MTEST    002174              211     214#    233
MTEST1   002200              209     215#    247     249     349     644     651
OCSR     001344              148#    303     366     383*    384*    385*    387*    390*    394     402*    403     412     427*
                             431*    445*    449*    451*    454*    464*    469*    478*    482*    484*    487*    499     522*
                             528     535     540*    542     570*    833     835     836
OCSR1    001346              149#    393*    400*    409*    422*    466*    571*
PC     -%000007               19#    203*    220*    296*    329*    348*    575*    586*    600*    608*    617*    621*    630*
                             648*    651*    652*    668*    669*    670*    674*    675     681*
PIRQ   = 177772               19#
PIRQVE=  000240               19#
PRIMEO   012362              214     781#
PR0    = 000000               19#
PR1    = 000040               19#
PR2    = 000100               19#
PR3    = 000140               19#
PR4    = 000200               19#
```

```
PR5    = 000240              19#
PR6    = 000300              19#
PR7    = 000340              19#
PS     = 177776              19#
PSW    = 177776              19#
PWRMSG  011356              675     676#
PWRVEC= 000024               19#     197*    675*
RBEGO   001502              191     194     197#
RDCHR  = 104411             651     685#
RDLIN  = 104412             222     679     685#
RDOCT  = 104413             336     344     685#
REMAIN  005300              373     565#
RESTRT  001476               48     195#
RESVEC= 000010               19#
RUNIT   002416              223*    224*    225     228     232     234     238     241     244     250#
R0     =%000000              19#     303*    305*    307     310*    312*    313*    318*    321*    324*    325*    327     533*
                            576*    577*    578     586*    609*    611*    613*    615*    623*    628*    670*    672*    674*
                            675*    679*    681*    685*
R1     =%000001              19#     304*    305     306*    309*    310     311     312     314*    318     322*    324     326
                            624*    626*    670*    672*    674*    675*    679*    682*
R2     =%000002              19#     323*    325     672*    675*    679*
R3     =%000003              19#     651*    671*    672*    675*
R4     =%000004              19#     671*    675*
R5     =%000005              19#     671*    672*    675*
R6     =%000006              19#     197*
R7     =%000007              19#
SP     -%000006              19#     197*    223     268     282*    299*    300*    333*    337     339     341*    345     347
                            369     423*    424*    435     438*    439*    448     458*    459*    472*    473*    481     488*
                            489*    586*    592*    597*    607*    620*    643     651*    652*    669*    670*    671*    672*
                            674*    675*    679*    681*    682*    685*    719*    757     761*    766*    768*
STACK  = 001100              19#     197     607     620
STKLMT= 177774               19#
SWR     001140               60#     197*    204     263     651*    652     669     675*
SWREG   000176               44#     197     204     651
SW0    = 000001              19#
SW00   = 000001              19#
SW01   = 000002              19#
SW02   = 000004              19#
SW03   = 000010              19#
SW04   = 000020              19#
SW05   = 000040              19#
SW06   = 000100              19#
SW07   = 000200              19#
SW08   = 000400              19#
SW09   = 001000              19#
SW1    = 000002              19#
SW10   = 002000              19#
SW11   = 004000              19#
SW12   = 010000              19#     263
SW13   = 020000              19#
SW14   = 040000              19#
SW15   = 100000              19#
SW2    = 000004              19#
SW3    = 000010              19#
```

```
SW4    = 000020              19#
SW5    = 000040              19#
SW6    = 000100              19#
SW7    = 000200              19#
SW8    = 000400              19#
SW9    = 001000              19#
TBITVE= 000014               19#
TCRLF   014662              830#
TEMP    001362              157#   293*   379*   509*   511*   622*   625    631*   838
TEMP1   001364              158#   189*   193*   195*   208
TESTER  001462               49    192#
TKVEC  = 000060              19#   651*
TPVEC  = 000064              19#
TRAPVE= 000034               19#   197*
TRFRO   012360              718*   778#   837
TRTO    012356              707*   708*   710    768    772    777#   837
TRTVEC= 000014               19#
TSTR0   001366              159#   520*   523
TSTR2   001370              160#   357*   519*   550*
TSTR4   001372              161#   358    533    554
TSTR6   001374              162#
TST1    003164              300    354#   582
TST10   003724              383    384#
TST11   004010              384    385#
TST12   004074              385    387#
TST13   004140              387    389#
TST14   004206              396    398#
TST15   004244              404    407#
TST16   004302              413    417#
TST17   004474              457#
TST2    003236              356    362    364#
TST20   004656              494#
TST21   004722              496    501    503#
TST22   005006              505    507    516#
TST23   005202              518    544    547#
TST24   005300              566#
TST3    003320              377#
TST4    003372              379#
TST5    003456              380#
TST6    003522              380    381#
TST7    003640              383#
TYPBN  = 104406              598    685#
TYPDS  = 104405              283    586    593    670    685#   762
TYPE   = 104401              204    212    214    221    248    253    281    284    332    335    340    343    586
                             591    596    599    651    669    670    671    672    675    681    682    685#   760
                             765    767    769
TYPOC  = 104402              334    342    651    670    685#
TYPON  = 104404              685#
TYPOS  = 104403              685#   766    768
UNITBD  001446              183#   576    662*   665*   833    834    835    836    837    838
VADDR   001400              164#   259    570    571    572    573
VECLST  001402              165#   313    578
VECOFF  001422              173#   314
VECOUT  014415              340    821#
```

```
VTMSG   013714            760     809#
VTMSG1  014000            767     811#
VTMSG2  014030            769     812#
VTMSG3  013750            765     810#
WHICHU  007652            575     661#    669
$APTHD  001000             59#
$ASTAT= ****** U          674
$ATYC   010762            674#
$ATY1   010736            674#
$ATY3   010744            674#    681
$ATY4   010754            669     674#
$AUTOB  001134             60#    204*    205     651
$BASE   001244             60#    254     304     333     339*
$BDADR  001122             60#
$BDDAT  001126             60#    254*    258     259*    358*    359*    361     377*    379*    380*    381*    383*    384*
                          385*    387*    394*    395     403*    412*    499*    500     523*    525     528*    530     535*
                          537     542*    543     554*    557     834     835
$BELL   007320            651#
$BIN    012062            682#*
$CDW1   001250             60#
$CDW2   001252             60#
$CHARC  012004            681#*
$CKSWR  006372            651#    685
$CMTAG  001100             60#    197
$CM3  = 000000             60#
$CNTLC  007324            651#
$CNTLG  007336            651#
$CNTLU  007331            651#
$CPUOP  001216             60#
$CRLF   001165             60#    651     669     670     681
$DBLK   010726            672#
$DEVCT  001200             60#
$DEVM   001246             60#
$DOAGN  005524            586#
$DTBL   010716            672#
$ENDAD  005514             57     204     586#    669
$ENDCT  005462            197     586#
$ENDMG  005533            586#
$ENULL  005530            586#    599
$ENV    001210             60#    204     261     669     674     681
$ENVM   001211             60#    197     674     681
$EOP    005426            274     287     586#
$EOPCT  005454            197*    586#
$ERFLG  001103             60#    652*    669*
$ERMAX  001115             60#    197*    652*
$ERROR  007706            197     669#
$ERRPC  001116             60#    669*    670     833     834     835     836     837     838
$ERRTB  001254             60#    670
$ERRTY  010110            669     670#
$ERTTL  001112             60#    218*    589     592     669*
$ESCAP  001162             60#    197*    652*    669
$ETABL  001210             60#
$ETEND  001254             59      60#
$FATAL  001172             60#    674*
```

```
$FFLG   011202          674#*
$FILLC  001156           60#    681
$FILLS  001155           60#    681
$GDADR  001120           60#
$GDDAT  001124           60#    360*   361    377*   379*   380*   381*   383*   384*   385*   387*   391*   395
                        401*    410*   497*   500    508*   510    513*   524*   525    529*   530    536*   537
                        541*    543    552*   553    557    560*   834    835
$GET42  005504          586#
$GTSWR  006462          651#    685
$HD   = 000000           15
$HIBTS  001000           59#
$HIOCT  011524          679#*
$ICNT   001104           60#    652*
$ILLUP  011350          675#
$INTAG  001135           60#    651*
$ITEMB  001114           60#    669*   670
$LF     001166           60#    651    669    681
$LFLG   011201          674#*
$LPADR  001106           60#    197*   377*   379*   381*   418*   551*   652*   776
$LPERR  001110           60#    197*   652*   669
$MADR1  001222           60#
$MADR2  001226           60#
$MADR3  001232           60#
$MADR4  001236           60#
$MAIL   001170           59     60#    197    204    652    669    681
$MAMS1  001220           60#
$MAMS2  001224           60#
$MAMS3  001230           60#
$MAMS4  001234           60#
$MBADR  001002           59#
$MFLG   011200          674#*
$MNEW   007354          651#
$MSGAD  001204           60#    674*
$MSGLG  001206           60#    674*
$MSGTY  001170           60#    674*
$MSWR   007343          651#
$MTYP1  001221           60#
$MTYP2  001225           60#
$MTYP3  001231           60#
$MTYP4  001235           60#
$MXCNT  007650          652#
$NULL   001154           60#    681
$NWTST= 000001          354#    364#   377#   379#   380#   381#   383#   384#   385#   387#   389#   398#   407#
                        417#    457#   494#   503#   516#   547#   566#
$OCNT   010506          671#*
$OMODE  010510          671#*
$OVER   007634          652#
$PASS   001176           60#    197*   217*   506    586*   652    758
$PASTM  001006           59#
$PURAD  011344          675#
$PURDN  011204          197     675#
$PURMG  011340          675#
$PURUP  011256          675#
$QUES   001164           60#    248    651    669    681
```

```
$RDCHR  006734          651#    685
$RDDEC= ****** U        685
$RDLIN  007024          651#    685
$RDOCT  011424          679#    685
$RDSZ = 000040          651#
$RTNAD  005526          586#
$R2A  = ****** U        685
$SAVRE= ****** U        685
$SAVR6  011354          675#*
$SCOPE  007366          197     652#
$SETUP= 000137          188#    197     204     586     651     652     669
$STUP = 177777          188#
$SVLAD  007600          652#
$SVPC = 000106           57#
$SWR  = 165400            8#     15      21      60     197     354     364     377     379     380     381     383     384
                        385     387     389     398     407     417     457     494     503     516     547     566     586
                        652     669     675
$SWREG  001212           60#    197
$SWRMK= 000000           21     652
$TESTN  001174           60#    652*
$TIMES  001160           60#    197*    380*    387*    503*    566*    586*    652*
$TKB    001146           60#    639     651
$TKCNT  006036          651#*
$TKINT  006104          203     651#
$TKQEN= 006104          651#
$TKQIN  006040          651#*
$TKQOU  006042          651#*
$TKQSR  006044          651#
$TKS    001144           60#    216*    380*    387*    588*    637     651*
$TKSRV  006154          651#
$TN   = 000025            9#     15      354#    356     362     364#    377#    379#    380#    381#    383#    384#    585#
                        387#    389#    396     398#    404     407#    413     417#    457#    494#    496     501     503#
                        505     507     516#    518     544     547#    566#
$TPB    001152           60#    681*
$TPFLG  001157           60#    681
$TPS    001150           60#    681
$TRAP   012064          197     685#
$TRAP2  012106          685#
$TRP  = 000014          685#
$TRPAD  012120          685#
$TSTM   001004           59#
$TSTNM  001102           60#    581*    586*    652*    669     721
$TTYIN  007260          651#
$TYPBN  012010          682#    685
$TYPDS  010512          672#    685
$TYPE   011526          199*    200*    201*    202*    674     681#    685
$TYPEC  011740          651     681#
$TYPEX  012006          681#
$TYPOC  010310          671#    685
$TYPON  010324          671#    685
$TYPOS  010264          671#    685
$UNIT   001202           60#    256*    260*    265     269     282     285     288     291     295*    567*    568     761
                        838
$UNITM  001010           59#
```

```
$USWR   001214              60#
$VECT1  001240              60#     309     318     341     347*
$VECT2  001242              60#
$XTSTR  007404             652#
$$GET4= 000000             586#
$OFILL  010507             671#*
$40CAT= ****** U           652     669
.       = 014776            25#      36      37#     40#     42#     46#     52#     57#     58#     59#     60#     197     586
                           651#     652     669     670#    672#    674#    675     677#    681     831#
.$ASTA= ****** U           674
.$X     = 001000            59#
```

```
COMMEN    19#
ENDCOM    19#
ERROR     19#  273  294  363  370  377  379  380  381  383  384  385  387  397  405
          414  432  450  483  502  527  532  539  545  559  724
ESCAPE    19#
FIXERO   655#  669
FLOATO   109#  379
FLOAT1    97#  377
GETPRI    19#
GETSWR    12#   19#  204#
HIBYTE    77#  381
MULT      19#
NEWTST    19#  354  364  377  379  380  381  383  384  385  387  389  398  407  417
          457  494  503  516  547  566
POP       19#  672  674  675  679
PUSH      19#  672  674  675  679
READRW   133#  383  384  385
REPORT    19#
RESETO   123#  380  387
SCOPE     19#  354  364  377  379  380  381  383  384  385  387  389  398  407  417
          457  494  503  516  547  566  586
SETPRI    19#  651
SETTRA   685#
SETUP     19#  197
SKIP      19#  356  362  368  377  379  380  381  383  384  385  387  396  404  413
          433  446  479  496  501  505  507  518  526  531  538  544  549  558  569
          759
SLASH     19#
SPACE     19#
STARS     19#   57   59   60  354  364  377  379  380  381  383  384  385  387  389
          398  407  417  457  494  503  516  547  566  586  651  652  669  670  671
          672  674  675  679  681  682  685
SWRSU     19#  197#
TRMTRP   685#
TYPBIN    19#
TYPDEC    19#  586  670
TYPNAM    19#  204
TYPNUM    19#
TYPOCS    19#  766  768
TYPOCT    19#  651  670
TYPTXT    19#
$$CMRE    60#
$$CMTM    60#
$$ESCA    19#
$$NEWT    19#  354  364  377  379  380  381  383  384  385  387  389  398  407  417
          457  494  503  516  547  566
$$SET    685#
$$SETM   197#
$$SKIP    19#  356  362  380  383  384  385  387  396  404  413  496  501  505  507
          518  544
.EQUAT    10#   19
.HEADE    10#   15
.SETUP    11#  188
.SWRHI    12#   21
```

```
.SWRLO     21#
.$ACT1     13#     57
.$APTB     13#     60#
.$APTH     13#     59
.$APT`     13#    674
.$CATC     10#
.$CMTA     10#     60
.$EOP      10#    586
.$ERRO     10#    669
.$ERRT     12#    670
.$PARM     11#
.$POWE     11#    675
.$RDOC     12#    679
.$READ     11#    651
.$SAVE     11#
.$SCOP     11#    652
.$SPAC     11#
.$SWDO     11#
.$TRAP     11#    685
.$TYPB     12#    682
.$TYPD     12#    672
.$TYPE     10#     11#    681
.$TYPO     10#    671
```

E 6

```
MOV     192  197  199  200  201  202  223  230  254  257  275  276  282  288  297
        300  303  304  305  309  310  312  313  314  318  321  322  323  324  325
        333  339  341  347  358  360  365  371  372  375  376  377  379  380  381
        383  384  385  387  391  394  403  412  418  419  420  424  436  439  452
        453  458  459  462  463  464  473  485  486  489  497  499  503  508  509
        510  519  523  524  528  529  533  535  536  541  542  550  551  552  553
        554  566  576  578  579  586  592  597  607  609  613  620  622  623  625
        639  651  652  661  662  669  670  671  672  674  675  679  681  682  685
        707  718  761  766  768  770  771  772  774
MOVB    197  204  297  393  400  409  422  651  652  669  671  672  674  679  681
        682  685
NEG     671  672
NOP     426  428  429  430  441  442  443  444  461  467  468  470  471  475  476
        477  491  555  556  562  583  586  763  764
RESET   215  380  387  586
ROL     671  679  682
RTI      54  197  202  301  425  440  460  474  490  651  652  669  671  672  675
        679  681  682  685  756
RTS     329  648  651  668  670  674  681  685
SEC     682
SUB     669  672  674  708
TRAP    685
TST     204  208  210  258  261  269  271  277  279  285  311  337  345  366  367
        369  506  589  643  651  652  669  670  671  672  674  679  681  685  758
TSTB    205  637  651  652  670  672  674  681
.ASCII   60  782  784  786  788  790  792  812  824  827
.ASCIZ   60  204  586  651  670  676  794  796  798  799  800  801  802  803  804
        805  806  807  808  809  810  811  813  814  815  816  817  818  819  820
        821  822  829
.BLKB   651
.BLKW   672
.BYTE    60  586  651  669  671  674  682  766  768  781  783  785  787  789  791
        793  795  823  825  826  828  830  839
.DSABL  651
.ENABL    7  651
.END    941
.ENDC    15   19   21   57   59   60  188  197  204  354  356  362  364  368  377
        379  380  381  383  384  385  387  389  396  398  404  407  413  417  433
        446  457  479  494  496  501  503  505  507  516  518  526  531  538  544
        547  549  558  566  569  586  651  652  669  670  671  672  674  675  679
        681  682  685  759  766  768
.EQUIV   19
.EVEN    60  204  651  670  674  677  831
.IF      15   19   21   57   59   60  188  197  204  354  356  362  364  368  377
        379  380  381  383  384  385  387  389  396  398  404  407  413  417  433
        446  457  479  494  496  501  503  505  507  516  518  526  531  538  544
        547  549  558  566  569  586  651  652  669  670  671  672  674  675  679
        681  682  685  759  766  768
.IFF     19   21   57   59   60  197  204  354  356  362  364  368  377  379  380
        381  383  384  385  387  389  396  398  404  407  413  417  433  446  457
        479  494  496  501  503  505  507  516  518  526  531  538  544  547  549
        558  566  569  586  651  652  669  670  671  672  674  675  679  681  682
        685  759  766  768
.IFT    204  651  652  669  679
```

ERRORS DETECTED: 0

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