IDENTIFICATION

PRODUCT CODE: AC-S039A-MC

PRODUCT NAME: CXDPVA0 DPV-11 MODULE

PRODUCT DATE: JUNE 1980

MAINTAINER: DEC/X11 SUPPORT GROUP

1. ABSTRACT

DPV IS AN IOMOD THAT EXERCISES UP TO SIX CONSECUTIVELY ADDRESSED SYNCHRONOUS LINE INTERFACES (DPV11) BY TRANSMITTING A STANDARD BINARY COUNT PATTERN IN EITHER INTERNAL OR EXTERNAL LOOPBACK. THE USER MAY ALSO SELECT EITHER BIT ORIENTED PROTOCOL (BOP) OR BYTE ORIENTED PROTOCOL (BCP). THE RECEIVED DATA IS COMPARED WITH THE TRANSMITTED DATA AND ANY ERRORS ARE REPORTED VIA THE CONSOLE TTY. ALL AVAILABLE INTERFACES (UP TO 6) ARE ACTIVATED AND RUNNING SIMULTANEOUSLY.

(NOTE: THIS MODULE IS A MODIFICATION OF CXDPBB WHICH IS THE DEC/X11 MODULE FOR THE DUP11).

2. REQUIREMENTS

HARDWARE: DPV11 SYNCHRONOUS INTERFACE
STORAGE:: THE DPV REQUIRES:
1. DECIMAL WORDS: 604
2. OCTAL WORDS: 1134
3. OCTAL BYTES: 2270

3. PASS DEFINITION

ONE PASS OF THE DPV MODULE CONSISTS OF TRANSMITTING AND RECEIVING 14,400 8-BIT CHARACTERS (TOTAL) PER ACTIVE DEVICE.

4. EXECUTION TIME

ONE DPV RUNNING ALONE ON A LSI 11/23 PROCESSOR TAKES APPROXIMATELY 1 MINUTE TO COMPLETE ONE PASS. THIS TIME INCREASES SLIGHTLY AS MORE DEVICES ARE ADDED.

5. CONFIGURATION PARAMETERS

DEFAULT PARAMETERS:

DEVADR: 000001, VECTOR:001, BR1:5, BR2:5, DEVCNT:1
THE DEVICE IS TESTED IN SDLC MODE AS A SECONDARY STATION, WITH A HARDWARE BCC CHECK USING THE CRC/CCITT POLYNOMIAL.

REQUIRED PARAMETERS: THE CSR AND VECTOR MUST BE SET UP BY THE USER.

6. DEVICE/OPTION SETUP

SR1
BIT 0 = 0 BIT ORIENTED PROTOCOL
BIT 0 = 1 BYTE ORIENTED PROTOCOL

BIT 1 = 0 INTERNAL LOOPBACK (MAINTENANCE MODE)
BIT 1 = 1 EXTERNAL LOOPBACK (ONBOARD CONNECTOR - H3260 RS423)

7. MODULE OPERATION

TEST SEQUENCE:

A. TEST UP TO 6 POSSIBLE DEVICES FOR SELECTION
B. STORE THE NO. OF DEVICES TO BE TESTED AND SET UP THE VECTORS AND PRIORITIES FOR THESE DEVICES
C. LOAD ALL REGISTERS--SET UP IN USER DEFINED MODE,TURN RECEIVER AND TRANSMITTER ON, AND INTERRUPT ENABLES FOR ALL ACTIVE DEVICES. ENABLE SELECTED DEVICES.
D. TRANSMITTER INTERRUPT SERVICE:
    1.) TEST FOR FALSE INTERRUPT (READY (0)); REPORT ERRORS
    2.) OUTPUT NEXT CHARACTER TO THE DEVICE
    3.) RETURN TO MONITOR TO WAIT FOR RECEIVER INTERRUPT.
E. RECEIVER INTERRUPT SERVICE:
    1.) TEST FOR FALSE INTERRUPT (DONE (0)); REPORT ERRORS
    2.) CHECK FOR DATA ERROR; REPORT ERRORS
    3.) IF LAST CHARACTER, CHECK FOR BCC ERROR.
    4.) RETURN TO MONITOR TO WAIT FOR TRANSMITTER INTERRUPT
F. REPEAT D AND E UNTIL ALL DEVICES HAVE BEEN PROCESSED.
G. TURN OFF ALL ACTIVE DEVICES AND DECREMENT ITERATION COUNT. IF NOT 0,RESTART AT B.
H. SIGNAL END PASS.

8. OPERATION OPTIONS

A. LOCATION DVID1 (DPV 14) MAY BE CHANGED TO SELECT ANY COMBINATION OF DEVICES BIT0=DEV0, BIT1=DEV1 .....BIT5=DEV5.

NOTE: IF DVID1 IS INITIALLY 0 DPV WILL BE DROPPED FROM TEST.

9. NON STANDARD PRINTOUTS

NONE: ALL PRINTOUTS HAVE STANDARD FORMATS AS DESCRIBED IN THE DEC/X11 DOCUMENT.

```
        8
        9 000000                        IOMOD   <DPVA >,1,1,5,5,0,14,70
          000000                        MODULE  140000,DPVA ,1,1,5,5,0,14,70
                                        .TITLE  DPVA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6      23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************************
          000000                BEGIN:
          000000     104  120  126     MODNAM: .ASCII  /DPVA / ;MODULE NAME.
          000003     101  040
          000005     000                XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
          000006  000001                ADDR:   1+0             ;1ST DEVICE ADDR.
          000010  000001                VECTOR: 1+0                     ;1ST DEVICE VECTOR.
          000012     240                BR1:    .BYTE   PRTY5+0         ;1ST BR LEVEL.
          000013     240                BR2:    .BYTE   PRTY5+0         ;2ND BR LEVEL.
          000014  000001                DVID1:  0+1                     ;DEVICE INDICATOR 1.
          000016  000000                SR1:    OPEN                    ;SWITCH REGISTER 1
          000020  000000                SR2:    OPEN                    ;SWITCH REGISTER 2
          000022  000000                SR3:    OPEN                    ;SWITCH REGISTER 3
          000024  000000                SR4:    OPEN                    ;SWITCH REGISTER 4
                                        ;*****************************************************************************
          000026  140000                STAT:   140000                  ;STATUS WORD.
          000030  000244'               INIT:   START                   ;MODULE START ADDR.
          000032  000224'               SPOINT: MODSP                   ;MODULE STACK POINTER.
          000034  000000                PASCNT: 0                       ;PASS COUNTER.
          000036  000014                ICONT:  14                      ;# OF ITERATIONS PER PASS=14
          000040  000000                ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
          000042  000000                SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
          000044  000000                HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
          000046  000000                SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
          000050  000000                HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
          000052  000000                SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
          000054  000000                RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
          000056                        CONFIG:                         ;RESERVED FOR MONITOR USE
          000056  000000                RES1:   0                       ;RESERVED FOR MONITOR USE
          000060  000000                RES2:   0                       ;RESERVED FOR MONITOR USE
          000062  000000                SVR0:   OPEN                    ;LOC TO SAVE R0.
          000064  000000                SVR1:   OPEN                    ;LOC TO SAVE R1.
          000066  000000                SVR2:   OPEN                    ;LOC TO SAVE R2.
          000070  000000                SVR3:   OPEN                    ;LOC TO SAVE R3.
          000072  000000                SVR4:   OPEN                    ;LOC TO SAVE R4.
          000074  000000                SVR5:   OPEN                    ;LOC TO SAVE R5.
          000076  000000                SVR6:   OPEN                    ;LOC TO SAVE R6.
          000100  000000                CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
          000102                        SBADR:                          ;ADDR OF GOOD DATA, OR
          000102  000000                ACSR:   OPEN                    ;CONTENTS OF CSR.
          000104                        WASADR:                         ;ADDR OF BAD DATA, OR
          000104  000000                ASTAT:  OPEN                    ;STATUS REG CONTENTS.
          000106                        ERRTYP:                         ;TYPE OF ERROR
          000106  000000                ASB:    OPEN                    ;EXPECTED DATA.
          000110  000000                AWAS:   OPEN                    ;ACTUAL DATA.
          000112  000330'               RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
          000114  000000                WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
          000116  000000                WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
          000120  000000                INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
          000122  000070                IDNUM:  70                      ;MODULE IDENTIFICATION NUMBER=70
```



```
                  000040                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                                .NLIST
                                                .WORD   0
                                                .LIST
                                                .ENDR
          000224                        MODSP:
                                        ;*****************************************************************************
```

```
                                ;BIT DEFINITIONS
          100000                BIT15=  100000
          040000                BIT14=  40000
          020000                BIT13=  20000
          010000                BIT12=  10000
          004000                BIT11=  4000
          002000                BIT10=  2000
          001000                BIT9=   1000
          000400                BIT8=   400
          000200                BIT7=   200
          000100                BIT6=   100
          000040                BIT5=   40
          000020                BIT4=   20
          000010                BIT3=   10
          000004                BIT2=   4
          000002                BIT1=   2
          000001                BIT0=   1


                                ;CONTROL AND STATUS REGISTER DEFINITIONS
                                ;--------------------------
                                        ;RXCSR = RECEIVER CONTROL AND STATUS     (16XXX0)
                                        ;RDSR  = RECEIVER DATA AND STATUS       (16XXX2) READ ONLY
                                        ;PCSAR = PARAMETER CONTROL SYNC/ADDRESS (16XXX2) WRITE ONLY
                                        ;TXCSR = TRANSMITTER CONTROL AND STATUS (16XXX4)
                                        ;TDSR  = TRANSMITTER DATA AND STATUS    (16XXX6)
          000000                RXCSR=  0               ;RECEIVER CONTROL AND STATUS REG
          000002                RDSR=   2               ;RECEIVER DATA BUFFER REGISTER
          000002                PCSAR=  2               ;PARAMETER CONTROL SYNC/ADDRESS
          000004                TXCSR=  4               ;TRANSMITTER CONTROL AND STATUS REGISTER
          000006                TDSR=   6               ;TRANSMITTER DATA BUFFER REGISTER


                                        ;RXCSR BIT DEFINITIONS
          004000                RXACT=  BIT11           ;RECEIVER ACTIVE
          002000                RXSTAT= BIT10           ;RECEIVER STATUS AVAILABLE
          000100                RXIE=   BIT6            ;RECEIVE INTERRUPT ENABLE
          000020                RXEN=   BIT4            ;RECEIVER ENABLE
          000004                RTS=    BIT2            ;REQUEST TO SEND

                                        ;RDSR BIT DEFINITIONS
          100000                ERR=    BIT15           ;RECEIVER ERROR
          004000                ROVER=  BIT11           ;RECEIVER OVERRUN
          002000                RABORT= BIT10           ;RECEIVER ABORT
          001000                REOM=   BIT9            ;RECEIVE END OF MESSAGE
          000400                RSOM=   BIT8            ;RECEIVE START OF MESSAGE

                                        ;PCSAR  BIT DEFINITIONS
          040000                BCP=    BIT14           ;PROTOCOL SELECTION. SET = BCP, CLEAR = BOP.
          010000                SECADR= BIT12           ;SECONDARY ADDRESS.
          001400                CRC16=  BIT8!BIT9       ;CRC16

                                        ;TXCSR BIT DEFINITIONS
          000100                TXIE=   BIT6            ;TRANSMITTER INTERRUPT ENABLE
          000020                TXEN=   BIT4            ;TRANSMITTER ENABLE
          000010                MAINT=  BIT3            ;MAINTENANCE MODE
```



```
          000004                TBE=    BIT2            ;TRANSMIT BUFFER EMPTY
          000002                TXACT=  BIT1            ;TRANSMITTER ACTIVE
          000001                RESET=  BIT0            ;DPV RESET

                                        ;TDSR BIT DEFINITIONS
          100000                TERR=   BIT15           ;TRANSMIT DATA LATE ERROR
          002000                TABORT= BIT10           ;TRANSMIT ABORT
          001000                TEOM=   BIT9            ;TRANSMIT END OF MESSAGE
          000400                TSOM=   BIT8            ;TRANSMIT START OF MESSAGE

                                        ;MISCELLANEOUS STORAGE
000224    000000                N.DPVS: OPEN            ;SOFTWARE MEMORY
000226    000000                TOTAL:  OPEN            ;STORAGE FOR DATA CHECK
000230    000000                COUNT:  OPEN            ;ITERATION COUNT
000232    000000                RXBF:   OPEN            ;REC BUFFER ADRS STORAGE
000234    000000                SAVBF:  OPEN            ;REC ISR TEMPORARY STORAGE
000236    000000                SELECT: OPEN            ;SOFTWARE POINTER TO ACTIVE DEVICES

000240    000000                MODE:   .WORD   0       ;PROTOCOL MODE (0 = BOP, 1= BCP)
000242    000000                LOOP:   .WORD   0       ;LOOPBACK FLAG (0 = MAINTENANCE, 1 = EXTERNAL)


                                .MACRO  $PUSH   $A,$B,$C,$D,$E,$F
                                .IIF NB <$A>,   MOV     $A,-(SP)        ;SAVE '$A ON THE STACK
                                .IIF NB <$B>,   MOV     $B,-(SP)        ;SAVE '$B ON THE STACK
                                .IIF NB <$C>,   MOV     $C,-(SP)        ;SAVE '$C ON THE STACK
                                .IIF NB <$D>,   MOV     $D,-(SP)        ;SAVE '$D ON THE STACK
                                .IIF NB <$E>,   MOV     $E,-(SP)        ;SAVE '$E ON THE STACK
                                .IIF NB <$F>,   MOV     $F,-(SP)        ;SAVE '$F ON THE STACK
                                .ENDM   $PUSH
                                .MACRO  $POP    $A,$B,$C,$D,$E,$F
                                .IIF NB <$A>,   MOV     (SP)+,$A        ;POP STACK TO '$A
                                .IIF NB <$B>,   MOV     (SP)+,$B        ;POP STACK TO '$B
                                .IIF NB <$C>,   MOV     (SP)+,$C        ;POP STACK TO '$C
                                .IIF NB <$D>,   MOV     (SP)+,$D        ;POP STACK TO '$D
                                .IIF NB <$E>,   MOV     (SP)+,$E        ;POP STACK TO '$E
                                .IIF NB <$F>,   MOV     (SP)+,$F        ;POP STACK TO '$F
                                .ENDM   $POP
```

```
                                                ;BEGIN TESTING FOR THE DPV-11
000244                                  START:
000244  032767  000001  177544                  BIT     #BIT0,SR1       ;WHAT PROTOCOL?
000252  001003                                  BNE     1$              ;IF SET, BCP
000254  005067  177760                          CLR     MODE            ;FLAG THAT BOP MODE REQUESTED.
000260  000403                                  BR      2$
000262                                  1$:
000262  012767  000001  177750                  MOV     #1,MODE         ;FLAG THAT BCP MODE REQUESTED.
000270                                  2$:
000270  032767  000002  177520                  BIT     #BIT1,SR1       ;WHAT LOOPBACK
000276  001003                                  BNE     3$              ;IF SET, EXTERNAL LOOPBACK.
000300  005067  177736                          CLR     LOOP            ;FLAG THAT MAINTENANCE MODE (INTERNAL
                                                                        ;LOOPBACK) REQUESTED.
000304  000403                                  BR      4$
000306                                  3$:
000306  012767  000001  177726                  MOV     #1,LOOP         ;FLAG THAT EXTERNAL LOOPBACK REQUESTED.
000314                                  4$:
000314  016767  177474  177714                  MOV     DVID1,SELECT    ;GET ACTIVE DEVICES
000322  001002                                  BNE     RESTRT          ;BR IF ANY ARE SELECTED
000324                                  DROP:
000324  104410  000000'                          END$,BEGIN              ;;INCORRECT NUMBER OF DPV11'S

                                                ;SETUP VECTORS FOR ACTIVE DEVICES
000330                                  RESTRT:
000330  032767  177700  177700                  BIT     #^C<77>,SELECT  ;MAKE SURE NO MORE THAN 6 SELECTED.
000336  001372                                  BNE     DROP            ;INVALID-NO MORE THAN 6 DEVICES!!
000340  016701  177672                          MOV     SELECT,R1       ;GET IMAGE OF RUNNING DEVICES
000344  001767                                  BEQ     DROP            ;BR IF ALL DPV'S DROPPED
000346  005067  177652                          CLR     N.DPVS          ;CLEAR THE # OF DPV'S (SOFTWARE)
000352  016702  177432                          MOV     VECTOR,R2       ;GET INITIAL VECTOR
000356  016700  177424                          MOV     ADDR,R0         ;GET INITIAL ADDRESS
000362  012703  001752'                        MOV     #LNKTAB,R3      ;SET ISR POINTER
000366  012767  002172' 177636                  MOV     #RXBF0,RXBF     ;SET UP BUFFER POINTERS
000374                                  1$:
000374  006201                                  ASR     R1              ;ACTIVE?
000376  103410                                  BCS     3$              ;BR IF YES
000400  001437                                  BEQ     SETUP1          ;BR IF DONE
000402  062703  000024                          ADD     #24,R3          ;POP ISR POINTER
000406                                  2$:
000406  062702  000010                          ADD     #10,R2          ;NEXT DEVICE VECTOR
000412  062700  000010                          ADD     #10,R0          ;NEXT DEVICE CSR
000416  000766                                  BR      1$              ;CONTINUE
000420                                  3$:
000420  005267  177600                          INC     N.DPVS          ;UPDATE THE # TO RUN
000424  010312                                  MOV     R3,(R2)         ;LOAD ISR POINTER (RECEIVER)
000426  116762  177360  000002                  MOVB    BR1,2(R2)       ;LOAD PRIORITY
000434  010063  000004                          MOV     R0,4(R3)        ;LOAD CSR POINTER
000440  005063  000010                          CLR     10(R3)          ;CLR REC BYTE COUNT
000444  062703  000012                          ADD     #12,R3          ;UPDATE POINTER
000450  010362  000004                          MOV     R3,4(R2)        ;LOAD TRANSMITTER ISR
000454  116762  177333  000006                  MOVB    BR2,6(R2)       ;LOAD PRIORITY
000462  010063  000004                          MOV     R0,4(R3)        ;LOAD CSR POINTER
000466  005063  000010                          CLR     10(R3)          ;CLEAR TX BYTE COUNT
000472  062703  000012                          ADD     #12,R3          ;NEXT ISR POINTER
000476  000743                                  BR      2$              ;CONTINUE
```



```
                                                ;SET UP BUFFERS
000500                                  SETUP1:
000500  016767  177520  177412                  MOV     N.DPVS,INTR     ;SET # OF INTERRUPTS
000506  016767  177512  177400                  MOV     N.DPVS,WDTO     ;SET # OF WORDS TO MEM
000514  016767  177504  177374                  MOV     N.DPVS,WDFR     ;SET # OF WORDS FROM MEM
000522  006367  177372                          ASL     INTR            ;DOUBLE INTERRUPTS
000526  012700  000006                          MOV     #6,R0           ;LOAD BUFFER SIZE
000532  012703  002156'                        MOV     #TXBF0,R3       ;LOAD BUFFER START
000536                                  1$:
000536  012723  000135                          MOV     #135,(R3)+      ;FIRST CHARACTER TO TRANSMIT.
000542  005300                                  DEC     R0              ;LOAD UNTIL DONE.
000544  001374                                  BNE     1$              ;BR IF MORE TO GO
000546  012700  000006                          MOV     #6,R0           ;LOAD BUFFER SIZE
000552  012703  002172'                        MOV     #RXBF0,R3       ;LOAD START ADDRESS
000556                                  2$:
000556  012723  000135                          MOV     #135,(R3)+      ;FIRST CHARACTER TO RECEIVE.
000562  005300                                  DEC     R0              ;LOAD BUFFER UNTIL DONE.
000564  001374                                  BNE     2$              ;BR IF MORE TO GO

                                                ;PRELIMINARY DEVICE SETUP
000566                                  SETUP2:
000566  016700  177214                          MOV     ADDR,R0         ;LOAD FIRST CSR
000572  016701  177440                          MOV     SELECT,R1       ;SET UP TO GET ACTIVE DEVICES
000576                                  1$:
000576  006201                                  ASR     R1              ;GET AN ACTIVE
000600  103404                                  BCS     2$              ;ONE
000602  001410                                  BEQ     ACTIV           ;BR IF DONE
000604  062700  000010                          ADD     #10,R0          ;UPDATE FOR THE NEXT ONE
000610  000772                                  BR      1$              ;CONTINUE
000612                                  2$:
000612  004767  001024                          JSR     PC,PRELIM       ;GO DO THE DEVICE SETUP
000616  062700  000010                          ADD     #10,R0          ;UPDATE FOR THE NEXT ONE
000622  000765                                  BR      1$              ;CONTINUE
000624                                  ACTIV:
000624  016701  177406                          MOV     SELECT,R1       ;GET THE ACTIVE DPV'S
000630  016767  177370  177370                  MOV     N.DPVS,TOTAL    ;SET UP FOR DATA CHECK
000636  016700  177144                          MOV     ADDR,R0         ;GET FIRST CSR
000642                                  1$:
000642  006201                                  ASR     R1              ;GET AN ACTIVE ONE
000644  103404                                  BCS     3$              ;BR IF ACTIVE
000646  001406                                  BEQ     WAIT            ;BR IF DONE
000650                                  2$:
000650  062700  000010                          ADD     #10,R0          ;UPDATE CSR
000654  000772                                  BR      1$              ;CONTINUE
000656                                  3$:
000656  004767  001052                          JSR     PC,READY        ;TURN ON DEVICE
000662  000772                                  BR      2$              ;CONTINUE

                                                ;DELAY AND SCAN FOR FINISH ROUTINE
000664                                  WAIT:
000664  012705  020000                          MOV     #20000,R5       ;SET UP FOR A LONG DELAY
                                                                        ;APPROX 30 SECONDS ON A LSI 11/23
                                                                        ;APPROX 60 SECONDS ON A LSI 11/2
000670                                  SCAN:
000670                                  1$:
000670  104407  000000'                          BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
```

```
      000674  104407  000000'                BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
  115 000700  005767  177322                 TST     TOTAL           ;GET THE # OF ACTIVE DPVS
  116 000704  001004                         BNE     3$              ;BR IF MORE TO GO
  117                                        ;---------------------------------------
  118                                        ;---------------------------------------
  119 000706  104413  000000'                ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                     ;MONITOR SHALL TEST END OF PASS
  120 000712                             2$:
  121 000712  000167  177412                 JMP     RESTRT          ;RESTART PROGRAM
  122 000716                             3$:
  123 000716  005305                         DEC     R5              ;STALL FOR ALL DPV'S TO FINISH
  124 000720  001363                         BNE     1$              ;BR IF DELAY NOT ZERO
  125 000722  104403  000000' 000734'        MSGN$,BEGIN,HUNG        ;ASCII MESSAGE CALL WITH COMMON HEADER
  126 000730  104410  000000'                END$,BEGIN              ;;DROP THE MODULE
  127 000734                             HUNG:
  128 000734  002222'                        MHUNG                   ;POINTER TO ASCII MESSAGE
  129 000736  177777                         -1                      ;TERMINATOR
```



```
    1                                      ;TRANSMITTER INTERRUPT SERVICE ROUTINE
    2 000740                           TXISR:
    3 000740  010046                       MOV     R0,-(SP)        ;SAVE R0 ON THE STACK
      000742  010146                       MOV     R1,-(SP)        ;SAVE R1 ON THE STACK
    4 000744  012500                       MOV     (R5)+,R0        ;GET DEVICE CSR AND POP OFFSET TO R5
    5 000746  012501                       MOV     (R5)+,R1        ;GET DATA ADDRESS AND LEAVE R5=COUNT ADDRESS
    6 000750  032760  100000  000006       BIT     #TERR,TDSR(R0)  ;DATA LATE ERROR?
    7 000756  001430                       BEQ     2$              ;BR IF NO
    8 000760  010067  177114               MOV     R0,CSRA         ;ADDRESS OF RXCSR FOR TYPEOUT
    9 000764  011067  177112               MOV     (R0),ACSR       ;CONTENTS OF RXCSR FOR TYPEOUT
   10 000770  016067  000006  177106       MOV     TDSR(R0),ASTAT  ;CONTENTS OF TDSR FOR TYPEOUT
   11 000776  012760  000001  000004       MOV     #RESET,TXCSR(R0) ;TURN OFF THE DEVICE
   12 001004  012601                       MOV     (SP)+,R1        ;POP STACK TO R1
      001006  012600                       MOV     (SP)+,R0        ;POP STACK TO R0
      001010  012605                       MOV     (SP)+,R5        ;POP STACK TO R5
   13                                      ;---------------------------------------------------------------
      001012  000004  000000' 001020'      PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                          ;---------------------------------------------------------------
   14 001020                           1$:
   15 001020  012767  000041  177060       MOV     #41,ERRTYP      ;XMITTER DATA LATE
   16                                      ;*****************************************************************
      001026  104406  000000' 000000      SOFER$,BEGIN,NULL       ;;TRANSMIT DATA LATE ERROR
                                          ;*****************************************************************
   17 001034  000167  177630               JMP     SCAN            ;CONTINUE SCANNING FOR END
   18 001040                           2$:
   19 001040  005215                       INC     (R5)            ;UPDATE THE COUNTER
   20 001042  022715  000001               CMP     #1,(R5)         ;IS THIS THE FIRST ONE?
   21 001046  001004                       BNE     3$              ;IF NOT, SEND DATA
   22 001050  052760  000400  000006       BIS     #TSOM,TDSR(R0)  ;IF YES, DON'T TRANSMIT DATA
   23                                                              ;SEND OUT A SECOND FLAG (OR SYNCH)
   24                                                              ;NOTE: IN BOP MODE ONLY 1 FLAG IS
   25                                                              ;NECESSARY.  IN BCP MODE, 2 SYNCHS
   26                                                              ;ARE NEEDED.  IN BOTH CASE WE SEND OUT
   27                                                              ;2 JUST FOR EASE OF CODING.
   28 001056  000414                       BR      4$              ;
   29 001060                           3$:
   30 001060  011160  000006               MOV     (R1),TDSR(R0)   ;PUSH OUT DATA.
   31 001064  105211                       INCB    (R1)            ;CHANGE DATA FOR NEXT CHARACTER.
   32 001066  022715  002261               CMP     #1201.,(R5)     ;CHECK FOR FINISH. (THE TRANSMIT COUNTER IS
   33                                                              ;1201. BECAUSE WE COMPENSATE FOR THE EXTRA
   34                                                              ;FLAG OR SYNCH).
   35 001072  001006                       BNE     4$              ;BR IF MORE TO GO
   36 001074  052760  001000  000006       BIS     #TEOM,TDSR(R0)  ;END OF MESSAGE.
   37 001102  042760  000100  000004       BIC     #TXIE,TXCSR(R0) ;TURN OFF INTERRUPTS
   38 001110                           4$:
   39 001110  012601                       MOV     (SP)+,R1        ;POP STACK TO R1
      001112  012600                       MOV     (SP)+,R0        ;POP STACK TO R0
      001114  012605                       MOV     (SP)+,R5        ;POP STACK TO R5
   40 001116  000002                       RTI                     ;RETURN FROM INTERRUPT
```

```
                                        ;RECEIVER INTERRUPT SERVICE ROUTINE
001120                          RXISR:
001120  010046                          MOV     R0,-(SP)        ;SAVE R0 ON THE STACK
001122  010146                          MOV     R1,-(SP)        ;SAVE R1 ON THE STACK
001124  012500                          MOV     (R5)+,R0        ;GET CSR
001126  012501                          MOV     (R5)+,R1        ;GET BUFFER ADDR AND LEAVE R5=BYTE COUNT
001130  105710                          TSTB    (R0)            ;IS RECEIVER DATA AVAILABLE? (BIT 7 OF  RXCSR)
001132  100430                          BMI     2$              ;OK IF SET - BR

001134  010067  176740                  MOV     R0,CSRA         ;ADDRESS OF RXCSR FOR HARD ERROR MESSAGE.
001140  011067  176736                  MOV     (R0),ACSR       ;CONTENTS OF RXCSR FOR HARD ERROR MESSAGE.
001144  016067  000002  176732          MOV     RDSR(R0),ASTAT  ;CONTENTS OF RDSR FOR HARD ERROR MESSAGE.
001152  012760  000001  000004          MOV     #RESET,TXCSR(R0) ;TURN OFF THE DEVICE

001160  012601                          MOV     (SP)+,R1        ;POP STACK TO R1
001162  012600                          MOV     (SP)+,R0        ;POP STACK TO R0
001164  012605                          MOV     (SP)+,R5        ;POP STACK TO R5
                                        ;---------------------------------------------------------------
001166  000004  000000' 001174'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;---------------------------------------------------------------
001174                          1$:
001174  012767  000011  176704          MOV     #11,ERRTYP      ;ILLEGAL  INTERRUPT
                                        ;*****************************************************************
001202  104405  000000' 000000         HRDER$,BEGIN,NULL       ;;FALSE INTERRUPT
                                        ;*****************************************************************
001210  000167  177454                  JMP     SCAN            ;CONTINUE SCANNING FOR END
001214                          2$:
001214  016067  000002  177012          MOV     RDSR(R0),SAVBF  ;SAVE THE CHARACTER IN THE BUFFER
                                                                ;WAS THERE AN OVERRUN OR ABORT?
001222  032767  006000  177004          BIT     #ROVERIRABORT,SAVBF
001230  001012                          BNE     5$              ;BRANCH TO ERROR IF AN OVERRUN OR ABORT REC.
001232  005215                          INC     (R5)            ;UPDATE THE CHARACTER COUNT.
001234  022715  002260                  CMP     #1200.,(R5)     ;DONE?
001240  001467                          BEQ     13$             ;BR IF DATA DONE
001242  121167  176766                  CMPB    (R1),SAVBF      ;IS THE DATA OK?
001246  001033                          BNE     10$             ;IF NOT, PROCEED.
001250  105211                          INCB    (R1)            ;UPDATE DATA
001252  000167  000354                  JMP     20$

001256                          5$:
001256  010067  176616                  MOV     R0,CSRA         ;ADDRESS OF RXCSR FOR HARD ERROR MESSAGE.
001262  011067  176614                  MOV     (R0),ACSR       ;CONTENTS OF RXCSR FOR HARD ERROR MESSAGE.
001266  016067  000002  176610          MOV     RDSR(R0),ASTAT  ;CONTENTS OF RDSR FOR HARD ERROR MESSAGE.
001274  012760  000001  000004          MOV     #RESET,TXCSR(R0) ;TURN OFF THE DEVICE

001302  012601                          MOV     (SP)+,R1        ;POP STACK TO R1
001304  012600                          MOV     (SP)+,R0        ;POP STACK TO R0
001306  012605                          MOV     (SP)+,R5        ;POP STACK TO R5
                                        ;---------------------------------------------------------------
001310  000004  000000' 001316'         PIRQ$,BEGIN,6$          ; QUEUE UP TO CONTINUE AT 6$ AND RTI
                                        ;---------------------------------------------------------------
001316                          6$:
001316  012767  000017  176562          MOV     #17,ERRTYP      ;OVERRUN OR ABORT.
                                        ;*****************************************************************
001324  104405  000000' 000000         HRDER$,BEGIN,NULL       ;;HARDWARE DATA ERROR
                                        ;*****************************************************************
001332  000167  177332                  JMP     SCAN            ;CONTINUE SCANNING FOR END
```



```
001336                          10$:
001336  010167  176540                  MOV     R1,SBADR        ;LOAD GOOD DATA ADRS
001342  010067  176532                  MOV     R0,CSRA         ;LOAD CSR
001346  012767  000234' 176530          MOV     #SAVBF,WASADR   ;LOAD BAD DATA ADRS
001354  111167  176526                  MOVB    (R1),ASB        ;LOAD GOOD DATA
001360  116767  176650  176522          MOVB    SAVBF,AWAS      ;LOAD BAD DATA
001366  012760  000001  000004          MOV     #RESET,TXCSR(R0) ;TURN OFF THE DEVICE
001374  012601                          MOV     (SP)+,R1        ;POP STACK TO R1
001376  012600                          MOV     (SP)+,R0        ;POP STACK TO R0
001400  012605                          MOV     (SP)+,R5        ;POP STACK TO R5
                                        ;---------------------------------------------------------------
001402  000004  000000' 001410'         PIRQ$,BEGIN,11$         ; QUEUE UP TO CONTINUE AT 11$ AND RTI
                                        ;---------------------------------------------------------------
001410                          11$:
                                        ;*****************************************************************
001410  104404  000000'                 DATER$,BEGIN            ;DATA ERROR!!!
                                        ;*****************************************************************
001414  000167  177250                  JMP     SCAN            ;CONTINUE SCANNING FOR END


001420                          13$:
001420  005767  176614                  TST     MODE            ;PROTOCOL?
001424  001405                          BEQ     14$             ;BR IF BOP MODE
001426  032767  100000  176600          BIT     #ERR,SAVBF      ;CHECK FOR CRC ERROR
001434  001071                          BNE     19$             ;BR IF NO ERROR
001436  000440                          BR      17$             ;CRC ERROR
001440                          14$:
001440  032767  100000  176566          BIT     #ERR,SAVBF      ;CHECK FOR CRC ERROR
001446  001034                          BNE     17$             ;BR IF ERROR
001450  032767  001000  176556          BIT     #REOM,SAVBF     ;WAS RECEIVE END OF MESSAGE RECEIVED?
001456  001060                          BNE     19$             ;IF YES, OK.

001460  010067  176414                  MOV     R0,CSRA         ;ADDRESS OF RXCSR FOR HARD ERROR MESSAGE.
001464  011067  176412                  MOV     (R0),ACSR       ;CONTENTS OF RXCSR FOR HARD ERROR MESSAGE.
001470  016767  176540  176406          MOV     SAVBF,ASTAT     ;CONTENTS OF RDSR FOR HARD ERROR MESSAGE.
001476  012760  000001  000004          MOV     #RESET,TXCSR(R0) ;TURN OFF THE DEVICE

001504  012601                          MOV     (SP)+,R1        ;POP STACK TO R1
001506  012600                          MOV     (SP)+,R0        ;POP STACK TO R0
001510  012605                          MOV     (SP)+,R5        ;POP STACK TO R5
                                        ;---------------------------------------------------------------
001512  000004  000000' 001520'         PIRQ$,BEGIN,16$         ; QUEUE UP TO CONTINUE AT 16$ AND RTI
                                        ;---------------------------------------------------------------
001520                          16$:
001520  012767  000017  176360          MOV     #17,ERRTYP      ;RECEIVE END OF MESSAGE NOT RECEIVED
                                        ;*****************************************************************
001526  104405  000000' 000000         HRDER$,BEGIN,NULL       ;;RECEIVER ERROR
                                        ;*****************************************************************
001534  000167  177130                  JMP     SCAN            ;CONTINUE SCANNING FOR END


001540                          17$:

001540  010067  176334                  MOV     R0,CSRA         ;ADDRESS OF RXCSR FOR HARD ERROR MESSAGE.
001544  011067  176332                  MOV     (R0),ACSR       ;CONTENTS OF RXCSR FOR HARD ERROR MESSAGE.
001550  016067  000002  176326          MOV     RDSR(R0),ASTAT  ;CONTENTS OF RDSR FOR HARD ERROR MESSAGE.
```

```
     90 001556  012760  000001  000004          MOV     #RESET,TXCSR(R0) ;TURN OFF THE DEVICE
     91
     92 001564  012601                          MOV     (SP)+,R1         ;POP STACK TO R1
        001566  012600                          MOV     (SP)+,R0         ;POP STACK TO R0
        001570  012605                          MOV     (SP)+,R5         ;POP STACK TO R5
     93
        001572  000004  000000' 001600'         PIRQ$,BEGIN,18$          ; QUEUE UP TO CONTINUE AT 18$ AND RTI
     94 001600                          18$:
     95 001600  012767  000043  176300          MOV     #43,ERRTYP       ;CRC ERROR
     96
        001606  104405  000000' 000000          HRDERS,BEGIN,NULL        ;;HARDWARE DETECTED CRC ERROR
     97 001614  000167  177050                  JMP     SCAN             ;CONTINUE SCANNING FOR END
     98 001620                          19$:
     99 001620  012760  000001  000004          MOV     #RESET,TXCSR(R0) ;TURN OFF THE DEVICE
    100 001626  005367  176374                  DEC     TOTAL            ;DECREMENT DEVICE COUNTER
    101 001632                          20$:
    102 001632  012601                          MOV     (SP)+,R1         ;POP STACK TO R1
        001634  012600                          MOV     (SP)+,R0         ;POP STACK TO R0
        001636  012605                          MOV     (SP)+,R5         ;POP STACK TO R5
    103 001640  000002                          RTI                      ;RETURN FROM INTERRUPT
```



```
      1
      2                                         ;SUBROUTINES
      3
      4                                         ;PRELIM SUBROUTINE SETS UP THE DPV
      5                                         ; R0 = CSR ADDRESS
      6 001642                          PRELIM:
      7 001642  052760  000001  000004          BIS     #RESET,TXCSR(R0);RESET THE DPV11
      8 001650  005767  176364                  TST     MODE             ;WHAT PROTOCOL?
      9 001654  001004                          BNE     1$               ;BRANCH IF BCP.
     10
     11                                                                  ;BOP MODE - CRC-CCITT AND SECONDARY ADDRESS
     12 001656  012760  010135  000002          MOV     #SECADR!135,PCSAR(R0)
     13 001664  000403                          BR      2$
     14 001666                          1$:
     15                                                                  ;BCP MODE - CRC16 AND SYNCH CHARACTER
     16 001666  012760  041626  000002          MOV     #BCP!CRC16!226,PCSAR(R0)
     17 001674                          2$:
     18 001674  005767  176342                  TST     LOOP             ;WHAT LOOPBACK?
     19 001700  001004                          BNE     3$               ;BRANCH IF EXTERNAL LOOPBACK.
     20 001702  052760  000010  000004          BIS     #MAINT,TXCSR(R0);TURN ON MAINTENANCE MODE (INTERNAL LOOP)
     21 001710  000403                          BR      4$
     22 001712                          3$:
     23 001712  052760  000004  000000          BIS     #RTS,RXCSR(R0)  ;SET RTS IN ORDER TO USE TURNAROUND.
     24 001720                          4$:
     25 001720  052710  000120                  BIS     #RXEN!RXIE,(R0) ;TURN ON RECEIVER AND INT. ENABLE
     26 001724  052760  000020  000004          BIS     #TXEN,TXCSR(R0) ;ENABLE THE TRANSMITTER
     27 001732  000207                          RTS     PC               ;RETURN
     28
     29                                         ;READY SUBROUTINE TURNS ON THE TRANSMITTER
     30                                         ; R0 = CSR ADDRESS
     31 001734                          READY:
     32 001734  052760  000400  000006          BIS     #TSOM,TDSR(R0)  ;TURN ON TRANSMITTER
     33 001742  052760  000100  000004          BIS     #TXIE,TXCSR(R0) ;TURN ON TRANSMITTER INT. ENABLE
     34 001750  000207                          RTS     PC               ;RETURN
     35
     36
```

```
                                        ;SERVICE CODE FOR LINKING A PARTICULAR DEVICE
                                        ;TO A COMMON TRANSMIT OR RECEIVE INTERRUPT SERVICE ROUTINE.

001752                          LNKTAB:
                                .MACRO  $JS,$Q
                                        JSR     R5,RXISR        ;ANSWER FOR DEVICE '$Q REC ISR
                                        0                       ;CSR FOR DEVICE '$Q REC ISR
                                        RXBF'$Q                 ;BUFFER LOC FOR DEVICE '$Q
                                        BCNT'$Q                 ;BYTE COUNT ADDRESS FOR RXBF'$Q

                                        JSR     R5,TXISR        ;ANSWER FOR DEVICE '$Q TX ISR
                                        0                       ;CSR FOR DEVICE '$Q TX ISR
                                        TXBF'$Q                 ;DATA ADDRESS FOR DEVICE '$Q
                                        OFSET'$Q                        ;BYTE COUNT ADDRESS FOR DEVICE '$Q

                                .ENDM   $JS

                                $JS     $Q
                                .ENDM
001752  004567  177142                  JSR     R5,RXISR        ;ANSWER FOR DEVICE 0 REC ISR
001756  000000                          0                       ;CSR FOR DEVICE 0 REC ISR
001760  002172'                         RXBF0                   ;BUFFER LOC FOR DEVICE 0
001762  002206'                         BCNT0                   ;BYTE COUNT ADDRESS FOR RXBF0

001764  004567  176750                  JSR     R5,TXISR        ;ANSWER FOR DEVICE 0 TX ISR
001770  000000                          0                       ;CSR FOR DEVICE 0 TX ISR
001772  002156'                         TXBF0                   ;DATA ADDRESS FOR DEVICE 0
001774  002142'                         OFSET0                  ;BYTE COUNT ADDRESS FOR DEVICE 0

001776  004567  177116                  JSR     R5,RXISR        ;ANSWER FOR DEVICE 1 REC ISR
002002  000000                          0                       ;CSR FOR DEVICE 1 REC ISR
002004  002174'                         RXBF1                   ;BUFFER LOC FOR DEVICE 1
002006  002210'                         BCNT1                   ;BYTE COUNT ADDRESS FOR RXBF1

002010  004567  176724                  JSR     R5,TXISR        ;ANSWER FOR DEVICE 1 TX ISR
002014  000000                          0                       ;CSR FOR DEVICE 1 TX ISR
002016  002160'                         TXBF1                   ;DATA ADDRESS FOR DEVICE 1
002020  002144'                         OFSET1                  ;BYTE COUNT ADDRESS FOR DEVICE 1

002022  004567  177072                  JSR     R5,RXISR        ;ANSWER FOR DEVICE 2 REC ISR
002026  000000                          0                       ;CSR FOR DEVICE 2 REC ISR
002030  002176'                         RXBF2                   ;BUFFER LOC FOR DEVICE 2
002032  002212'                         BCNT2                   ;BYTE COUNT ADDRESS FOR RXBF2

002034  004567  176700                  JSR     R5,TXISR        ;ANSWER FOR DEVICE 2 TX ISR
002040  000000                          0                       ;CSR FOR DEVICE 2 TX ISR
002042  002162'                         TXBF2                   ;DATA ADDRESS FOR DEVICE 2
002044  002146'                         OFSET2                  ;BYTE COUNT ADDRESS FOR DEVICE 2

002046  004567  177046                  JSR     R5,RXISR        ;ANSWER FOR DEVICE 3 REC ISR
002052  000000                          0                       ;CSR FOR DEVICE 3 REC ISR
002054  002200'                         RXBF3                   ;BUFFER LOC FOR DEVICE 3
002056  002214'                         BCNT3                   ;BYTE COUNT ADDRESS FOR RXBF3

002060  004567  176654                  JSR     R5,TXISR        ;ANSWER FOR DEVICE 3 TX ISR
002064  000000                          0                       ;CSR FOR DEVICE 3 TX ISR
002066  002164'                         TXBF3                   ;DATA ADDRESS FOR DEVICE 3
```



```
002070  002150'                         OFSET3                  ;BYTE COUNT ADDRESS FOR DEVICE 3

002072  004567  177022                  JSR     R5,RXISR        ;ANSWER FOR DEVICE 4 REC ISR
002076  000000                          0                       ;CSR FOR DEVICE 4 REC ISR
002100  002202'                         RXBF4                   ;BUFFER LOC FOR DEVICE 4
002102  002216'                         BCNT4                   ;BYTE COUNT ADDRESS FOR RXBF4

002104  004567  176630                  JSR     R5,TXISR        ;ANSWER FOR DEVICE 4 TX ISR
002110  000000                          0                       ;CSR FOR DEVICE 4 TX ISR
002112  002166'                         TXBF4                   ;DATA ADDRESS FOR DEVICE 4
002114  002152'                         OFSET4                  ;BYTE COUNT ADDRESS FOR DEVICE 4

002116  004567  176776                  JSR     R5,RXISR        ;ANSWER FOR DEVICE 5 REC ISR
002122  000000                          0                       ;CSR FOR DEVICE 5 REC ISR
002124  002204'                         RXBF5                   ;BUFFER LOC FOR DEVICE 5
002126  002220'                         BCNT5                   ;BYTE COUNT ADDRESS FOR RXBF5

002130  004567  176604                  JSR     R5,TXISR        ;ANSWER FOR DEVICE 5 TX ISR
002134  000000                          0                       ;CSR FOR DEVICE 5 TX ISR
002136  002170'                         TXBF5                   ;DATA ADDRESS FOR DEVICE 5
002140  002154'                         OFSET5                  ;BYTE COUNT ADDRESS FOR DEVICE 5
```

```
                                        ;BUFFER AREAS
                                
                                .MACRO  $OF,$Q
                                OFSET'$Q:       .WORD   0       ;TRANSMITTER BYTE COUNT POINTER FOR DEVICE '$Q
                                .ENDM
                                $OF     $Q
                                .ENDM
002142  000000                  OFSET0: .WORD   0       ;TRANSMITTER BYTE COUNT POINTER FOR DEVICE 0
002144  000000                  OFSET1: .WORD   0       ;TRANSMITTER BYTE COUNT POINTER FOR DEVICE 1
002146  000000                  OFSET2: .WORD   0       ;TRANSMITTER BYTE COUNT POINTER FOR DEVICE 2
002150  000000                  OFSET3: .WORD   0       ;TRANSMITTER BYTE COUNT POINTER FOR DEVICE 3
002152  000000                  OFSET4: .WORD   0       ;TRANSMITTER BYTE COUNT POINTER FOR DEVICE 4
002154  000000                  OFSET5: .WORD   0       ;TRANSMITTER BYTE COUNT POINTER FOR DEVICE 5

                                .MACRO  $TB,$Q
                                TXBF'$Q:        .WORD   0       ;TRANSMITTER BUFFER FOR DEVICE '$Q
                                .ENDM
                                $TB     $Q
                                .ENDM
002156  000000                  TXBF0:  .WORD   0       ;TRANSMITTER BUFFER FOR DEVICE 0
002160  000000                  TXBF1:  .WORD   0       ;TRANSMITTER BUFFER FOR DEVICE 1
002162  000000                  TXBF2:  .WORD   0       ;TRANSMITTER BUFFER FOR DEVICE 2
002164  000000                  TXBF3:  .WORD   0       ;TRANSMITTER BUFFER FOR DEVICE 3
002166  000000                  TXBF4:  .WORD   0       ;TRANSMITTER BUFFER FOR DEVICE 4
002170  000000                  TXBF5:  .WORD   0       ;TRANSMITTER BUFFER FOR DEVICE 5

                                .MACRO  $RB,$Q
                                RXBF'$Q:        .WORD   0       ;RECEIVER BUFFER FOR DEVICE #'$Q
                                .ENDM
                                $RB     $Q
                                .ENDM
002172  000000                  RXBF0:  .WORD   0       ;RECEIVER BUFFER FOR DEVICE #0
002174  000000                  RXBF1:  .WORD   0       ;RECEIVER BUFFER FOR DEVICE #1
002176  000000                  RXBF2:  .WORD   0       ;RECEIVER BUFFER FOR DEVICE #2
002200  000000                  RXBF3:  .WORD   0       ;RECEIVER BUFFER FOR DEVICE #3
002202  000000                  RXBF4:  .WORD   0       ;RECEIVER BUFFER FOR DEVICE #4
002204  000000                  RXBF5:  .WORD   0       ;RECEIVER BUFFER FOR DEVICE #5

                                .MACRO  $BC,$Q
                                BCNT'$Q:        .WORD   0       ;RECEIVER BYTE COUNT POINTER FOR DEVICE '$Q
                                .ENDM
                                $BC     $Q
                                .ENDM
002206  000000                  BCNT0:  .WORD   0       ;RECEIVER BYTE COUNT POINTER FOR DEVICE 0
002210  000000                  BCNT1:  .WORD   0       ;RECEIVER BYTE COUNT POINTER FOR DEVICE 1
002212  000000                  BCNT2:  .WORD   0       ;RECEIVER BYTE COUNT POINTER FOR DEVICE 2
002214  000000                  BCNT3:  .WORD   0       ;RECEIVER BYTE COUNT POINTER FOR DEVICE 3
002216  000000                  BCNT4:  .WORD   0       ;RECEIVER BYTE COUNT POINTER FOR DEVICE 4
002220  000000                  BCNT5:  .WORD   0       ;RECEIVER BYTE COUNT POINTER FOR DEVICE 5


                                        ;ASCII MESSAGES
002222  045  104  120  MHUNG:  .ASCIZ  '%DPV11 MODULE IS HUNG - SEE LISTING%'
```



```
                                .EVEN
000001                          .END
```

SYMBOL TABLE

```
ACSR    000102R     COUNT   000230R     OFSET0  002142R     RES1    000056R     SR4     000024R
ACTIV   000624R     CRC16 = 001400      OFSET1  002144R     RES2    000060R     START   000244R
ADDR    000006R     CSRA    000100R     OFSET2  002146R     ROVER = 004000      STAT    000026R
ADDR22= 001000      DATCKS= 104411      OFSET3  002150R     RSUM  = 000400      SVR0    000062R
ASB     000106R     DATERS= 104404      OFSET4  002152R     RSTRT   000112R     SVR1    000064R
ASTAT   000104R     DROP    000324R     OFSET5  002154R     RTS   = 000004      SVR2    000066R
AWAS    000110R     DVID1   000014R     OPEN  = 000000      RXACT = 004000      SVR3    000070R
BCNT0   002206R     ENDITS= 104413      OTOA$ = 104420      RXBF    000232R     SVR4    000072R
BCNT1   002210R     ENDS  = 104410      PASCNT  000034R     RXBF0   002172R     SVR5    000074R
BCNT2   002212R     ERR   = 100000      PCSAR = 000002      RXBF1   002174R     SVR6    000076R
BCNT3   002214R     ERRTYP  000106R     PIRQ$ = 000004      RXBF2   002176R     SYSCNT  000052R
BCNT4   002216R     EXITS = 104400      POPSP = 005726      RXBF3   002200R     TABORT= 002000
BCNT5   002220R     GETPAS= 104415      POPSP2= 022626      RXBF4   002202R     TBE   = 000004
BCP   = 040000      GWBUFS= 104414      PRELIM  001642R     RXBF5   002204R     TDSR  = 000006
BEGIN   000000R     HRDCNT  000044R     PRTY  = 000000      RXCSR = 000000      TEOM  = 001000
BIT0  = 000001      HRDERS= 104405      PRTY0 = 000000      RXEN  = 000020      TERR  = 100000
BIT1  = 000002      HRDPAS  000050R     PRTY1 = 000040      RXIE  = 000100      TOTAL   000226R
BIT10 = 002000      HUNG    000734R     PRTY2 = 000100      RXISR   001120R     TRPDFD= 000022
BIT11 = 004000      ICONT   000036R     PRTY3 = 000140      RXSTAT= 002000      TSOM  = 000400
BIT12 = 010000      ICOUNT  000040R     PRTY4 = 000200      R6    =%000006      TXACT = 000002
BIT13 = 020000      IDNUM   000122R     PRTY5 = 000240      R7    =%000007      TXBF0   002156R
BIT14 = 040000      INIT    000030R     PRTY6 = 000300      SAVBF   000234R     TXBF1   002160R
BIT15 = 100000      INTR    000120R     PRTY7 = 000340      SBADR   000102R     TXBF2   002162R
BIT2  = 000004      LNKTAB  001752R     PS    = 177776      SCAN    000670R     TXBF3   002164R
BIT3  = 000010      LOOP    000242R     PSW   = 177776      SECADR= 010000      TXBF4   002166R
BIT4  = 000020      MAINT = 000010      PUSH  = 005746      SELECT  000236R     TXBF5   002170R
BIT5  = 000040      MAP22S= 104416      PUSH2 = 024646      SETUP1  000500R     TXCSR = 000004
BIT6  = 000100      MHUNG   002222R     PWRFLG= 000002      SETUP2  000566R     TXEN  = 000020
BIT7  = 000200      MODE    000240R     RABORT= 002000      SOFCNT  000042R     TXIE  = 000100
BIT8  = 000400      MODNAM  000000R     RAND$ = 104417      SOFERS= 104406      TXISR   000740R
BIT9  = 001000      MODSP   000224R     RANNUM  000054R     SOFPAS  000046R     VECTOR  000010R
BREAKS= 104407      MSGN$ = 104403      RDSR  = 000002      SPOINT  000032R     WAIT    000664R
BR1     000012R     MSGS$ = 104402      READY   001734R     SPSIZ = 000040      WASADR  000104R
BR2     000013R     MSG$  = 104401      REUM  = 001000      SR1     000016R     WDFR    000116R
BTOD$ = 104421      NULL  = 000000      RESET = 000001      SR2     000020R     WDTO    000114R
CDATA$= 104412      N.DPVS  000224R     RESTRT  000330R     SR3     000022R     XFLAG   000005R
CONFIG  000056R
```

```
. ABS.  000000     000
        002270     001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  5509 WORDS  ( 22 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  69 PAGES
XDPVA0,CXDPVA.SEQ/C=#DDXCOM.P11,CXDPVA.P11
```


CROSS REFERENCE TABLE (CREF V01-05)

```
ACSR      4-9#      7-9*      8-10*     8-35*     8-74*     8-88*
ACTIV     6-85      6-92#
ADDR      4-9#      6-33      6-80      6-95
ADDR22    4-9#
ASB       4-9#      8-51*
ASTAT     4-9#      7-10*     8-11*     8-36*     8-75*     8-89*
AWAS      4-9#      8-52*
BCNT0    10-20     11-32#
BCNT1    10-20     11-32#
BCNT2    10-20     11-32#
BCNT3    10-20     11-32#
BCNT4    10-20     11-32#
BCNT5    10-20     11-32#
BCP       5-50#     9-16
BEGIN     4-9#      6-23      6-114     6-114     6-119     6-125     6-126     7-13      7-16      8-15      8-18      8-40      8-43      8-55
          8-57      8-79      8-82      8-93      8-96
BIT0      4-9#      5-18#     5-60      6-5
BIT1      4-9#      5-17#     5-59      6-12
BIT10     4-9#      5-8#      5-37      5-45      5-64
BIT11     4-9#      5-7#      5-36      5-44
BIT12     4-9#      5-6#      5-51
BIT13     4-9#      5-5#
BIT14     4-9#      5-4#      5-50
BIT15     4-9#      5-3#      5-43      5-63
BIT2      4-9#      5-16#     5-40      5-58
BIT3      4-9#      5-15#     5-57
BIT4      4-9#      5-14#     5-39      5-56
BIT5      4-9#      5-13#
BIT6      4-9#      5-12#     5-38      5-55
BIT7      4-9#      5-11#
BIT8      4-9#      5-10#     5-47      5-52      5-66
BIT9      4-9#      5-9#      5-46      5-52      5-65
BR1       4-9#      6-48
BR2       4-9#      6-53
BREAKS    4-9#      6-114     6-114
BTOD$     4-9#
CDATA$    4-9#
CONFIG    4-9#
COUNT     5-71#
CRC16     5-52#     9-16
CSRA      4-9#      7-8*      8-9*      8-34*     8-49*     8-73*     8-87*
DATCK$    4-9#
DATER$    4-9#      8-57
DROP      6-22#     6-28      6-30
DVID1     4-9#      6-20
ENDS      4-9#      6-23      6-126
ENDITS    4-9#      6-119
ERR       5-43#     8-64      8-68
ERRTYP    4-9#      7-15*     8-17*     8-42*     8-81*     8-95*
EXITS     4-9#
GETPAS    4-9#
GWBUFS    4-9#
HRDCNT    4-9#
HRDERS    4-9#      8-18      8-43      8-82      8-96
HRDPAS    4-9#
HUNG      6-125     6-127#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
ICONT    4-9#
ICOUNT   4-9#
IDNUM    4-9#
INIT     4-9#
INITK    4-9#     6-61*    6-64*
LNKTAB   6-34     10-4#
LOOP     5-77#    6-14*    6-18*    9-18
MAINT    5-57#    9-20
MAP22$   4-9#
MHUNG    6-128    11-40#
MODE     5-76#    6-7*     6-10*    8-62     9-8
MODNAM   4-9#
MODSP    4-9      4-9#
MSGS     4-9#
MSGN$    4-9#     6-125
MSGS$    4-9#
N.DPVS   5-69#    6-31*    6-46*    6-61     6-62     6-63     6-94
NULL     4-9#     7-16     8-18     8-43     8-82     8-96
OFSET0   10-20    11-8#
OFSET1   10-20    11-8#
OFSET2   10-20    11-8#
OFSET3   10-20    11-8#
OFSET4   10-20    11-8#
OFSET5   10-20    11-8#
OPEN     4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9
         4-9      4-9      4-9      4-9      4-9      4-9      4-9#     5-69     5-70     5-71     5-72     5-73     5-74
OTOAS    4-9#
PASCNT   4-9#
PCSAR    5-30#    9-12*    9-16*
PIRQS    4-9#     7-13     8-15     8-40     8-55     8-79     8-93
POPSP    4-9#
POPSP2   4-9#
PRELIM   6-89     9-6#
PRTY     4-9#
PRTY0    4-9#
PRTY1    4-9#
PRTY2    4-9#
PRTY3    4-9#
PRTY4    4-9#
PRTY5    4-9      4-9      4-9#
PRTY6    4-9#
PRTY7    4-9#
PS       4-9#
PSW      4-9#
PUSH     4-9#
PUSH2    4-9#
PWRFLG   4-9#
R6       4-9#
R7       4-9#
RABORT   5-45#    8-23
RAND$    4-9#
RANNUM   4-9#
RDSR     5-29#    8-11     8-21     8-36     8-89
READY    6-104    9-31#
REOM     5-46#    8-70
RES1     4-9#
RES2     4-9#
```


CROSS REFERENCE TABLE (CREF V01-05)

```
RESET    5-60#    7-11     8-12     8-37     8-53     8-76     8-90     8-99     9-7
RESTRT   4-9      6-21     6-26#    6-121
ROVER    5-44#    8-23
RSOM     5-47#
RSTRT    4-9#
RTS      5-40#    9-23
RXACT    5-36#
RXBF     5-72#    6-35*
RXBF0    6-35     6-72     10-20    11-24#
RXBF1    10-20    11-24#
RXBF2    10-20    11-24#
RXBF3    10-20    11-24#
RXBF4    10-20    11-24#
RXBF5    10-20    11-24#
RXCSR    5-28#    9-23*
RXEN     5-39#    9-25
RXIE     5-38#    9-25
RXISR    8-2#     10-20    10-20    10-20    10-20    10-20    10-20
RXSTAT   5-37#
SAVBF    5-73#    8-21*    8-23     8-28     8-50     8-52     8-64     8-68     8-70     8-75
SBADR    4-9#     8-48*
SCAN     6-112#   7-17     8-19     8-44     8-58     8-83     8-97
SECADR   5-51#    9-12
SELECT   5-74#    6-20*    6-27     6-29     6-81     6-93
SETUP1   6-39     6-60#
SETUP2   6-79#
SOFCNT   4-9#
SOFERR   4-9#     7-16
SOFPAS   4-9#
SPOINT   4-9#
SPSIZ    1-23#    4-9
SR1      4-9#     6-5      6-12
SR2      4-9#
SR3      4-9#
SR4      4-9#
START    4-9      6-4#
STAT     4-9#
SVR0     4-9#
SVR1     4-9#
SVR2     4-9#
SVR3     4-9#
SVR4     4-9#
SVR5     4-9#
SVR6     4-9#
SYSCNT   4-9#
TABORT   5-64#
TBE      5-58#
TDSR     5-32#    7-6      7-10     7-22*    7-30*    7-36*    9-32*
TEOM     5-65#    7-36
TERR     5-63#    7-6
TOTAL    5-70#    6-94*    6-115    8-100*
TRPDFD   4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9
         4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9
         4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9#     4-9#     4-9#     4-9#     4-9#     4-9#
         4-9#     4-9#     4-9#     4-9#     4-9#     4-9#     4-9#     4-9#     4-9#     4-9#     4-9#     4-9#     4-9#
TSOM     5-66#    7-22     9-32
TXACT    5-59#
```

```
TXBF0    6-66    10-20    11-16#
TXBF1   10-20    11-16#
TXBF2   10-20    11-16#
TXBF3   10-20    11-16#
TXBF4   10-20    11-16#
TXBF5   10-20    11-16#
TXCSR    5-31#    7-11*    7-37*    8-12*    8-37*    8-53*    8-76*    8-90*    8-99*    9-7*    9-20*    9-26*    9-33*
TXEN     5-56#    9-26
TXIE     5-55#    7-37     9-33
TXISR    7-2#    10-20    10-20    10-20    10-20    10-20    10-20
VECTOR   4-9#     6-32
WAIT     6-99     6-108#
WASAVR   4-9#     8-50*
WDFR     4-9#     6-63*
WDTO     4-9#     6-62*
XFLAG    4-9#
```



```
$BC      11-27#   11-32    11-32    11-32    11-32    11-32    11-32
$JS      10-5#    10-20    10-20    10-20    10-20    10-20    10-20
$QF      11-3#    11-8     11-8     11-8     11-8     11-8     11-8
$POP      5-88#    7-12     7-39     8-14     8-39     8-54     8-76     8-92     8-102
$PUSH     5-80#    7-3      8-3
$RB      11-19#   11-24    11-24    11-24    11-24    11-24    11-24
$TB      11-11#   11-16    11-16    11-16    11-16    11-16    11-16
BKMOD     1-111#
BREAK     1-201#   6-114
BTOD      1-220#
CKDATA    1-256#
DATACK    1-265#
DATERR    1-154#   8-57
DFSEVN    1-288#   4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9      4-9
DSEVNT    1-298#   4-9      4-9      4-9      4-9      4-9      4-9
END       1-191#   6-23     6-126
ENDIT     1-182#   6-119
ENDMOD    1-187#
EQUATS    1-306#   4-9
EXIT      1-136#
GETPA     1-247#
GWBUFF    1-235#
HRDER     1-144#   8-18     8-43     8-82     8-96
IOMOD     1-107#   4-9
IOMODP    1-131#
IOMODR    1-127#
IOMODX    1-123#
MAP22     1-251#
MODULE    1-24#    4-9
MSG       1-170#
MSGN      1-174#   6-125
MSGS      1-178#
NBKMOD    1-119#
OTOA      1-206#
PIRQ      1-195#   7-13     8-15     8-40     8-55     8-79     8-93
RAND      1-140#
SBKMOD    1-115#
SOFER     1-160#   7-16
```

```
000000                          .REPT 0


                                        IDENTIFICATION
                                        --------------


                                PRODUCT CODE:   AC-S054A-MC

                                PRODUCT NAME:   CXVTCA0 VTV30-K CSS DEC/X MOD

                                DATE CREATED:   20 MAR 80

                                AUTHOR:         PAUL TAYLOR

                                MAINTAINER:     COMPUTER SPECIAL SYSTEMS,
                                               DIGITAL EQUIPMENT CO.
                                               LTD.,
                                               READING, BERKS, U.K.
```

```
                                PROGRAM HISTORY:

                        VO.   AUTHOR       DATE        PURPOSE

                        01    P. TAYLOR    MAR-80      ORIGINAL


                        1.          ABSTRACT

                        VTCA IS A BKMOD THAT EXERCISES THE VTV30-K COLOUR
                        GRAPHICS CONTROLLER BY DISPLAYING MOVING DIAGONAL LINES
                        OF CHARACTERS ACROSS THE DISPLAY.  COLOUR INFORMATION IS
                        DISPLAYED IN VERTICAL STRIPES.


                        2.          REQUIREMENTS

                        2.1         HARDWARE

                        EACH MODULE WILL EXERCISE ONE VTV30-K DISPLAY
                        CONTROLLER.


                        2.2         STORAGE

                        VTCA REQUIRES 1286 (DECIMAL) WORDS.


                        3.          PASS DEFINITION

                        ONE PASS OF VTCA CONSISTS OF 43 CHANGES OF PICTURE.


                        4.          EXECUTION TIME

                        ONE PASS OF VTCA TAKES TWENTY SECONDS WHEN RUNNING STAND
                        ALONE ON AN LSI-11.


                        5.          CONFIGURATION REQUIREMENTS

                        5.1         DEFAULT PARAMETERS

                        DVADR:      174000


                        5.2         REQUIRED PARAMETERS

                        SR1 MUST BE SET UP TO INDICATE THE TYPE OF CONTROLLER
                        BEING TESTED:-

                                    M7368               SR1 = 2
                                    M7369               SR1 = 3
```

```
                                M7370               SR1 = 0
                                M7371               SR1 = 1
                                M7372               SR1 = 2

IN ADDITION THE VALUE 4 MUST BE ADDED TO SR1
IF A T.V. ENCODER IS USED.


6.          DEVICE/OPTION SETUP

THE DEVICE MUST BE ABLE TO CONNECT TO A COLOUR T.V.
MONITOR.


7.          MODULE OPERATION

A.  SET UP THE DEVICE CSR AND DBUF ADDRESSES.

B.  CHECK FOR 525/625 LINE OPERATION AND SET UP THE
    MAXIMUM X AND Y CO-ORDINATES.

C.  IF THE DEVICE HAS A RAM CHARACTER SET, LOAD THE
    CAHRACTER SET FROM PROGRAM MEMORY.

D.  STARTING AT THE TOP LEFT HAND CORNER OF THE DISPLAY,
    PUT OUT INCREMENTING CHARACTER CODES BEGINNING ON
    ROW 0.  THIS WILL APPEAR AS DIAGONAL LINES OF
    IDENTICAL CHARACTERS RUNNING FROM TOP RIGHT TO
    BOTTOM LEFT, THE CHARACTER CODES INCREMENTING FROM
    LEFT TO RIGHT.

E.  THE COLOUR CODES ARE DISPLAYED AS VERTICAL LINES.
    STARTING WITH RED BACKGROUND, THE FOREGROUND COLOURS
    OF RED, GREEN, THEN BLUE ARE DISPLAYED.  THE
    BACKGROUND COLOUR IS THEN INCREMENTED AND THE
    FOREGROUND COLOUR SEQUENCE IS REPEATED.

F.  ONCE THE WHOLE DISPLAY HAS BEEN FILLED, INCREMENT
    THE STARTING CHARACTER CODE, AND CHECK FOR END OF
    PASS.  THUS, UNTIL END OF PASS OCCURS, THE DISPLAY
    WILL BE UPDATED STARTING WITH THE NEXT CHARACTER
    CODE.  THIS WILL APPEAR AS IF THE DIAGONAL LINES ARE
    MOVING FROM RIGHT TO LEFT.

G.  IF NO END OF PASS OCCURS, RETURN TO D.  OTHERWISE,
    RETURN TO A.


8.          OPERATION OPTIONS

THE SR1 SETTINGS MUST BE SET UP TO MATCH THE HARDWARE
SETTINGS.


9.          NON STANDARD PRINTOUT
```

```
   520                                  NONE.
   521                                  .ENDR
```

```
523  000000'                            BKMOD   <VTCA>,174000,,,,,43.,
(1)  000000'                            MODULE  40020,VTCA,174000,,,,,43.,
(2)                                     .TITLE  VTCA DEC/X11 SYSTEM EXERCISER MODULE
(2)                                ;    DDXCOM  VERSION  6       23-MAY-78
(2)                                     .LIST   BIN
(2)                                ;*****************************************************************************
(2)  000000'                       BEGIN:
(2)  000000' 052126  040503   040  MODNAM: .ASCII  /VTCA / ;MODULE NAME.
(2)  000005'    000                XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
(2)  000006' 174000                ADDR:   174000+0                ;1ST DEVICE ADDR.
(2)  000010' 000000                VECTOR: +0                      ;1ST DEVICE VECTOR.
(2)  000012'    000                BR1:    .BYTE   PRTY+0          ;1ST BR LEVEL.
(2)  000013'    000                BR2:    .BYTE   PRTY+0          ;2ND BR LEVEL.
(2)  000014' 000001                DVID1:  +1                      ;DEVICE INDICATOR 1.
(2)  000016' 000000                SR1:    OPEN                    ;SWITCH REGISTER 1
(2)  000020' 000000                SR2:    OPEN                    ;SWITCH REGISTER 2
(2)  000022' 000000                SR3:    OPEN                    ;SWITCH REGISTER 3
(2)  000024' 000000                SR4:    OPEN                    ;SWITCH REGISTER 4
(2)                                ;*****************************************************************************
(2)  000026' 040020                STAT:   40020                   ;STATUS WORD.
(2)  000030' 000222'               INIT:   START                   ;MODULE START ADDR.
(2)  000032' 000222'               SPOINT: MODSP                   ;MODULE STACK POINTER.
(2)  000034' 000000                PASCNT: 0                       ;PASS COUNTER.
(2)  000036' 000053                ICONT:  43.                     ;# OF ITERATIONS PER PASS=43.
(2)  000040' 000000                ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
(2)  000042' 000000                SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
(2)  000044' 000000                HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
(2)  000046' 000000                SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
(2)  000050' 000000                HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
(2)  000052' 000000                SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
(2)  000054' 000000                RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
(2)  000056'                       CONFIG:                         ;RESERVED FOR MONITOR USE
(2)  000056' 000000                RES1:   0                       ;RESERVED FOR MONITOR USE
(2)  000060' 000000                RES2:   0                       ;RESERVED FOR MONITOR USE
(2)  000062' 000000                SVR0:   OPEN                    ;LOC TO SAVE R0.
(2)  000064' 000000                SVR1:   OPEN                    ;LOC TO SAVE R1.
(2)  000066' 000000                SVR2:   OPEN                    ;LOC TO SAVE R2.
(2)  000070' 000000                SVR3:   OPEN                    ;LOC TO SAVE R3.
(2)  000072' 000000                SVR4:   OPEN                    ;LOC TO SAVE R4.
(2)  000074' 000000                SVR5:   OPEN                    ;LOC TO SAVE R5.
(2)  000076' 000000                SVR6:   OPEN                    ;LOC TO SAVE R6.
(2)  000100' 000000                CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
(2)  000102'                       SBADR:                          ;ADDR OF GOOD DATA, OR
(2)  000102' 000000                ACSR:   OPEN                    ;CONTENTS OF CSR.
(2)  000104'                       WASADR:                         ;ADDR OF BAD DATA, OR
(2)  000104' 000000                ASTAT:  OPEN                    ;STATUS REG CONTENTS.
(2)  000106'                       ERRTYP:                         ;TYPE OF ERROR
(2)  000106' 000000                ASB:    OPEN                    ;EXPECTED DATA.
(2)  000110' 000000                AWAS:   OPEN                    ;ACTUAL DATA.
(2)  000112' 000222'               RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
(2)  000114' 000000                WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
(2)  000116' 000000                WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
(2)  000120' 000000                INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
(2)  000122'                       IDNUM:                          ;MODULE IDENTIFICATION NUMBER=
(2)           000040                       .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
(2)                                        .NLIST
```



```
(2)                                        .WORD   0
(2)                                        .LIST
(3)                                        .ENDR
(2)  000222'                       MODSP:
(2)                                ;*****************************************************************************
```

```
  525  000222'                          START:
  526  000222' 016700  177560           RESTRT: MOV     ADDR,R0         ; GET DEVICE ADDRESS
  527  000226' 010067  000522                   MOV     R0,DEVCSR       ; SAVE IT
  528  000232' 005720                           TST     (R0)+           ; CHECK FOR NXM
  529  000234' 010067  000516                   MOV     R0,DEVBUF       ; SAVE DBUF ADDRESS
  530  000240' 005710                           TST     (R0)            ; CHECKIT
  531  000242' 012767  000060  000530           MOV     #48.,XMAX       ; SET XMAX
  532  000250' 012767  000037  000524           MOV     #31.,YMAX       ; ASSUME 625 OPERATION
  533  000256' 032767  000001  177532           BIT     #1,SR1          ; 525 SELECTED ?
  534  000264' 001403                           BEQ     T0601           ; NO
  535  000266' 012767  000033  000506           MOV     #27.,YMAX       ; YES- SELECT ITS LENGTH
  536  000274' 032767  000002  177514  T0601:   BIT     #2,SR1          ; IS THERE A PROM ON BOARD
  537  000302' 001045                           BNE     T0604           ; YES DON'T LOAD CHSR
  538                                   ;
  539                                   ; RAM CHARACTER SET NEEDS LOADING
  540                                   ;
  541  000304' 012700  001014'                  MOV     #P02CHR,R0      ; GET START OF DATA TO LOAD
  542  000310' 032767  000004  177500           BIT     #4,SR1          ; T.V. ENCODED?
  543  000316' 001402                           BEQ     5$              ; ....NO IF BRANCH.
  544  000320' 012700  003014'                  MOV     #P03CHR,R0      ; PICK UP T.V. CHAR SET.
  545                                   ;
  546  000324' 005067  000442           5$:     CLR     CHAR            ; START AT CHARACTER ZERO
  547  000330' 005001                   T0602:  CLR     R1              ; 8 LINES PER CHARACTER
  548  000332' 016702  000434           T0603:  MOV     CHAR,R2         ; GET CHAR CODE
  549  000336' 006302                           ASL     R2
  550  000340' 006302                           ASL     R2              ; SET IN RIGHT PLACE
  551  000342' 006302                           ASL     R2
  552  000344' 050102                           BIS     R1,R2           ; ADD IN LINE NUMBER
  553  000346' 052702  100000                   BIS     #100000,R2      ; SELECT CHSR
  554  000352' 010277  000376                   MOV     R2,@DEVCSR      ; LOAD CSR
  555  000356' 112003                           MOVB    (R0)+,R3        ; GET DATA
  556  000360' 000303                           SWAB    R3
  557  000362' 105003                           CLRB    R3              ; ENSURE 8 BITS LOADED
  558  000364' 000303                           SWAB    R3
  559  000366' 010377  000364                   MOV     R3,@DEVBUF      ; LOAD DATA OUT
  560  000372' 005201                           INC     R1              ; UPDATE LINE NUMBER
  561  000374' 020127  000007                   CMP     R1,#7           ; ALL LINES DONE
  562  000400' 003754                           BLE     T0603           ; NO
  563  000402' 005267  000364                   INC     CHAR            ; UPDATE CHAR COUNT
  564  000406' 026727  000360  000177           CMP     CHAR,#177       ; ALL DONE
  565  000414' 003745                           BLE     T0602           ; NOT YET
  566                                   ;
  567                                   ; NOW START PUTTING UP PICTURE
  568                                   ;
  569  000416'                          T0604:
  570  000416' 012767  040000  000340           MOV     #40000,VIDEON   ; ENABLE VIDEO
  571  000424' 005067  000346           T0605:  CLR     CARY            ; START AT FIRST ROW
  572  000430' 016767  000352  000346           MOV     BASE1,BASE      ; SET THIS PICTURE START
  573  000436' 016767  000342  000326   T0606:  MOV     BASE,CHAR       ; SET START FOR THIS ROW
  574  000444' 005067  000324                   CLR     CARX            ; START OF COLUMN
  575  000450' 005067  000312           T0607:  CLR     FLASH           ; NO FLASH
  576  000454' 005067  000310           T0608:  CLR     BLINK           ; NO BLINK
  577  000460' 012767  000001  000274   T0609:  MOV     #1,BAKGND       ; BLACK ON
  578  000466' 012767  000001  000264   T0610:  MOV     #1,FORGND       ; BLACK
  579  000474' 016767  000274  000306   T0611:  MOV     CARX,SARDAT     ; SET UP OUR SAR REGISTER
  580  000502' 016701  000270                   MOV     CARY,R1
```



```
  581  000506' 000301                           SWAB    R1              ; GET Y ADDRESS IN TOP BYTE
  582  000510' 105001                           CLRB    R1
  583  000512' 006201                           ASR     R1
  584  000514' 006201                           ASR     R1              ; SHIFT IT DOWN
  585  000516' 042701  174000                   BIC     #174000,R1      ; CLEAR RUBBISH
  586  000522' 056701  000236                   BIS     VIDEON,R1       ; SET VIDEO ON
  587  000526' 050167  000256                   BIS     R1,SARDAT       ; THEN SET IT UP
  588  000532' 016777  000252  000214           MOV     SARDAT,@DEVCSR  ; SET OUR X AND Y ADDRESSES
  589  000540' 016767  000214  000244           MOV     FORGND,SDRDAT   ; START WITH FOREGROUND
  590  000546' 016701  000210                   MOV     BAKGND,R1       ; GET BACKGROUND COLOUR
  591  000552' 006301                           ASL     R1
  592  000554' 006301                           ASL     R1              ; SET IT IN CORRECT POSITION
  593  000556' 006301                           ASL     R1
  594  000560' 050167  000226                   BIS     R1,SDRDAT       ; ADD TO COMPOSITE
  595  000564' 056767  000200  000220           BIS     BLINK,SDRDAT    ; SET BLINK IF REQUIRED
  596  000572' 056767  000170  000212           BIS     FLASH,SDRDAT    ; AND FLASH
  597  000600' 016701  000166                   MOV     CHAR,R1         ; GET CHARACTER
  598  000604' 000301                           SWAB    R1              ; THE SET IN THE CORRECT POSITION
  599  000606' 105001                           CLRB    R1
  600  000610' 050167  000176                   BIS     R1,SDRDAT       ; ADD IT IN
  601  000614' 016777  000172  000134           MOV     SDRDAT,@DEVBUF  ; OUT
  602  000622' 005267  000144                   INC     CHAR            ; SELECT NEXT ONE
  603  000626' 042767  177600  000136           BIC     #177600,CHAR    ; IN RANGE 0-177
  604  000634' 005267  000134                   INC     CARX            ; GET NEXT COORD
  605  000640' 026767  000130  000132           CMP     CARX,XMAX       ; END OF ROW ?
  606  000646' 003424                           BLE     T0612           ; NOT YET
  607  000650' 005267  000130                   INC     BASE            ; ELSE SET NEW ROW STARTER
  608  000654' 042767  177600  000122           BIC     #177600,BASE    ; IN RANGE 0-177
  609  000662' 005267  000110                   INC     CARY            ; UPDATE ROW
  610  000666' 026767  000104  000106           CMP     CARY,YMAX       ; END OF DISPLAY ?
  611  000674' 003660                           BLE     T0606           ; NOT YET
  612  000676' 005267  000104                   INC     BASE1           ; SET NEW PICTURE STARTER
  613  000702' 042767  177600  000076           BIC     #177600,BASE1   ; IN RANGE 0-177
  614                                   ;
  615                                   ;
  616                                   ; SIGNAL END OF PASS
  617                                   ;
  618  000710' 104413  000000'                  ENDITS,BEGIN            ;SIGNAL END OF ITERATION.
  (1)                                                                   ;MONITOR SHALL TEST END OF PASS
  619  000714' 000167  177504                   JMP     T0605           ; KEEP GOING
  620                                   ;
  621  000720' 006367  000034           T0612:  ASL     FORGND          ; GET NEW FOREGROUND
  622  000724' 026727  000030  000004           CMP     FORGND,#4       ; ALL DONE
  623  000732' 003660                           BLE     T0611           ; NO
  624  000734' 005267  000022                   INC     BAKGND          ; GET NEXT BACK GROUND
  625  000740' 026727  000016  000007           CMP     BAKGND,#7       ; ALL DONE
  626  000746' 003647                           BLE     T0610           ; NO
  627  000750' 000167  177474                   JMP     T0607           ; YES
```

```
                                        .SBTTL  DATA TABLES
                                ;
000754' 160010                  DEVCSR: 160010                  ; CSR ADDRESS
000756' 160012                  DEVBUF: 160012                  ; BUFFER ADDRESS
000760' 000000                  FORGND: 0                       ; FOREGROUND
000762' 000000                  BAKGND: 0                       ; BACK GROUND
000764' 000000                  VIDEON: 0                       ; VIDEO ON
000766' 000000                  FLASH:  0                       ; FLASH
000770' 000000                  BLINK:  0                       ; BLINK
000772' 000000                  CHAR:   0                       ; CHARACTER CODE TO DISPLAY
000774' 000000                  CARX:   0                       ; CURRENT X COORD
000776' 000000                  CARY:   0                       ; CURRENT Y COORRD
001000' 000000                  XMAX:   0                       ; MAX X COORD
001002' 000000                  YMAX:   0                       ; MAX Y COORD
001004' 000000                  BASE:   0                       ; LINE START CODE
001006' 000000                  BASE1:  0                       ; PICTURE START CHAR
001010' 000000                  SARDAT: 0                       ; TEMP CSR DATA
001012' 000000                  SDRDAT: 0                       ; TEMP DBUF DATA
```



```
                                        .SBTTL  STANDARD CHARACTER SET
                                        .NLIST BEX
001014'                         P02CHR:
001014'  360  360  360                  .BYTE   360,360,360,360,360,360,360,360
001024'  010  010  010                  .BYTE   010,010,010,010,010,010,010,010
001034'  030  030  030                  .BYTE   030,030,030,030,030,030,030,030
001044'  074  074  074                  .BYTE   074,074,074,074,074,074,074,074
001054'  000  000  000                  .BYTE   000,000,000,000,377,000,000,000
001064'  000  000  000                  .BYTE   000,000,000,000,377,377,000,000
001074'  000  000  377                  .BYTE   000,000,377,377,377,377,000,000
001104'  010  010  010                  .BYTE   010,010,010,010,377,010,010,010
001114'  030  030  030                  .BYTE   030,030,030,377,377,030,030,030
001124'  074  074  377                  .BYTE   074,074,377,377,377,377,074,074 ;10
001134'  001  002  004                  .BYTE   001,002,004,010,020,040,100,200
001144'  200  100  040                  .BYTE   200,100,040,020,010,004,002,001
001154'  000  014  022                  .BYTE   000,014,022,040,170,040,100,176
001164'  201  102  044                  .BYTE   201,102,044,030,030,044,102,201
001174'  074  176  377                  .BYTE   074,176,377,377,377,377,176,074
001204'  030  074  176                  .BYTE   030,074,176,377,377,176,074,034
001214'  010  010  024                  .BYTE   010,010,024,042,301,042,024,010
001224'  010  010  034                  .BYTE   010,010,034,076,377,076,034,010
001234'  252  252  252                  .BYTE   252,252,252,252,252,252,252,252
001244'  314  314  314                  .BYTE   314,314,314,314,314,314,314,314 ;20
001254'  377  000  377                  .BYTE   377,000,377,000,377,000,377,000
001264'  377  377  000                  .BYTE   377,377,000,000,377,377,000,000
001274'  377  203  205                  .BYTE   377,203,205,211,221,241,301,377
001304'  377  201  201                  .BYTE   377,201,201,201,201,201,201,377
001314'  377  376  374                  .BYTE   377,376,374,370,360,340,300,200
001324'  377  177  077                  .BYTE   377,177,077,037,017,007,003,001
001334'  200  300  340                  .BYTE   200,300,340,360,370,374,376,377
001344'  001  003  007                  .BYTE   001,003,007,017,037,077,177,377
001354'  003  007  016                  .BYTE   003,007,016,034,070,160,340,300
001364'  300  340  260                  .BYTE   300,340,260,070,034,016,007,003 ;30
001374'  303  347  176                  .BYTE   303,347,176,074,074,176,347,303
001404'  000  160  120                  .BYTE   000,160,120,160,000,000,000,000
001414'  000  000  000                  .BYTE   000,000,000,000,000,000,000,000
001424'  010  010  010                  .BYTE   010,010,010,010,010,000,010,000
001434'  024  024  024                  .BYTE   024,024,024,000,000,000,000,000
001444'  024  024  066                  .BYTE   024,024,066,000,066,024,024,000
001454'  010  036  040                  .BYTE   010,036,040,034,002,074,010,000
001464'  000  062  062                  .BYTE     0, 62, 62,  4, 10, 20, 46, 46
001474'  000  020  050                  .BYTE     0, 20, 50, 50, 20, 52, 44, 32
001504'  000  030  030                  .BYTE     0, 30, 30, 30,  0,  0,  0,  0 ;40(')
001514'  000  010  020                  .BYTE     0, 10, 20, 40, 40, 40, 20, 10
001524'  000  010  004                  .BYTE     0, 10,  4,  2,  2,  2,  4, 10
001534'  000  052  034                  .BYTE     0, 52, 34, 76, 34, 52,  0,  0
001544'  000  000  010                  .BYTE     0,  0, 10, 10, 76, 10, 10,  0
001554'  000  000  000                  .BYTE     0,  0,  0,  0, 30, 30, 10, 20
001564'  000  000  000                  .BYTE     0,  0,  0,  0, 76,  0,  0,  0
001574'  000  000  000                  .BYTE     0,  0,  0,  0,  0,  0, 30, 30
001604'  000  002  002                  .BYTE     0,  2,  2,  4, 10, 20, 40, 40
001614'  000  034  042                  .BYTE   000,034,042,046,052,062,042,034
001624'  000  010  030                  .BYTE   000,010,030,010,010,010,010,034 ;50(1)
001634'  000  034  042                  .BYTE   000,034,042,002,034,040,040,076
001644'  000  034  042                  .BYTE   000,034,042,002,014,002,042,034
001654'  000  004  014                  .BYTE   000,004,014,024,044,074,004,004
```

```
  704  001664'  000  076  040          .BYTE   0, 76, 40, 74,  2,  2, 42, 34
  705  001674'  000  014  020          .BYTE   0, 14, 20, 40, 74, 42, 42, 34
  706  001704'  000  076  002          .BYTE   0, 76,  2,  4, 10, 20, 20, 20
  707  001714'  000  034  042          .BYTE   0, 34, 42, 42, 34, 42, 42, 34
  708  001724'  000  034  042          .BYTE   0, 34, 42, 42, 36,  2,  4, 30
  709  001734'  000  000  030          .BYTE   0,  0, 30, 30,  0, 30, 30,  0
  710  001744'  000  030  030          .BYTE   0, 30, 30,  0, 30, 30, 10, 20 ;60(;)
  711  001754'  000  004  010          .BYTE   0,  4, 10, 20, 40, 20, 10,  4
  712  001764'  000  000  000          .BYTE   0,  0,  0, 76,  0, 76,  0,  0
  713  001774'  000  020  010          .BYTE   0, 20, 10,  4,  2,  4, 10, 20
  714  002004'  000  030  044          .BYTE   0, 30, 44,  4, 10, 10,  0, 10
  715  002014'  000  074  102          .BYTE   0, 74,102,132,132,104, 40,  0
  716  002024'  000  010  024          .BYTE   0, 10, 24, 42, 42, 76, 42, 42
  717  002034'  000  074  022          .BYTE   0, 74, 22, 22, 34, 22, 22, 74
  718  002044'  000  034  042          .BYTE   0, 34, 42, 40, 40, 40, 42, 34
  719  002054'  000  074  022          .BYTE   0, 74, 22, 22, 22, 22, 22, 74
  720  002064'  000  076  040          .BYTE   0, 76, 40, 40, 70, 40, 40, 76 ;70(E)
  721  002074'  000  076  040          .BYTE   0, 76, 40, 40, 70, 40, 40, 40
  722  002104'  000  036  040          .BYTE   0, 36, 40, 40, 46, 42, 42, 36
  723  002114'  000  042  042          .BYTE   0, 42, 42, 42, 76, 42, 42, 42
  724  002124'  000  034  010          .BYTE   0, 34, 10, 10, 10, 10, 10, 34
  725  002134'  000  002  002          .BYTE   0,  2,  2,  2,  2,  2, 42, 34
  726  002144'  000  042  044          .BYTE   0, 42, 44, 50, 60, 50, 44, 42
  727  002154'  000  040  040          .BYTE   0, 40, 40, 40, 40, 40, 40, 76
  728  002164'  000  042  066          .BYTE   0, 42, 66, 52, 42, 42, 42, 42
  729  002174'  000  042  062          .BYTE   0, 42, 62, 52, 46, 42, 42, 42
  730  002204'  000  034  042          .BYTE   0, 34, 42, 42, 42, 42, 42, 34 ;80(O)
  731  002214'  000  074  042          .BYTE   0, 74, 42, 42, 74, 40, 40, 40
  732  002224'  000  034  042          .BYTE   0, 34, 42, 42, 42, 52, 44, 32
  733  002234'  000  074  042          .BYTE   0, 74, 42, 42, 74, 50, 44, 42
  734  002244'  000  036  040          .BYTE   0, 36, 40, 40, 34,  2,  2, 74
  735  002254'  000  076  010          .BYTE   0, 76, 10, 10, 10, 10, 10, 10
  736  002264'  000  042  042          .BYTE   0, 42, 42, 42, 42, 42, 42, 34
  737  002274'  000  042  042          .BYTE   0, 42, 42, 42, 24, 24, 10, 10
  738  002304'  000  042  042          .BYTE   0, 42, 42, 42, 42, 52, 66, 42
  739  002314'  000  042  042          .BYTE   0, 42, 42, 24, 10, 24, 42, 42
  740  002324'  000  042  042          .BYTE   0, 42, 42, 24, 10, 10, 10, 10 ;90(Y)
  741  002334'  000  076  002          .BYTE   0, 76,  2,  4, 10, 20, 40, 76
  742  002344'  000  070  040          .BYTE   000,070,040,040,040,040,040,070
  743  002354'  000  040  040          .BYTE   000,040,040,020,010,004,002,002
  744  002364'  000  016  002          .BYTE   000,016,002,002,002,002,002,016
  745  002374'  030  074  176          .BYTE   030,074,176,377,030,030,030,030
  746  002404'  010  014  016          .BYTE   010,014,016,377,377,016,014,010
  747  002414'  030  030  030          .BYTE   030,030,030,030,377,176,074,030
  748  002424'  020  060  160          .BYTE   020,060,160,377,377,160,060,020
  749  002434'  000  377  377          .BYTE   000,377,377,377,377,377,377,377 ;100
  750  002444'  000  000  377          .BYTE   000,000,377,377,377,377,377,377
  751  002454'  000  000  000          .BYTE   000,000,000,377,377,377,377,377
  752  002464'  000  000  000          .BYTE   000,000,000,000,377,377,377,377
  753  002474'  000  000  000          .BYTE   000,000,000,000,000,377,377,377
  754  002504'  000  000  000          .BYTE   000,000,000,000,000,000,377,377
  755  002514'  000  000  000          .BYTE   000,000,000,000,000,000,000,377
  756  002524'  376  376  376          .BYTE   376,376,376,376,376,376,376,376
  757  002534'  374  374  374          .BYTE   374,374,374,374,374,374,374,374
  758  002544'  370  370  370          .BYTE   370,370,370,370,370,370,370,370
  759  002554'  360  360  360          .BYTE   360,360,360,360,360,360,360,360 ;110
```



```
  760  002564'  340  340  340          .BYTE   340,340,340,340,340,340,340,340
  761  002574'  300  300  300          .BYTE   300,300,300,300,300,300,300,300
  762  002604'  200  200  200          .BYTE   200,200,200,200,200,200,200,200
  763  002614'  200  200  200          .BYTE   200,200,200,200,200,200,200,377
  764  002624'  377  001  001          .BYTE   377,001,001,001,001,001,001,001
  765  002634'  001  001  001          .BYTE   001,001,001,001,001,001,001,377
  766  002644'  200  200  200          .BYTE   200,200,200,200,200,200,200,377
  767  002654'  000  000  000          .BYTE   000,000,000,000,000,000,000,200
  768  002664'  200  000  000          .BYTE   200,000,000,000,000,000,000,000
  769  002674'  001  000  000          .BYTE   001,000,000,000,000,000,000,000 ;120
  770  002704'  000  000  000          .BYTE   000,000,000,000,000,000,000,001
  771  002714'  000  010  024          .BYTE   000,010,024,042,343,000,000,000
  772  002724'  010  010  030          .BYTE   010,010,030,040,100,040,030,010
  773  002734'  000  074  146          .BYTE   000,074,146,303,201,000,000,000
  774  002744'  030  060  140          .BYTE   030,060,140,100,100,140,060,030
  775  002754'  000  000  000          .BYTE   000,000,000,000,017,010,010,010
  776  002764'  010  010  010          .BYTE   010,010,010,010,370,000,000,000
  777  002774'  010  010  010          .BYTE   010,010,010,010,013,000,000,000
  778  003004'  000  000  000          .BYTE   000,000,000,000,370,010,010,010
```

```
                                        .SBTTL  T.V. CHARACTER SET
                                        ;
003014'  360  360  360  P03CHR: .BYTE   360,360,360,360,360,360,360,360
003024'  030  030  030          .BYTE   030,030,030,030,030,030,030,030
003034'  030  030  030          .BYTE   030,030,030,030,030,030,030,030
003044'  074  074  074          .BYTE   074,074,074,074,074,074,074,074
003054'  000  000  000          .BYTE   000,000,000,000,377,000,000,000
003064'  000  000  000          .BYTE   000,000,000,000,377,377,000,000
003074'  000  000  377          .BYTE   000,000,377,377,377,377,000,000
003104'  030  030  030          .BYTE   030,030,030,030,377,030,030,030
003114'  030  030  030          .BYTE   030,030,030,377,377,030,030,030
003124'  074  074  377          .BYTE   074,074,377,377,377,377,074,074 ; 9
003134'  003  006  014          .BYTE   003,006,014,030,060,140,300,200
003144'  200  300  140          .BYTE   200,300,140,060,030,014,006,003
003154'  000  034  062          .BYTE   000,034,062,060,170,060,140,176
003164'  203  306  154          .BYTE   203,306,154,070,070,154,306,203
003174'  074  176  377          .BYTE   074,176,377,377,377,377,176,074
003204'  030  074  176          .BYTE   030,074,176,377,377,176,074,030
003214'  030  030  074          .BYTE   030,030,074,146,303,146,074,030
003224'  030  030  074          .BYTE   030,030,074,176,377,176,074,030
003234'  146  146  146          .BYTE   146,146,146,146,146,146,146,146
003244'  314  314  314          .BYTE   314,314,314,314,314,314,314,314
003254'  377  000  377          .BYTE   377,000,377,000,377,000,377,000 ; 20
003264'  377  377  000          .BYTE   377,377,000,000,377,377,000,000
003274'  377  303  307          .BYTE   377,303,307,317,333,363,343,377
003304'  377  303  303          .BYTE   377,303,303,303,303,303,303,377
003314'  377  376  374          .BYTE   377,376,374,370,360,340,300,200
003324'  377  377  177          .BYTE   377,377,177,077,037,017,007,003
003334'  200  300  340          .BYTE   200,300,340,360,370,374,376,377
003344'  003  007  017          .BYTE   003,007,017,037,077,177,377,377
003354'  003  007  016          .BYTE   003,007,016,034,070,160,340,300
003364'  300  340  160          .BYTE   300,340,160,070,034,016,007,003
003374'  303  347  176          .BYTE   303,347,176,074,074,176,347,303
003404'  000  160  120          .BYTE   000,160,120,160,000,000,000,000
003414'  000  000  000          .BYTE   000,000,000,000,000,000,000,000
003424'  000  030  030          .BYTE   000,030,030,030,030,030,000,030
003434'  000  066  066          .BYTE   000,066,066,000,000,000,000,000
003444'  066  066  167          .BYTE   066,066,167,000,167,066,066,000
003454'  030  076  140          .BYTE   030,076,140,074,006,076,030,000
003464'  000  146  146          .BYTE   000,146,146,014,030,060,146,146
003474'  000  060  154          .BYTE   000,060,154,154,060,156,154,076
003504'  000  030  030          .BYTE   000,030,030,030,000,000,000,000
003514'  000  030  060          .BYTE   000,030,060,140,140,140,060,030 ; 40
003524'  000  030  014          .BYTE   000,030,014,006,006,006,014,030
003534'  000  066  034          .BYTE   000,066,034,076,034,066,000,000
003544'  000  000  030          .BYTE   000,000,030,030,076,030,030,000
003554'  000  000  000          .BYTE   000,000,000,000,000,030,030,060
003564'  000  000  000          .BYTE   000,000,000,000,076,000,000,000
003574'  000  000  000          .BYTE   000,000,000,000,000,000,030,030
003604'  000  006  006          .BYTE   000,006,006,014,030,060,140,140
003614'  000  074  146          .BYTE   000,074,146,146,146,146,146,074
003624'  000  030  070          .BYTE   000,030,070,030,030,030,030,074
003634'  000  074  146          .BYTE   000,074,146,006,036,060,140,176 ; 50
003644'  000  074  146          .BYTE   000,074,146,006,030,006,146,074
003654'  000  014  034          .BYTE   000,014,034,074,154,176,014,014
003664'  000  176  140          .BYTE   000,176,140,174,006,006,146,074
```



```
003674'  000  034  060          .BYTE   000,034,060,140,174,146,146,074
003704'  000  176  006          .BYTE   000,176,006,014,030,060,060,060
003714'  000  074  146          .BYTE   000,074,146,146,030,146,146,074
003724'  000  074  146          .BYTE   000,074,146,146,036,006,014,070
003734'  000  000  030          .BYTE   000,000,030,030,000,030,030,000
003744'  000  000  030          .BYTE   000,000,030,030,000,030,030,060
003754'  000  014  030          .BYTE   000,014,030,060,140,060,030,014 ; 60
003764'  000  000  000          .BYTE   000,000,000,076,000,076,000,000
003774'  000  060  030          .BYTE   000,060,030,014,006,014,030,060
004004'  000  074  146          .BYTE   000,074,146,006,034,030,000,030
004014'  000  030  146          .BYTE   000,030,146,156,156,146,060,000
004024'  000  030  066          .BYTE   000,030,066,146,146,176,146,146
004034'  000  174  146          .BYTE   000,174,146,146,170,146,146,174
004044'  000  074  146          .BYTE   000,074,146,140,140,140,146,074
004054'  000  174  146          .BYTE   000,174,146,146,146,146,146,174
004064'  000  176  140          .BYTE   000,176,140,140,170,140,140,176
004074'  000  176  140          .BYTE   000,176,140,140,170,140,140,140 ;70
004104'  000  076  140          .BYTE   000,076,140,140,156,146,146,076
004114'  000  146  146          .BYTE   000,146,146,146,176,146,146,146
004124'  000  074  030          .BYTE   000,074,030,030,030,030,030,074
004134'  000  006  006          .BYTE   000,006,006,006,006,006,146,074
004144'  000  146  154          .BYTE   000,146,154,170,160,170,154,146
004154'  000  140  140          .BYTE   000,140,140,140,140,140,140,176
004164'  000  146  176          .BYTE   000,146,176,176,146,146,146,146
004174'  000  146  146          .BYTE   000,146,146,166,156,146,146,146
004204'  000  074  146          .BYTE   000,074,146,146,146,146,146,074
004214'  000  174  146          .BYTE   000,174,146,146,174,140,140,140
004224'  000  074  146          .BYTE   000,074,146,146,146,156,156,166
004234'  000  174  146          .BYTE   000,174,146,146,174,170,154,146
004244'  000  076  140          .BYTE   000,076,140,140,074,006,006,174
004254'  000  176  030          .BYTE   000,176,030,030,030,030,030,030
004264'  000  146  146          .BYTE   000,146,146,146,146,146,146,074
004274'  000  146  146          .BYTE   000,146,146,146,146,074,074,030
004304'  000  146  146          .BYTE   000,146,146,146,146,176,176,146
004314'  000  146  146          .BYTE   000,146,146,074,030,074,146,146
004324'  000  146  146          .BYTE   000,146,146,074,030,030,030,030
004334'  000  176  006          .BYTE   000,176,006,014,030,060,140,176
004344'  000  170  140          .BYTE   000,170,140,140,140,140,140,170
004354'  000  140  140          .BYTE   000,140,140,060,030,014,006,006
004364'  000  036  006          .BYTE   000,036,006,006,006,006,006,036
004374'  030  074  176          .BYTE   030,074,176,377,030,030,030,030
004404'  030  034  036          .BYTE   030,034,036,377,377,036,034,030
004414'  030  030  030          .BYTE   030,030,030,030,377,176,074,030
004424'  030  070  170          .BYTE   030,070,170,377,377,170,070,030
004434'  000  377  377          .BYTE   000,377,377,377,377,377,377,377
004444'  000  000  377          .BYTE   000,000,377,377,377,377,377,377
004454'  000  000  000          .BYTE   000,000,000,377,377,377,377,377
004464'  000  000  000          .BYTE   000,000,000,000,377,377,377,377
004474'  000  000  000          .BYTE   000,000,000,000,000,377,377,377
004504'  000  000  000          .BYTE   000,000,000,000,000,000,377,377
004514'  000  000  000          .BYTE   000,000,000,000,000,000,000,377
004524'  374  374  374          .BYTE   374,374,374,374,374,374,374,374
004534'  374  374  374          .BYTE   374,374,374,374,374,374,374,374
004544'  370  370  370          .BYTE   370,370,370,370,370,370,370,370
004554'  360  360  360          .BYTE   360,360,360,360,360,360,360,360
004564'  340  340  340          .BYTE   340,340,340,340,340,340,340,340
```

```
   892  004574'   300   300   300          .BYTE   300,300,300,300,300,300,300,300 ; 110
   893  004604'   300   300   300          .BYTE   300,300,300,300,300,300,300,300
   894  004614'   300   300   300          .BYTE   300,300,300,300,300,300,300,377
   895  004624'   377   003   003          .BYTE   377,003,003,003,003,003,003,003
   896  004634'   003   003   003          .BYTE   003,003,003,003,003,003,003,377
   897  004644'   377   300   300          .BYTE   377,300,300,300,300,300,300,300
   898  004654'   000   000   000          .BYTE   000,000,000,000,000,000,000,300
   899  004664'   300   000   000          .BYTE   300,000,000,000,000,000,000,000
   900  004674'   003   000   000          .BYTE   003,000,000,000,000,000,000,000
   901  004704'   000   000   000          .BYTE   000,000,000,000,000,000,000,003
   902  004714'   000   030   074          .BYTE   000,030,074,146,347,000,000,000
   903  004724'   030   030   070          .BYTE   030,030,070,060,140,060,070,030
   904  004734'   000   074   146          .BYTE   000,074,146,303,303,000,000,000
   905  004744'   030   070   160          .BYTE   030,070,160,140,140,160,070,030
   906  004754'   000   000   000          .BYTE   000,000,000,000,037,030,030,030
   907  004764'   030   030   030          .BYTE   030,030,030,030,370,000,000,000
   908  004774'   030   030   030          .BYTE   030,030,030,030,037,000,000,000
   909  005004'   000   000   000          .BYTE   000,000,000,000,370,030,030,030
   910                                     .EVEN
   911                                     .LIST   BEX
   912          000001                     .END
```



```
ACSR     000102R        BR1      000012R        INIT     000030R        P02CHR   001014R        SVR4     000072R
ADDR     000006R        BR2      000013R        INTR     000120R        P03CHR   003014R        SVR5     000074R
ADDR22=  001000         RTODS =  104421         MAP22S=  104416         RANDS =  104417         SVR6     000076R
ASB      000106R        CARX     000774R        MODNAM   000000R        RANNUM   000054R        SYSCNT   000052R
ASTAT    000104R        CARY     000776R        MODSP    000222R        RESTRT   000222R        TRPDFD=  000022
AWAS     000110R        CDATAS=  104412         MSGNS =  104403         RES1     000056R        T0601    000274R
BAKGND   000762R        CHAR     000772R        MSGSS =  104402         RES2     000060R        T0602    000330R
BASE     001004R        CONFIG   000056R        MSGS  =  104401         RSTRT    000112R        T0603    000332R
BASE1    001006R        CSRA     000100R        NULL  =  000006         R6     =%000006         T0604    000416R
BEGIN    000000R        DATCKS=  104411         OPEN  =  000000         R7     =%000007         T0605    000424R
BIT0  =  000001         DATERS=  104404         OTOAS =  104420         SARDAT   001010R        T0606    000436R
BIT1  =  000002         DEVBUF   000756R        PASCNT   000034R        SBADR    000102R        T0607    000450R
BIT10 =  002000         DEVCSR   000754R        PIRQS =  000004         SDRDAT   001012R        T0608    000454R
BIT11 =  004000         DVID1    000014R        POPSP =  005726         SOFCNT   000042R        T0609    000460R
BIT12 =  010000         ENDITS=  104413         POPSP2=  022626         SOFERS=  104406         T0610    000466R
BIT13 =  020000         ENDS  =  104410         PRTY  =  000000         SOFPAS   000046R        T0611    000474R
BIT14 =  040000         ERRTYP   000106R        PRTY0 =  000000         SPOINT   000032R        T0612    000720R
BIT15 =  100000         EXITS =  104400         PRTY1 =  000040         SPSIZ =  000040         VECTOR   000010R
BIT2  =  000004         FLASH    000766R        PRTY2 =  000100         SR1      000016R        VIDEON   000764R
BIT3  =  000010         FORGND   0007b0R        PRTY3 =  000140         SR2      000020R        WASADR   000104R
BIT4  =  000020         GETPAS=  104415         PRTY4 =  000200         SR3      000022R        WDFR     000116R
BIT5  =  000040         GWBUFS=  104414         PRTY5 =  000240         SR4      000024R        WDTO     000114R
BIT6  =  000100         HRDCNT   000044R        PRTY6 =  000300         START    000222R        XFLAG    000005R
BIT7  =  000200         HRDERS=  104405         PRTY7 =  000340         STAT     000026R        XMAX     001000R
BIT8  =  000400         HRDPAS   000050R        PS    =  177776         SVR0     000062R        YMAX     001002R
BIT9  =  001000         ICUNT    000036R        PSW   =  177776         SVR1     000064R        .      = 005014R
BLINK    000770R        ICOUNT   000040R        PUSH  =  005746         SVR2     000066R
BREAKS=  104407         IDNUM    000122R        PUSH2 =  024646         SVR3     000070R

. ABS.  000000     000
        005014     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XVTCA0.OBJ,XVTCA0.LPT=DDXCOM.P11,XVTCA0.P11
 RUN-TIME:  4  7  .3 SECONDS
 RUN-TIME RATIO: 28/13=2.1
 CORE USED:  8K  (15 PAGES)
```

```
.REM_


                        IDENTIFICATION
                        --------------

        PRODUCT CODE:   AC-S248A-MC

        PRODUCT NAME:   CXKWGA0 CSS KW11C DEC/X MOD

        DATE CREATED:   JUL 15,1980

        MAINTAINER:     CSS LOW VOLUME PRODUCTS

        AUTHOR(S):      VIJAY ANANDWALA

        UPDATE AUTHOR(S):


        THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
        NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
        EQUIPMENT CORPORATION.  DIGITAL EQUIPMENT CORPORATION ASSUMES
        NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

        THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A
        LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE
        TERMS OF SUCH LICENSE.

        DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR
        THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS
        NOT SUPPLIED BY DIGITAL.

            COPYRIGHT (C)  1980 BY DIGITAL EQUIPMENT CORPORATION
```

```
                        ABSTRACT:
                        --------
                                "CXKWG" IS A DEC/X11 SOFTWARE MODULE WHICH EXERCISES A
                                KW11-C CLOCK.
                                THE MAIN CHARACTERISTICS OF THIS MODULE ARE:
                                1.SET/READ THE TIME OF THE CLOCK.
                                2.SET UPDATE INTERRUPT
                                3.SET TIME INTERVAL INTERRUPT
                        REQUIREMENTS:
                        -------------
                                PDP11 PROCESSOR,KW11-C CLOCK,TERMINAL,16K MEMORY

                        PASS DEFINATION:
                        ---------------
                                ONE PASS OF THE CXKWG CONSISTS OF 10. ITERATIONS OF
                                BASIC TEST SEQUENCE.(SET/READ TIME,UPDATE INTERRUPT,
                                TIME INTERVAL INTERRUPT).

                        EXECUTION TIME:
                        --------------
                                ONE PASS OF CXKWG ALONE ON PDP11/34 TAKES APPROXIMATELY
                                1 MINUTE.

                        CONFIGARATION REQUIREMENTS:
                        --------------------------
                                DEFAULT PARAMETERS:
                                DEVADR:160200, VECTOR: 300, BR1: 6, BR2: 6,DEVCNT:1,ICONST:10.

                        PRINTOUT:
                        --------
                                PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN THE DEC/X11
                                DOCUMENT.
                        NOTE:
                                IN HARD AND SOFT ERROR WHERE
                                ERRTYP:  1 = ACTUAL READ STATUS OF CSR NOT MATCH WITH EXPECTED.
                                         2 = ACTUAL READ VALUE OF TIME NOT MATCH WITH EXPECTED.
```

```
  442
  443
  444  000000'                        IOMOD   <KWGA >,160200,300,6,6,0,10.,0
  (1)  000000'                                MODULE  140000,KWGA ,160200,300,6,6,0,10.,0
  (2)                                         .TITLE  KWGA DEC/X11 SYSTEM EXERCISER MODULE
  (2)                                 ;       DDXCOM  VERSION  6      23-MAY-78
  (2)                                         .LIST   BIN
  (2)                                 ;*****************************************************************************
  (2)  000000'                        BEGIN:
  (2)  000000' 053513  040507     040 MODNAM: .ASCII  /KWGA / ;MODULE NAME.
  (2)  000005'    000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
  (2)  000006' 160200                 ADDR:   160200+0                ;1ST DEVICE ADDR.
  (2)  000010' 000300                 VECTOR: 300+0                   ;1ST DEVICE VECTOR.
  (2)  000012'    300                 BR1:    .BYTE   PRTY6+0         ;1ST BR LEVEL.
  (2)  000013'    300                 BR2:    .BYTE   PRTY6+0         ;2ND BR LEVEL.
  (2)  000014' 000001                 DVID1:  0+1                     ;DEVICE INDICATOR 1.
  (2)  000016' 000000                 SR1:    OPEN                    ;SWITCH REGISTER 1
  (2)  000020' 000000                 SR2:    OPEN                    ;SWITCH REGISTER 2
  (2)  000022' 000000                 SR3:    OPEN                    ;SWITCH REGISTER 3
  (2)  000024' 000000                 SR4:    OPEN                    ;SWITCH REGISTER 4
  (2)                                 ;*****************************************************************************
  (2)  000026' 140000                 STAT:   140000                  ;STATUS WORD.
  (2)  000030' 000244'                INIT:   START                   ;MODULE START ADDR.
  (2)  000032' 000224'                SPOINT: MODSP                   ;MODULE STACK POINTER.
  (2)  000034' 000000                 PASCNT: 0                       ;PASS COUNTER.
  (2)  000036' 000012                 ICONT:  10.                     ;# OF ITERATIONS PER PASS=10.
  (2)  000040' 000000                 ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
  (2)  000042' 000000                 SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
  (2)  000044' 000000                 HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
  (2)  000046' 000000                 SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
  (2)  000050' 000000                 HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
  (2)  000052' 000000                 SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
  (2)  000054' 000000                 RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
  (2)  000056'                        CONFIG:
  (2)  000056' 000000                 RES1:   0                       ;RESERVED FOR MONITOR USE
  (2)  000060' 000000                 RES2:   0                       ;RESERVED FOR MONITOR USE
  (2)  000062' 000000                 SVR0:   OPEN                    ;LOC TO SAVE R0.
  (2)  000064' 000000                 SVR1:   OPEN                    ;LOC TO SAVE R1.
  (2)  000066' 000000                 SVR2:   OPEN                    ;LOC TO SAVE R2.
  (2)  000070' 000000                 SVR3:   OPEN                    ;LOC TO SAVE R3.
  (2)  000072' 000000                 SVR4:   OPEN                    ;LOC TO SAVE R4.
  (2)  000074' 000000                 SVR5:   OPEN                    ;LOC TO SAVE R5.
  (2)  000076' 000000                 SVR6:   OPEN                    ;LOC TO SAVE R6.
  (2)  000100' 000000                 CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
  (2)  000102'                        SBADR:                          ;ADDR OF GOOD DATA, OR
  (2)  000102' 000000                 ACSR:   OPEN                    ;CONTENTS OF CSR.
  (2)  000104'                        WASADR:                         ;ADDR OF BAD DATA, OR
  (2)  000104' 000000                 ASTAT:  OPEN                    ;STATUS REG CONTENTS.
  (2)  000106'                        ERRTYP:                         ;TYPE OF ERROR
  (2)  000106' 000000                 ASB:    OPEN                    ;EXPECTED DATA.
  (2)  000110' 000000                 AWAS:   OPEN                    ;ACTUAL DATA.
  (2)  000112' 000430'                RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
  (2)  000114' 000000                 WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
  (2)  000116' 000000                 WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
  (2)  000120' 000000                 INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
  (2)  000122' 000000                 IDNUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
```

```
(2)          000040                          .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
(2)                                          .NLIST
(2)                                          .WORD   0
(2)                                          .LIST
(3)                                          .ENDR
(2)  000224'                             MODSP:
(2)                                      ;*************************************************************************
445                                      ;DEFINATIONS AND OTHER GOODIES
446  000224' 000000                      CSR:    OPEN                    ;CONTROL STATUS REGISTER
447  000226' 000000                      RG0:    OPEN                    ;YEAR,MONTH & DAY REGISTER
448  000230' 000000                      RG2:    OPEN                    ;HOUR & MINUTE REGISTER
449  000232' 000000                      RG4:    OPEN                    ;SECOND & TIMEINTERVAL REGISTER
450  000234' 000000                      CNT:    OPEN            ;COUNTER
451  000236' 000000                      GOOD:   OPEN
452  000240' 000000                      BAD:    OPEN
453  000242' 000000                      DREG: OPEN
454  000244' 016700  177536              START:  MOV     ADDR,   R0
455  000250' 010067  177752                      MOV     R0,     RG0
456  000254' 062700  000002                      ADD     #2,     R0
457  000260' 010067  177744                      MOV     R0,     RG2
458  000264' 062700  000002                      ADD     #2,     R0
459  000270' 010067  177736                      MOV     R0,     RG4
460  000274' 062700  000002                      ADD     #2,     R0
461  000300' 010067  177720                      MOV     R0,     CSR
462  000304' 016700  177500                      MOV     VECTOR, R0
463  000310' 012720  001016'                     MOV     #DATINT,(R0)+   ;ADDRESS OF THE INTERRUPT ROUTINE
464  000314' 116710  177472                      MOVB    BR1,    (R0)    ;PRIORITY
465  000320' 005720                              TST     (R0)+
466  000322' 012720  001262'                     MOV     #TIMINT,(R0)+   ;ADDRESS OF TIME INTERRUPT
467  000326' 116710  177461                      MOVB    BR2,    (R0)
468  000332' 005077  177666                      CLR     @CSR
469  000336' 012777  105201  177662              MOV     #105201,@RG0    ;SETUP PRESET
470  000344' 004767  001144                      JSR     PC,     WAIT    ;WAIT FOR RDY AND 'DE'BIT TO SET
471  000350' 032777  020000  177646              BIT     #BIT13, @CSR    ;IS 'VI' BIT SET?
472  000356' 001422                              BEQ     1$              ;NO;BR
473  000360' 005077  177640                      CLR     @CSR            ;CLEAR CSR
474  000364' 012777  013470  177636              MOV     #13470, @RG2    ;PRESET FOR HOUR+MIN
475  000372' 004767  001116                      JSR     PC,     WAIT    ;WAIT FOR RDY AND DE TO SET
476  000376' 032777  020000  177620              BIT     #BIT13, @CSR    ;IS 'VI' BIT SET?
477  000404' 001407                              BEQ     1$              ;NO;EXIT
478  000406' 005077  177612                      CLR     @CSR            ;CLEAR CSR
479  000412' 012777  010045  177612              MOV     #10045, @RG4    ;PRESET SECOND
480  000420' 004767  001070                      JSR     PC,     WAIT    ;WAIT FOR 'DE'AND RDY BIT TO SET
481  000424' 005077  177574              1$:     CLR     @CSR            ;CLEAR CSR
482
483  000430' 012700  000646'             RESTRT: MOV     #TABLE, R0              ;;POINTER TO THE TABLE
484  000434' 022700  000742'             1$:     CMP     #TABEND,R0              ;;IS IT END OF TABLE?
485  000440' 003004                              BGT     2$                      ;;BRANCH IF NOT
486  000442' 005067  177440                      CLR     ASB                     ;;CLEAR THE LOCATION
487  000446' 000167  000272                      JMP     TABEND+2                ;;EXIT
488  000452' 005077  177546              2$:     CLR     @CSR
489  000456' 012002                              MOV     (R0)+,  R2              ;;MASK TO SET BIT IN CSR
490  000460' 012004                              MOV     (R0)+,  R4              ;;ADDRESS OF CSR
491  000462' 010274  000000                      MOV     R2,     @(R4)           ;;VALUE AT THE ADDRESS
492  000466' 012002                              MOV     (R0)+,  R2              ;;VALUE TO BE WRITTEN IN DEV REG
493  000470' 012004                              MOV     (R0)+,  R4              ;;ADDRESS OF DEV REG
```

```
   494  000472' 010274  000000                MOV     R2,     @(R4)           ;;VALUE IN THE DEV REG
   495  000476' 004767  001012                JSR     PC,     WAIT            ;;WAIT FOR READY BIT TO SET
   496  000502' 005077  177516                CLR     @CSR                    ;;CLEAR CSR
   497  000506' 022027  177777                CMP     (R0)+,  #-1             ;;IS IT END OF SET?
   498  000512' 001350                        BNE     1$                      ;;GET MORE VALUES FROM TABLE
   499
   500  000514' 012002                3$:     MOV     (R0)+,  R2              ;;MASK TO READ DEV REGISTER
   501  000516' 012004                        MOV     (R0)+,  R4              ;;ADDRESS OF CSR
   502  000520' 010274  000000                MOV     R2,     @(R4)           ;;ACTUAL TRANSFER
   503  000524' 012067  177506                MOV     (R0)+,  GOOD            ;;EXPECTED VALUE
   504  000530' 012004                        MOV     (R0)+,  R4              ;;ADDRESS OF DEVICE REG
   505  000532' 017467  000000  177500        MOV     @(R4),  BAD             ;;READ VALUE IN BAD
   506  000540' 026767  177472  177472        CMP     GOOD,   BAD             ;;EXPECTED=ACTUAL?
   507  000546' 001433                        BEQ     4$                      ;;BRANCH IF YES
   508  000550' 017467  000000  177462        MOV     @(R4),  BAD             ;;READ AGAIN
   509  000556' 026767  177454  177454        CMP     GOOD,   BAD             ;;NOW THEY ARE EQUAL?
   510  000564' 001424                        BEQ     4$                      ;;BRANCH IF YES
   511  000566' 017767  177432  177306        MOV     @CSR,   CSRA            ;;CONTENT OF CSR
   512  000574' 016767  177424  177276        MOV     CSR,    CSRA            ;;ADDRESS OF CSR
   513  000602' 012767  000236' 177272        MOV     #GOOD,  SBADR           ;;EXPECTED ADDRESS
   514  000610' 012767  000240' 177266        MOV     #BAD,   WASADR          ;;ADDRESS OF ACTUAL DATA
   515  000616' 016767  177414  177262        MOV     GOOD,   ASB             ;;ACTUAL DATA SHOULD BE
   516  000624' 016767  177410  177256        MOV     BAD,    AWAS            ;;ACTUAL DATA WAS
   517                                        ;*****************************************************************
(1)  000632' 104404  000000'                DATER$,BEGIN            ;DATA ERROR!!!
(1)                                         ;*****************************************************************
   518  000636' 022027  177777        4$:     CMP     (R0)+,  #-1             ;;IS IT END OF SUBSET?
   519  000642' 001324                        BNE     3$                      ;;READ MORE VALUE
   520  000644' 000673                        BR      1$                      ;;GET MORE VALUES
   521
   522  000646' 040000                TABLE:  BIT14                           ;;MASK TO SET YEAR BIT IN CSR
   523  000650' 000224'                       CSR                             ;;ADDRESS OF CSR
   524  000652' 017073                        17073                           ;;YEAR 1979
   525  000654' 000226'                       RG0                             ;;ADDRESS OF YEAR REGISTER
   526  000656' 177777                        177777                          ;;END OF SUBSET
   527
   528  000660' 000004                        BIT2                            ;;SET BIT TO READ YEAR
   529  000662' 000224'                       CSR                             ;;ADDRESS OF CSR
   530  000664' 017073                        17073                           ;;EXPECTED VALUE OF YEAR
   531  000666' 000226'                       RG0
   532  000670' 177777                        177777                          ;;END OF SUBSET
   533
   534  000672' 100000                        BIT15                           ;;SET TIME
   535  000674' 000224'                       CSR
   536  000676' 002004                        2004                            ;;FEBRUARY 4
   537  000700' 000226'                       RG0
   538  000702' 177777                        177777
   539  000704' 000000                        00000                           ;;
   540  000706' 000224'                       CSR
   541  000710' 002004                        2004
   542  000712' 000226'                       RG0
   543  000714' 177777                        177777
   544
   545  000716' 100000                        BIT15
   546  000720' 000224'                       CSR
   547  000722' 000000                        00000
```

```
000724' 000230'                         RG2
000726' 177777                          177777
000730' 000000                          000000
000732' 000224'                         CSR
000734' 000000                          00000
000736' 000230'                         RG2
000740' 177777                          177777
000742' 000000                  TABEND: 000000

                                ;THIS ROUTINE SETS UPDATE INTERRUPT
                                ;BIT15 IN CSR IS SET TO WRITE THE TIME
                                ;BIT6 IS SET IN CSR TO ENABLE THE INTERRUPT
                                ;
000744' 012700  001206'         SETINT: MOV     #TABLE2,R0              ;;POINTER TO THE TABLE
000750' 022700  001216'         ONE:    CMP     #TABEN2,R0              ;;IS IT END OF TABLE
000754' 003002                          BGT     TWO                     ;;BRANCH IF NOT END
000756' 000167  000236                  JMP     TABEN2+2                ;;EXIT,OTHERWISE
000762' 005077  177236          TWO:    CLR     @CSR                    ;;CLEAR CSR
000766' 012777  100100  177230          MOV     #BIT15+BIT6,@CSR        ;;SET TIME WITH INTRPT ENABL
000774' 012067  177236                  MOV     (R0)+,  GOOD            ;;WRITE VALUE
001000' 013067  177236                  MOV     @(R0)+, DREG            ;;ADDRESS OF DEV REG
001004' 016777  177226  177230          MOV     GOOD,   @DREG           ;;
001012' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.

001016' 017767  177202  177056  DATINT: MOV     @CSR,   ACSR            ;;CURRENT CONTENT OF CSR
001024' 005077  177174                  CLR     @CSR                    ;;CLEAR CSR
                                        ;---------------------------------------------------------------
(1) 001030' 000004  000000' 001036'     PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
(1)                                     ;---------------------------------------------------------------
001036' 017767  177200  177174  1$:     MOV     @DREG,  BAD             ;;READ THE CURRENT VALUE
001044' 042767  040000  177030          BIC     #BIT14, ACSR            ;;CLEAR 'TI' IF SET
001052' 022767  100300  177022          CMP     #100300,        ACSR    ;;EXPECTED STATUS OF CSR
001060' 001416                          BEQ     2$                      ;;BRANCH IF EQUAL
001062' 016767  177136  177010          MOV     CSR,    CSRA            ;;ADDRESS OF CSR
001070' 012767  100300  177006          MOV     #100300,ASTAT           ;;EXPECTED STATUS
001076' 012767  000001  177002          MOV     #1,     ERRTYP          ;;TYPE OF ERROR
                                        ;*****************************************************************
(1) 001104' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;EXPECTED STATUS OF CSR DIDNOT MATCH WITH ACTUAL STATUS
(1)                                     ;*****************************************************************
001112' 000167  177632                  JMP     ONE
001116' 026767  177114  177114  2$:     CMP     GOOD,   BAD             ;;EXPECTED=ACTUAL?
001124' 001711                          BEQ     ONE
001126' 017767  177110  177104          MOV     @DREG,  BAD             ;;TRY TO READ AGAIN
001134' 026767  177076  177076          CMP     GOOD,   BAD             ;;ARE THEY EQUAL?
001142' 001702                          BEQ     ONE
001144' 016767  177072  176726          MOV     DREG,   CSRA            ;;ADDRESS OF DEV REG
001152' 017767  177064  176722          MOV     @DREG,  ACSR            ;;CONTENT OF REG
001160' 016767  177052  176716          MOV     GOOD,   ASTAT           ;;EXPECTED STATUS
001166' 012767  000002  176712          MOV     #2,     ERRTYP          ;;TYPE OF ERROR
                                        ;*****************************************************************
(1) 001174' 104406  000000' 000000      SUFER$,BEGIN,NULL       ;READ VALUE NOT MATCH WITH THE SET VALUE OF TIME
(1)                                     ;*****************************************************************
001202' 000167  177542                  JMP     ONE
001206' 006005                  TABLE2: 6005                            ;;DECEMBER 5
001210' 000226'                         RG0                             ;;ADDRESS OF REG
001212' 006000                          6000                            ;;12HH,00MIN
```

```
001214' 000230'                         RG2                             ;;ADDRESS OF DEV REG
001216' 000000                  TABEN2: 00000                           ;;END OF TABLE
                                ;
                                ;
                                ;THIS ROUTINE SETS TIME INTERVAL INTERRUPT
                                ;
                                ;
001220' 012700  001470'         DELTA:  MOV     #TIMTAB,R0              ;;POINTER TO THE TABLE
001224' 022700  001502'         D1:     CMP     #TIMEND,R0              ;;IS IT END OF TABLE?
001230' 001515                          BEQ     D2                      ;;BRANCH IF END OF TABLE
001232' 012067  177000                  MOV     (R0)+,  GOOD            ;;GET VALUE FROM THE TABLE
001236' 005077  176762                  CLR     @CSR                    ;;CLEAR CSR
001242' 012777  000050  176754          MOV     #BIT5+BIT3,@CSR         ;;SET TIME INT AND INTRPT ENBL
001250' 016777  176762  176754          MOV     GOOD,   @RG4            ;;INTO SECOND REG
001256' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
                                ;
001262' 017767  176736  176612  TIMINT: MOV     @CSR,   ACSR            ;;STATUS OF CSR
001270' 005077  176730                  CLR     @CSR
001274' 012777  000010  176722          MOV     #BIT3,  @CSR            ;;READ DELTAT
001302' 017767  176724  176730          MOV     @RG4,   BAD             ;;READ TIME INTERVAL VALUE
                                        ;---------------------------------------------------------------
(1) 001310' 000004  000000' 001316'     PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
(1)                                     ;---------------------------------------------------------------
001316' 042767  137727  176556  1$:     BIC     #137727,ACSR            ;;CLEAR UNWANTED BITS
001324' 022767  040050  176550          CMP     #40050, ACSR            ;;EXPCT STATUS OF CSR
001332' 001415                          BEQ     2$                      ;;BRANCH IF EQUAL
001334' 016767  176664  176536          MOV     CSR,    CSRA            ;;
001342' 012767  040050  176534          MOV     #40050, ASTAT           ;;EXPECTED STATUS OF CSR
001350' 012767  000001  176530          MOV     #1,     ERRTYP          ;;TYPE OF ERROR
                                        ;*****************************************************************
(1) 001356' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;STATUS DIDN'T MATCH
(1)                                     ;*****************************************************************
001364' 000717                          BR      D1                      ;;GET NEXT VALUE
001366' 042767  040000  176642  2$:     BIC     #40000, GOOD            ;;CLEAR TIME INTERVAL ENABLE BIT
001374' 016702  176636                  MOV     GOOD,   R2              ;;GET EXPECTED VALUE
001400' 016704  176634                  MOV     BAD,    R4              ;; DELAT VALUE AT TIME OF INTRPT
001404' 160204                          SUB     R2,     R4              ;;GET THE DIFFERENCE
001406' 002001                          BGE     3$                      ;;BRANCH IF DIFFERENCE POSITIVE OR ZERO
001410' 005404                          NEG     R4                      ;;MAKE POSITIVE
001412' 020427  000001          3$:     CMP     R4,     #1              ;;IS DIFF. MORE THAN 1?
001416' 003702                          BLE     D1                      ;;GET NEXT VALUE
001420' 017767  176600  176454          MOV     @CSR,   ACSR            ;;CURRENT STATUS OF CSR
001426' 012767  000236' 176446          MOV     #GOOD,  SBADR           ;;ADDRESS OF EXP DATA
001434' 012767  000240' 176442          MOV     #BAD,   WASADR          ;;ADDRESS OF ACTUAL DATA
001442' 016767  176570  176436          MOV     GOOD,   ASB             ;;ACTUAL DATA SHOLD BE
001450' 016767  176564  176432          MOV     BAD,    AWAS            ;;ACTUAL DATA WAS
                                        ;*****************************************************************
001456' 104404  000000'                 DATER$,BEGIN            ;DATA ERROR!!!
(1)                                     ;*****************************************************************
001462' 000660                          BR      D1                      ;;GET NEXT VALUE
                                ;
001464' 000167  000014          D2:     JMP     TIMEND+2
```

```
  648  001470' 047777                  TIMTAB: 47777                           ;;
  649  001472' 040304                          40304
  650  001474' 040506                          40506
  651  001476' 042043                          42043
  652  001500' 042333                          42333
  653  001502' 000000                  TIMEND: 00000
  654
  655  001504' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
  (1)                                                                  ;MONITOR SHALL TEST END OF PASS
  656  001510' 000167  176714                  JMP     RESTRT
  657
  658                                  ;THIS ROUTINE WAIT FOR READY BIT TO SET
  659                                  ;IF READY BIT DIDN'T SET IN GIVEN TIME
  660                                  ;THE MODULE WILL BE DROPPED
  661                                  ;
  662  001514' 012767  077777  176512  WAIT:   MOV     #77777,CNT              ;;SET UP TIMER
  663
  664  001522'                         1$:
  (1)  001522' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
  (1)  001526' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
  665  001532' 032777  100000  176464          BIT     #BIT15, @CSR            ;;IS RDY BIT SET?
  666  001540' 001024                          BNE     3$                      ;;BRANCH IF YES
  667  001542' 005367  176466          2$:     DEC     CNT                     ;;DECREMENT COUNT
  668  001546' 001365                          BNE     1$                      ;;WAIT SOME MORE TIME AND TRY AGAIN
  669  001550' 016767  176450  176322          MOV     CSR,    CSRA            ;;CSR ADDRESS
  670  001556' 017767  176442  176316          MOV     @CSR,   ACSR            ;;CONTENT OF CSR
  671  001564' 012767  100200  176312          MOV     #100200,ASTAT           ;;EXPECTED STATUS OF CSR
  672  001572' 012767  000001  176306          MOV     #1,     ERRTYP          ;;
  673                                          ;*****************************************************************
  (1)  001600' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;'DE' AND/OR 'RDY' BIT FAIL TO SET IN CSR
  (1)                                          ;*****************************************************************
  674  001606' 104410  000000'                 END$,BEGIN              ;MODULE DROPPED;READY BIT FAIL TO SET IN TIME
  675  001612' 032777  000200  176404  3$:     BIT     #BIT7,  @CSR            ;;IS RDYBIT SET?
  676  001620' 001001                          BNE     4$                      ;;YES:RETURN
  677  001622' 000773                          BR      3$                      ;;DECREMENT COUNTER
  678  001624' 000207                  4$:     RTS     PC                      ;;EXIT
  679
  680          000001                          .END
```

```
ACSR     000102R          444#    511*    572*    576*    577     590*    617*    623*    624     639*    670*
ADDR     000006R          444#    454
ADDR22=  001000           444#
ASB      000106R          444#    486*    515*    642*
ASTAT    000104R          444#    580*    591*    627*    671*
AWAS     000110R          444#    516*    643*
BAD      000240R          452#    505*    506     508*    509     514     516     575*    584     586*    587     620*    633
                          641     643
BEGIN    000000R          444#    517     570     574     582     593     615     621     629     644     655     664     673
                          674
BIT0   = 000001           444#
BIT1   = 000002           444#
BIT10  = 002000           444#
BIT11  = 004000           444#
BIT12  = 010000           444#
BIT13  = 020000           444#    471     476
BIT14  = 040000           444#    522     576
BIT15  = 100000           444#    534     545     566     665
BIT2   = 000004           444#    528
BIT3   = 000010           444#    613     619
BIT4   = 000020           444#
BIT5   = 000040           444#    613
BIT6   = 000100           444#    566
BIT7   = 000200           444#    675
BIT8   = 000400           444#
BIT9   = 001000           444#
BREAK$= 104407            444#    664
BR1      000012R          444#    464
BR2      000013R          444#    467
BTOD$  = 104421           444#
CDATA$= 104412            444#
CNT      000234R          450#    662*    667*
CONFIG   000056R          444#
CSR      000224R          446#    461*    468*    471     473*    476     478*    481*    488*    496*    511     512     523
                          529     535     540     546     551     565*    566*    572     573*    579     612*    613*    617
                          618*    619*    626     639     665     669     670     675
CSRA     000100R          444#    512*    579*    589*    626*    669*
DATCK$= 104411            444#
DATER$= 104404            444#    517     644
DATINT   001016R          463     572#
DELTA    001220R          608#
DREG     000242R          453#    568*    569*    575     586     589     590
DVID1    000014R          444#
D1       001224R          609#    630     638     645
D2       001464R          610     647#
ENDIT$= 104413            444#    655
END$   = 104410           444#    674
ERRTYP   000106R          444#    581*    592*    628*    672*
EXIT$  = 104400           444#    570     615
GETPA$= 104415            444#
GOOD     000236R          451#    503*    506     509     513     515     567*    569     584     587     591     611*    614
                          631*    632     640     642
GWBUF$= 104414            444#
HRDCNT   000044R          444#
HRDER$= 104405            444#    582     629     673
HRDPAS   000050R          444#
```

```
ICONT   000036R          444#
ICOUNT  000040R          444#
IDNUM   000122R          444#
INIT    000030R          444#
INTR    000120R          444#
MAP22$= 104416           444#
MODNAM  000000R          444#
MODSP   000224R          444#
MSGN$ = 104403           444#
MSGS$ = 104402           444#
MSG$  = 104401           444#
NULL  = 000000           444#  582     593     629     673
ONE     000750R          562#  583     585     588     594
OPEN  = 000000           444#  446     447     448     449     450     451     452     453
OTOA$ = 104420           444#
PASCNT  000034R          444#
PIRQ$ = 000004           444#  574     621
POPSP = 005726           444#
POPSP2= 022626           444#
PRTY  = 000000           444#
PRTY0 = 000000           444#
PRTY1 = 000040           444#
PRTY2 = 000100           444#
PRTY3 = 000140           444#
PRTY4 = 000200           444#
PRTY5 = 000240           444#
PRTY6 = 000300           444#
PRTY7 = 000340           444#
PS    = 177776           444#
PSW   = 177776           444#
PUSH  = 005746           444#
PUSH2 = 024646           444#
RAND$ = 104417           444#
RANNUM  000054R          444#
RESTRT  000430R          444     483#    656
RES1    000056R          444#
RES2    000060R          444#
RG0     000226R          447#  455*    469*    525     531     537     542     596
RG2     000230R          448#  457*    474*    548     553     598
RG4     000232R          449#  459*    479*    614*    620
RSTRT   000112R          444#
SBADR   000102R          444#  513*    640*
SETINT  000744R          561#
SOFCNT  000042R          444#
SOFER$= 104406           444#  593
SOFPAS  000046R          444#
SPOINT  000032R          444#
SPSIZ = 000040             7#  444
SR1     000016R          444#
SR2     000020R          444#
SR3     000022R          444#
SR4     000024R          444#
START   000244R          444     454#
STAT    000026R          444#
SVR0    000062R          444#
SVR1    000064R          444#
```

```
SVR2    000066R          444#
SVR3    000070R          444#
SVR4    000072R          444#
SVR5    000074R          444#
SVR6    000076R          444#
SYSCNT  000052R          444#
TABEND  000742R          484     487     555#
TABEN2  001216R          562     564     599#
TABLE   000646R          483     522#
TABLE2  001206R          561     595#
TIMEND  001502R          609     647     653#
TIMINT  001262R          466     617#
TIMTAB  001470R          608     648#
TRPDFD= 000022           444#
TWO     000762R          563     565#
VECTOR  000010R          444#    462
WAIT    001514R          470     475     480     495     662#
WASADR  000104R          444#    514*    641*
WDFR    000116R          444#
WDTO    000114R          444#
XFLAG   000005R          444#
```

```
BKMOD      95#
BREAK     185#    664
BTOD      204#
CKDATA    240#
DATACK    249#
DATERR    138#    517     644
DFSEVN    272#    444
DSEVNT    282#    444
END       175#    674
ENDIT     166#    655
ENDMOD    171#
EQUATS    288#    444
EXIT      120#    570     615
GETPA     231#
GWBUFF    219#
HRDER     128#    582     629     673
IOMOD      91#    444
IOMODP    115#
IOMODR    111#
IOMODX    107#
MAP22     235#
MODULE      8#    444
MSG       154#
MSGN      158#
MSGS      162#
NBKMOD    103#
OTOA      190#
PIRQ      179#    574     621
RAND      124#
SBKMOD     99#
SOFER     144#    593


. ABS.  000000     000
        001626     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XKwGA0,XKWGA0.LST/CRF=DDXCOM.C6,CXKWGA.SRC
 RUN-TIME:  3  4  .7 SECONDS
 RUN-TIME RATIO:  44/9=4.8
 CORE USED:  7K  (13 PAGES)
```

IDENTIFICATION

PRODUCT CODE: AC-F658A-MC

PRODUCT NAME: CXVTVA0 VTV30J/H-VT30H MODULE

PRODUCT DATE: OCT 1, 1979

MAINTAINER: COMPUTER SPECIAL SYSTEMS
DIGITAL EQUIPMENT CO. LTD.
READING
BERKS. U.K.

VTV30-H/J OR VT30-H DEC/X11 MODULE

PROGRAM DESCRIPTION

1. ABSTRACT

VTV IS AN IOMOD WHICH EXERCISES A VTV30-H/J OR VT30-H COLOUR VIDEO DISPLAY CONTROLLER. THE EXERCISE CONSISTS OF CONTINUALLY UPDATING THE CONTENTS OF THE PICTURE STORE SO AS TO PRODUCE A MOVING PATTERN ON THE SCREEN OF THE T.V. MONITOR. NO SOFTWARE CHECKING IS CARRIED OUT, BUT ERRORS IN THE PICTURE STORE OR CHARACTER STORE AREAS WILL BE CLEARLY VISIBLE ON THE DISPLAY.

2. REQUIREMENTS

2.1 HARDWARE

VTV30-H/J OR VT30-H CONTROLLER WITH T.V. MONITOR

2.2 STORAGE

VTV REQUIRES 500 WORDS OF STORAGE.

3. PASS DEFINITION

ONE PASS OF THE VTV MODULE CONSISTS OF SIX COMPLETE SCREEN COLOUR CHANGES.

4. EXECUTION TIME

ONE PASS OF THE VTV MODULE TAKES APPROXIMATELY ONE MINUTE WHEN RUNNING ALONE.

5. CONFIGURATION REQUIREMENTS

5.1 DEFAULT PARAMETERS

DVADR: 164000
VECTOR: 170
BR1: 4

5.2 REQUIRED PARAMETERS

NONE.

6. DEVICE/OPTION SETUP

ENSURE THAT THE CONTROLLER MODULES ARE CORRECTLY INSTALLED AND CONNECTED TO THE COLOUR T.V. MONITOR. THE MONITOR SHOULD BE POWERED UP.

7. MODULE OPERATION

TEST SEQUENCE

A. SET UP DEVICE REGISTER ADDRESSES AND MODULE VARIABLES.

B. SET DISPLAY CHARACTER MATRIX (6 X 6 OR 8 X 8).

C. LOAD ENTIRE CHARACTER SET WITH DUMMY CHARACTERS (252 PATTERN).

D. LOAD SPECIAL PATTERNS FOR USE IN DISPLAY INTO CHARACTERS 0 - 26 (6 X 6) OR CHARACTERS 0 - 36 (8 X 8).

E. SET INITIAL COLOUR AND CHARACTER COMBINATION.

F. CHECK IF ALL COLOURS DONE. IF SO, REPORT END OF PASS AND GO TO E.

G. DRAW PATTERN ON SCREEN UNTIL FULL OF FOREGROUND COLOUR.

H. INCREMENT COLOUR COUNTER, GO TO F.

VISUAL OPERATION

THE PICTURE STORE IS PRESET TO BLUE BACKGROUND. A RED TRIANGLE IS THEN DRAWN, STARTING AT THE BOTTOM CENTRE OF THE SCREEN AND EXPANDING UPWARDS UNTIL THE SCREEN IS FULL OF RED FOREGROUND. THE SEQUENCE IS THEN REPEATED USING GREEN FOREGROUND ON RED BACKGROUND, BLUE ON GREEN, GREEN ON BLUE, RED ON GREEEN AND BLUE ON RED. AT END OF PASS, THE DISPLAY IS DISABLED THEN RESTART TAKES PLACE AT THE BEGINNING OF THE COLOUR SEQUENCE.

AT ALL TIMES, THE EDGES OF THE TRIANGLE SHOULD APPEAR SMOOTH AND MOVING STEADILY. THE APPEARANCE OF ANY VERTICALLY STRIPED CHARACTERS ON THE SCREEN INDICATES THAT AN ILLEGAL CHARACTER IS BEING DISPLAYED AT THAT POSITION. ANY OTHER PICTURE STORE OR CHARACTER STORE MALFUNCTIONS SHOULD BE JUST AS OBVIOUS.

8. OPERATION OPTIONS

SR1 = 0, 6 X 6 CHARACTER MATRIX.

SR1 = 1, 8 X 8 CHARACTER MATRIX.

9. NON STANDARD PRINTOUT

NONE.

```
1                       .ENDR
2 000000                IOMOD   <VTVA >,164000,170,4,0,0,1,0
  000000( ) 1212        .OTUNE  140000,VTVA ,164000,170,4,0,0,1,0
                        .TITLE  VTVA DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION  6      28-NOV-78
                ;       CSSUK
                        .LIST   BIN
                ;*****************************************************************
  000000                BEGIN:
  000000     126 MODNAM: .ASCII  /VTVA / ;MODULE NAME.
  000001     124
  000002     126
  000003     101
  000004     040
  000005     000 XFLAG:  .BYTE   OPEN    ;USED TO KEEP TRACK OF WBUFF USAGE
  000006 164000 ADDR:    164000+0        ;1ST DEVICE ADDR.
  000010 000170 VECTOR:  170+0           ;1ST DEVICE VECTOR.
  000012     200 BR1:    .BYTE   PRTY4+0 ;1ST BR LEVEL.
  000013     000 BR2:    .BYTE   PRTY0+0 ;2ND BR LEVEL.
  000014 000001 DVID1:   0+1             ;DEVICE INDICATOR 1.
  000016 000000 SR1:     OPEN            ;SWITCH REGISTER 1
  000020 000000 SR2:     OPEN            ;SWITCH REGISTER 2
  000022 000000 SR3:     OPEN            ;SWITCH REGISTER 3
  000024 000000 SR4:     OPEN            ;SWITCH REGISTER 4
                ;*****************************************************************
  000026 140000 STAT:    140000          ;STATUS WORD.
  000030 000620'INIT:    START           ;MODULE START ADDR.
  000032 000224'SPOINT:  MODSP           ;MODULE STACK POINTER.
  000034 000000 PASCNT:  0               ;PASS COUNTER.
  000036 000001 ICONT:   1               ;# OF ITERATIONS PER PASS=1
  000040 000000 ICOUNT:  0               ;LOC TO COUNT ITERATIONS
  000042 000000 SOFCNT:  0               ;LOC TO SAVE TOTAL SOFT ERRORS
  000044 000000 HRDCNT:  0               ;LOC TO SAVE TOTAL HARD ERRORS
  000046 000000 SOFPAS:  0               ;LOC TO SAVE SOFT ERRORS PER PASS
  000050 000000 HRDPAS:  0               ;LOC TO SAVE HARD ERRORS PER PASS
  000052 000000 SYSCNT:  0               ;# OF SYS ERRORS ACCUMULATED
  000054 000000 RANNUM:0                 ;HOLDS RANDOM# WHEN RAND MACRO IS CALLED
  000056                CONFIG:          ;RESERVED FOR MONITOR USE
  000056 000000 RES1:    0               ;RESERVED FOR MONITOR USE
  000060 000000 RES2:    0               ;RESERVED FOR MONITOR USE
  000062 000000 SVR0:    OPEN            ;LOC TO SAVE R0.
  000064 000000 SVR1:    OPEN            ;LOC TO SAVE R1.
  000066 000000 SVR2:    OPEN            ;LOC TO SAVE R2.
  000070 000000 SVR3:    OPEN            ;LOC TO SAVE R3.
  000072 000000 SVR4:    OPEN            ;LOC TO SAVE R4.
  000074 000000 SVR5:    OPEN            ;LOC TO SAVE R5.
  000076 000000 SVR6:    OPEN            ;LOC TO SAVE R6.
  000100 000000 CSRA:    OPEN            ;ADDR OF CURRENT CSR.
  000102                SBADR:           ;ADDR OF GOOD DATA, OR
  000102 000000 ACSR:    OPEN            ;CONTENTS OF CSR.
  000104                WASADR:          ;ADDR OF BAD DATA, OR
  000104 000000 ASTAT:   OPEN            ;STATUS REG CONTENTS.
  000106                ERRTYP:          ;TYPE OF ERROR
  000106 000000 ASB:     OPEN            ;EXPECTED DATA.
  000110 000000 AWAS:    OPEN            ;ACTUAL DATA.
  000112 001216'RSTRT:   RESTRT          ;RESTART ADDRESS AFTER END OF PASS
  000114 000000 WDTO:    OPEN            ;WORDS TO MEMORY PER ITERATION
```



```
  000116 000000 WDFR:    OPEN            ;WORDS FROM MEMORY PER ITERATION
  000120 000000 INTR:    OPEN            ;# OF INTERRUPTS PER ITERATION
  000122 000000 IDNUM:   0               ;MODULE IDENTIFICATION NUMBER=0
         000040         .REPT   SPSIZ    ;MODULE STACK STARTS HERE.
                        .NLIST
                        .WORD   0
                        .LIST
                        .ENDR
  000224                MODSP:
                        ;*****************************************************************
```

```
                        ;
                        ;CONSTANTS AND VARIABLES USED IN PROGRAM.
                        ;
                        .NLIST  BEX
000224 000000 ACNT:     OPEN            ;VARIABLES LOADED AT RUN-TIME.
000226 000000 BCNT:     OPEN
000230 000000 CHARA:    OPEN
000232 000000 CHARB:    OPEN
000234 000000 ACHAR:    OPEN
00236 000000 BCHAR:     OPEN
00240 000000 CCHAR:     OPEN
00242 000000 DCHAR:     OPEN
00244 000000 MAXY:      OPEN
00246 000000 MAXX:      OPEN
00250 000000 LX:        OPEN
00252 000000 RX:        OPEN
00254 000000 CHROW:     OPEN
00256 000000 CHCNT:     OPEN
00260 000014 M6X6:      14              ;CONSTANTS FOR 6X6 CHARACTER MATRIX.
00262 000013            13
00264 000350'           CHARA6
00266 000371'           CHARB6
00270 000412'           ACHAR6
00272 000420'           BCHAR6
00274 000426'           CCHAR6
00276 000434'           DCHAR6
00300 000057            57
00302 000120            120
00304 000047            47
00306 000050            50
00310 000006            6
00312 000005            5
00314 000020 M8X8:      20              ;CONSTANTS FOR 8X8 CHARACTER MATRIX.
00316 000017            17
00320 000442'           CHARA8
00322 000471'           CHARB8
00324 000520'           ACHAR8
00326 000530'           BCHAR8
00330 000540'           CCHAR8
00332 000550'           DCHAR8
00334 000045            45
00336 000100            100
00340 000037            37
00342 000040            40
00344 000010            8.
00346 000007            7
00350     000 CHARA6:   .BYTE   0,0,0,0,0,0     ;6X6 CHARACTER DATA
00356     004           .BYTE   4,14,34,74,174,374
00364     374           .BYTE   374,374,374,374,374
00371     000 CHARB6:   .BYTE   0,0,0,0,0,0
00376     200           .BYTE   200,300,340,360,370,374
00404     374           .BYTE   374,374,374,374,374
                        .EVEN
00412     001 ACHAR6:   .BYTE   1,2,3,4,5,6     ;6X6 CHARACTER TABLE.
00420     007 BCHAR6:   .BYTE   7,10,11,12,13,13
00426     014 CCHAR6:   .BYTE   14,15,16,17,20,21
00434     022 DCHAR6:   .BYTE   22,23,24,25,26,26
```



```
00442     000 CHARA8:   .BYTE   0,0,0,0,0,0,0,0 ;8X8 CHARACTER DATA
00452     001           .BYTE   1,3,7,17,37,77,177,377
00462     377           .BYTE   377,377,377,377,377,377,377
00471     000 CHARB8:   .BYTE   0,0,0,0,0,0,0,0
00500     200           .BYTE   200,300,340,360,370,374,376,377
00510     377           .BYTE   377,377,377,377,377,377,377
                        .EVEN
00520     001 ACHAR8:   .BYTE   1,2,3,4,5,6,7,10        ;8X8 CHARACTER TABLE
00530     011 BCHAR8:   .BYTE   11,12,13,14,15,16,17,17
00540     020 CCHAR8:   .BYTE   20,21,22,23,24,25,26,27
00550     030 DCHAR8:   .BYTE   30,31,32,33,34,35,36,36
00560     241 COL:      .BYTE   241,212,224,242,221,214
                        .EVEN
                        .LIST   BEX
00566 001702'MSG1:      MSG66
00570 177777            177777
00572 001725'MSG2:      MSG88
00574 177777            177777
00576 000000 ITCNT:     0
00600 000000 CSR:       OPEN            ;CONTROL & STATUS REGISTER ADDRESS
00602 000000 DBUF:      OPEN            ;DATA BUFFER REGISTER ADDRESS
00604 000000 CAR:       OPEN            ;CURSOR REGISTER ADDRESS
00606 000000 CHSR:      OPEN            ;CHARACTER STORE REGISTER ADDRESS
00610 000000 START1:    OPEN            ;STARTING COORDINATE FOR LEFT DIAGONAL
00612 000000 START2:    OPEN            ;       DITTO           FOR RIGHT DIAGONAL
00614 000000 XY1:       OPEN            ;RUNNING COORDINATE FOR LEFT DIAGONAL
00616 000000 XY2:       OPEN            ;       DITTO           FOR RIGHT DIAGONAL
```

```
                        ;
                        ;INITIALISE AND LOAD CHARACTER SET.
                        ;
000620 016700 177162 START: MOV     ADDR,R0         ;GET BASE ADDRESS
000624 010067 177750         MOV     R0,CSR          ;LOAD UP TABLE; CONTROL & STATUS
000630 005720                TST     (R0)+
000632 010067 177744         MOV     R0,DBUF         ;DATA BUFFER
000636 005720                TST     (R0)+           ;
000640 010067 177740         MOV     R0,CAR          ;CURSOR REGISTER
000644 005720                TST     (R0)+           ;
000646 010067 177734         MOV     R0,CHSR         ;CHARACTER STORE REGISTER
000652 016700 177132         MOV     VECTOR,R0       ;NOW LOAD INTERRUPT VECTOR.
000656 012720 001164'        MOV     #RDYINT,(R0)+   ;SERVICE ROUTINE
000662 005010                CLR     (R0)
000664 116710 177122         MOVB    BR1,(R0)        ;AND PRIORITY.
000670 032767 000001 177120  BIT     #1,SR1          ;SEE WHICH MMATRIX REQUIRED.
000676 001417                BEQ     2$              ;BRANCH IF 6 X 6.
000700 005077 177674         CLR     @CSR            ;SET UP FOR 8 X 8.
000704 012700 000224'        MOV     #ACNT,R0        ;GET START OF VARIABLE TABLE
000710 012701 000314'        MOV     #M8X8,R1        ;GET START OF CONSTANT TABLE
000714 012702 000016         MOV     #14.,R2         ;COUNT OF TABLE ENTRIES
000720 012120 000000' 1$:    MOV     (P1)+,(R0)+     ;LOAD AN ENTRY
000722 005302                DEC     R2              ;COUNT.
000724 001375                BNE     1$              ;LOOP UNTIL TABLE FINISHED.
000726 104403                MSGN$,BEGIN,MSG2        ;ASCII MESSAGE CALL WITH COMMON HEADER
000730 000572'
000732 000000'
000734 000417                BR      4$              ;THEN EXIT.
000736 012777 001400 177634 2$:  MOV     #1400,@CSR      ;SET UP FOR 6 X 6
000744 012700 000224'        MOV     #ACNT,R0        ;GET START OF VARIABLE TABLE.
000750 012701 000260'        MOV     #M6X6,R1        ;GET START OF CONSTANT TABLE.
000754 012702 000016         MOV     #14.,R2         ;COUNT OF TABLE ENTRIES.
000760 012120 000000' 3$:    MOV     (R1)+,(R0)+     ;LOAD AN ENTRY.
000762 005302                DEC     R2              ;COUNT.
000764 001375                BNE     3$              ;LOOP UNTIL TABLE FINISHED.
000766 104403                MSGN$,BEGIN,MSG1        ;ASCII MESSAGE CALL WITH COMMON HEADER
000770 000000'
000772 000566'
```



```
000774 005077 177604 4$:     CLR     @CAR            ;ZERO CURSOR
001000 016700 177602         MOV     CHSR,R0         ;GET HIGH BYTE ADDRESS
001004 105060 000001         CLRB    1(R0)           ;CLR HIGH BYTE OF CHSR.
                        ;
                        ;LOAD ALL DUMMY CHARACTERS FIRST.
                        ;
001010 012700 000200 LDCH:   MOV     #128.,R0        ;NUMBER OF CHARACTERS
001014 016701 177234 1$:     MOV     CHROW,R1        ;BYTES PER CHARACTER
001020 112777 000252 177560 2$:  MOVB    #252,@CHSR      ;LOAD A BYTE.
001026 005301                DEC     R1              ;CHARACTER DONE?
001030 001373                BNE     2$              ;BRANCH IF NOT
001032 005300                DEC     R0              ;ALL CHARACTERS DONE?
001034 001367                BNE     1$              ;BRANCH IF NOT.
                        ;
                        ;DUMMY CHARACTERS LOADED, NOW LOAD SPECIALS.
                        ;
001036 016700 177544         MOV     CHSR,R0         ;GET HIGH BYTE ADDRESS
001042 105060 000001         CLRB    1(R0)           ;ZERO CHAR ADDRESS.
001046 016700 177156         MOV     CHARA,R0        ;GET DATA POINTER.
001052 016702 177146         MOV     ACNT,R2         ;NUMBER OF CHARACTERS.
001056 010001 3$:    MOV     R0,R1           ;COPY POINTER.
001060 016703 177170         MOV     CHROW,R3        ;BYTES PER CHARACTER.
001064 112177 177516 4$:     MOVB    (R1)+,@CHSR     ;LOAD A BYTE.
001070 005303                DEC     R3              ;CHARACTER DONE?
001072 001374                BNE     4$              ;BRANCH IF NOT.
001074 005200                INC     R0              ;UPDATE POINTER
001076 005302                DEC     R2              ;THIS LOT DONE?
001100 001366                BNE     3$              ;BRANCH IF NOT.
001102 016700 177124         MOV     CHARB,R0        ;NEXT POINTER
001106 016702 177114         MOV     BCNT,R2         ;NUMBER OF CHARACTERS.
001112 010001 5$:    MOV     R0,R1           ;COPY POINTER.
001114 016703 177134         MOV     CHROW,R3        ;BYTES PER CHARACTER.
001120 112177 177462 6$:     MOVB    (R1)+,@CHSR     ;LOAD A BYTE
001124 005303                DEC     R3              ;CHAR DONE?
001126 001374                BNE     6$
001130 005200                INC     R0              ;UPDATE POINTER.
001132 005302                DEC     R2              ;ALL DONE?
001134 001366                BNE     5$              ;BRANCH IF NOT
001136 112777 000242         MOVB    #242,@DBUF      ;SET GREEN ON BLUE.
```

```
          177436
74 01144 112777           MOVB    #300,@DBUF      ;CLEAR BLINK CONTROL.
          000300
          177430
75 01152 052777           BIS     #110,@CSR       ;SET INT. ENAB. AND PRESET.
          000110
          177420
76 01160 104400           EXIT$,BEGIN     ;EXIT TO MONITOR, WAIT FOR INTERRUPT.
   01162 000000'
77 01164 042777 RDYINT:   BIC     #100,@CSR       ;CLEAR INT. ENABLE.
          000100
          177406
78                        ;-------------------------------------------------------
   01172 000004           PIRQ$,BEGIN,1$  ; QUEUE  CONTINUE AT 1$ AND RTI
   01174 000000'
   01176 001200'
                          ;-------------------------------------------------------
79 01200 032777 1$:       BIT     #BIT10,@CSR     ;CHECK FOR 525 LINES.
          002000
          177372
80 01206 001403           BEQ     RESTRT          ;IF NOT, OK
81 01210 162767           SUB     #10,MAXY        ;YES,REDUCE MAXIMUM Y COORD.
          000010
          177026
```



```
1 001216 052777 RESTRT:   BIS     #1,@CSR         ;TURN DISPLAY ON.
          000001
          177354
2 001224 012777           MOV     #TIMINT,@VECTOR ;SET FOR TIMER SERVICE
          001356'
          176556
3 001232 005067           CLR     ITCNT           ;RESET COLOUR COUNTER.
          177340
4 001236 026727 PASS:     CMP     ITCNT,#6        ;ALL COLOURS DONE YET?
          177334
          000006
5 001244 001005           BNE     1$              ;IF NOT, CARRY ON.
6 001246 042777           BIC     #40101,@CSR     ;DISPLAY OFF, DISABLE INTS.
          040101
          177324
7 001254 104413           ENDIT$,BEGIN    ;SIGNAL END OF ITERATION.
  001256 000000'
                                    ;MONITOR SHALL TEST END OF PASS
8 001260 016701 1$:       MOV     ITCNT,R1        ;GET COLOUR INDEX.
          177312
9 001264 116177           MOVB    COL(R1),@DBUF   ;SET COLOUR.
          000560'
          177310
10 01272 005267           INC     ITCNT           ;UPDATE COUNTER FOR NEXT TIME.
          177300
11 01276 116767           MOVB    LX,START1       ;SET 1ST X COORD (LEFT).
          176746
          177304
12 01304 116767           MOVB    MAXY,START1+1   ;SET 1ST Y COORD (LEFT).
          176734
          177277
13 01312 116767           MOVB    RX,START2       ;SET 1ST X COORD (RIGHT).
          176734
          177272
14 01320 116767           MOVB    MAXY,START2+1   ;SET 1ST Y COORD (RIGHT).
          176720
          177265
15 01326 005001 STROW:    CLR     R1              ;P1 IS NOW CHARACTER INDEX.
16 01330 016767 FIRST:    MOV     START1,XY1      ;SET RUNNING COORDINATES
          177254
          177256
17 01336 016767           MOV     START2,XY2      ;
          177250
          177252
18 01344 052777           BIS     #BIT14,@CSR     ;ENABLE TIMER
          040000
          177226
19 01352 104400           EXIT$,BEGIN     ;EXIT TO MONITOR, WAIT FOR INTERRUPT.
   01354 000000'
20 01356 042777 TIMINT:   BIC     #BIT15!BIT14,@CSR       ;DISABLE TIMER.
          140000
          177214
21                        ;-------------------------------------------------------
   01364 000004           PIRQ$,BEGIN,CONT        ; QUEUE  CONTINUE AT CONT AND RTI
   01366 000000'
   01370 001372'
                          ;-------------------------------------------------------
```

```
22 01372 126767 CONT:   CMPB    XY1,MAXX        ;AT MAXIMUM X ?
         177216
         176646
23 01400 103013         BHIS    11$
24 01402 016777         MOV     XY1,@CAR        ;SET FOR 1ST LEFT
         177206
         177174
25 01410 105777 18:     TSTB    @CSR            ;CHECK READY
         177164
26 01414 100375         BPL     1$              ;WAIT IF NECESSARY.
27 01416 016702         MOV     ACHAR,R2        ;GET CHARACTER ADDRESS
         176612
28 01422 060102         ADD     R1,R2           ;ADD INDEX
29 01424 111277         MOVB    (R2),@DBUF      ;AND SEND IT.
         177152
30 01430 126767 11$:    CMPB    XY2,MAXX        ;AT MAXIMUM X ?
         177162
         176610
31 01436 103013         BHIS    12$
32 01440 016777         MOV     XY2,@CAR        ;MOVE CURSOR TO 1ST RIGHT.
         177152
         177136
33 01446 105777 28:     TSTB    @CSR            ;CHECK READY
         177126
34 01452 100375         BPL     2$
35 01454 016702         MOV     CCHAR,R2        ;GET CHARACTER ADDRESS
         176560
36 01460 060102         ADD     R1,R2           ;ADD INDEX
37 01462 111277         MOVB    (R2),@DBUF      ;AND SEND IT.
         177114
38 01466 126767 12$:    CMPB    XY1,LX          ;IS X AT CENTRE?
         177122
         176554
39 01474 001452         BEQ     NXCHAR          ;IF SO, SKIP 2ND PAIR.
40 01476 105267         INCB    XY1             ;UPDATE COORDINATES
         177112
41 01502 105367         DECB    XY2             ;FOR 2ND PAIR.
         177110
42 01506 126767         CMPB    XY1,MAXX        ;AT MAXIMUM X ?
         177102
         176532
43 01514 103013         BHIS    13$
44 01516 016777         MOV     XY1,@CAR        ;SET CURSOR FOR LEFT.
         177072
         177060
45 01524 105777 38:     TSTB    @CSR            ;CHECK FOR READY
         177050
46 01530 100375         BPL     3$
47 01532 016702         MOV     BCHAR,R2        ;GET CHARACTER ADDRESS
         176500
48 01536 060102         ADD     R1,R2           ;ADD INDEX
49 01540 111277         MOVB    (R2),@DBUF      ;AND SEND IT.
         177036
50 01544 126767 13$:    CMPB    XY2,MAXX        ;AT MAXIMUM X ?
         177046
         176474
51 01552 103013         BHIS    14$
```



```
52 01554 016777         MOV     XY2,@CAR        ;SET CURSOR FOR RIGHT.
         177036
         177022
53 01562 105777 48:     TSTB    @CSR            ;CHECK FOR READY.
         177012
54 01566 100375         BPL     4$
55 01570 016702         MOV     DCHAR,R2        ;GET CHARACTER ADDRESS
         176446
56 01574 060102         ADD     R1,R2           ;ADD INDEX
57 01576 111277         MOVB    (R2),@DBUF      ;AND SEND IT.
         177000
58 01602 105767 14$:    TSTB    XY1+1           ;Y REACHED TOP?
         177007
59 01606 001405         BEQ     NXCHAR          ;IF SO, CHANGE CHARS.
60 01610 105367         DECB    XY1+1           ;NO, UPDATE COORDINATES
         177001
61 01614 105367         DECB    XY2+1           ;
         176777
62 01620 000664         BR      CONT            ;AND CONTINUE ROW.
63 01622 020167 NXCHAR: CMP     R1,CHCNT        ;ALL CHARACTERS DONE?
         176430
64 01626 001402         BEQ     NXTROW          ;IF SO, START NEW ROW.
65 01630 005201         INC     R1              ;NO, NEXT SET OF CHARS
66 01632 000636         BR      FIRST           ;GO BACK TO ROW START.
67 01634 126767 NXTROW: CMPB    START2,MAXX     ;RUN OUT OF X'S YET?
         176752
         176404
68 01642 001405         BEQ     NEXTY           ;IF SO, CHANGE Y STARTS.
69 01644 105367         DECB    START1          ;NO, CHANGE X STARTS.
         176740
70 01650 105267         INCB    START2
         176736
71 01654 000624         BR      STROW           ;START NEW ROW.
72 01656 105767 NEXTY:  TSTB    START1+1        ;Y START REACHED TOP?
         176727
73 01662 001002         BNE     1$
74 01664 000167         JMP     PASS            ;IF SO, GO TO PASS COUNT.
         177346
75 01670 105367 1$:     DECB    START1+1        ;NO, CHANGE Y STARTS
         176715
76 01674 105367         DECB    START2+1        ;
         176713
77 01700 000612         BR      STROW           ;START NEW ROW.
78                      .NLIST BEX
79 01702     115 MSG66: .ASCIZ  'MATRIX IS 6 BY 6.%'
80 01725     115 MSG88: .ASCIZ  'MATRIX IS 8 BY 8.%'
81                      .LIST BEX
82                      .EVEN
83       000001'        .END                    ;END OF PROGRAM.
```

```
ACHAR   000234R     ACHAR6  000412R     ACHAR8  000520R
ACNT    000224R     ACSR    000102R     ADDR    000006R
ADDR22= 001000      ASB     000106R     ASTAT   000104R
AWAS    000110R     BCHAR   000236R     BCHAR6  000420R
BCHAR8  000530R     BCNT    000226R     BEGIN   000000R
BIT0  = 000001      BIT1  = 000002      BIT10 = 002000
BIT11 = 004000      BIT12 = 010000      BIT13 = 020000
BIT14 = 040000      BIT15 = 100000      BIT2  = 000004
BIT3  = 000010      BIT4  = 000020      BIT5  = 000040
BIT6  = 000100      BIT7  = 000200      BIT8  = 000400
BIT9  = 001000      BREAK$= 104407      BR1     000012R
BR2     000013R     BTOD$ = 104421      CAR     000604R
CCHAR   000240R     CCHAR6  000426R     CCHAR8  000540R
CDATA$= 104412      CHARA   000230R     CHARA6  000350R
CHARA8  000442R     CHARB   000232R     CHARB6  000371R
CHARB8  000471R     CHCNT   000256R     CHROW   000254R
CH8R    000606R     COL     000560R     CONFIG  000056R
CONT    001372R     CSR     000600R     CSRA    000100R
DATCK$= 104411      DATER$= 104404      DBUF    000602R
DCHAR   000242R     DCHAR6  000434R     DCHAR8  000550R
DVID1   000014R     ENDIT$= 104413      END$  = 104410
ERRTYP  000106R     EXIT$ = 104400      FIRST   001330R
GETPA$= 104415      GWBUF$= 104414      HRDCNT  000044R
HRDER$= 104405      HRDPAS  000050R     ICONT   000036R
ICOUNT  000040R     IDNUM   000122R     INIT    000030R
INTR    000120R     ITCNT   000576R     LDCH    001010R
LX      000250R     MAP22$= 104416      MAXX    000246R
MAXY    000244R     MODNAM  000000R     MODSP   000224R
MSGN$ = 104403      MSGS$ = 104402      MSG$  = 104401
MSG1    000566R     MSG2    000572R     MSG66   001702R
MSG88   001725R     M6X6    000260R     M8X8    000314R
NEXTY   001656R     NULL  = 000000      NXCHAR  001622R
NXTROW  001634R     OPEN  = 000000      OTOA$ = 104420
PASCNT  000034R     PASS    001236R     PC    =%000007
PIRQ$ = 000004      POPSP = 005726      POPSP2= 022626
PRTY  = 000000      PRTY0 = 000000      PRTY1 = 000040
PRTY2 = 000100      PRTY3 = 000140      PRTY4 = 000200
PRTY5 = 000240      PRTY6 = 000300      PRTY7 = 000340
PS    = 177776      PSW   = 177776      PUSH  = 005746
PUSH2 = 024646      RAND$ = 104417      RANNUM  000054R
RDYINT  001164R     RESTRT  001216R     RES1    000056R
RES2    000060R     RSTRT   000112R     RX      000252R
R0    =%000000      R1    =%000001      R2    =%000002
R3    =%000003      R4    =%000004      R5    =%000005
R6    =%000006      R7    =%000007      SRADR   000102R
SOFCNT  000042R     SOFER$= 104406      SOFPAS  000046R
SP    =%000006      SPOINT  000032R     SPSIZ = 000040
SR1     000016R     SR2     000020R     SR3     000022R
SR4     000024R     START   000620R     START1  000610R
START2  000612R     STAT    000026R     STROW   001326R
SVR0    000062R     SVR1    000064R     SVR2    000066R
SVR3    000070R     SVR4    000072R     SVR5    000074R
SVR6    000076R     SYSCNT  000052R     TIMINT  001356R
TRPDFD= 000022      VECTOR  000010R     WASADR  000104R
WDFR    000116R     WDTO    000114R     XFLAG   000005R
XY1     000614R     XY2     000616R
. ABS.  000000      000
```



```
        001750      001
ERRORS DETECTED: 0
FREE CORE: 11488. WORDS

XVTVA0,OBJ,XVTVA0,SEQ=[40,0]NEWDDX.P11,[40,30]XVTVA0.P11
```

```
                        .REM %
```

IDENTIFICATION

PRODUCT CODE:	AC-F542A-MC
PRODUCT NAME:	CXBMIA0 BOOT ROM MODULE
PROGRAM DATE:	MAY 1979
MAINTAINER:	DEC/X11 SUPPORT GROUP

```
                        1. ABSTRACT

                                BMI IS A BKMOD THAT DOES A READ CRC AND
                                LPC CHECK ON THE ROM SELECTED BY SR1.


                        2. REQUIREMENTS

                                HARDWARE: ANY PDP-11 WITH A ROM.
                                STORAGE : BMI REQUIRES
                                                1. DECIMAL WORDS: 417
                                                2. OCTAL WORDS  :641
                                                3. OCTAL BYTES  :1502


                        3. PASS DEFINITION

                                ONE PASS CONSISTS OF DOING A CRC AND LPC CHECK ON
                                THE ROM 3000(8) TIMES.


                        4. EXECUTION TIME

                                BMI RUNNING ALONE TAKES APPROXIMATELY ONE MINUTE
                                45 SECONDS.


                        5. CONFIGURATION REQUIREMENTS

                                SR1 IS USED TO SELECTTHE VERSION OF THE ROM BOOTSTRAP TO BE TESTED
                                ACCORDING TO THE FOLLOWING TABLE. NOTE: THESE SETTINGS ARE
                                OCTAL NUMBERS. THEY ARE NOT PARTICULAR SWITCHES SET TO A
                                ONE. FOR EXAMPLE, TO SELECT THE M9301-YH VERSION, SET
                                SWITCHES #3 AND #1 IN SR1.THIS CORRESPONDS TO AN OCTAL 12.

                                        SWR     MODULE VERSION
                                        ---     --------------

                                        1       M9301-YA
                                        2       M9301-YB
                                        3       M9301-YC
                                        4       M9400-YA (OR YC)
                                        5       M9301-YF
                                        6       M7942-YB
                                        7       M9301-YD
                                        10      M9400-YH (OR YK)
                                        11      M9311
                                        12      M9301-YH
                                        13      M9301-YE
                                        14      M9301-YJ
                                        15      M9400-YN
                                        16      UNUSED
```

```
474
475                                     6. DEVICE/OPTION SETUP
476
477                                             NONE
478
479
480                                     7. MODULE OPERATION
481
482                                             READS EACH ROM LOCATION AND CALCULATES A CRC WORD
483                                             AND LPC WORD FOR THE SELECTED ROM. IT COMPARES BOTH
484                                             WORDS AGAINST THE EXPECTED VALUE THAT IS GETS FROM
485                                             THE TABLE.
486
487
488                                     8. OPERATING OPTIONS
489
490                                             NONE
491
492
493                                     9. NON-STANDARD PRINTOUTS
494
495                                             NONE
496
497
498
499                                     ;
```

```
                                        PKMOD   <BMIA >,,,,,,300,13
                                                MODULE  40020,BMIA ,,,,,,300,13
                                                .TITLE  BMIA DEC/X11 SYSTEM EXERCISER MODULE
                                        ;       DDXCOM  VERSION  6       23-MAY-78
                                                .LIST   BIN
                                        ;*****************************************************************************
000000'                                 BEGIN:
000000' 046502  040511      040 MODNAM: .ASCII  /BMIA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000000                  ADDR:   +0              ;1ST DEVICE ADDR.
000010' 000000                  VECTOR: +0              ;1ST DEVICE VECTOR.
000012'     000                 BR1:    .BYTE   PRTY+0          ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY+0          ;2ND BR LEVEL.
000014' 000001                  DVID1:  +1              ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN            ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN            ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN            ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN            ;SWITCH REGISTER 4
                                        ;*****************************************************************************
000026' 040020                  STAT:   40020           ;STATUS WORD.
000030' 000252'                 INIT:   START           ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP           ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0               ;PASS COUNTER.
000036' 000300                  ICONT:  300             ;# OF ITERATIONS PER PASS=300
000040' 000000                  ICOUNT: 0               ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0               ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0               ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0               ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0               ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0               ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0               ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                 ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0               ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0               ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN            ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN            ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN            ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN            ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN            ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN            ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN            ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN            ;ADDR OF CURRENT CSR.
000102'                         SBADR:                  ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN            ;CONTENTS OF CSR.
000104'                         WASADR:                 ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN            ;STATUS REG CONTENTS.
000106'                         ERRTYP:                 ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN            ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN            ;ACTUAL DATA.
000112' 000252'                 RSTRT:  RESTRT          ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN            ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN            ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN            ;# OF INTERRUPTS PER ITERATION
000122' 000013                  IDNUM:  13              ;MODULE IDENTIFICATION NUMBER=13
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
000040
```



```
                                                .NLIST
                                                .WORD   0
                                                .LIST
                                                .ENDR
000224'                                 MODSP:
                                        ;*****************************************************************************
000224' 173000                  ROMSA1: 173000
000226' 001000                  DATLN1: 512.
000230' 165000                  ROMSA2: 165000
000232' 001000                  DATLN2: 512.
000234' 000000                  XORS:   0
000236' 000000                  EXCRC:  0
000240' 000000                  EXLPC:  0
000242' 000000                  ACTCRC: 0
000244' 000000                  ACTLPC: 0
000246' 000000                  PARCNT: 0
000250' 000000                  TYPOUT: 0

000252'                         START:
000252'                         RESTRT:
000252' 016700  177540          1$:     MOV     SR1,R0                  ;GET SR1
000256' 001005                          BNE     ST2
000260' 104403  000000' 001442'         MSGN$,BEGIN,ADR ;ASCII MESSAGE CALL WITH COMMON HEADER
000266' 104410  000000'                 END$,BEGIN              ;
000272' 006300                  ST2:    ASL     R0
000274' 016067  001142' 177736          MOV     TXLPC(R0),EXLPC         ;FETCH EXPECT. LPC
000302' 016067  001074' 177726          MOV     TXCRC(R0),EXCRC         ;FETCH EXPECTED CRC
000310' 016067  001212' 177710          MOV     TDLN1(R0),DATLN1        ;FETCH 1ST LENGTH
000316' 016067  001260' 177700          MOV     TRMSA1(R0),ROMSA1       ;FETCH 1ST STARTING ADDR.
000324' 016067  001326' 177700          MOV     TDLN2(R0),DATLN2        ;FETCH 2ND LENGTH
000332' 016067  001374' 177670          MOV     TRMSA2(R0),ROMSA2       ;FETCH 2ND STARTING ADDR
000340' 005067  177676          1$:     CLR     ACTCRC                  ;CLEAR STORAGE FOR ACTUAL CRC
000344' 005067  177674                  CLR     ACTLPC                  ;CLEAR STORAGE FOR ACTUAL LPC
000350' 016700  177652                  MOV     DATLN1,R0               ;SET LENGTH OF 1ST ROM SPACE
000354' 001413                          BEQ     CH0                     ;IF NO VERSION SELECTED:  BR
000356' 016701  177642                  MOV     ROMSA1,R1               ;POINT TO START OF 1ST ROM SPACE
000362' 004767  000200                  JSR     PC,CRC                  ;COMPUTE FIRST HALF OF CRC
000366' 016701  177632                  MOV     ROMSA1,R1               ;POINT TO START OF 1ST ROM ADDRS.
000372' 016700  177630                  MOV     DATLN1,R0               ;SET LENGTH OF 1ST ROM ADDRS.
000376' 006200                          ASR     R0                      ;CONVERT TO WORDS
000400' 004767  000432                  JSR     PC,LPC                  ;COMPUTE FIRST HALF OF CRC
000404' 016701  177620          CH0:    MOV     ROMSA2,R1               ;POINT TO 2ND ROM ADDRS.
000410' 016700  177616                  MOV     DATLN2,R0               ;SET LENGTH OF 2ND ROM ADDRS.
000414' 001411                          BEQ     CH1                     ;BR IF THIS SPACE NOT USED
000416' 004767  000144                  JSR     PC,CRC                  ;COMPUTE REMAINDER OF CRC
000422' 016701  177602                  MOV     ROMSA2,R1               ;POINT TO START OF 2ND ROM ADDRS.
000426' 016700  177600                  MOV     DATLN2,R0               ;SET LENGTH OF 2ND ROM ADDRS.
000432' 006200                          ASR     R0                      ;CONVERT TO WORDS
000434' 004767  000376                  JSR     PC,LPC                  ;COMPUTE REMAINDER OF LPC
000440' 026767  177572  177574  CH1:    CMP     EXCRC,ACTCRC            ;COMPUTED = EXPECTED ?
000446' 001420                          BEQ     CK1                     ;IF SO:  BR
000450' 016767  177550  177424          MOV     ROMSA1,ACSR
000456' 016767  177542  177416          MOV     ROMSA1,SBADR
000464' 005067  177414                  CLR     WASADR
000470' 016767  177542  177410          MOV     EXCRC,ASB
000476' 016767  177540  177404          MOV     ACTCRC,AWAS
```

```
                                        ;*****************************************************************
(1)  000504' 104404  000000'            DATER$,BEGIN            ;DATA ERROR!!!
(1)                                     ;*****************************************************************
                CK1:    CMP     ACTLPC,EXLPC            ;COMPARE EXPT. LPC=ACTUAL LPC
                        BEQ     PASS                    ;IF SO:  BR
                        MOV     ROMSA1,ACSR
                        MOV     ROMSA1,SRADR
                        CLR     WASADR
                        MOV     EXLPC,ASE
                        MOV     ACTLPC,AWAS
                                        ;*****************************************************************
(1)  000554' 104404  000000'            DATER$,BEGIN            ;DATA ERROR!!!
(1)                                     ;*****************************************************************
000560'         PASS:
(1)  000560' 104413  000000'            ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
(1)                                                             ;MONITOR SHALL TEST END OF PASS
000564' 000632                          BR      RESTPT
                CRC:    MOV     ACTCRC,XORS
                CL0:    MOVB    (R1),R4                 ;GET CHAR.
                        CMP     #173024,R1              ;LOCATION EFFECTED BY SWITCHES
                        BNE     CL3                     ;IF NOT:  BR
                        DEC     R0                      ;FIX COUNTERS
                        DEC     R0
                        TST     (R1)+                   ;FIX POINTER
                        BP      CL0                     ;CONTINUE
                CL3:    JSR     PC,PARITY               ;GO GET PARITY
                        JSR     PC,XOR                  ;XOR CHAR
                        CLC
                        ROR     R4                      ;ROTATE 1 POS. RIGHT
                        BCC     CL2                     ;IF NO CARRY:  BR
                        BIS     #400,R4                 ;SET BIT NINE
                        CLC
                CL1:    MOV     R4,R5                   ;SAVE CHAR
                        BIC     #177703,R5
                        COM     R5
                        BIC     #177703,R5
                        BIC     #74,R4
                        BIS     R5,R4
                CL2:    MOV     R4,XORS
                        DEC     R0
                        BEQ     CLLAST                  ;IF LAST CAR.:  BR
                        JMP     CL0                     ;GET NEXT CHAR.
                CLLAST: MOV     XORS,R4
                        COM     XORS
                        BIC     #177050,XORS
                        BIC     #177727,R4              ;COMPLEMENT ALL BUT BITS 3 & 5
                        BIS     R4,XORS
                        MOV     XORS,ACTCRC
                        RTS     PC
                PARITY: CLR     PARCNT                  ;CLEAR BIT COUNTER
                        MOV     #10,R3                  ;SET NO. OF BITS
                CLP0:   BIT     #1,R4                   ;SEE IF ONE BIT
                        BEQ     CLP1                    ;IF NOT:  BR
```



```
                        INC     PARCNT                  ;BUMP COUNTER
                CLP1:   CLC
                        ROR     R4                      ;ROTATE TO NEXT BIT
                        DEC     R3
                        BNE     CLP0                    ;CONTINUE FOR ALL BITS
                        MOVB    (R1)+,R4
                        BIC     #177400,R4
                        BIT     #1,PARCNT               ;SEE IF ODD # OF ONE BITS
                        BNE     CLP2                    ;IF SO:  BR
                        BIS     #400,R4                 ;SET PARITY BIT
                CLP2:   RTS     PC                      ;EXIT

                XOR:    MOV     R4,-(SP)                ;XOR SUBROUTINE; R4 WITH XORS
                        BIC     XORS,(SP)
                        BIC     R4,XORS
                        BIS     (SP)+,XORS
                        MOV     XORS,R4
                        RTS     PC

                LPC:    MOV     ACTLPC,XORS
                LPC1:   MOV     (R1)+,R4
                        CMP     #173026,R1              ;LOCATION EFFECTED BY SWITCHES
                        BEQ     LPC2                    ;IF SO:  SKIP LOC. BY BRANCHING
                        JSR     PC,XOR
                LPC2:   DEC     R0
                        BNE     LPC1
                        MOV     XORS,ACTLPC
                        RTS     PC


                TXCRC:  -1              ;TABLE OF CRC VALUES
                        571             ;M9301 - YA VERSION
                        457             ;M9301 - YB VERSION
                        243             ;M9301 - YC VERSION
                        635             ;M9400 - YA(OR YC) VERSION
                        207             ;M9301 - YF VERSION
                        670             ;M7942 - YB VERSION
                        132             ;M9301- YD VERSION
                        374             ;M9400 - YH (OR YK) VERSION
                        710             ;M9311 VERSION
                        536             ;M9301 - YH VERSION
                        752             ;M9301 - YE VERSION
                        633             ;M9301 - YJ VERSION
                        650             ;M9400 - YN VERSION
                        -1
                        -1
                        -1
                        -1
                        -1

                TXLPC:  -1              ;TABLE OF LPC VALUES
                        133725          ;M9301 - YA VERSION
                        17563           ;M9301 - YB VERSION
                        141744          ;M9301 - YC VERSION
                        47613           ;M9400 - YA(OR YC) VERSION
                        114175          ;M9301 - YF VERSION
```

```
660  001156' 146126                          146126          ;M7942 - YB VERSION
661  001160' 132161                          132161          ;M9301 - YD VERSION
662  001162' 143466                          143466          ;M9400 - YH(OR YK) VERSION
663  001164' 036743                          036743          ;M9311 VERSION
664  001166' 125411                          125411          ;M9301 - YH VERSION
665  001170' 066246                          066246          ;M9301 - YE VERSION
666  001172' 132367                          132367          ;M9301 - YJ VERSION
667  001174' 030210                          30210           ;M9400 - YN VERSION
668  001176' 177777                          -1
669  001200' 177777                          -1
670  001202' 177777                          -1
671  001204' 177777                          -1
672  001206' 177777                          -1
673  001210' 177777                          -1
674
675  001212' 177777              TDLN1:      -1              ;TABLE OF THE LENGTH (BYTES) OF 1ST ROM ADDRESS SPACE
676  001214' 001000                          1000            ;M9301 - YA VERSION
677  001216' 001000                          1000            ;M9301 - YB VERISION
678  001220' 001000                          1000            ;M9301 - YC VERSION
679  001222' 001000                          1000            ;M9400 - YA(OR YC) VERSION
680  001224' 001000                          1000            ;M9301 - YF VERSION
681  001226' 004000                          4000            ;M7942 - YB VERSION
682  001230' 001000                          1000            ;M9301 - YD VERSION
683  001232' 001000                          1000            ;M9400 - YH(OR YK) VERSION
684  001234' 001000                          1000            ;M9311 VERSION
685  001236' 000734                          734             ;M9301 - YH VERSION
686  001240' 001000                          1000            ;M9301 - YE VERSION
687  001242' 001000                          1000            ;M9301 - YJ VERSION
688  001244' 001000                          1000            ;M9400 - YN VERSION
689  001246' 177777                          -1
690  001250' 177777                          -1
691  001252' 177777                          -1
692  001254' 177777                          -1
693  001256' 177777                          -1
694
695  001260' 177777              TRMSA1:     -1              ;TABLE OF THE STARTING ADDRESS OF 1ST ROM SPACE
696  001262' 173000                          173000          ;M9301 - YA VERSION
697  001264' 173000                          173000          ;M9301 - YB VERSION
698  001266' 173000                          173000          ;M9301 - YC VERSION
699  001270' 173000                          173000          ;M9400 - YA(OR YC) VERSION
700  001272' 173000                          173000          ;M9301 - YF VERSION
701  001274' 170000                          170000          ;M7942 - YB VERSION
702  001276' 173000                          173000          ;M9301 - YD VERSION
703  001300' 173000                          173000          ;M9400 - YH(OR YK) VERSION
704  001302' 163000                          163000          ;M9311 VERSION
705  001304' 173000                          173000          ;M9301 - YH VERSION
706  001306' 173000                          173000          ;M9301 - YE VERSION
707  001310' 173000                          173000          ;M9301 - YJ VERSION
708  001312' 173000                          173000          ;M9301 - YN VERSION
709  001314' 177777                          -1
710  001316' 177777                          -1
711  001320' 177777                          -1
712  001322' 177777                          -1
713  001324' 177777                          -1
714
715  001326' 177777              TDLN2:      -1              ;TABLE OF THE LENGTH (BYTES) OF 2ND ROM ADDRESS SPACE
```



```
716  001330' 001000                          1000            ;M9301 - YA VERSION
717  001332' 001000                          1000            ;M9301 - YB VERSION
718  001334' 001000                          1000            ;M9301 - YC VERSION
719  001336' 001000                          1000            ;M9400 -YA(OR YC) VERSION
720  001340' 001000                          1000            ;M9301 -YF VERSION
721  001342' 000000                          0               ;M7942 - YB VERSION
722  001344' 001000                          1000            ;M9301 - YD VERSION
723  001346' 001000                          1000            ;M9400 - YH(OR YK) VERSION
724  001350' 001000                          1000            ;M9311 VERSION
725  001352' 000764                          764             ;M9301 - YH VERSION
726  001354' 001000                          1000            ;M9301 - YE VERSION
727  001356' 001000                          1000            ;M9301 - YJ VERSION
728  001360' 001000                          1000            ;M9400 - YN VERSION
729  001362' 177777                          -1
730  001364' 177777                          -1
731  001366' 177777                          -1
732  001370' 177777                          -1
733  001372' 177777                          -1
734
735  001374' 177777              TRMSA2:     -1              ;TABLE OF THE STARTING ADDRESS OF 2ND ROM ADDRESS SPACE
736  001376' 165000                          165000          ;M9301 - YA VERSION
737  001400' 165000                          165000          ;M9301 - YB VERSION
738  001402' 165000                          165000          ;M9301 - YC VERSION
739  001404' 165000                          165000          ;M9400 - YA(OR YC) VERSION
740  001406' 165000                          165000          ;M9301 - YF VERSION
741  001410' 000000                          0               ;M7942 - YB VERSION
742
743  001412' 165000                          165000          ;M9301 - YD VERSION
744  001414' 165000                          165000          ;M9400 - YH(OR YK) VERSION
745  001416' 166000                          166000          ;M9311 VERSION
746  001420' 165000                          165000          ;M9301 - YH VERSION
747  001422' 165000                          165000          ;M9301 - YE VERSION
748  001424' 165000                          165000          ;M9301 - YJ VERSION
749  001426' 165000                          165000          ;M9400 - YN VERSION
750  001430' 177777                          -1
751  001432' 177777                          -1
752  001434' 177777                          -1
753  001436' 177777                          -1
754  001440' 177777                          -1
755
756  001442' 001446'             ADR:        MES1
757  001444' 177777                          177777
758  001446' 051445  030522  047040  MES1:    .ASCIZ  /%SR1 NOT SET TO SELECT ROM*/
       001454' 052117  051440  052105
       001462' 052040  020117  042523
       001470' 042514  052103  051040
       001476' 046517  000047
759                                         .EVEN
760          000001                 .END
```

```
ACSR    000102R     CDATA$= 104412     HRDPAS  000050R     PRTY3 = 000140     STAT    000026R
ACTCRC  000242R     CH0     000404R     ICONT   000036R     PRTY4 = 000200     ST2     000272R
ACTLPC  000244R     CH1     000440R     ICOUNT  000040R     PRTY5 = 000240     SVR0    000062R
ADDR    000006R     CK1     000510R     IDNUM   000122R     PRTY6 = 000300     SVR1    000064R
ADDR22= 001000      CLLAST  000676R     INIT    000030R     PRTY7 = 000340     SVR2    000066R
ADR     001442R     CLP0    000744R     INTR    000120R     PS    = 177776     SVR3    000070R
ASB     000106R     CLP1    000756R     LPC     001036R     PS#   = 177776     SVR4    000072R
ASTAT   000104R     CLP2    001010R     LPC1    001044R     PUSH  = 005746     SVR5    000074R
AWAS    000110R     CL0     000574R     LPC2    001060R     PUSH2 = 024646     SVR6    000076R
BEGIN   000000R     CL1     000640R     MAP22$= 104416      RAND$ = 104417     SYSCNT  000052R
BIT0  = 000001      CL2     000662R     MES1    001446R     RANNUM  000054R    TDLN1   001212R
BIT1  = 000002      CL3     000614R     MODNAM  000000R     RESTRT  000252R    TDLN2   001326R
BIT10 = 002000      CONFIG  000056R     MODSP   000224R     RES1    000056R    TRMSA1  001260R
BIT11 = 004000      CPC     000566R     MSGN$ = 104403      RES2    000060R    TRMSA2  001374R
BIT12 = 010000      CSRA    000100R     MSGS$ = 104402      ROMSA1  000224R    TRPDFD= 000022
BIT13 = 020000      DATCK$= 104411      MSG$  = 104401      ROMSA2  000230R    TXCRC   001074R
BIT14 = 040000      DATER$= 104404      NULL  = 000000      RSTRT   000112R    TXLPC   001142R
BIT15 = 100000      DATLN1  000226R     OPEN  = 000000      R6    =%000006     TYPOUT  000250R
BIT2  = 000004      DATLN2  000232R     OTOA$ = 104420      R7    =%000007     VECTOR  000010R
BIT3  = 000010      DVID1   000014R     PAPCNT  000246R     SBADR   000102R    WASADR  000104R
BIT4  = 000020      ENDIT$= 104413      PARITY  000734R     SOFCNT  000042R    WDFR    000116R
BIT5  = 000040      END$  = 104410      PASCNT  000034R     SOFER$= 104406     WDTO    000114R
BIT6  = 000100      ERRTYP  000106R     PASS    000560R     SOFPAS  000046R    XFLAG   000005R
BIT7  = 000200      EXCRC   000236R     PIRQ$ = 000004      SPOINT  000032R    XOR     001012R
BIT8  = 000400      EXIT$ = 104400      POPSP = 005726      SPSIZ = 000040     XORS    000234R
BIT9  = 001000      EXLPC   000240R     POPSP2= 022626      SR1     000016R    .     = 001502R
BREAK$= 104407      GETPA$= 104415      PRTY  = 000000      SR2     000020R
BR1     000012R     GWBUF$= 104414      PRTY0 = 000000      SR3     000022R
BR2     000013R     HRDCNT  000044R     PRTY1 = 000040      SR4     000024R
BTOD$ = 104421      HRDER$= 104405      PRTY2 = 000100      START   000252R

. ABS.  000000  000
        001502  001
```

```
ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

PACK:XBMIA0,PACK:XBMIA0=DDXCOM,PACK:XBMIA0
RUN-TIME:  3  4  .4 SECONDS
RUN-TIME RATIO: 47/8=5.4
CORE USED:  7K  (13 PAGES)
```

.REM_

IDENTIFICATION

PRODUCT CODE: AC-F428A-MC

PRODUCT NAME: CXMNEA0 MACRO MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

THE MNE IS AN IOMOD THAT EXERCISES THE MNCDO DIGITAL OUTPUT. THE MODULE CONSISTS OF A READ-WRITE SECTION OF THE INTERNAL DATA PATH'S OF THE MNCDO LOGIC. UP TO 8 MNCDO'S CAN BE EXERCISED WITH THIS MODULE. THE "MNB" MODULE CAN BE ENABLED TO USE THE MNCDO TO WRAP-AROUND DATA INTO THE MNCDI, IF YOU HAVE SELECTED THAT OPTION, YOU SHOULD DESELECT "MNE" MODULE.

2.0 REQUIREMENTS

HARDWARE: ONE MNCDO (DIGITAL OUT).

STORAGE:: MNE REQUIRES:
DECIMAL WORDS: 407
OCTAL WORDS: 627
OCTAL BYTES: 1456

3.0 PASS DEFINITION

1000 OCTAL PASSES THRU THE LOGIC AND INTERRUPT TESTS.

4.0 EXECUTION TIME

ONE PASS OF THE MNE MODULE RUNNING ALONE TAKES APPROXIMATELY ONE MINUTE.

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADP: 171260, VECTOR 340, BR1: 4

DEVCNT: 1, SR1: N/A

REQUIRED PARAMETERS:

NONE.

6.0 DEVICE/OUTPUT SET-UP

NONE IF THE "MNB" MODULE IS NOT SELECTED TO RUN WRAP-AROUND MODE.
THIS MODULE MUST BE DESELECTED IF "MNB" IS IN WRAP-AROUND MODE
AND THE BUS ADDRESS AND VECTOR IS THE DEFAULT VALUE.

7.0 MODULE OPERATION

THE MODULE PERFORMES THE FOLLOWING TESTS:

FLOAT A 1 ACROSS THE DATA OUTPUT REGISTER
FLOAT A 0 ACROSS THE DATA OUTPUT REGISTER
VERIFY BYTE OPERATION OF THE DATA OUTPUT REGISTER
READ-WRITE TEST OF BIT 6 OF THE STAUTS REGISTER
READ-WRITE TEST OF BIT 4 OF THE STATUS REGISTER
READ-WRITE TEST OF BIT 3 OF THE STATUS REGISTER
OUTPUT DONE FLAG CAN SET
OUTPUT DONE FLAG CAN BE WRITTEN TO A ZERO
OUTPUT DONE FLAG CLEARS WHEN THE DATA OUTPUT REGISTER IS LOADED
OUTPUT DONE FLAG GENERATES AN INTERRUPT

8.0 OPERATION OPTIONS

LOCATION DVID1 CAN BE MODIFIED TO SELECT ADDITIONAL UNITS.

9.0 NON-STANDARD PRINTOUTS

ALL PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN THE DEC/X11
DOCUMENT.

-

```
                                        .LIST   ME
                                        .NLIST  MC,CND,MD
                                        .TITLE  MNEA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6      23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 047115  040505      040 MODNAM: .ASCII  /MNEA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF %BUFF USAGE
000006' 171260                  ADDR:   171260+0                ;1ST DEVICE ADDR.
000010' 000340                  VECTOR: 340+0                   ;1ST DEVICE VECTOR.
000012'     200                 BR1:    .BYTE   PRTY4+0         ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;*****************************************************************************
000026' 140000                  STAT:   140000                  ;STATUS WORD.
000030' 000244'                 INIT:   START                   ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                       ;PASS COUNTER.
000036' 001000                  ICONT:  1000                    ;# OF ITERATIONS PER PASS=1000
000040' 000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                         ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000102'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000104'                         WASADR:                         ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000106'                         ERRTYP:                         ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000112' 000244'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
000122' 000000                  IDNUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
000224'                         MODSP:
                                ;*****************************************************************************
```



```
000224' 000006'                 OCSR:   ADDR
000226' 000007'                 OCSR1:  ADDR+1                  ;HIGH BYTE ADDRESS

000230' 000010'                 DOR:    ADDR+2
000232' 000011'                 DOR1:   ADDR+3                  ;HIGH BYTE ADDRESS

000234' 000010'                 DODINV: VECTOR
000236' 000012'                 DODINS: VECTOR+2

000240' 000000                  TEMP:   0
000242' 000001                  TEMP1:  1
                  010000                BITDAT=BIT12            ;MAINT INPUT INHIBIT
                  004000                BITEXT=BIT11            ;MAINT INPUT STROBE
000244'                         RESTRT:
000244'                         START:
000244' 012767  000001  177770  CONT:   MOV     #BIT0,TEMP1     ;LOAD UNIT SELECT POINTER
000252' 012700  000224'                 MOV     #OCSR,R0        ;LOAD ADDRESS POINTER
000256' 016701  177524                  MOV     ADDR,R1         ;LOAD INITIAL BUS ADDRESS
000262' 010120                  1$:     MOV     R1,(R0)+        ;LOAD DEVICE ADDRESS
000264' 005201                          INC     R1              ;UPDATE BUS ADDRESS VALUE
000266' 020027  000234'                 CMP     R0,#DOR1+2      ;TEST IF DONE WITH BUS ADDRESSES
000272' 001373                          BNE     1$              ;BR IF NOT
000274' 016701  177510                  MOV     VECTOR,R1       ;LOAD VECTOR POINTER
000300' 010120                  3$:     MOV     R1,(R0)+        ;LOAD DEVICE VECTOR ADDRESS
000302' 005721                          TST     (R1)+           ;UPDATE BUS VECTOR VALUE
000304' 010110                          MOV     R1,(R0)         ;LOAD 2ND ADDR.

                                        ;VERIFY A MNCDO ADDRESS RESPONSE

000306' 016767  177712  177564  DO0:    MOV     OCSR,CSPA       ;LOAD BUS ADDRESS IF AN ERROR
000314' 005777  177704                  TST     @OCSR           ;TEST OUTPUT STATUS REGISTER
000320' 005777  177704                  TST     @DOR            ;TEST OUTPUT DATA REGISTER
```

```
000324'                         DO1:
                        ;FLOAT A 1 ACROSS THE MNCDO DATA REGISTER
000324' 012767  000001  177550  MOV     #BIT0,ACSR              ;LOAD EXPECT BIT
000332' 016777  177544  177670  1$:     MOV     ACSR,@DOR               ;LOAD MNCDO DATA REGISTER
000340' 017767  177664  177536          MOV     @DOP,ASTAT              ;READ MNCDO DATA REGISTER
000346' 026767  177530  177530          CMP     ACSR,ASTAT              ;COMPARE
000354' 001403                          BEQ     2$                      ;BR IF SAME
                        ;*****************************************************************
000356' 104405  000000' 000000          HRDER$,BEGIN,NULL       ; MNCDO DATA REGISTER FAILED TO HOLD A FLOATING 1
                        ;*****************************************************************
000364'                         2$:
000364' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000370' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000374' 006367  177502                  ASL     ACSR                    ;CHANGE THE DATA
000400' 001354                          BNE     1$                      ;BR IF MORE DATA
000402'                         DO2:
                        ;FLOAT A 0 ACROSS THE MNCDO DATA REGISTER
000402' 012767  000001  177630          MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
000410' 016767  177624  177464  1$:     MOV     TEMP,ACSR               ;LOAD EXPECTED
000416' 005167  177460                  COM     ACSR                    ;COMPLEMENT
000422' 016777  177454  177600          MOV     ACSR,@DOR               ;LOAD MNCDO DATA REGISTER
000430' 017767  177574  177446          MOV     @DOP,ASTAT              ;READ MNCDO DATA REGISTER
000436' 026767  177440  177440          CMP     ACSR,ASTAT              ;COMPARE
000444' 001403                          BEQ     2$                      ;BR IF SAME
                        ;*****************************************************************
000446' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;MNCDO DATA REGISTER FAILED TO HOLD A FLOATING 0
                        ;*****************************************************************
000454'                         2$:
000454' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000460' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000464' 006367  177550                  ASL     TEMP                    ;CHANGE THE DATA
000470' 001347                          BNE     1$                      ;BR IF MORE DATA
000472'                         DO3:
                        ;VERIFY BYTE OPERATION ON THE MNCDO DATA REGISTER
000472' 012777  177777  177530  1$:     MOV     #-1,@DOR                ;LOAD MNCDO DATA REGISTER
000500' 012767  000377  177374          MOV     #377,ACSR               ;LOAD EXPECTED
000506' 105077  177520                  CLRB    @DOR1                   ;CLEAR HIGH BYTE
000512' 017767  177512  177364          MOV     @DOR,ASTAT              ;READ MNCDO DATA REGISTER
000520' 026767  177356  177356          CMP     ACSR,ASTAT              ;COMPARE
000526' 001403                          BEQ     2$                      ;BR IF SAME
                        ;*****************************************************************
000530' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLEARING HIGH BYTE CHANGED LOW BYTE
                        ;*****************************************************************
000536' 012777  177777  177464  2$:     MOV     #-1,@DOR                ;LOAD MNCDO DATA REGISTER
000544' 012767  177400  177330          MOV     #177400,ACSR            ;LOAD EXPECTED
000552' 105077  177452                  CLRB    @DOR                    ;CLEAR LOW BYTE
000556' 017767  177446  177320          MOV     @DOR,ASTAT              ;READ MNCDO DATA REGISTER
000564' 026767  177312  177312          CMP     ACSR,ASTAT              ;COMPARE
000572' 001403                          BEQ     3$                      ;BR IF SAME
                        ;*****************************************************************
000574' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLEARING LOW BYTE CHANGED HIGH BYTE
                        ;*****************************************************************
000602'                         3$:
000602' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000606' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000612'                         DO4:
```



```
                        ;TEST THAT BIT6 OF MNCDO STATUS REGISTER IS READ-WRITE
000612' 012767  000100  177262          MOV     #BIT6,ACSP              ;LOAD EXPECTED
000620' 016777  177256  177376          MOV     ACSR,@OCSP              ;LOAD BIT6 INTO MNCDO STATUS REGISTER
000626' 017767  177372  177250          MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER
000634' 026767  177242  177242          CMP     ACSR,ASTAT              ;TEST THAT IT SET
000642' 001403                          BEQ     1$                      ;BR IF SAME
                        ;*****************************************************************
000644' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT6 OF MNCDO STATUS REGISTER FAILED TO SET
                        ;*****************************************************************
000652' 046777  177224  177344  1$:     BIC     ACSR,@OCSR              ;CLEAR THAT BIT
000660' 017767  177340  177216          MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER AGAIN
000666' 026767  177210  177210          CMP     ACSR,ASTAT              ;TEST THE BIT
000674' 001003                          BNE     2$                      ;BR IF CLEARED
                        ;*****************************************************************
000676' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT6 OF MNCDO STATUS REGISTER FAILED TO CLEAR
                        ;*****************************************************************
000704'                         2$:
000704' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000710' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000714'                         DO5:
                        ;TEST THAT BIT4 OF MNCDO STATUS REGISTER IS READ-WRITE
000714' 012767  000020  177160          MOV     #BIT4,ACSR              ;LOAD EXPECTED
000722' 016777  177154  177274          MOV     ACSP,@OCSR              ;LOAD BIT4 INTO MNCDO STATUS REGISTER
000730' 017767  177270  177146          MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER
000736' 026767  177140  177140          CMP     ACSR,ASTAT              ;TEST THAT IT SET
000744' 001403                          BEQ     1$                      ;BR IF SAME
                        ;*****************************************************************
000746' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT4 OF MNCDO STATUS REGISTER FAILED TO SET
                        ;*****************************************************************
000754' 046777  177122  177242  1$:     BIC     ACSR,@OCSR              ;CLEAR THAT BIT
000762' 017767  177236  177114          MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER AGAIN
000770' 026767  177106  177106          CMP     ACSR,ASTAT              ;TEST THE BIT
000776' 001003                          BNE     2$                      ;BR IF CLEARED
                        ;*****************************************************************
001000' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT4 OF MNCDO STATUS REGISTER FAILED TO CLEAR
                        ;*****************************************************************
001006'                         2$:
001006' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001012' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```

```
001016'                         DO6:
                                        ;TEST THAT BIT3 OF MNCDO STATUS REGISTER IS READ-WRITE
001016' 012767  000010  177056          MOV     #BIT3,ACSR              ;LOAD EXPECTED
001024' 016777  177052  177172          MOV     ACSR,@OCSR              ;LOAD BIT3 INTO MNCDO STATUS REGISTER
001032' 017767  177166  177044          MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER
001040' 026767  177036  177036          CMP     ACSR,ASTAT              ;TEST THAT IT SET
001046' 001403                          BEQ     1$                      ;BR IF SAME
                                        ;*****************************************************************
001050' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT3 OF MNCDO STATUS REGISTER FAILED TO SET
                                        ;*****************************************************************
001056' 046777  177020  177140  1$:     BIC     ACSR,@OCSR              ;CLEAR THAT BIT
001064' 017767  177134  177012          MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER AGAIN
001072' 026767  177004  177004          CMP     ACSR,ASTAT              ;TEST THE BIT
001100' 001003                          BNE     2$                      ;BR IF CLEARED
                                        ;*****************************************************************
001102' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT3 OF MNCDO STATUS REGISTER FAILED TO CLEAR
                                        ;*****************************************************************
001110'                         2$:
001110' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001114' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.

                                        ;VERIFY THAT MNCDO DONE FLAG SETS
001120' 005077  177100          DO7:    CLR     @OCSR                   ;CLEAR CLEARED FLAG
001124' 012767  000200  176750          MOV     #BIT7,ACSR              ;LOAD EXPECTED
001132' 105077  177072                  CLRB    @DOR                    ;ENABLE
001136' 112777  000001  177062          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
001144' 017767  177054  176732          MOV     @OCSR,ASTAT             ;READ OUTPUT STATUS REGISTER
001152' 026767  176724  176724          CMP     ACSR,ASTAT              ;COMMPARE
001160' 001403                          BEQ     DO10                    ;BR IF SAME
                                        ;*****************************************************************
001162' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;OUTPUT DONE FLAG FAILED TO SET
                                        ;*****************************************************************
                                        ;VERIFY THAT MNCDO DONE FLAG CLEARS WHEN WRITTEN TO A 0
001170' 105077  177034          DO10:   CLRB    @DOR                    ;ENABLE
001174' 112777  000001  177024          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
001202' 005067  176674                  CLR     ACSR                    ;CLEAR EXPECTED
001206' 005077  177012                  CLR     @OCSR                   ;CLEAR OUTPUT DONE FLAG
001212' 017767  177006  176664          MOV     @OCSR,ASTAT             ;READ STATUS
001220' 001403                          BEQ     DO11                    ;BR IF SAME
                                        ;*****************************************************************
001222' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;WRITING OUTPUT FLAG TO A ZERO FAILED TO CLEAR OUTPUT DO
                                        ;*****************************************************************


                                        ;VERIFY THAT MNCDO DONE FLAG CLEARS WHEN OUTPUT DATA REGISTER IS WRITTEN
001230' 105077  176774          DO11:   CLRB    @DOR                    ;ENABLE
001234' 112777  000001  176764          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
001242' 005067  176634                  CLR     ACSR                    ;CLEAR EXPECTED
001246' 105077  176756                  CLRB    @DOR                    ;WRITE THE OUTPUT DATA REGISTER
001252' 017767  176746  176624          MOV     @OCSR,ASTAT             ;READ OUTPUT STATUS REGISTER
001260' 001403                          BEQ     DO12                    ;BR IF CLEARED
                                        ;*****************************************************************
001262' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;OUTPUT DONE FLAG FAILED TO CLEAR
                                        ;*****************************************************************
                                        ;WHEN OUTPUT DATA REGISTER WAS WRITTEN
```



```
                                        ;INTERRUPT TEST -- VERIFY MNCDO DOES INTERRUPT
001270'                         DO12:
001270' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001274' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001300' 012777  001346' 176726          MOV     #1$,@DODINV             ;LOAD RETURN VECTOR
001306' 116777  176500  176722          MOVB    BR1,@DODINS             ;LOAD RETURN LEVEL
001314' 105077  176710                  CLRB    @DOR                    ;ENABLE
001320' 112777  000001  176700          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
001326' 052777  000100  176670          BIS     #BIT6,@OCSR             ;ENABLE INTR.
001334' 000240                          NOP
001336' 000240                          NOP
001340' 000240                          NOP
001342' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.

001346' 005077  176652          1$:     CLR     @OCSR
                                        ;---------------------------------------------------------------
001352' 000004  000000' 001360'         PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3$ AND RTI
                                        ;---------------------------------------------------------------
001360' 016777  176652  176646  3$:     MOV     DODINS,@DODINV          ;RESET VECTORS
001366' 005077  176644                  CLR     @DODINS
001372' 005077  176626                  CLR     @OCSR


                                ;DETERMINE IF MORE SEQUENTIAL MINC-DO'S ARE TO BE TESTED
001376' 006367  176640          FINISH: ASL     TEMP1                   ;MOVE LEFT
001402' 022767  000400  176632          CMP     #BIT8,TEMP1             ;TEST IF LAST VALID UNIT
001410' 001420                          BEQ     3$                      ;BR IF YES
001412' 036767  176624  176374          BIT     TEMP1,DVID1             ;TEST IF SELECTED
001420' 001003                          BNE     1$
001422' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                ;MONITOR SHALL TEST END OF PASS
001426' 000411                          BR      3$
001430' 012700  000224'         1$:     MOV     #OCSR,R0                ;GET ADDRESS POINTER
001434' 062720  000004          2$:     ADD     #4,(R0)+                ;UPDATE CONTNETS
001440' 020027  000240'                 CMP     R0,#TEMP                ;TEST IF DONE ALL LOC.
001444' 001373                          BNE     2$
001446' 000167  176634                  JMP     DO0                     ;RUN NEXT UNIT
001452' 000167  176566          3$:     JMP     CONT

        000001                          .END
```

```
ACSR    000102R          184#  232*  233   235   243*  248*  249*  250   252   265*  268   274*  277
                         287*  288   290   295   297   307*  308   310   315   317   327*  328   330
                         335   337   348*  352   360*  371*
ADDR    000006R          150*  198   199   201   202   215
ADDR22= 001000           197#
ASB     000106R          188#
ASTAT   000104R          186#  234*  235   251*  252   267*  268   276*  277   289*  290   296*  297
                         309*  310   316*  317   329*  330   336*  337   351*  352   362*  373*
AWAS    000110R          189#
BEGIN   000000R          147#  238   241   242   255   258   259   271   280   283   284   293   300
                         303   304   313   320   323   324   333   340   343   344   355   365   376
                         381   382   391   395   407
BITDAT= 010000           209#
BITEXT= 004000           210#
BIT0  = 000001           197#  213   232   247   350   359   370   386
BIT1  = 000002           197#
BIT10 = 002000           197#
BIT11 = 004000           197#  210
BIT12 = 010000           197#  209
BIT13 = 020000           197#
BIT14 = 040000           197#
BIT15 = 100000           197#
BIT2  = 000004           197#
BIT3  = 000010           197#  327
BIT4  = 000020           197#  307
BIT5  = 000040           197#
BIT6  = 000100           197#  287   387
BIT7  = 000200           197#  348
BIT8  = 000400           197#  403
BIT9  = 001000           197#
BREAK$= 104407           197#  241   242   258   259   283   284   303   304   323   324   343   344
                         381   382
BR1     000012R          152#  384
BR2     000013R          153#
BTOD$ = 104421           197#
CDATA$= 104412           197#
CONFIG  000056R          172#
CONT    000244R          213#  415
CSRA    000100R          182#  227*
DATCK$= 104411           197#
DATER$= 104404           197#
DODIN$  000236R          205#  384*  397   398*
DODINV  000234R          204#  383*  397*
DOR     000230R          201#  229   233*  234   250*  251   264*  267   273*  275*  276   349*  358*
                         369*  372*  385*
DOR1    000232R          202#  218   266*
DO0     000306R          227#  414
DO1     000324R          230#
DO10    001170R          353   358#
DO11    001230R          363   369#
DO12    001270R          374   380#
DO2     000402R          245#
DO3     000472R          262#
DO4     000612R          285#
DO5     000714R          305#
DO6     001016R          325#
```



```
DO7     001120R          347#
DVID1   000014R          154#  405
ENDIT$= 104413           197#  407
END$  = 104410           197#
ERRTYP  000106R          187#
EXIT$ = 104400           197#  391
FINISH  001376R          402#
GETPA$= 104415           197#
GWBUF$= 104414           197#
HRDCNT  000044R          167#
HRDER$= 104405           197#  238   255   271   280   293   300   313   320   333   340   355   365
                         376
HRDPAS  000050R          169#
ICONT   000036R          164#
ICOUNT  000040R          165#
IDNUM   000122R          194#
INIT    000030R          161#
INTR    000120R          193#
MAP22$= 104416           197#
MODNAM  000000R          148#
MODSP   000224R          162   195#
MSGN$ = 104403           197#
MSGS$ = 104402           197#
MSG$  = 104401           197#
NULL  = 000000           197#  238   255   271   280   293   300   313   320   333   340   355   365
                         376
OCSR    000224R          198#  214   227   228   288*  289   295*  296   308*  309   315*  316   328*
                         329   335*  336   347*  351   361*  362   373   387*  393*  399*  410
OCSR1   000226R          199#  350*  359*  370*  386*
OPEN  = 000000           149   155   156   157   158   175   176   177   178   179   180   181   182
                         184   186   188   189   191   192   193   197*
OTOA$ = 104420           197#
PASCNT  000034R          163#
PIRQ$ = 000004           197#  395
POPSP = 005726           197#
POPSP2= 022626           197#
PRTY  = 000000           197#
PRTY0 = 000000           153   197#
PRTY1 = 000040           197#
PRTY2 = 000100           197#
PRTY3 = 000140           197#
PRTY4 = 000200           152   197#
PRTY5 = 000240           197#
PRTY6 = 000300           197#
PRTY7 = 000340           197#
PS    = 177776           197#
PSW   = 177776           197#
PUSH  = 005746           197#
PUSH2 = 024646           197#
RAND$ = 104417           197#
RANNUM  000054R          171#
RESTRT  000244R          190   211#
RES1    000056R          173#
RES2    000060R          174#
RSTRT   000112R          190#
$BADR   000102R          183#
```

```
SOFCNT  000042R          166#
SOFERS= 104406           197#
SOFPAS  000046R          168#
SPOINT  000032R          162#
SPSIZ = 000040             1#    195
SR1     000016R          155#
SR2     000020R          156#
SR3     000022R          157#
SR4     000024R          158#
START   000244R          161     212#
STAT    000026R          160#
SVR0    000062R          175#
SVR1    000064R          176#
SVR2    000066R          177#
SVR3    000070R          178#
SVR4    000072R          179#
SVR5    000074R          180#
SVR6    000076R          181#
SYSCNT  000052R          170#
TEMP    000240R          207#    247#    248     260#    412
TEMP1   000242R          208#    213#    402#    403     405
TRPDFD= 000022           197#
VECTOR  000010R          151#    204     205     220
WASADR  000104R          185#
WDFR    000116R          192#
WDTO    000114R          191#
XFLAG   000005R          149#

. ABS.  000000      000
        001456      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XMNEA0,XMNEA0/SOL/CRF:SYM=DDXCOM,XMNEA0
 RUN-TIME:  1  1  .2 SECONDS
 RUN-TIME RATIO: 64/3=16.3
 CORE USED:  7K  (13 PAGES)
```

.REM _

IDENTIFICATION

PRODUCT CODE: AC-F425A-MC

PRODUCT NAME: CXMNDA0 MNCDA (D/A) MOD

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1. ABSTRACT:

"MND" IS A BKMOD THAT EXEPCISES THE MNCDA
DIGITAL TO ANALOG CONVERTER. A CONFIDENCE LOGIC TEST IS
EXECUTED ON THE DAC0, DAC1, DAC2 AND DAC3 REGISTERS.
ALL LOGIC ERRORS ARE REPORTED TO THE CONSOLE TELETYPE.
"MND" WILL TEST MULTIPLE UNITS AS SELECTED BY DEVCNT/DVID1.

2. REQUIREMENTS:

HARDWARE: ONE MNCDA (D/A)

STORAGE:: MND REQUIRES:
DECIMAL WORDS: 399
OCTAL WORDS: 617
OCTAL BYTES: 1436

3. PASS DEFINITION:

ONE PASS OF THE MND MODULE CONSISTS OF FLOATING A 1 AND A 0
ACCROSS THE FOUR D TO A REGISTERS 3000(8) TIMES.
THIS RESULTS IN 340,000 BUS REFERENCES TO THE MNCDA OPTION

4. EXECUTION TIME:

VARIES WITH THE NUMBER OF OTHER DEVICES BEING RUN.
THIS SHOULD TAKE AN AVERAGE OF THIRTY SECONDS TO COMPLETE ONE
PASS WHEN RUNNING ALONE.

5. CONFIGURATION PARAMETERS:

DEFAULT PARAMETERS:

DVA: 171060, VCT: N/A, BR1: N/A, DEVCNT: 1, SR1: N/A

REQUIRED PARAMETERS:

NONE

6. DEVICE OPTION SETUP:

NONE.

7. MODULE OPERATION:

START/RESTART:

THIS CODE WILL USE THE VALUE CONTAINED IN LOCATION "ADDR" TO BE THE BASE ADDRESS OF THE MNCDA. THE BUS ADDRESS OF EACH DAC IS PRIMED IN THIS ROUTINE. THE INITIAL PASS COUNTER IS ALSO PRESET.

TSDAC0:

THE ABILITY OF DAC 0 REGISTER TO HOLD A FLOATING 1 PATTERN IS VERIFIED IN THIS CODE. BIT 11 OF THE REGISTER IS INITILLY SET (4000) AND THEN ROTATED TO THE RIGHT. UPON COMPLETION, THE SAME PROCEDURE IS REPEATED EXCEPT THE INITIAL VALUE IS 3777.

TSDAC1:, TSDAC2:, TSDAC3:

SAME AS TSDAC0

DUAL:

THIS ROUTINE WILL LOAD DIFFERENT DATA INTO EACH REGISTER AND VERIFY INDEPENDANT ADDRESS SELECTION.

DONE:

IN THIS ROUTINE, THE LOCATION "PASSX" IS DECREMENTED TO DETERMINE IF THE MODULE HAS BEEN EXERCISED. IF NOT THE PROGRAM WILL LOOP TO LOCATION "LOOPA" AND REPEAT THE SEQUENCE. WHEN THE PASS COUNT HAS BEEN COMPLETED, THE "END OF PASS" IS REPORTED.

8. OPERATOR OPTIONS:

LOCATION (PASSCT) CAN BE MODIFIED TO VARY THE NUMBER OF LOOPS THRU TEST BEFORE END OF PASS IS REPORTED (TO ACCOMODATE SYSTEM CONFIGURATION).

9. NON-STANDARD PRINTOUTS:

NONE: ALL PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN THE DEC/X11 DOCUMENT

```
                        RKMOD   <MNDA >,171060,0,0,0,0,3000,0
                        MODULE  40020,MNDA ,171060,0,0,0,0,3000,0
                        .TITLE  MNDA DEC/X11 SYSTEM EXERCISER MODULE
;                       DDXCOM  VERSION  6      23-MAY-78
                        .LIST   BIN
                        ;*****************************************************************
BEGIN:
000000' 047115  040504      040 MODNAM: .ASCII  /MNDA / ;MODULE NAME.
000005'     000 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 171060 ADDR:    171060+0                ;1ST DEVICE ADDR.
000010' 000000 VECTOR:  0+0                     ;1ST DEVICE VECTOR.
000012'     000 BR1:    .BYTE   PRTY0+0         ;1ST BR LEVEL.
000013'     000 BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000001 DVID1:   0+1                     ;DEVICE INDICATOR 1.
000016' 000000 SR1:     OPEN                    ;SWITCH REGISTER 1
000020' 000000 SR2:     OPEN                    ;SWITCH REGISTER 2
000022' 000000 SR3:     OPEN                    ;SWITCH REGISTER 3
000024' 000000 SR4:     OPEN                    ;SWITCH REGISTER 4
                        ;*****************************************************************
000026' 040020 STAT:    40020                   ;STATUS WORD.
000030' 000236' INIT:   START                   ;MODULE START ADDR.
000032' 000224' SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000 PASCNT:  0                       ;PASS COUNTER.
000036' 003000 ICONT:   3000                    ;# OF ITERATIONS PER PASS=3000
000040' 000000 ICOUNT:  0                       ;LOC TO COUNT ITERATIONS
000042' 000000 SOFCNT:  0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000 HRDCNT:  0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000 SOFPAS:  0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000 HRDPAS:  0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000 SYSCNT:  0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000 RANNUM:  0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056' CONFIG:                                 ;RESERVED FOR MONITOR USE
000056' 000000 RES1:    0                       ;RESERVED FOR MONITOR USE
000060' 000000 RES2:    0                       ;RESERVED FOR MONITOR USE
000062' 000000 SVR0:    OPEN                    ;LOC TO SAVE R0.
000064' 000000 SVR1:    OPEN                    ;LOC TO SAVE R1.
000066' 000000 SVR2:    OPEN                    ;LOC TO SAVE R2.
000070' 000000 SVR3:    OPEN                    ;LOC TO SAVE R3.
000072' 000000 SVR4:    OPEN                    ;LOC TO SAVE R4.
000074' 000000 SVP5:    OPEN                    ;LOC TO SAVE P5.
000076' 000000 SVR6:    OPEN                    ;LOC TO SAVE R6.
000100' 000000 CSRA:    OPEN                    ;ADDR OF CURRENT CSR.
000102' SRADR:                                  ;ADDR OF GOOD DATA, OR
000102' 000000 ACSR:    OPEN                    ;CONTENTS OF CSR.
000104' WASADR:                                 ;ADDR OF BAD DATA, OR
000104' 000000 ASTAT:   OPEN                    ;STATUS REG CONTENTS.
000106' ERRTYP:                                 ;TYPE OF ERROR
000106' 000000 ASB:     OPEN                    ;EXPECTED DATA.
000110' 000000 AWAS:    OPEN                    ;ACTUAL DATA.
000112' 000236' RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000 WDTO:    OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000 WDFR:    OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000 INTP:    OPEN                    ;# OF INTERRUPTS PER ITERATION
000122' 000000 IDNUM:   0                       ;MODULE IDENTIFICATION NUMBER=0
            000040          .REPT   SPSIZ       ;MODULE STACK STARTS HERE.
                            .NLIST
```



```
                            .WORD   0
                            .LIST
                            .ENDR
000224' MODSP:
                        ;*****************************************************************


                        ;DEVICE BUS ADDRESS

000224' 000006' DAC0:   ADDR                    ;BUS ADDRESS OF DAC 0
000226' 000010' DAC1:   ADDR+2                  ;BUS ADDRESS FOR DAC 1
000230' 000012' DAC2:   ADDR+4                  ;               2
000232' 000014' DAC3:   ADDR+6                  ;               3
000234' 000001 TEMP:    BIT0
                        ;INITILIZATION CODE
                        START:
000236' 012767  000001  177770 RESTRT: MOV      #BIT0,TEMP      ;LOAD UNIT POINTER
000244' 016767  177536  177752 LOOPA:  MOV      ADDR,DAC0       ;LOAD BUS ADDRESS
000252' 016767  177746  177746         MOV      DAC0,DAC1       ; FOR
000260' 062767  000002  177740         ADD      #2,DAC1
000266' 016767  177734  177734         MOV      DAC1,DAC2       ;  DIFFERENT
000274' 062767  000002  177726         ADD      #2,DAC2
000302' 016767  177722  177722         MOV      DAC2,DAC3       ;   DAC
000310' 062767  000002  177714         ADD      #2,DAC3         ;BUS ADDRESSES
000316' 012767  010000  177556 TSDAC0: MOV      #BIT12,ACSR     ;LOAD EXPECTED
000324' 012777  004000  177672         MOV      #BIT11,@DAC0    ;LOAD DAC0 REGISTER
000332' 016767  177666  177540         MOV      DAC0,CSRA       ;LOAD BUS ADDRESS
000340' 006267  177536         1$:     ASR      ACSR            ;SHIFT THE EXPECTED
000344' 001043                         BNE      4$              ;BR IF DONE
000346' 017767  177652  177530         MOV      @DAC0,ASTAT     ;READ THE REGISTER
000354' 026767  177522  177522         CMP      ACSR,ASTAT      ;COMPARE
000362' 001403                         BEQ      2$              ;BR IF SAME
                        ;*****************************************************************
000364' 104405  000000' 000000         HRDER$,BEGIN,NULL       ; DAC0 FAILED TO HOLD THE FLOATING 1 PATTERN
                        ;*****************************************************************
000372' 005167  177504         2$:     COM      ACSR           ;COMPLEMENT DATA
000376' 005177  177622                 COM      @DAC0          ;COMPLEMENT DATA IN DAC0
000402' 042767  170000  177472         BIC      #170000,ACSR   ;MASK OFF UNUSED BITS
000410' 017767  177610  177466         MOV      @DAC0,ASTAT    ;READ DAC0
000416' 026767  177460  177460         CMP      ACSR,ASTAT     ;COMPARE
000424' 001403                         BEQ      3$             ;BR IF SAME
                        ;*****************************************************************
000426' 104405  000000' 000000         HRDER$,BEGIN,NULL       ;DAC0 FAILED TO HOLD THE FLOATING 0 PATTERN
                        ;*****************************************************************
000434' 005167  177442         3$:     COM      ACSR           ;COMPLEMENT EXPECTED
000440' 005177  177560                 COM      @DAC0          ;COMPLEMENT DATA
000444' 042767  170000  177430         BIC      #170000,ACSR   ;MASK EXPECTED DATA
000452' 000732                         BR       1$             ;TEST MORE BITS

000454'                        4$:
```

```
000454' 012767 010000 177420 TSDAC1: MOV     #BIT12,ACSR             ;LOAD EXPECTED
000462' 012777 004000 177536         MOV     #BIT11,@DAC1            ;LOAD DAC1 REGISTER
000470' 016767 177532 177402         MOV     DAC1,CSRA               ;LOAD BUS ADDRESS
000476' 006267 177400        1$:     ASR     ACSR                    ;SHIFT THE EXPECIED
000502' 001043                       BNE     4$                      ;BR IF DONE
000504' 017767 177516 177372         MOV     @DAC1,ASTAT             ;READ THE REGISTER
000512' 026767 177364 177364         CMP     ACSR,ASTAT              ;COMPARE
000520' 001403                       BEQ     2$                      ;BR IF SAME
                                     ;*************************************************************
000522' 104405 000000' 000000        HRDER$,BEGIN,NULL       ; DAC1 FAILED TO HOLD THE FLOATING 1 PATTERN
                                     ;*************************************************************
000530' 005167 177346        2$:     COM     ACSR                    ;COMPLEMENT DATA
000534' 005177 177466                COM     @DAC1                   ;COMPLEMENT DATA IN DAC1
000540' 042767 170000 177334         BIC     #170000,ACSR            ;MASK OFF UNUSED BITS
000546' 017767 177454 177330         MOV     @DAC1,ASTAT             ;READ DAC1
000554' 026767 177322 177322         CMP     ACSR,ASTAT              ;COMPARE
000562' 001403                       BEQ     3$                      ;BR IF SAME
                                     ;*************************************************************
000564' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;DAC1 FAILED TO HOLD THE FLOATING 0 PATTERN
                                     ;*************************************************************
000572' 005167 177304        3$:     COM     ACSR                    ;COMPLEMENT EXPECTED
000576' 005177 177424                COM     @DAC1                   ;COMPLEMENT DATA
000602' 042767 170000 177272         BIC     #170000,ACSR            ;MASK EXPECTED DATA
000610' 000732                       BR      1$                      ;TEST MORE BITS

000612'                      4$:
000612' 012767 010000 177262 TSDAC2: MOV     #BIT12,ACSR             ;LOAD EXPECTED
000620' 012777 004000 177402         MOV     #BIT11,@DAC2            ;LOAD DAC2 REGISTER
000626' 016767 177376 177244         MOV     DAC2,CSRA               ;LOAD BUS ADDRESS
000634' 006267 177242        1$:     ASR     ACSR                    ;SHIFT THE EXPECTED
000640' 001043                       BNE     4$                      ;BR IF DONE
000642' 017767 177362 177234         MOV     @DAC2,ASTAT             ;READ THE REGISTER
000650' 026767 177226 177226         CMP     ACSR,ASTAT              ;COMPARE
000656' 001403                       BEQ     2$                      ;BR IF SAME
                                     ;*************************************************************
000660' 104405 000000' 000000        HRDER$,BEGIN,NULL       ; DAC2 FAILED TO HOLD THE FLOATING 1 PATTERN
                                     ;*************************************************************
000666' 005167 177210        2$:     COM     ACSR                    ;COMPLEMENT DATA
000672' 005177 177332                COM     @DAC2                   ;COMPLEMENT DATA IN DAC2
000676' 042767 170000 177176         BIC     #170000,ACSR            ;MASK OFF UNUSED BITS
000704' 017767 177320 177172         MOV     @DAC2,ASTAT             ;READ DAC2
000712' 026767 177164 177164         CMP     ACSR,ASTAT              ;COMPARE
000720' 001403                       BEQ     3$                      ;BR IF SAME
                                     ;*************************************************************
000722' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;DAC2 FAILED TO HOLD THE FLOATING 0 PATTERN
                                     ;*************************************************************
000730' 005167 177146        3$:     COM     ACSR                    ;COMPLEMENT EXPECTED
000734' 005177 177270                COM     @DAC2                   ;COMPLEMENT DATA
000740' 042767 170000 177134         BIC     #170000,ACSR            ;MASK EXPECTED DATA
000746' 000732                       BR      1$                      ;TEST MORE BITS

000750'                      4$:
```



```
000750' 012767 010000 177124 TSDAC3: MOV     #BIT12,ACSR             ;LOAD EXPECTED
000756' 012777 004000 177246         MOV     #BIT11,@DAC3            ;LOAD DAC3 REGISTER
000764' 016767 177242 177106         MOV     DAC3,CSRA               ;LOAD BUS ADDRESS
000772' 006267 177104        1$:     ASR     ACSR                    ;SHIFT THE EXPECTED
000776' 001043                       BNE     4$                      ;BR IF DONE
001000' 017767 177226 177076         MOV     @DAC3,ASTAT             ;READ THE REGISTER
001006' 026767 177070 177070         CMP     ACSR,ASTAT              ;COMPARE
001014' 001403                       BEQ     2$                      ;BR IF SAME
                                     ;*************************************************************
001016' 104405 000000' 000000        HRDER$,BEGIN,NULL       ; DAC3 FAILED TO HOLD THE FLOATING 1 PATTERN
                                     ;*************************************************************
001024' 005167 177052        2$:     COM     ACSR                    ;COMPLEMENT DATA
001030' 005177 177176                COM     @DAC3                   ;COMPLEMENT DATA IN DAC3
001034' 042767 170000 177040         BIC     #170000,ACSR            ;MASK OFF UNUSED BITS
001042' 017767 177164 177034         MOV     @DAC3,ASTAT             ;READ DAC3
001050' 026767 177026 177026         CMP     ACSR,ASTAT              ;COMPARE
001056' 001403                       BEQ     3$                      ;BR IF SAME
                                     ;*************************************************************
001060' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;DAC3 FAILED TO HOLD THE FLOATING 0 PATTERN
                                     ;*************************************************************
001066' 005167 177010        3$:     COM     ACSR                    ;COMPLEMENT EXPECTED
001072' 005177 177134                COM     @DAC3                   ;COMPLEMENT DATA
001076' 042767 170000 176776         BIC     #170000,ACSR            ;MASK EXPECTED DATA
001104' 000732                       BR      1$                      ;TEST MORE BITS

001106'                      4$:

                                     ;TEST FOR DUAL ADDRESSING

001106' 012777 000000 177110 DUAL:   MOV     #0,@DAC0                ;LOAD DAC 0
001114' 012777 002525 177104         MOV     #2525,@DAC1             ;LOAD DAC 1
001122' 012777 005252 177100         MOV     #5252,@DAC2             ;LOAD DAC 2
001130' 012777 007777 177074         MOV     #7777,@DAC3             ;LOAD DAC 3

001136' 016767 177062 176734         MOV     DAC0,CSRA               ;LOAD DAC 0 BUS ADDRESS
001144' 017767 177054 176732         MOV     @DAC0,ASTAT             ;READ DAC 0
001152' 012767 000000 176722         MOV     #0,ACSR                 ;LOAD EXPECTED
001160' 026767 176716 176716         CMP     ACSR,ASTAT              ;COMPARE
001166' 001403                       BEQ     1$                      ;BR IF SAME
                                     ;*************************************************************
001170' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;DUAL ADDRESS ERROR ON DAC 0
                                     ;*************************************************************

001176' 016767 177024 176674 1$:     MOV     DAC1,CSRA               ;LOAD DAC 1 BUS ADDRESS
001204' 017767 177016 176672         MOV     @DAC1,ASTAT             ;READ DAC 1
001212' 012767 002525 176662         MOV     #2525,ACSR              ;LOAD EXPECTED
001220' 026767 176656 176656         CMP     ACSR,ASTAT              ;COMPARE
001226' 001403                       BEQ     2$                      ;BR IF SAME
                                     ;*************************************************************
001230' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;DUAL ADDRESS ERROR ON DAC 1
                                     ;*************************************************************
```

```
001236' 016767  176766  176634  28$:    MOV     DAC2,CSRA               ;LOAD DAC 2 BUS ADDRESS
001244' 017767  176760  176632            MOV     @DAC2,ASTAT             ;READ DAC 2
001252' 012767  005252  176622            MOV     #5252,ACSR              ;LOAD EXPECTED
001260' 026767  176616  176616            CMP     ACSR,ASTAT              ;COMPARE
001266' 001403                            BEQ     3$                      ;BR IF SAME

                                          ;*************************************************************
001270' 104405  000000' 000000            HRDER$,BEGIN,NULL       ;DUAL ADDRESS ERROR ON DAC 2
                                          ;*************************************************************


001276' 016767  176730  176574  38$:     MOV     DAC3,CSRA               ;LOAD DAC 3 BUS ADDRESS
001304' 017767  176722  176572            MOV     @DAC3,ASTAT             ;READ DAC 3
001312' 012767  007777  176562            MOV     #7777,ACSR              ;LOAD EXPECTED
001320' 026767  176556  176556            CMP     ACSR,ASTAT              ;COMPARE
001326' 001403                            BEQ     4$                      ;BR IF SAME

                                          ;*************************************************************
001330' 104405  000000' 000000            HRDER$,BEGIN,NULL       ;DUAL ADDRESS ERROR ON DAC 3
                                          ;*************************************************************


001336'                         48:

001336' 006367  176672          DONE:   ASL     TEMP            ;CHANGE UNIT POINTER
001342' 022767  000400  176664          CMP     #BIT8,TEMP      ;TEST IF LAST UNIT
001350' 001404                          BEQ     3$              ;BR IF YES
001352' 036767  176656  176434          BIT     TEMP,DVID1      ;TEST IF UNIT IS SELECTED
001360' 001006                          BNE     1$              ;BR IF UNIT IS TO BE TESTED
001362' 012767  000001  176644  3$:     MOV     #BIT0,TEMP      ;RELOAD UNIT POINTER
001370' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION,
                                                                ;MONITOR SHALL TEST END OF PASS
001374' 000416                          BR      2$              ;BR IF NOT DONE
001376' 062767  000010  176620  1$:     ADD     #10,DAC0        ;UPDATE ADDRESS
001404' 062767  000010  176614          ADD     #10,DAC1
001412' 062767  000010  176610          ADD     #10,DAC2
001420' 062767  000010  176604          ADD     #10,DAC3
001426' 000167  176664                  JMP     TSDAC0
001432' 000167  176606          2$:     JMP     LOOPA

                000001                  .END
```



```
ACSR    000102R   194*  231*  234*  237   242*  244*  246   251*  253*  257*  260*  263   268*
                  270*  272   277*  279*  283*  286*  289   294*  296*  298   303*  305*  309*
                  312*  315   320*  322*  324   329*  331*  345*  346   354*  355   363*  364
                  372*  373
ADDR    000006R   166*  216   217   218   219   224
ADDR22= 001000    212*
ASB     000106R   198*
ASTAT   000104R   196*  236*  237   245*  246   262*  263   271*  272   288*  289   297*  298
                  314*  315   323*  324   344*  346   353*  355   362*  364   371*  373
AWAS    000110R   199*
BEGIN   000000R   157*  240   249   266   275   292   301   318   327   349   358   367   376
                  387
BIT0  = 000001    212*  220   223   386
BIT1  = 000002    212*
BIT10 = 002000    212*
BIT11 = 004000    212*  232   258   284   310
BIT12 = 010000    212*  231   257   283   309
BIT13 = 020000    212*
BIT14 = 040000    212*
BIT15 = 100000    212*
BIT2  = 000004    212*
BIT3  = 000010    212*
BIT4  = 000020    212*
BIT5  = 000040    212*
BIT6  = 000100    212*
BIT7  = 000200    212*
BIT8  = 000400    212*  382
BIT9  = 001000    212*
BREAK$= 104407    212*
BR1     000012R   162*
BR2     000013R   163*
BTOD$ = 104421    212*
CDATA$= 104412    212*
CONFIG  000056R   182*
CSRA    000100R   192*  233*  259*  285*  311*  343*  352*  361*  370*
DAC0    000224R   216*  224*  225   232*  233   236   243*  245   252*  338*  343   344   390*
DAC1    000226R   217*  225*  226*  227   258*  259   262   269*  271   278*  339*  352   353
                  391*
DAC2    000230R   218*  227*  228*  229   284*  285   288   295*  297   304*  340*  361   362
                  392*
DAC3    000232R   219*  229*  230*  310*  311   314   321*  323   330*  341*  370   371   393*
DATCK$= 104411    212*
DATER$= 104404    212*
DONE    001336R   381*
DUAL    001106R   338*
DVID1   000014R   164*  384
ENDIT$= 104413    212*  387
END$  = 104410    212*
ERRTYP  000106R   197*
EXIT$ = 104400    212*
GETPA$= 104415    212*
GWBUF$= 104414    212*
HRDCNT  000044R   177*
HRDER$= 104405    212*  240   249   266   275   292   301   318   327   349   358   367   376
HRDPAS  000050R   179*
ICONT   000036R   174*
```

```
ICOUNT  000040R          175#
IDNUM   000122R          204#
INIT    000030R          171#
INTR    000120R          203#
LOOPA   000244R          224#    395
MAP22$= 104416           212#
MODNAM  000000R          158#
MODSP   000224R          172     210#
MSGN$ = 104403           212#
MSGS$ = 104402           212#
MSG$  = 104401           212#
NULL  = 000000           212#    240     249     266     275     292     301     318     327     349     358     367     376
OPEN  = 000000           159     165     166     167     168     185     186     187     188     189     190     191     192
                         194     196     198     199     201     202     203     212#
OTOA$ = 104420           212#
PASCNT  000034R          173#
PIRQ$ = 000004           212#
POPSP = 005726           212#
POPSP2= 022626           212#
PRTY  = 000000           212#
PRTY0 = 000000           162     163     212#
PRTY1 = 000040           212#
PRTY2 = 000100           212#
PRTY3 = 000140           212#
PRTY4 = 000200           212#
PRTY5 = 000240           212#
PRTY6 = 000300           212#
PRTY7 = 000340           212#
PS    = 177776           212#
PSW   = 177776           212#
PUSH  = 005746           212#
PUSH2 = 024646           212#
RAND$ = 104417           212#
RANNUM  000054R          181#
RESTRT  000236R          200     223#
RES1    000056R          183#
RES2    000060R          184#
RSTRT   000112R          200#
SBADR   000102R          193#
SOFCNT  000042R          176#
SOFER$= 104406           212#
SOFPAS  000046R          178#
SPOINT  000032R          172#
SPSIZ = 000040             1#    205
SR1     000016R          165#
SR2     000020R          166#
SR3     000022R          167#
SR4     000024R          168#
START   000236R          171     222#
STAT    000026R          170#
SVR0    000062R          185#
SVR1    000064R          186#
SVR2    000066R          187#
SVR3    000070R          188#
SVR4    000072R          189#
SVR5    000074R          190#
```



```
SVR6    000076R          191#
SYSCNT  000052R          180#
TEMP    000234R          220#    223#    381#    382     384     386#
TRPDFD= 000022           212#
TSDAC0  000316R          231#    394
TSDAC1  000454R          257#
TSDAC2  000612R          283#
TSDAC3  000750R          309#
VECTOR  000010R          161#
WA$ADR  000104R          195#
WDFR    000116R          202#
WDTO    000114R          201#
XFLAG   000005R          159#


. ABS.  000000     000
        001436     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XMNDA0,XMNDA0/SOL/CRF:SYM=DDXCOM,XMNDA0
 RUN-TIME:  1  1  .2 SECONDS
 RUN-TIME RATIO:  80/3=23.
 CORE USED:  7K  (13 PAGES)
```

.REM_

IDENTIFICATION

PRODUCT CODE: AC-F422A-MC

PRODUCT NAME: CXMNCA0 MNCKW MODULE

PRODUCT DATE: FEBRUARY 1979

MAINTAINER: RAY SHOOP

1.0 ABSTRACT

THE MNC IS AN IOMOD THAT EXERCISES THE MNCKW REAL TIME CLOCK. ON START IT EXERCISES THE CSRS AND PRESET BUFFERS OF THE CLOCK. THEN AND FOR ALL RESTARTS, IT EXERCISES THE CLOCK AT EACH ONE OF ITS BASIC RATES. UP TO 8 MNCKW WILL BE EXERCISED WITH THIS MODULE. THE "MNA" (A/D) MODULE CAN BE ENABLED TO USE THE CLOCK TO START THE A/D. IF YOU HAVE ENABLED "MNA" TO USE THE CLOCK, YOU SHOULD DESELECT THIS EXERCISER MODULE.

2.0 REQUIREMENTS

HARDWARE: ONE MNCKW (CLOCK).

STORAGE:: MNC REQUIRES:
DECIMAL WORDS: 508
OCTAL WORDS: 775
OCTAL BYTES: 1772

3.0 PASS DEFINITION

ONE PASS OF THE MNC MODULE CONSISTS OF GENERATING INTERRUPTS FOR ONE SECOND AT EACH CLOCK RATE, UNTIL 60 SECONDS HAVE ELAPSED.

4.0 EXECUTION TIME

ONE PASS OF THE MNC MODULE RUNNING ALONE TAKES APPROXIMATELY ONE MINUTE.

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 171020, VECTOR 440, BR1: 4

DEVCNT: 1, SR1: 0

REQUIRED PARAMETERS:

NONE.

6.0 DEVICE/OUTPUT SET-UP

THE FRONT PANEL SWITCHES MUST SELECT THE SCHMITT TRIGGER INPUT (PULL OUT "ST1" AND "ST2" SWITCHES AND ROTATE TO THE END).

7.0 MODULE OPERATION

TEST SEQUENCE:

1. (START) PIT EXERCISE CSR, PRESET REGISTER OF CLOCK.

2. (RESTRT) COUNT TESTS USING INTERRUPTS COUNT INTERRUPTS WILL OCCUR IN ONE SECOND AND ADVANCE THE TEST TO THE NEXT RATE.

   AFTER A RATE HAS BEEN SELECTED, A CHECK IS MADE TO SEE IF THE OPERATOR HAS INHIBITED THAT RATE FROM TEST. IF NOT, CONTROL IS TRANSFERRED TO THE PARTICULAR RATE ROUTINE (LISTED BELOW). EACH RATE ROUTINE MUST PRELOAD THE BUFFER REGISTER OF THE CLOCK TO THE COUNT THAT WILL CAUSE IT TO INTERRUPT IN ONE SECOND. AFTER THE BUFFER IS LOADED, THE CSR IS LOADED WITH THE PROPER BITS THAT SELECT THE RATE.

   A. COUNT TEST CLOCK RATE: 1MHZ.

   B. COUNT TEST CLOCK RATE: 100KHZ.

   C. COUNT TEST CLOCK RATE: 10KHZ.

   D. COUNT TEST CLOCK RATE: 1KHZ.

   E. COUNT TEST CLOCK RATE: 100HZ.

   F. COUNT TEST CLOCK RATE: PSEUDO RANDOM (1 OF 3 RATES).

8.0 OPERATION OPTIONS

VALID SR1 VALUES

SR1 BIT	ENABLE/DISABLE	FUNCTION
0	0	ENABLE TESTING 1MHZ
	1	DISABLE TESTING 1MHZ
1	0	ENABLE TESTING 100KHZ
	1	DISABLE TESTING 100KHZ
2	0	ENABLE TESTING 10KHZ
	1	DISABLE TESTING 10KHZ
3	0	ENABLE TESTING 1KHZ
	1	DISABLE TESTING 1KHZ
4	0	ENABLE TESTING 100HZ
	1	DISABLE TESTING 100HZ
5	0	*ENABLE TESTING RANDOM
	1	DISABLE TESTING RANDOM

*NOTE: IF RANDOM RATE SELECTED, THEN AN SR1 BIT DISABLING A PARTICULAR RATE WILL BE IGNORED.

9.0 NON-STANDARD PRINTOUTS

ALL PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN THE DEC/X11 DOCUMENT.

```
                                        IOMOD   <MNCA >,171020,440,4,0,0,60.,,0
                                        MODULE  140000,MNCA ,171020,440,4,0,0,60.,,0
                                        .TITLE  MNCA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6       23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************
000000'                         BEGIN:
000000' 047115  040503      040 MODNAM: .ASCII  /MNCA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF MBUFF USAGE
000006' 171020                  ADDR:   171020+0                ;1ST DEVICE ADDR.
000010' 000440                  VECTOR: 440+0                   ;1ST DEVICE VECTOR.
000012'     200                 BR1:    .BYTE   PRTY4+0         ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;*****************************************************************
000026' 140000                  STAT:   140000                  ;STATUS WORD.
000030' 000254'                 INIT:   START                   ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                       ;PASS COUNTER.
000036' 000074                  ICONT:  60.                     ;# OF ITERATIONS PER PASS=60.
000040' 000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                         ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000102'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000104'                         WASADR:                         ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000106'                         ERRTYP:                         ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000112' 000254'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
000122' 000000                  IONUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
000040
                                        .NLIST
```



```
                                        .WORD   0
                                        .LIST
                                        .ENDR
000224'                         MODSP:
                                ;*****************************************************************
000224' 000001                  TEMP:   1                       ;MASK OF CURRENT UNIT

                                ;MODULE REQUIRED REGISTERS - SET UP BY THIS MODULE.

000226' 171020                  ASR:    .WORD   171020          ;CLOCK A STATUS REG.
000230' 171022                  ABR:    .WORD   171022          ;CLOCK A BUFFER REG.

000232' 000440                  AVECT:  .WORD   440             ;CLOCK A INTERRUPT VECTOR.
000234' 000442                  AVECT2: .WORD   442

000236' 000444                  BVECT:  .WORD   444             ;CLOCK INTERRUPT VECTOR.
000240' 000446                  BVECT2: .WORD   446

000242' 000001                  RATEP:  .WORD   1               ;POINTS TO CURRENT RATE
000244' 000000                  OFF:    .WORD   0               ;OFFSET TO TAKE US TO RATE ROUTINE
000246' 000000                  RANA:   .WORD   0               ;RANDOM NUMBER.
000250' 000000                  RANB:   .WORD   0               ;RANDOM NUMBER.
000252' 000000                  AIFLG:  .WORD   0               ;FLAG TO SHOW THAT CLOCK A HAS INTERRUPTED.

000254'                         RESTRT:
000254' 004767  000002          START:  JSR     PC,START0       ;PRIME THE ADDRESSES
000260' 000442                          BR      LOG1            ;RUN LOGIC TEST

000262' 012767  000001  177734  START0: MOV     #BIT0,TEMP      ;LOAD CURRENT UNIT MASK
000270' 016767  177512  177730  START1: MOV     ADDR,ASR        ;GET BASE ADDR.
000276' 016767  177506  177726          MOV     VECTOR,AVECT    ;GET BASE VECTOR ADDR.
000304' 016767  177716  177716  START2: MOV     ASR,ABR         ;NOW WE'RE GONNA FIX
000312' 062767  000002  177710          ADD     #2,ABR          ;ALL CLOCK ADDRESSES BASED ON ASR.
000320' 016700  177706                  MOV     AVECT,R0        ;NOW FIX VECTOR ADDRESSES
000324' 062700  000004                  ADD     #4,R0
000330' 010067  177702                  MOV     R0,BVECT
000334' 016767  177672  177672          MOV     AVECT,AVECT2
000342' 062767  000002  177664          ADD     #2,AVECT2
000350' 016767  177662  177662          MOV     BVECT,BVECT2
000356' 062767  000002  177654          ADD     #2,BVECT2
000364' 000207                          RTS     PC              ;EXIT
```

```
                                        ;*
                                        ;*LOGIC TEST #1 BE SURE A CLOCK EXISTS AT THE
                                        ;*SPECIFIED ADDR. IF NO CLOCK, THEN A
                                        ;*DEC/X11 SYS ERROR WILL OCCUR.
                                        ;*

000366' 005777  177634          LOG1:   TST     @ASR            ;ADDRESS THE CLOCK, IF SYS ERROR
                                        ;OCCURS, THEN CLOCK DID NOT
                                        ;RETURN SLAVE-SYN WHEN
                                        ;ADDRESSED.

                                        ;*
                                        ;*LOGIC TEST #2, MAKE SURE CLOCK CSR BITS
                                        ;*14,11,6,5,2 AND 0 CAN BE SET + CLEARED.
                                        ;*

000372' 012767  044125  177504  LOG2:   MOV     #044125,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000400' 016777  177500  177620          MOV     ASTAT,@ASR      ;SET THEM IN CSR OF CLOCK A.
000406' 017767  177614  177466          MOV     @ASR,ACSR       ;READ THEM BACK
000414' 026767  177464  177460          CMP     ASTAT,ACSR      ;DID THEY ALL SET?

000422' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
000424' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000430' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000434' 016767  177566  177436  1$:     MOV     ASR,CSRA        ;RECORD CSR'S ADDR
                                        ;****************************************************************
000442' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;PATTERN 044125 FAILED
                                        ;****************************************************************

000450' 005077  177552          2$:     CLR     @ASR            ;TRY CLEARING THE BITS
000454' 017767  177546  177420          MOV     @ASR,ACSR       ;READ IT BACK.
000462' 001414                          BEQ     LOG3            ;IF ZERO CSR GOOD.
000464' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000470' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000474' 005067  177404          3$:     CLR     ASTAT           ;EXPECT ZERO CSR.
000500' 016767  177522  177372          MOV     ASR,CSRA        ;RECORD CSR'S ADDR.
                                        ;****************************************************************
000506' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;CSR FAILED TO CLEAR
                                        ;****************************************************************
```



```
                                        ;*
                                        ;*LOGIC TEST #3, MAKE SURE CLOCK CSR BITS
                                        ;*13,5,3 AND 1 CAN BE SET + CLEARED
                                        ;*

000514' 012767  020052  177362  LOG3:   MOV     #020052,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000522' 016777  177356  177476          MOV     ASTAT,@ASR      ;SET THEM IN CSR OF CLOCK A.
000530' 017767  177472  177344          MOV     @ASR,ACSR       ;READ THEM BACK
000536' 026767  177342  177336          CMP     ASTAT,ACSR      ;DID THEY ALL SET?
000544' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
000546' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000552' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000556' 016767  177444  177314  1$:     MOV     ASR,CSRA        ;RECORD CSR'S ADDR.
                                        ;****************************************************************
000564' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;CSR PATTERN 020052 FAILED
                                        ;****************************************************************

000572' 005077  177430          2$:     CLR     @ASR            ;TRY CLEARING THE BITS
000576' 017767  177424  177276          MOV     @ASR,ACSR       ;READ IT BACK.
000604' 001414                          BEQ     LOG4            ;IF ZERO CSR GOOD.
000606' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000612' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000616' 005067  177262          3$:     CLR     ASTAT           ;EXPECT ZERO CSR.
000622' 016767  177400  177250          MOV     ASR,CSRA        ;RECORD CSR'S ADDR.
                                        ;****************************************************************
000630' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;CSR FAILED TO CLEAR
                                        ;****************************************************************

                                        ;*
                                        ;*LOGIC TEST #4, MAKE SURE CLOCK BUFFER REG
                                        ;*PATTERN 125252 CAN BE SET + CLEARED.
                                        ;*

000636' 012767  125252  177240  LOG4:   MOV     #125252,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000644' 016777  177234  177356          MOV     ASTAT,@ABR      ;SET THEM IN BUFFER REG OF CLOCK.
000652' 017767  177352  177222          MOV     @ABR,ACSR       ;READ THEM BACK
000660' 026767  177220  177214          CMP     ASTAT,ACSR      ;DID THEY ALL SET?
000666' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
000670' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000674' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000700' 016767  177324  177172  1$:     MOV     ABR,CSRA
                                        ;****************************************************************
000706' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;BUFFER REG PATTERN 125252 FAILED
                                        ;****************************************************************

000714' 005077  177310          2$:     CLR     @ABR            ;TRY CLEARING THE BITS
000720' 017767  177304  177154          MOV     @ABR,ACSR       ;READ IT BACK.
000726' 001414                          BEQ     LOG5            ;IF ZERO BUFFER GOOD.
000730' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000734' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000740' 005067  177140          3$:     CLR     ASTAT           ;EXPECT ZERO BUFFER.
000744' 016767  177260  177126          MOV     ABR,CSRA        ;RECORD ADDR. OF BUFFER REG.
                                        ;****************************************************************
000752' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;BUFFER REG FAILED TO CLEAR
                                        ;****************************************************************

                                        ;*LOGIC TEST #5, MAKE SURE CLOCK BUFFER REG
                                        ;*PATTERN 052525 CAN BE SET + CLEARED
000760' 012767  052525  177116  LOG5:   MOV     #052525,ASTAT   ;GENERATE + RECORD PATTERN TO BE USED.
000766' 016777  177112  177234          MOV     ASTAT,@ABR      ;SET THEM IN BUFFER OF CLOCK A.
```

```
001774' 017767  177230  177100          MOV     @ABR,ACSR       ;READ THEM BACK
001002' 026767  177076  177072          CMP     ASTAT,ACSR      ;DID THEY ALL SET?
001010' 001412                          BEQ     2$              ;YES - GO TO NEXT TEST.
001012' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001016' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001022' 016767  177202  177050  1$:     MOV     ABR,CSRA        ;RECORD BUFFER REG ADDR.
                                        ;*************************************************************
001030' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;BUFF REG PATTERN 052525 FAILED
                                        ;*************************************************************
001036' 005077  177166          2$:     CLR     @ABR            ;TRY CLEARING THE BITS
001042' 017767  177162  177032          MOV     @ABR,ACSR       ;READ IT BACK.
001050' 001414                          BEQ     REST0           ;IF ZERO BUFFER GOOD,
001052' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001056' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001062' 005067  177016          3$:     CLR     ASTAT           ;EXPECT ZERO BUFFER.
001066' 016767  177136  177004          MOV     ABR,CSRA        ;RECORD BUFFER REG A ADDR.
                                        ;*************************************************************
001074' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;BUFFER REG FAILED TO CLEAR
                                        ;*************************************************************
001102' 006367  177116          REST0:  ASL     TEMP            ;CHECK FOR NEXT UNIT
001106' 022767  000400  177110          CMP     #BIT8,TEMP      ;TEST IF VALID UNIT
001114' 001420                          BEQ     REST1           ;BR IF NOT
001116' 036767  177102  176670          BIT     TEMP,DVID1      ;TEST IF THIS UNIT IS SELECTED
001124' 001414                          BEQ     REST1           ;BR IF NOT
001126' 005077  177074                  CLR     @ASR            ;ENSURE CLOCK IS STOPPED
001132' 062767  000004  177066          ADD     #4,ASR          ;UPDATE STATUS ADDRESS
001140' 062767  000010  177064          ADD     #10,AVECT       ;UPDATE VECTOR ADDRESS
001146' 004767  177132                  JSR     PC,START2       ;FIX OTHER VALUES
001152' 000167  177210                  JMP     LOG1            ;CONTINUE AT LOG1
001156' 012767  000100  177056  REST1:  MOV     #BIT6,RATEP     ;FIRST PASS THRU LOOP, OFFSET=0, RATES WILL=1.
001164' 012777  001574' 177040  REST2:  MOV     #INSERV,@AVECT  ;SET UP CLOCK A'S INTER. VECTOR.
001172' 116777  176614  177034  REST3:  MOVB    BR1,@AVECT2     ;SET PRIORITY ON CLOCK A'S INTR.
001200' 106167  177036          LOOP:   ROLB    RATEP           ;GET NEXT RATE.
001204' 100005                          BPL     1$              ;IF NOT END THEN CONTINUE.
001206' 005067  177032                  CLR     OFF             ;CLEAR THE OFFSET
001212' 012767  000001  177022          MOV     #1,RATEP        ;LOOK AT FIRST RATE
001220' 062767  000002  177016  1$:     ADD     #2,OFF
001226' 022767  000016  177010          CMP     #16,OFF         ;TEST IF =16
001234' 001761                          BEQ     LOOP            ;BR
001236' 036767  177000  176552          BIT     RATEP,SR1       ;IS THIS RATE INHIBITED?
001244' 001355                          BNE     LOOP
001246' 005067  177000                  CLR     AIFLG           ;CLR FLAG INDICATING CLOCK A HAS INTERRUPTED.
001252' 016701  176766                  MOV     OFF,R1          ;PICK UP OFFSET
001256' 000171  001262'                 JMP     @LISTP(R1)      ;GO SET THE RATE + START THE CLOCK.
```



```
                                        ;THE FOLLOWING (LISTP) ARE POINTERS TO VARIOUS RATE
                                        ;ROUTINES. THEY ARE USED BY "LOOP". "LOOP" GENERATES
                                        ;AN OFFSET OF A RATE WE WISH TO EXERCISE. THE OFFSET
                                        ;IS STORED IN R1. WE INDEX "LISTP" BY R1 (JMP @LISTP(R1))
                                        ;TO GET THE ADDRESS OF THE RATE ROUTINE TO EXERCISE.
                                        ;

001262' 000001          LISTP:  .WORD   1               ;
001264' 001340'                 .WORD   RATE0           ;POINTER TO 1MHZ ROUTINE
001266' 001362'                 .WORD   RATE1           ;POINTER TO 100KHZ ROUTINE
001270' 001404'                 .WORD   RATE2           ;POINTER TO 10KHZ ROUTINE
001272' 001426'                 .WORD   RATE3           ;POINTER TO 1KHZ ROUTINE
001274' 001450'                 .WORD   RATE4           ;POINTER TO 100HZ ROUTINE
001276' 001472'                 .WORD   RATE5           ;POINTER TO RANDOM ROUTINE
001300' 000001                  .WORD   1

                                        ;
                                        ;THE FOLLOWING (RATEAL) ARE THE PRESET VALUES THAT THE
                                        ;VARIOUS RATE ROUTINES NEED. THEY ARE LOADED INTO
                                        ;CLOCK A'S PRESET BUFFER. "RATEAL" IS INDEXED BY
                                        ;AN OFFSET IN R1 BY THE RATE ROUTINES TO GET THE
                                        ;PRESET VALUE
                                        ;

001302' 000001          RATEAL: .WORD   1               ;OFFSET ZERO, NO RATE.
001304' 036260                  .WORD   -50000.         ;VALUE FOR 1MHZ PRESET.
001306' 036260                  .WORD   -50000.         ;PRESET VALUE FOR 100 KHZ
001310' 154360                  .WORD   -10000.         ;PRESET VALUE FOR 10 KHZ
001312' 176030                  .WORD   -1000.          ;PRESET VALUE FOR 1 KHZ
001314' 177634                  .WORD   -100.           ;PRESET VALUE FOR 100 HZ
001316' 000000                  .WORD   0               ;PRESET VALUE FOR RANDOM
001320' 000001                  .WORD   1

                                        ;
                                        ;THE FOLLOWING (RSAL) IS USED BY THE RANDOM
                                        ;RATE ROUTINE (RATE5). THEY ARE THE VALUES NEEDED
                                        ;TO BE PUT INTO THE CLOCK'S CSR FOR A PARTICULAR RATE.
                                        ;

001322' 000000          RSAL:   .WORD   0               ;OFFSET ZERO, NO RATE.
001324' 000113                  .WORD   113             ;1 MHZ, GO., MODE 1
001326' 000123                  .WORD   123             ;100 KHZ, GO., MODE 1
001330' 000131                  .WORD   131             ;10 KHZ, GO.
001332' 000141                  .WORD   141             ;1 KHZ, GO.
001334' 000151                  .WORD   151             ;100 HZ, GO.
001336' 000000                  .WORD   0
```

```
                                        ;*THIS ROUTINE PRESETS CLOCK A FOR
                                        ;*1 MHZ RATE    CLOCK A INTRS IN 1/20 SEC., 25 TIMES.
                                        ;*
                                        ;*

001340'                         RATE0:
001340' 005077  176662                  CLR     @ASR            ;CLEAR CLOCK A,
001344' 016177  001302' 176656          MOV     RATEAL(P1),@ABR ;PRESET COUNT IN CLOCK A,
001352' 016177  001322' 176646          MOV     RSAL(R1),@ASR   ;START CLOCK A,
                                                                ;NOW WAIT FOR INTERRUPT
001360' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERRRUPT.

                                        ;*
                                        ;*THIS ROUTINE PRESETS CLOCK A FOR
                                        ;*100 KHZ RATE CLOCK A INTRS IN ,5 SEC., TWICE.
                                        ;*
                                        ;*

001362'                         RATE1:
001362' 005077  176640                  CLR     @ASR            ;CLEAR CLOCK A,
001366' 016177  001302' 176634          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A,
001374' 016177  001322' 176624          MOV     RSAL(R1),@ASR   ;START CLOCK A,
                                                                ;NOW WAIT FOR INTERRUPT,
001402' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.

                                        ;*
                                        ;*THIS ROUTINE PRESETS CLOCK A FOR
                                        ;*10 KHZ RATE   CLOCK A INTRS IN 1.0 SEC.
                                        ;*
                                        ;*

001404'                         RATE2:
001404' 005077  176616                  CLR     @ASR            ;CLEAR CLOCK A,
001410' 016177  001302' 176612          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A,
001416' 016177  001322' 176602          MOV     RSAL(R1),@ASR   ;START CLOCK A,
                                                                ;NOW WAIT FOR INTERRUPT,
001424' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAST FOR INTERRUPT.

                                        ;*
                                        ;*THIS ROUTINE PRESETS CLOCK FOR
                                        ;*1 KHZ RATE    CLOCK A INTRS IN 1.0 SEC.
                                        ;*
                                        ;*

001426'                         RATE3:
001426' 005077  176574                  CLR     @ASR            ;CLEAR CLOCK A,
001432' 016177  001302' 176570          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A,
001440' 016177  001322' 176560          MOV     RSAL(R1),@ASR   ;START CLOCK A,
                                                                ;NOW WAIT FOR INTERRUPT,
001446' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
```



```
                                        ;*THIS ROUTINE PRESETS CLOCK FOR
                                        ;*100 HZ RATE   CLOCK A INTRS IN 1.0 SEC.
                                        ;*
                                        ;*

001450'                         RATE4:
001450' 005077  176552                  CLR     @ASR            ;CLEAR CLOCK A,
001454' 016177  001302' 176546          MOV     RATEAL(R1),@ABR ;PRESET COUNT IN CLOCK A,
001462' 016177  001322' 176536          MOV     RSAL(R1),@ASR   ;START CLOCK A,
                                                                ;NOW WAIT FOR INTERRUPT,
001470' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.

                                        ;*THIS ROUTINE PRESETS CLOCK FOR
                                        ;*RANDOM RATES

001472' 004767  000246          RATE5:  JSR     PC,RANDOM       ;GET 2 RANDOM NUMBERS.

001476' 042767  177771  176542          BIC     #177771,RANA    ;MAKE NUMBER < 10.
001504' 042767  177771  176536          BIC     #177771,RANB    ;MAKE 2ND NUMBER < 10.

                                        ;NUMBERS MUST BE 2, 4, OR 6

001512' 005767  176530          3$:     TST     RANA            ;IS NUMBER ZERO?
001516' 001003                          BNE     4$              ;NO - GO AHEAD.
001520' 062767  000002  176520          ADD     #2,RANA         ;MAKE IT NON-ZERO.
001526' 005767  176516          4$:     TST     RANB            ;IS NUMBER ZERO?
001532' 001003                          BNE     5$              ;NO GO AHEAD.
001534' 062767  000002  176506          ADD     #2,RANB         ;MAKE IT NON-ZERO.
001542'                         5$:
001542' 005077  176460                  CLR     @ASR            ;CLEAR CLOCK A
001546' 016701  176474                  MOV     RANA,R1
001552' 010167  176466                  MOV     R1,OFF          ;RECORD THE OFFSET
001556' 016177  001302' 176444          MOV     RATEAL(R1),@ABR ;PRESET CLOCK A.
001564' 016177  001322' 176434          MOV     RSAL(R1),@ASR   ;START CLOCK A.

001572' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT
```

```
                                        ;*INTERRUPT SERVICE ROUTINE FOR CLOCK.
001574' 005267  176452          INSERV: INC     AIFLG           ;INDICATE CLOCK A HAS INTERRUPTED
001600' 042777  000200  176420          BIC     #200,@ASR       ;CLEAR OVERFLOW FLAG.
001606' 026727  176432  000002  28:     CMP     OFF,#2          ;ARE WE RUNNING 1MHZ RATE?
001614' 001005                          BNE     3$              ;IF NOT, 3$
001616' 026727  176430  000024          CMP     AIFLG,#20.      ;1 MHZ, 1 SEC. UP?
001624' 001412                          BEQ     4$              ;YES - STOP
001626' 000002                          RTI                     ;NO, ALLOW ANOTHER COUNT.
001630' 026727  176410  000004  3$:     CMP     OFF,#4          ;100 KHZ RATE?
001636' 001005                          BNE     4$              ;NO - 4$
001640' 026727  176406  000002          CMP     AIFLG,#2        ;YES, COUNTED TWICE?
001646' 001401                          BEQ     4$              ;YES - 4$,
001650' 000002                          RTI                     ;NO - COUNT ONE MORE TIME.
001652' 005077  176350          4$:     CLR     @ASR            ;STOP CLOCK A.
                                        ;-----------------------------------------------------------
001656' 000004  000000' 001664'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;-----------------------------------------------------------
001664' 006367  176334          1$:     ASL     TEMP            ;CHECK NEXT UNIT
001670' 022767  000400  176326          CMP     #BIT8,TEMP      ;TEST IF VALID UNIT
001676' 001004                          BNE     11$             ;BR IF OK
001700'                         10$:
001700' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                ;MONITOR SHALL TEST END OF PASS
001704' 000167  177254                  JMP     REST2
001710' 036767  176310  176076  11$:    BIT     TEMP,DVID1      ;TEST IF UNIT SELECTED TO RUN ?
001716' 001770                          BEQ     10$             ;BR IF NOT
001720' 062767  000004  176300          ADD     #4,ASR          ;UPDATE ADDRESS
001726' 062767  000010  176276          ADD     #10,AVECT       ;UPDATE VECTOR
001734' 004767  176344                  JSR     PC,START2       ;FIX ADDRESSES
001740' 000167  177226                  JMP     REST3
                                        ;*THIS SUBROUTINE GENERATES TWO RANDOM NUMBERS
                                        ;*CALL = JSR PC, RANDOM
                                        ;*RETURNS WITH NUMBERS IN RANDA AND RANDB
001744' 066767  176276  176276  RANDOM: ADD     RANA,RANB
001752' 005567  176270                  ADC     RANA
001756' 066767  176266  176262          ADD     RANB,RANA
001764' 005567  176260                  ADC     RANB
001770' 000207                          RTS     PC
        000001                          .END
```



```
ABR     000230R          246#  267*  268*  347*  348   353   357*  358   363   370*  371   376   380*
                         381   386   468*  481*  494*  507*  518*  543*
ACSR    000102R          223#  295*  296   306*  322*  323   332*  348*  349   358*  371*  372   381*
ADDR    000006R          189#  265
ADDR22= 001000           241#
AIFLG   000252R          258#  412*  548*  552   557
ASB     000106R          227#
ASR     000226R          245#  265*  267   283   294*  295   301   305*  306   311   321*  322   327
                         331*  332   337   395*  396*  467*  469*  480*  482*  493*  495*  506*  508*
                         517*  519*  540*  544*  549*  560*  573*
ASTAT   000104R          225#  293*  294   296   310*  320*  321   323   336*  346*  347   349   362*
                         369*  370   372   385*
AVECT   000232R          248#  266*  269   272   397*  401*  574*
AVECT2  000234R          249#  272*  273*  402*
AWAS    000110R          228#
BEGIN   000000R          186#  299   300   303   308   309   313   325   326   329   334   335   339
                         351   352   355   360   361   365   374   375   378   383   384   388   562
                         568
BIT0  = 000001           241#  264
BIT1  = 000002           241#
BIT10 = 002000           241#
BIT11 = 004000           241#
BIT12 = 010000           241#
BIT13 = 020000           241#
BIT14 = 040000           241#
BIT15 = 100000           241#
BIT2  = 000004           241#
BIT3  = 000010           241#
BIT4  = 000020           241#
BIT5  = 000040           241#
BIT6  = 000100           241#  400
BIT7  = 000200           241#
BIT8  = 000400           241#  391   565
BIT9  = 001000           241#
BREAK$= 104407           241#  299   300   308   309   325   326   334   335   351   352   360   361
                         374   375   383   384
BR1     000012R          191#  402
BR2     000013R          192#
BTOD$ = 104421           241#
BVECT   000236R          251#  271*  274
BVECT2  000240R          252#  274*  275*
CDATA$= 104412           241#
CONFIG  000056R          211#
CSRA    000100R          221#  301*  311*  327*  337*  353*  363*  376*  386*
DATCK$= 104411           241#
DATER$= 104401           241#
DVID1   000014R          193#  393   571
ENDIT$= 104413           241#  568
END$  = 104410           241#
ERRTYP  000106R          226#
EXIT$ = 104400           241#  471   484   497   510   521   546
GETPA$= 104415           241#
GWBUF$= 104414           241#
HRDCNT  000044R          206#
HRDER$= 104405           241#  303   313   329   339   355   365   378   388
HRDPAS  000050R          208#
```

```
ICONT   000036R          203#
ICOUNT  000040R          204#
IDNUM   000122R          233#
INIT    000030R          200#
INSERV  001574R          401     548#
INTR    000120R          232#
LISTP   001262R          414     422#
LOG1    000366R          262     283#    399
LOG2    000372R          293#
LOG3    000514R          307     320#
LOG4    000636R          333     346#
LOG5    000760R          359     369#
LOOP    001200R          403#    409     411
MAP22$= 104416           241#
MODNAM  000000R          187#
MODSP   000224R          201     239#
MSGN$ = 104403           241#
MSGS$ = 104402           241#
MSG$  = 104401           241#
NULL  = 000000           241#    303     313     329     339     355     365     378     388
OFF     000244R          255#    405#    407#    408     413     542#    550     555
OPEN  = 000000           188     194     195     196     197     214     215     216     217     218     219     220     221
                         223     225     227     228     230     231     232     241#
OTOA$ = 104420           241#
PASCNT  000034R          202#
PIRQ$ = 000004           241#    562
POPSP = 005726           241#
POPSP2= 022626           241#
PRTY  = 000000           241#
PRTY0 = 000000           192     241#
PRTY1 = 000040           241#
PRTY2 = 000100           241#
PRTY3 = 000140           241#
PRTY4 = 000200           191     241#
PRTY5 = 000240           241#
PRTY6 = 000300           241#
PRTY7 = 000340           241#
PS    = 177776           241#
PSW   = 177776           241#
PUSH  = 005746           241#
PUSH2 = 024646           241#
RANA    000246R          256#    528#    533     535#    541     580     581#    582#
RANB    000250R          257#    529#    536     538#    580#    582     583#
RANDOM  001744R          526     580#
RAND$ = 104417           241#
RANNUM  000054R          210#
RATEAL  001302R          439#    468     481     494     507     518     543
RATEP   000242R          254#    400#    403#    406#    410
RATE0   001340R          423     466#
RATE1   001362R          424     479#
RATE2   001404R          425     492#
RATE3   001426R          426     505#
RATE4   001450R          427     516#
RATE5   001472R          428     526#
RESTRT  000254R          229     260#
REST0   001102R          382     390#
```



```
REST1   001156R          392     394     400#
REST2   001164R          401#    570
REST3   001172R          402#    576
RES1    000056R          212#
RES2    000060P          213#
RSAL    001322R          454#    469     482     495     508     519     544
RSTRT   000112P          229#
SBADR   000102R          222#
SOFCNT  000042R          205#
SOFER$= 104406           241#
SOFPAS  000046R          207#
SPOINT  000032R          201#
SPSIZ = 000040             1#    234
SR1     000016R          194#    410
SR2     000020R          195#
SR3     000022R          196#
SR4     000024R          197#
START   000254R          200     261#
START0  000262R          261     264#
START1  000270R          265#
START2  000304R          267#    398     575
STAT    000026R          199#
SVR0    000062R          214#
SVR1    000064R          215#
SVR2    000066R          216#
SVR3    000070R          217#
SVR4    000072R          218#
SVR5    000074R          219#
SVR6    000076R          220#
SYSCNT  000052R          209#
TEMP    000224R          241#    264#    390#    391     393     564#    565     571
TRPDFD= 000022           241#
VECTOR  000010R          190#    266
WASADR  000104R          224#
WDFR    000116R          231#
WDTO    000114R          230#
XFLAG   000005R          188#

. ABS.  000000      000
        001772      001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

PACK:XMNCA0,PACK:XMNCA0/SOL/CRF:SYM=DDXCOM,PACK:XMNCA0
RUN-TIME:  3  5  .7 SECONDS
RUN-TIME RATIO:  43/10=4.1
CORE USED:  7K  (13 PAGES)
```

.REM_

IDENTIFICATION

PRODUCT CODE: AC-F419A-MC

PRODUCT NAME: CXMNBA0 MNCDI MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

THE MNB IS AN IOMOD THAT EXERCISES THE MNCDI DIGITAL INPUT.
THE SOFTWARE MODULE CONSISTS OF THREE SECTIONS. THE FIRST IS
THE DEFAULT CONDITION WHEN SR1 = 0. READ-WRITE TESTS ARE PERFORMED
TO VERIFY THE INTERNAL DATA PATH'S OF THE MNCDI LOGIC.
THE SECOND IS EXECUTED WHEN A MNCDO OUTPUT IS CONNECTED TO THE
MNCDI INPUT. THIS PROVIDES ADDITIONAL TEST OF THE CONTROL
SIGNALS. THE THIRD IS A WRAP-AROUND DATA TEST VERIFING THE DATA
OUTPUT AND INPUT CIRCUITS. UP TO 8 MNCDI'S CAN BE EXERCISED
WITH THIS MODULE. IF SR1 = 1, BE SURE TO "DESELECT" THE MNE
(DIGITAL OUT) MODULE.

2.0 REQUIREMENTS

HARDWARE: ONE MNCDI (DIGITAL IN).
ONE MNCDO (DIGITAL OUT) <OPTIONAL>
STORAGE: MNB REQUIRES:
DECIMAL WORDS: 1573
OCTAL WORDS: 3045
OCTAL BYTES: 6112

3.0 PASS DEFINITION

WHEN SR1 = 0, ONE PASS OF THE MNB MODULE CONSISTS OF GENERATION
4000 (8) INTERRUPTS.
WHEN SR1 = 1, ONE PASS OF THE MNB MODULE CONSISTS OF GENERATING
5000 (8) INTERRUPTS.

4.0 EXECUTION TIME

WHEN SR1 = 0, ONE PASS OF THE MNB MODULE RUNNING ALONE TAKES
APPROXIMATELY ONE MINUTE.
WHEN SR1 = 1, ONE PASS OF THE MNB MODULE RUNNING ALONE TAKES
APPROXIMATELY TWO MINUTES.

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 171160, VECTOR 130, BR1: 4 DEVCNT: 1, SR1: 0

REQUIRED PARAMETERS:

NONE IF SR1 = 0 IF SR1 = 1 THEN:
$BASE1 MUST CONTAIN THE MNCDO BUS ADDRESS
$VECT1 MUST CONTAIN THE MNCDO INTERRUPT VECTOR AND BE
SURE TO DESELECT THE "MNE" (DIGITAL OUT) MODULE.

6.0 DEVICE/OUTPUT SET-UP

THE FRONT PANEL SWITCHES MUST BE IN THE "-" POSITION,
NO ADDITIONAL IF SR1 = 0 IF SR1 = 1 THEN:
THE WRAP-AROUND CABLE MUST BE INSTALLED TO A MNCDO AND
BE SURE TO DESELECT THE "MNE" (DIGITAL OUT) MODULE.

7.0 MODULE OPERATION

THE FOLLOWING TESTS ARE PERFORMED ON THE MNCDI (SR1=0)

FLOAT A 1 ACROSS THE STIMULUS BIT REGISTER
FLOAT A 0 ACROSS THE STIMULUS BIT REGISTER
BYTE OPERATION OF THE STIMULUS BIT REGISTER
READ-WRITE TESTS OF BITS 1 - 6,8,9,12 AND 14 IN THE STATUS REGISTER
BYTE OPERATION OF THE STATUS REGISTER
MAINT. STROBE SETS INPUT READY FLAG
INPUT READY CAN BE WRITTEN TO A ZERO
INPUT READY WILL NOT SET IF NO "SBR MATCH"
OVERRUN FLAG SETS
OVERRUN FLAG CAN BE WRITTEN TO A ZERO
INVERT DATA FUNCTIONS CORRECTLY
EACH BIT OF THE INPUT REGISTER CAN BE CLEARED
INPUT READY FLAG INTERRUPT TEST
OVERRUN FLAG INTERRUPT TEST

8.0 OPERATION OPTIONS

SR1 = 0	RUN MNCDI LOGIC TEST
SR1 = 1	RUN MNCDI LOGIC TEST RUN MNCDO LOGIC TEST RUN MNCDO TO MNCDI WRAPAROUND CONTROL TEST RUN MNCDO TO MNCDI WRAPAROUND DATA TEST

IF DEVCNT (DVID1) CONTAINS MORE THAT A 1, MULTIPLE MNCDI
WILL BE TESTED. IF DEVCNT (DVID1) CONTAINS MORE THAN 1 AND SR1=1,
THERE MUST AT LEAST THE SAME NUMBER OF MNCDO'S CONNECTED TO MNCDI'S.
THE FIRST ADDRESS MNCDO UNIT MUST BE CONNECTED TO THE FIRST
ADDRESS MNCDI WITH ADDITIONAL UNITS ALSO PAIRED TOGETHER.

9.0 NON-STANDARD PRINTOUTS

ALL PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN THE DEC/X11
DOCUMENT.

```
                                        .LIST   ME
                                        .NLIST  MC,CND,MD
                                        .TITLE  MNBA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       IDXCOM  VERSION  6      23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************
000000'                         BEGIN:
000000' 047115  040502      040 MODNAM: .ASCII  /MNBA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 171160                  ADDR:   171160+0                ;1ST DEVICE ADDR.
000010' 000130                  VECTOR: 130+0                   ;1ST DEVICE VECTOR.
000012'     200                 BR1:    .BYTE   PRTY4+0         ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;*****************************************************************
000026' 140000                  STAT:   140000                  ;STATUS WORD.
000030' 000300'                 INIT:   START                   ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                       ;PASS COUNTER.
000036' 002000                  ICONT:  2000                    ;# OF ITERATIONS PER PASS=2000
000040' 000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                         ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000102'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000104'                         WASADR:                         ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000106'                         ERRTYP:                         ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000112' 000306'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
000122' 000000                  IDNUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
```



```
000224'                         MODSP:
                                ;*****************************************************************

000224' 171260                  $BASE1: 171260          ;INITIAL BUS ADDRESS OF THE MNCDO <IF SR1=1>
000226' 000340                  $VECT1: 340             ;INITIAL INTERRUPT VECTOR OF THE MNCDO <IF SR1=1>

                                ;OUTPUT ADDRESS
000230' 000000                  OCSR:   0
000232' 000000                  OCSR1:  0                       ;HIGH BYTE ADDRESS

000234' 000000                  DOR:    0
000236' 000000                  DOR1:   0                       ;HIGH BYTE ADDRESS

                                ;INPUT ADDRESS
000240' 000000                  ICSR:   0
000242' 000000                  ICSR1:  0                       ;HIGH BYTE ADDRESS
000244' 000000                  DIR:    0
000246' 000000                  DIR1:   0
000250' 000000                  SBR:    0
000252' 000000                  SBR1:   0                       ;HIGH BYTE ADDRESS

000254' 000000                  DODINV: 0
000256' 000000                  DODINS: 0

000260' 000000                  DIDINV: 0
000262' 000000                  DIDINS: 0
000264' 000000                  DIEINV: 0
000266' 000000                  DIEINS: 0
000270' 000000                  SRMINE: 0                       ;MY COPY OF SR1
000272' 000000                  TEMP:   0
000274' 000000                  TEMP1:  0
000276' 000001                  TEMP2:  BIT0
                                        BITDAT=BIT12            ;MAINT INPUT INHIBIT
                                        BITEXT=BIT11            ;MAINT INPUT STROBE
```

```
000300' 016767 177512 177762 START: MOV     $R1,$RMINE      ;COPY SR1 FOR MY USE
                             ;INITILIZE THE BUS ADDRESSES AND VECTORS
000306' 012767 000001 177762 RESTRT: MOV     #BIT0,TEMP2     ;LOAD UNIT SELECT FLAG
000314' 012700 000230'               MOV     #OCSR,R0        ;LOAD ADDRESS POINTER
000320' 016701 177700                MOV     $BASE1,R1       ;LOAD INITIAL BUS ADDRESS
000324' 010120                1$:    MOV     R1,(R0)+        ;LOAD DEVICE ADDRESS
000326' 005201                       INC     R1              ;UPDATE BUS ADDRESS VALUE
000330' 020027 000240'               CMP     R0,#ICSR        ;TEST IF DONE WITH BUS ADDRESSES
000334' 001373                       BNE     1$              ;BR IF NOT
000336' 016701 177444                MOV     ADDR,R1         ;LOAD INPUT ADDRESS
000342' 010120                2$:    MOV     R1,(R0)+        ;LOAD THE ADDRESS
000344' 005201                       INC     R1              ;UPDATE ADDRESS
000346' 020027 000254'               CMP     R0,#DODINV      ;TEST IF AT END
000352' 001373                       BNE     2$              ;BR IF NOT
000354' 016701 177646                MOV     $VECT1,R1       ;LOAD VECTOR POINTER
000360' 010120                3$:    MOV     R1,(R0)+        ;LOAD DEVICE VECTOR ADDRESS
000362' 005721                       TST     (R1)+           ;UPDATE BUS VECTOR VALUE
000364' 020027 000260'               CMP     R0,#DODINS+2    ;TEST IF DONE WITH BUS VECTORS
000370' 001373                       BNE     3$              ;BR IF NOT
000372' 016701 177412                MOV     VECTOR,R1       ;LOAD INPUT VECTOR
000376' 010120                4$:    MOV     R1,(R0)+        ;LOAD THE VECTOR
000400' 005721                       TST     (R1)+           ;BUMP THE ADDRESS
000402' 020027 000270'               CMP     R0,#DIEINS+2    ;TEST IF DONE
000406' 001373                       BNE     4$              ;BR IF NOT

                             ;TEST IF UNIT IS TO RUN
000410' 036767 177662 177376 CONT1:  BIT     TEMP2,DVID1     ;IS UNIT SELECTED ?
000416' 001040                       BNE     INPUT           ;BR IF SELECTED
                             ;UNIT IS NOT SELECTED -- CORRECT THE ADDRESSES
000420' 012700 000230'       CONT2:  MOV     #OCSR,R0        ;GET ADDRESS
000424' 012701 000004                MOV     #4,R1           ;GET NEW OFFSET VALUE
000430' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000432' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000434' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000436' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000440' 012701 000010                MOV     #10,R1          ;RELOAD NEW OFFSET
000444' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000446' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000450' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000452' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000454' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000456' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000460' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000462' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000464' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000466' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000470' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
000472' 060110                       ADD     R1,(R0)         ;UPDATE THE VALUE
                             ;DETERMINE IF NEXT UNIT IS TO BE TESTED
000474' 006367 177576                ASL     TEMP2           ;CHANGE UNITS
000500' 022767 000400 177570         CMP     #BIT8,TEMP2     ;TEST IF LAST UNIT
000506' 001340                       BNE     CONT1           ;BR IF NOT
000510' 104413 000000'               ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                             ;MONITOR SHALL TEST END OF PASS
000514' 000167 177566                JMP     RESTRT
```



```
                             ;VERIFY A MNCDI BUS ADDRESS RESPONSE
000520' 016767 177514 177352 INPUT:  MOV     ICSR,CSRA               ;LOAD ADDR.
000526' 005777 177506                TST     #ICSR                   ;TEST INPUT STATUS
000532' 005777 177506                TST     #DIR                    ;TEST INPUT DATA REGISTER
000536' 005777 177506                TST     #SBR                    ;TEST STIM. BUFFER REGISTER
000542'                      DI1:
                             ;FLOAT A 1 ACROSS THE MNCDI STIMULUS BIT REGISTER
000542' 012767 000001 177332         MOV     #BIT0,ACSR              ;LOAD EXPECT BIT
000550' 016777 177326 177472 1$:     MOV     ACSR,@SBR               ;LOAD MNCDI STIMULUS BIT REGISTER
000556' 017767 177466 177320         MOV     @SBR,ASTAT              ;READ MNCDI STIMULUS BIT REGISTER
000564' 026767 177312 177312         CMP     ACSR,ASTAT              ;COMPARE
000572' 001403                       BEQ     2$                      ;BR IF SAME
                             ;*****************************************************************
000574' 104405 000000' 000000        HRDER$,BEGIN,NULL       ; MNCDI STIMULUS BIT REGISTER FAILED TO HOLD A FLOATING
                             ;*****************************************************************
000602'                      2$:
000602' 104407 000000'               BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000606' 104407 000000'               BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000612' 006367 177264                ASL     ACSR                    ;CHANGE THE DATA
000616' 001354                       BNE     1$                      ;BR IF MORE DATA
000620'                      DI2:
                             ;FLOAT A 0 ACROSS THE MNCDI STIMULUS BIT REGISTER
000620' 012767 000001 177444         MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
000626' 016767 177440 177246 1$:     MOV     TEMP,ACSR               ;LOAD EXPECTED
000634' 005167 177242                COM     ACSR                    ;COMPLEMENT
000640' 016777 177236 177402         MOV     ACSR,@SBR               ;LOAD MNCDI STIMULUS BIT REGISTER
000646' 017767 177376 177230         MOV     @SBR,ASTAT              ;READ MNCDI STIMULUS BIT REGISTER
000654' 026767 177222 177222         CMP     ACSR,ASTAT              ;COMPARE
000662' 001403                       BEQ     2$                      ;BR IF SAME
                             ;*****************************************************************
000664' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;MNCDI STIMULUS BIT REGISTER FAILED TO HOLD A FLOATING 0
                             ;*****************************************************************
000672'                      2$:
000672' 104407 000000'               BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000676' 104407 000000'               BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000702' 006367 177364                ASL     TEMP                    ;CHANGE THE DATA
000706' 001347                       BNE     1$                      ;BR IF MORE DATA
000710'                      DI3:
                             ;VERIFY BYTE OPERATION ON THE MNCDI STIMULUS BIT REGISTER
000710' 012777 177777 177332 1$:     MOV     #-1,@SBR                ;LOAD MNCDI STIMULUS BIT REGISTER
000716' 012767 000377 177156         MOV     #377,ACSR               ;LOAD EXPECTED
000724' 105077 177322                CLRB    @SBR1                   ;CLEAR HIGH BYTE
000730' 017767 177314 177146         MOV     @SBR,ASTAT              ;READ MNCDI STIMULUS BIT REGISTER
000736' 026767 177140 177140         CMP     ACSR,ASTAT              ;COMPARE
000744' 001403                       BEQ     2$                      ;BR IF SAME
                             ;*****************************************************************
000746' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;CLEARING HIGH BYTE CHANGED LOW BYTE
                             ;*****************************************************************
000754' 012777 177777 177266 2$:     MOV     #-1,@SBR                ;LOAD MNCDI STIMULUS BIT REGISTER
000762' 012767 177400 177112         MOV     #177400,ACSR            ;LOAD EXPECTED
000770' 105077 177254                CLRB    @SBR                    ;CLEAR LOW BYTE
000774' 017767 177250 177102         MOV     @SBR,ASTAT              ;READ MNCDI STIMULUS BIT REGISTER
001002' 026767 177074 177074         CMP     ACSR,ASTAT              ;COMPARE
001010' 001403                       BEQ     3$                      ;BR IF SAME
                             ;*****************************************************************
001012' 104405 000000' 000000        HRDER$,BEGIN,NULL       ;CLEARING LOW BYTE CHANGED HIGH BYTE
```

```
                                        ;*****************************************************************
001020'                             38:
001020' 104407  000000'             BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001024' 104407  000000'             BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001030'                             DI4:
                                    ;TEST THAT BIT1 OF MNCDI STATUS REGISTER IS READ-WRITE
001030' 012767  000002  177044      MOV     #BIT1,ACSR              ;LOAD EXPECTED
001036' 016777  177040  177174      MOV     ACSR,@ICSR              ;LOAD BIT1 INTO MNCDI STATUS REGISTER
001044' 017767  177170  177032      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
001052' 042767  000200  177024      BIC     #BIT7,ASTAT             ;CLEAR BIT 7
001060' 026767  177016  177016      CMP     ACSR,ASTAT              ;TEST THAT IT SET
001066' 001403                      BEQ     1$                      ;BR IF SAME
                                    ;*****************************************************************
001070' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT1 OF MNCDI STATUS REGISTER FAILED TO SET
                                    ;*****************************************************************
001076' 046777  177000  177134  1$: BIC     ACSR,@ICSR              ;CLEAR THAT BIT
001104' 017767  177130  176772      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
001112' 026767  176764  176764      CMP     ACSR,ASTAT              ;TEST THE BIT
001120' 001003                      BNE     2$                      ;BR IF CLEARED
                                    ;*****************************************************************
001122' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT1 OF MNCDI STATUS REGISTER FAILED TO CLEAR
                                    ;*****************************************************************
001130'                             2$:
001130' 104407  000000'             BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001134' 104407  000000'             BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001140'                             DI5:
                                    ;TEST THAT BIT2 OF MNCDI STATUS REGISTER IS READ-WRITE
001140' 012767  000004  176734      MOV     #BIT2,ACSR              ;LOAD EXPECTED
001146' 016777  176730  177064      MOV     ACSR,@ICSR              ;LOAD BIT2 INTO MNCDI STATUS REGISTER
001154' 017767  177060  176722      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
001162' 042767  000200  176714      BIC     #BIT7,ASTAT             ;CLEAR BIT 7
001170' 026767  176706  176706      CMP     ACSR,ASTAT              ;TEST THAT IT SET
001176' 001403                      BEQ     1$                      ;BR IF SAME
                                    ;*****************************************************************
001200' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT2 OF MNCDI STATUS REGISTER FAILED TO SET
                                    ;*****************************************************************
001206' 046777  176670  177024  1$: BIC     ACSR,@ICSR              ;CLEAR THAT BIT
001214' 017767  177020  176662      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
001222' 026767  176654  176654      CMP     ACSR,ASTAT              ;TEST THE BIT
001230' 001003                      BNE     2$                      ;BR IF CLEARED
                                    ;*****************************************************************
001232' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT2 OF MNCDI STATUS REGISTER FAILED TO CLEAR
                                    ;*****************************************************************
001240'                             2$:
001240' 104407  000000'             BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001244' 104407  000000'             BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```



```
001250'                             DI6:
                                    ;TEST THAT BIT3 OF MNCDI STATUS REGISTER IS READ-WRITE
001250' 012767  000010  176624      MOV     #BIT3,ACSR              ;LOAD EXPECTED
001256' 016777  176620  176754      MOV     ACSR,@ICSR              ;LOAD BIT3 INTO MNCDI STATUS REGISTER
001264' 017767  176750  176612      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
001272' 042767  000200  176604      BIC     #BIT7,ASTAT             ;CLEAR BIT 7
001300' 026767  176576  176576      CMP     ACSR,ASTAT              ;TEST THAT IT SET
001306' 001403                      BEQ     1$                      ;BR IF SAME
                                    ;*****************************************************************
001310' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT3 OF MNCDI STATUS REGISTER FAILED TO SET
                                    ;*****************************************************************
001316' 046777  176560  176714  1$: BIC     ACSR,@ICSR              ;CLEAR THAT BIT
001324' 017767  176710  176552      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
001332' 026767  176544  176544      CMP     ACSR,ASTAT              ;TEST THE BIT
001340' 001003                      BNE     2$                      ;BR IF CLEARED
                                    ;*****************************************************************
001342' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT3 OF MNCDI STATUS REGISTER FAILED TO CLEAR
                                    ;*****************************************************************
001350'                             2$:
001350' 104407  000000'             BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001354' 104407  000000'             BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001360'                             DI7:
                                    ;TEST THAT BIT4 OF MNCDI STATUS REGISTER IS READ-WRITE
001360' 012767  000020  176514      MOV     #BIT4,ACSR              ;LOAD EXPECTED
001366' 016777  176510  176644      MOV     ACSR,@ICSR              ;LOAD BIT4 INTO MNCDI STATUS REGISTER
001374' 017767  176640  176502      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
001402' 042767  000200  176474      BIC     #BIT7,ASTAT             ;CLEAR BIT 7
001410' 026767  176466  176466      CMP     ACSR,ASTAT              ;TEST THAT IT SET
001416' 001403                      BEQ     1$                      ;BR IF SAME
                                    ;*****************************************************************
001420' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT4 OF MNCDI STATUS REGISTER FAILED TO SET
                                    ;*****************************************************************
001426' 046777  176450  176604  1$: BIC     ACSR,@ICSR              ;CLEAR THAT BIT
001434' 017767  176600  176442      MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
001442' 026767  176434  176434      CMP     ACSR,ASTAT              ;TEST THE BIT
001450' 001003                      BNE     2$                      ;BR IF CLEARED
                                    ;*****************************************************************
001452' 104405  000000' 000000      HRDER$,BEGIN,NULL       ;BIT4 OF MNCDI STATUS REGISTER FAILED TO CLEAR
                                    ;*****************************************************************
001460'                             2$:
001460' 104407  000000'             BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001464' 104407  000000'             BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```

```
001470'                         DI10:
                        ;TEST THAT BIT5 OF MNCDI STATUS REGISTER IS READ-WRITE
001470' 012767  000040  176404          MOV     #BIT5,ACSR              ;LOAD EXPECTED
001476' 016777  176400  176534          MOV     ACSR,@ICSR              ;LOAD BIT5 INTO MNCDI STATUS REGISTER
001504' 017767  176530  176372          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
001512' 042767  000200  176364          BIC     #BIT7,ASTAT             ;CLEAR BIT 7
001520' 026767  176356  176356          CMP     ACSR,ASTAT              ;TEST THAT IT SET
001526' 001403                          BEQ     1$                      ;BR IF SAME
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT5 OF MNCDI STATUS REGISTER FAILED TO SET
                        ;*************************************************************
001530' 104405  000000' 000000  1$:     BIC     ACSR,@ICSR              ;CLEAR THAT BIT
001536' 046777  176340  176474          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
001544' 017767  176470  176332          CMP     ACSR,ASTAT              ;TEST THE BIT
001552' 026767  176324  176324          BNE     2$                      ;BR IF CLEARED
001560' 001003
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT5 OF MNCDI STATUS REGISTER FAILED TO CLEAR
001562' 104405  000000' 000000  ;*************************************************************

001570'                         2$:
001570' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001574' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001600'                         DI11:
                        ;TEST THAT BIT6 OF MNCDI STATUS REGISTER IS READ-WRITE
001600' 012767  000100  176274          MOV     #BIT6,ACSR              ;LOAD EXPECTED
001606' 016777  176270  176424          MOV     ACSR,@ICSR              ;LOAD BIT6 INTO MNCDI STATUS REGISTER
001614' 017767  176420  176262          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
001622' 042767  000200  176254          BIC     #BIT7,ASTAT             ;CLEAR BIT 7
001630' 026767  176246  176246          CMP     ACSR,ASTAT              ;TEST THAT IT SET
001636' 001403                          BEQ     1$                      ;BR IF SAME
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT6 OF MNCDI STATUS REGISTER FAILED TO SET
001640' 104405  000000' 000000  ;*************************************************************

001646' 046777  176230  176364  1$:     BIC     ACSR,@ICSR              ;CLEAR THAT BIT
001654' 017767  176360  176222          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
001662' 026767  176214  176214          CMP     ACSR,ASTAT              ;TEST THE BIT
001670' 001003                          BNE     2$                      ;BR IF CLEARED
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT6 OF MNCDI STATUS REGISTER FAILED TO CLEAR
001672' 104405  000000' 000000  ;*************************************************************

001700'                         2$:
001700' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001704' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```



```
001710'                         DI12:
                        ;TEST THAT BIT8 OF MNCDI STATUS REGISTER IS READ-WRITE
001710' 012767  000400  176164          MOV     #BIT8,ACSR              ;LOAD EXPECTED
001716' 016777  176160  176314          MOV     ACSR,@ICSR              ;LOAD BIT8 INTO MNCDI STATUS REGISTER
001724' 017767  176310  176152          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
001732' 042767  000200  176144          BIC     #BIT7,ASTAT             ;CLEAR BIT 7
001740' 026767  176136  176136          CMP     ACSR,ASTAT              ;TEST THAT IT SET
001746' 001403                          BEQ     1$                      ;BR IF SAME
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT8 OF MNCDI STATUS REGISTER FAILED TO SET
001750' 104405  000000' 000000  ;*************************************************************

001756' 046777  176120  176254  1$:     BIC     ACSR,@ICSR              ;CLEAR THAT BIT
001764' 017767  176250  176112          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
001772' 026767  176104  176104          CMP     ACSR,ASTAT              ;TEST THE BIT
002000' 001003                          BNE     2$                      ;BR IF CLEARED
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT8 OF MNCDI STATUS REGISTER FAILED TO CLEAR
002002' 104405  000000' 000000  ;*************************************************************

002010'                         2$:
002010' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
002014' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002020'                         DI13:
                        ;TEST THAT BIT9 OF MNCDI STATUS REGISTER IS READ-WRITE
002020' 012767  001000  176054          MOV     #BIT9,ACSR              ;LOAD EXPECTED
002026' 016777  176050  176204          MOV     ACSR,@ICSR              ;LOAD BIT9 INTO MNCDI STATUS REGISTER
002034' 017767  176200  176042          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
002042' 042767  000200  176034          BIC     #BIT7,ASTAT             ;CLEAR BIT 7
002050' 026767  176026  176026          CMP     ACSR,ASTAT              ;TEST THAT IT SET
002056' 001403                          BEQ     1$                      ;BR IF SAME
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT9 OF MNCDI STATUS REGISTER FAILED TO SET
002060' 104405  000000' 000000  ;*************************************************************

002066' 046777  176010  176144  1$:     BIC     ACSR,@ICSR              ;CLEAR THAT BIT
002074' 017767  176140  176002          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
002102' 026767  175774  175774          CMP     ACSR,ASTAT              ;TEST THE BIT
002110' 001003                          BNE     2$                      ;BR IF CLEARED
                        ;*************************************************************
                        HRDER$,BEGIN,NULL       ;BIT9 OF MNCDI STATUS REGISTER FAILED TO CLEAR
002112' 104405  000000' 000000  ;*************************************************************

002120'                         2$:
002120' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
002124' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```

```
  526  002130'                          DI14:
  527
  528  002130' 012767  010000  175744          MOV     #BIT12,ACSR             ;LOAD EXPECTED
  529  002136' 016777  175740  176074          MOV     ACSR,@ICSR              ;LOAD BIT12 INTO MNCDI STATUS REGISTER
  530  002144' 017767  176070  175732          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
  531  002152' 042767  000200  175724          BIC     #BIT7,ASTAT             ;CLEAR BIT 7
  532  002160' 026767  175716  175716          CMP     ACSR,ASTAT              ;TEST THAT IT SET
  533  002166' 001403                          BEQ     1$                      ;BR IF SAME
  534                                          ;*****************************************************************
  535  002170' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT12 OF MNCDI STATUS REGISTER FAILED TO SET
  536                                          ;*****************************************************************
  537  002176' 046777  175700  176034  1$:     BIC     ACSR,@ICSR              ;CLEAR THAT BIT
  538  002204' 017767  176030  175672          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
  539  002212' 026767  175664  175664          CMP     ACSR,ASTAT              ;TEST THE BIT
  540  002220' 001003                          BNE     2$                      ;BR IF CLEARED
  541                                          ;*****************************************************************
  542  002222' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT12 OF MNCDI STATUS REGISTER FAILED TO CLEAR
  543                                          ;*****************************************************************
  544  002230'                          2$:
  545  002230' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
  546  002234' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
  547  002240'                          DI15:
  548                                          ;TEST THAT BIT14 OF MNCDI STATUS REGISTER IS READ-WRITE
  549  002240' 012767  040000  175634          MOV     #BIT14,ACSR             ;LOAD EXPECTED
  550  002246' 016777  175630  175764          MOV     ACSR,@ICSR              ;LOAD BIT14 INTO MNCDI STATUS REGISTER
  551  002254' 017767  175760  175622          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER
  552  002262' 042767  000200  175614          BIC     #BIT7,ASTAT             ;CLEAR BIT 7
  553  002270' 026767  175606  175606          CMP     ACSR,ASTAT              ;TEST THAT IT SET
  554  002276' 001403                          BEQ     1$                      ;BR IF SAME
  555                                          ;*****************************************************************
  556  002300' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT14 OF MNCDI STATUS REGISTER FAILED TO SET
  557                                          ;*****************************************************************
  558  002306' 046777  175570  175724  1$:     BIC     ACSR,@ICSR              ;CLEAR THAT BIT
  559  002314' 017767  175720  175562          MOV     @ICSR,ASTAT             ;READ MNCDI STATUS REGISTER AGAIN
  560  002322' 026767  175554  175554          CMP     ACSR,ASTAT              ;TEST THE BIT
  561  002330' 001003                          BNE     2$                      ;BR IF CLEARED
  562                                          ;*****************************************************************
  563  002332' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;BIT14 OF MNCDI STATUS REGISTER FAILED TO CLEAR
  564                                          ;*****************************************************************
  565  002340'                          2$:
  566  002340' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
  567  002344' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```



```
  568
  569  002350' 012777  040424  175662  DI16:   MOV     #40424,@ICSR            ;LOAD INPUT REG. BIT
  570  002356' 105077  175660                  CLRB    #ICSR1                  ;CLEAR HIGH BYTE
  571  002362' 012767  000024  175512          MOV     #BIT4!BIT2,ACSR         ;LOAD EXPECTED
  572  002370' 017767  175644  175506          MOV     #ICSR,ASTAT             ;READ INPUT STATUS REG.
  573  002376' 042767  000200  175500          BIC     #BIT7,ASTAT             ;REMOVE BIT 7
  574  002404' 026767  175472  175472          CMP     ACSR,ASTAT              ;COMPARE
  575  002412' 001403                          BEQ     DI17
  576                                          ;*****************************************************************
  577  002414' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLEARING HIGH BYTE CHANGED LOW BYTE
  578                                          ;*****************************************************************
  579
  580                                          ;VERIFY LOW BYTE OPERATION ON THE INPUT STATUS REGISTER
  581  002422' 012777  040426  175610  DI17:   MOV     #40426,@ICSR            ;LOAD INPUT REG.
  582  002430' 105077  175604                  CLRB    @ICSR                   ;CLEAR LOW BYTE
  583  002434' 012767  040400  175440          MOV     #40400,ACSR             ;LOAD EXPECTED
  584  002442' 017767  175572  175434          MOV     @ICSR,ASTAT             ;READ INPUT STATUS REG.
  585  002450' 026767  175426  175426          CMP     ACSR,ASTAT              ;COMPARE
  586  002456' 001403                          BEQ     DI20                    ;BR IF SAME
  587                                          ;*****************************************************************
  588  002460' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLEARING LOW BYTE CHANGED HIGH BYTE
  589                                          ;*****************************************************************
  590
  591                                          ;VERIFY THAT MAINT. STROBE SETS "INPUT DATA READY"
  592  002466' 005077  175546          DI20:   CLR     @ICSR                   ;ENSURE CLEAR FLAG
  593  002472' 012767  000200  175402          MOV     #BIT7,ACSR              ;LOAD EXPECTED DATA
  594  002500' 012777  004200  175532          MOV     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
  595  002506' 017767  175526  175370          MOV     @ICSR,ASTAT             ;READ INPUT STATUS REGISTER
  596  002514' 026767  175362  175362          CMP     ACSR,ASTAT              ;COMPARE RESULTS
  597  002522' 001403                          BEQ     DI21                    ;BR IF SAME
  598                                          ;*****************************************************************
  599  002524' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;MAINT. STROBE FAILED TO SET "INPUT DATA READY"
  600                                          ;*****************************************************************
```

```
                                        ;VERIFY THAT "INPUT DATA READY" CAN BE WRITTEN TO A ZERO
002532' 005067  175344          DI21:   CLR     ACSR                    ;LOAD EXPECTED DATA
002536' 012777  004000  175474          MOV     #RITEXT,@ICSR           ;GENERATE MAINT. STROBE
002544' 005077  175470                  CLR     @ICSR                   ;CLEAR DATA READY FLAG
002550' 017767  175464  175326          MOV     @ICSR,ASTAT             ;READ INPUT STATUS REGISTER
002556' 026767  175320  175320          CMP     ACSR,ASTAT              ;COMPARE
002564' 001403                          BEQ     DI22                    ;BR IF SAME
                                        ;*****************************************************************
002566' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;"INPUT DATA READY" FAILED TO BE WRITTEN TO A ZERO
                                        ;*****************************************************************


                                        ;VERIFY THAT "INPUT DATA READY" WILL NOT SET IF IN STIMILUS MODE AND NO SBR MATC
002574' 005077  175450          DI22:   CLR     @SBR                    ;CLEAR SBR REGISTER
002600' 012777  177777  175436          MOV     #-1,@DIR                ;CLEAR INPUT REGISTER
002606' 012777  000004  175424          MOV     #RIT2,@ICSR             ;SET STILILUS MODE
002614' 052777  004000  175416          BIS     #RITEXT,@ICSR           ;GENERATE MAINT. STROBE
002622' 012767  000004  175252          MOV     #RIT2,ACSR              ;LOAD EXPECTED
002630' 017767  175404  175246          MOV     @ICSR,ASTAT             ;READ STATUS
002636' 026767  175240  175240          CMP     ACSR,ASTAT              ;COMPARE
002644' 001403                          BEQ     1$                      ;BR IF SAME
                                        ;*****************************************************************
002646' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INPUT STROBE SET INPUT READY WHEN IN STIMILUS MODE
                                        ;*****************************************************************

002654'                         1$:
002654' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
002660' 104407  000000'                 BREAK$,BFGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
                                        ;VERIFY THAT "OVERRUN ERROR" SETS
002664' 012767  100202  175210  DI23:   MOV     #RIT15!BIT7!BIT1,ACSR   ;LOAD EXPECTED
002672' 012777  000002  175340          MOV     #RIT1,@ICSR             ;SET STROBE MODE
002700' 052777  004200  175332          BIS     #RITEXT!BIT7,@ICSR      ;GENERATE MAINT, STROBE
002706' 052777  104200  175324          BIS     #RIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE AGAIN
002714' 017767  175320  175162          MOV     @ICSR,ASTAT             ;READ INPUT STATUS REGISTER
002722' 026767  175154  175154          CMP     ACSR,ASTAT              ;COMPARE
002730' 001403                          BEQ     DI24                    ;BR IF SAME
                                        ;*****************************************************************
002732' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;"OVER RUN" FAILED TO SET
                                        ;*****************************************************************


                                        ;VERIFY THAT "OVERRUN ERROR" CAN BE WRITTEN TO A ZERO
002740' 012767  000202  175134  DI24:   MOV     #RIT7!BIT1,ACSR ;LOAD EXPECTED VALUE
002746' 012777  000002  175264          MOV     #BIT1,@ICSR             ;SET STROBE MODE
002754' 052777  004200  175256          BIS     #RITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
002762' 052777  104200  175250          BIS     #RIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE AGAN
002770' 105077  175246                  CLRB    @ICSR1                  ;CLEAR HIGH BYTE OF THE INPUT STATUS REGISTER
002774' 017767  175240  175102          MOV     @ICSR,ASTAT             ;READ INPUT STATUS REGISTER
003002' 026767  175074  175074          CMP     ACSR,ASTAT              ;COMPARE
003010' 001403                          BEQ     DI25                    ;BR IF SAME
                                        ;*****************************************************************
003012' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;"OVERRUN ERROR" FAILED TO BE WRITTEN TO A ZERO
                                        ;*****************************************************************


```



```
                                        ;VERIFY INVERT DATA FUNCTION
003020' 005067  175056          DI25:   CLR     ACSR                    ;LOADEXPECTED
003024' 012777  010000  175206          MOV     #RITDAT,@ICSR           ;SET INPUT INHIBIT
003032' 017767  175206  175044          MOV     @DIR,ASTAT              ;READ INPUT
003040' 026767  175036  175036          CMP     ACSR,ASTAT              ;COMPARE
003046' 001403                          BEQ     1$                      ;BR IF SAME
                                        ;*****************************************************************
003050' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INPUT INHIBIT FAILED TO INHIBIT INPUT
                                        ;*****************************************************************
003056' 012777  010060  175154  1$:     MOV     #RITDAT!BIT5!BIT4,@ICSR ;SET INVERT DATA AND INPUT INHIBIT
003064' 012767  177777  175010          MOV     #-1,ACSR                ;LOAD EXPECTED
003072' 017767  175146  175004          MOV     @DIR,ASTAT              ;READ INPUT
003100' 001003                          BNE     2$                      ;BR IF NON-ZERO
                                        ;*****************************************************************
003102' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INVERT DATA FUNCTION FAILED
                                        ;*****************************************************************
003110' 026767  174766  174766  2$:     CMP     ACSR,ASTAT              ;COMPARE DATA
003116' 001403                          BEQ     3$                      ;BR IF SAME
                                        ;*****************************************************************
003120' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INVERT DATA - DATA PATH ERROR
                                        ;*****************************************************************


003126' 012777  010000  175104  3$:     MOV     #BITDAT,@ICSR           ;SET INPUT INHIBIT
003134' 005067  174742                  CLR     ACSR                    ;CLEAR EXPECTED
003140' 017767  175100  174736          MOV     @DIR,ASTAT              ;READ INPUT
003146' 026767  174730  174730          CMP     ACSR,ASTAT              ;COMPARE
003154' 001403                          BEQ     DI26                    ;BR IF SAME
                                        ;*****************************************************************
003156' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INVERT DATA FUNCTION OR INPUT INHIBIT FAILED
                                        ;*****************************************************************


                                        ;VERIFY EACH BIT OF THE MNCDI INPUT DATA REGISTER CAN BE CLEARED
003164' 012767  000001  175100  DI26:   MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
003172' 012777  010060  175040  1$:     MOV     #RITDAT!BIT4!BIT5,@ICSR ;LOAD INHIBIT INPUT AND INVERT DATA
003200' 016767  175066  174674          MOV     TEMP,ACSR               ;LOAD EXPECTED
003206' 005167  174670                  COM     ACSR                    ;MAKE OPPOSITE
003212' 017700  175026                  MOV     @DIR,R0                 ;READ INPUT
003216' 016777  175050  175020          MOV     TEMP,@DIR               ;CLEAR THE INPUT BIT
003224' 042777  000040  175006          BIC     #BIT5,@ICSR             ;REM INVERT DATA BITOVE
003232' 052777  000002  175000          BIS     #BIT1,@ICSR             ;ENABLE EXT, STROBE TO PREVENT  DATA INPUT BEING
003240' 017767  175000  174636          MOV     @DIR,ASTAT              ;READ INPUT REG.
003246' 026767  174630  174630          CMP     ACSR,ASTAT              ;COMPARE
003254' 001403                          BEQ     2$                      ;BR IF SAME
                                        ;*****************************************************************
003256' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INPUT REGISTER BIT FAILED TO CLEAR
                                        ;*****************************************************************

003264'                         2$:
003264' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
003270' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
003274' 006367  174772                  ASL     TEMP                    ;SHIFT THE DATA
003300' 001334                          BNE     1$                      ;TRY MORE BITS
```

```
                                        ;VERIFY THAT A 2ND STROBE PULSE WILL NOT CHANGE THE DIR DATA
003302' 012777  010062  174730  DI27:   MOV     #BIT12!BIT5!BIT4!BIT1,@ICSR     ;DISABLE INPUTS, ENABLE INVERT DATA, EXT
003310' 052777  004200  174722          BIS     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
003316' 042777  000040  174714          BIC     #BIT5,@ICSR             ;REMOVE INVERT DATA
003324' 052777  004200  174706          BIS     #BITEXT!BIT7,@ICSR      ;SET MAINT. STROBE AGAIN
003332' 012767  177777  174542          MOV     #-1,ACSR                ;LOAD EXPECTED DATA
003340' 017767  174700  174536          MOV     @DIR,ASTAT              ;READ REGISTER
003346' 026767  174530  174530          CMP     ACSR,ASTAT              ;COMPARE
003354' 001403                          BEQ     DI30                    ;BR IF SAME
                                        ;*****************************************************************
003356' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;DATA READY FAILED TO INHIBIT 2ND
                                        ;*****************************************************************
                                                                ;STROBE FROM CHAINING THE DIR
                                        ;INTERRUPT TEST -- VERIFY MNCDI INTERRUPTS VIA DATA READY VECTOR
003364' 012777  003430' 174666  DI30:   MOV     #1$,@DIDINV             ;LOAD RETURN VECTOR
003372' 116777  174414  174662          MOVB    BR1,@DIDIN$             ;LOAD RETURN LEVEL
003400' 012777  000102  174632          MOV     #BIT6!BIT1,@ICSR        ;SET STROBE MODE
003406' 052777  004200  174624          BIS     #BITEXT!BIT7,@ICSR      ;GENERATE MAINT. STROBE
003414' 000240                          NOP
003416' 000240                          NOP
003420' 000240                          NOP
003422' 000240                          NOP
003424' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.

003430' 005077  174604          1$:     CLR     @ICSR
                                        ;-----------------------------------------------------------------
003434' 000004  000000' 003442'         PIRQ$,BEGIN,2$          ; QUEUE UP TO CONTINUE AT 2$ AND RTI
                                        ;-----------------------------------------------------------------
003442' 012777  000262' 174610  2$:     MOV     #DIDIN$,@DIDINV
003450' 005077  174606                  CLR     @DIDIN$

                                        ;INTERRUPT TEST -- VERIFY MNCDI INTERRUPTS VIA OVERRUN ERROR
003454' 012777  003524' 174602  DI31:   MOV     #1$,@DIEINV             ;LOAD RETURN VECTOR
003462' 116777  174324  174576          MOVB    BR1,@DIEIN$             ;LOAD RETURN STATUS
003470' 012777  040002  174542          MOV     #BIT14!BIT1,@ICSR       ;ENABLE INTR. AND STROBE MODE
003476' 052777  104200  174534          BIS     #BIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE
003504' 052777  104200  174526          BIS     #BIT15!BITEXT!BIT7,@ICSR        ;GENERATE MAINT. STROBE AGAIN TO SET OVE
003512' 000240                          NOP
003514' 000240                          NOP
003516' 000240                          NOP
003520' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
003524' 005077  174510          1$:     CLR     @ICSR
                                        ;-----------------------------------------------------------------
003530' 000004  000000' 003536'         PIRQ$,BEGIN,2$          ; QUEUE UP TO CONTINUE AT 2$ AND RTI
                                        ;-----------------------------------------------------------------
003536' 012777  000266' 174520  2$:     MOV     #DIEIN$,@DIEINV
003544' 005077  174516                  CLR     @DIEIN$
003550' 036767  174522  174512          BIT     TEMP2,SRMINE            ;TEST IF INPUT ONLY
003556' 001002                          BNE     DO0                     ;BR IF WRAP-AROUND MODE
003560' 000167  174634                  JMP     CONT2                   ;NO -- TRY NEXT UNIT
```



```
                                        ;VERIFY CORRECT MNCDO ADDRESS RESPONSE
003564' 016767  174440  174306  DO0:    MOV     OCSR,CSRA               ;LOAD BUS ADDRESS
003572' 005777  174432                  TST     @OCSR                   ;TEST OUTPUT STATUS REGISTER
003576' 005777  174432                  TST     @DOR                    ;TEST OUTPUT DATA REGISTER
003602'                         DO1:
                                        ;FLOAT A 1 ACROSS THE MNCDO DATA REGISTER
003602' 012767  000001  174272          MOV     #BIT0,ACSR              ;LOAD EXPECT BIT
003610' 016777  174266  174416  1$:     MOV     ACSR,@DOR               ;LOAD MNCDO DATA REGISTER
003616' 017767  174412  174260          MOV     @DOR,ASTAT              ;READ MNCDO DATA REGISTER
003624' 026767  174252  174252          CMP     ACSR,ASTAT              ;COMPARE
003632' 001403                          BEQ     2$                      ;BR IF SAME
                                        ;*****************************************************************
003634' 104405  000000' 000000          HRDER$,BEGIN,NULL       ; MNCDO DATA REGISTER FAILED TO HOLD A FLOATING 1
                                        ;*****************************************************************
003642'                         2$:
003642' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
003646' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
003652' 006367  174224                  ASL     ACSR                    ;CHANGE THE DATA
003656' 001354                          BNE     1$                      ;BR IF MORE DATA
003660'                         DO2:
                                        ;FLOAT A 0 ACROSS THE MNCDO DATA REGISTER
003660' 012767  000001  174404          MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
003666' 016767  174400  174206  1$:     MOV     TEMP,ACSR               ;LOAD EXPECTED
003674' 005167  174202                  COM     ACSR                    ;COMPLEMENT
003700' 016777  174176  174326          MOV     ACSR,@DOR               ;LOAD MNCDO DATA REGISTER
003706' 017767  174322  174170          MOV     @DOR,ASTAT              ;READ MNCDO DATA REGISTER
003714' 026767  174162  174162          CMP     ACSR,ASTAT              ;COMPARE
003722' 001403                          BEQ     2$                      ;BR IF SAME
                                        ;*****************************************************************
003724' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;MNCDO DATA REGISTER FAILED TO HOLD A FLOATING 0
                                        ;*****************************************************************
003732'                         2$:
003732' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
003736' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
003742' 006367  174324                  ASL     TEMP                    ;CHANGE THE DATA
003746' 001347                          BNE     1$                      ;BR IF MORE DATA
003750'                         DO3:
                                        ;VERIFY BYTE OPERATION ON THE MNCDO DATA REGISTER
003750' 012777  177777  174256  1$:     MOV     #-1,@DOR                ;LOAD MNCDO DATA REGISTER
003756' 012767  000377  174116          MOV     #377,ACSR               ;LOAD EXPECTED
003764' 105077  174246                  CLRB    @DOR1                   ;CLEAR HIGH BYTE
003770' 017767  174240  174106          MOV     @DOR,ASTAT              ;READ MNCDO DATA REGISTER
003776' 026767  174100  174100          CMP     ACSR,ASTAT              ;COMPARE
004004' 001403                          BEQ     2$                      ;BR IF SAME
                                        ;*****************************************************************
004006' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLEARING HIGH BYTE CHANGED LOW BYTE
                                        ;*****************************************************************
004014' 012777  177777  174212  2$:     MOV     #-1,@DOR                ;LOAD MNCDO DATA REGISTER
004022' 012767  177400  174052          MOV     #177400,ACSR            ;LOAD EXPECTED
004030' 105077  174200                  CLRB    @DOR                    ;CLEAR LOW BYTE
004034' 017767  174174  174042          MOV     @DOR,ASTAT              ;READ MNCDO DATA REGISTER
004042' 026767  174034  174034          CMP     ACSR,ASTAT              ;COMPARE
004050' 001403                          BEQ     3$                      ;BR IF SAME
                                        ;*****************************************************************
004052' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLEARING LOW BYTE CHANGED HIGH BYTE
                                        ;*****************************************************************
```

```
                        3$:
                                BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
                                BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
                        DO4:
                                ;TEST THAT BIT6 OF MNCDO STATUS REGISTER IS READ-WRITE
                                MOV     #BIT6,ACSR              ;LOAD EXPECTED
                                MOV     ACSR,@OCSR              ;LOAD BIT6 INTO MNCDO STATUS REGISTER
                                MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER
                                BIC     #BIT7,ASTAT             ;CLEAR BIT 7
                                CMP     ACSR,ASTAT              ;TEST THAT IT SET
                                BEQ     1$                      ;BR IF SAME
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;BIT6 OF MNCDO STATUS REGISTER FAILED TO SET
                                ;*****************************************************************
                        1$:     BIC     ACSR,@OCSR              ;CLEAR THAT BIT
                                MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER AGAIN
                                CMP     ACSR,ASTAT              ;TEST THE BIT
                                BNE     2$                      ;BR IF CLEARED
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;BIT6 OF MNCDO STATUS REGISTER FAILED TO CLEAR
                                ;*****************************************************************
                        2$:
                                BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
                                BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
                        DO5:
                                ;TEST THAT BIT4 OF MNCDO STATUS REGISTER IS READ-WRITE
                                MOV     #BIT4,ACSR              ;LOAD EXPECTED
                                MOV     ACSR,@OCSR              ;LOAD BIT4 INTO MNCDO STATUS REGISTER
                                MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER
                                BIC     #BIT7,ASTAT             ;CLEAR BIT 7
                                CMP     ACSR,ASTAT              ;TEST THAT IT SET
                                BEQ     1$                      ;BR IF SAME
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;BIT4 OF MNCDO STATUS REGISTER FAILED TO SET
                                ;*****************************************************************
                        1$:     BIC     ACSR,@OCSR              ;CLEAR THAT BIT
                                MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER AGAIN
                                CMP     ACSR,ASTAT              ;TEST THE BIT
                                BNE     2$                      ;BR IF CLEARED
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;BIT4 OF MNCDO STATUS REGISTER FAILED TO CLEAR
                                ;*****************************************************************
                        2$:
                                BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
                                BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```



```
                        DO6:
                                ;TEST THAT BIT3 OF MNCDO STATUS REGISTER IS READ-WRITE
                                MOV     #BIT3,ACSR              ;LOAD EXPECTED
                                MOV     ACSR,@OCSR              ;LOAD BIT3 INTO MNCDO STATUS REGISTER
                                MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER
                                BIC     #BIT7,ASTAT             ;CLEAR BIT 7
                                CMP     ACSR,ASTAT              ;TEST THAT IT SET
                                BEQ     1$                      ;BR IF SAME
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;BIT3 OF MNCDO STATUS REGISTER FAILED TO SET
                                ;*****************************************************************
                        1$:     BIC     ACSR,@OCSR              ;CLEAR THAT BIT
                                MOV     @OCSR,ASTAT             ;READ MNCDO STATUS REGISTER AGAIN
                                CMP     ACSR,ASTAT              ;TEST THE BIT
                                BNE     2$                      ;BR IF CLEARED
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;BIT3 OF MNCDO STATUS REGISTER FAILED TO CLEAR
                                ;*****************************************************************
                        2$:
                                BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
                                BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
                                ;VERIFY THAT MNCDO DONE FLAG SETS
                        DO7:    CLR     @OCSR                   ;CLEAR CLEARED FLAG
                                MOV     #BIT7,ACSR              ;LOAD EXPECTED
                                CLRB    @DOR                    ;ENABLE
                                MOVB    #BIT0,@OCSR1            ;GENERATE MAINT, REPLY
                                MOV     @OCSR,ASTAT             ;READ OUTPUT STATUS REGISTER
                                CMP     ACSR,ASTAT              ;COMMPARE
                                BEQ     DO10                    ;BR IF SAME
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;OUTPUT DONE FLAG FAILED TO SET
                                ;*****************************************************************
                                ;VERIFY THAT MNCDO DONE FLAG CLEARS WHEN WRITTEN TO A 0
                        DO10:   CLRB    @DOR                    ;ENABLE
                                MOVB    #BIT0,@OCSR1            ;GENERATE MAINT, REPLY
                                CLR     ACSR                    ;CLEAR EXPECTED
                                CLR     @OCSR                   ;CLEAR OUTPUT DONE FLAG
                                MOV     @OCSR,ASTAT             ;READ STATUS
                                BEQ     DO11                    ;BR IF SAME
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;CLEARING THE OUTPUT FLAG FAILED TO CLEAR OUTPUT DONE FL
                                ;*****************************************************************
                                ;VERIFY THAT MNCDO DONE FLAG CLEARS WHEN OUTPUT DATA REGISTER IS WRITTEN
                        DO11:   CLRB    @DOR                    ;ENABLE
                                MOVB    #BIT0,@OCSR1            ;GENERATE MAINT, REPLY
                                CLR     ACSR                    ;CLEAR EXPECTED
                                CLR     @DOR                    ;WRITE THE OUTPUT DATA REGISTER
                                MOV     @OCSR,ASTAT             ;READ OUTPUT STATUS REGISTER
                                BEQ     DO12                    ;BR IF CLEARED
                                ;*****************************************************************
                                HRDER$,BEGIN,NULL       ;OUTPUT DONE FLAG FAILED TO CLEAR WHEN THE DOR REGISTER
                                ;*****************************************************************
```

```
                                        ;INTERRUPT TEST -- VERIFY MNCDO DOES INTERRUPT
004570' 012777  004636' 173456  DO12$:  MOV     #1$,@DODINV             ;LOAD RETURN VECTOR
004576' 116777  173210  173452          MOVB    BR1,@DODIN$             ;LOAD RETURN LEVEL
004604' 105077  173424                  CLRB    @DOR                    ;ENABLE
004610' 112777  000001  173414          MOVB    #BIT0,@OCSR1            ;GENERATE MAINT. REPLY
004616' 052777  000100  173404          BIS     #BIT6,@OCSR             ;ENABLE INTR.
004624' 000240                          NOP
004626' 000240                          NOP
004630' 000240                          NOP
004632' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.

004636' 005077  173366          1$:     CLR     @OCSR
                                        ;---------------------------------------------------------------
004642' 000004  000000' 004650'         PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3$ AND RTI
                                        ;---------------------------------------------------------------
004650' 016777  173402  173376  3$:     MOV     DODIN$,@DODINV          ;RESET VECTORS
004656' 005077  173374                  CLR     @DODIN$
004662' 005077  173342                  CLR     @OCSR
```



```
                                        ;WRAP AROUND TESTS

                                        ;VERFIY MODE 00  -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
004666' 012777  177777  173350  DI32$:  MOV     #-1,@DIR                ;CLEAR INPUT REG.
004674' 005077  173350                  CLR     @SBR                    ;CLEAR STIM. REG.
004700' 005077  173334                  CLR     @ICSR                   ;CLEAR INPUT DATA READY FLAG
004704' 005077  173320                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
004710' 012777  025252  173316          MOV     #25252,@DOR             ;WRITE TO THE OUTPUT DATA REG.
004716' 012767  000200  173156          MOV     #BIT7,ACSR              ;LOAD EXPECTED
004724' 017767  173310  173152          MOV     @ICSR,ASTAT             ;READ STATUS
004732' 026767  173144  173144          CMP     ACSR,ASTAT              ;COMPARE
004740' 001406                          BEQ     DI33                    ;BR IF SAME
                                        ;*****************************************************************
004742' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;MODE 00  -- EXT, STROBE FAILED TO SET INPUT DATA READY
                                        ;*****************************************************************
004750' 104403  000000' 006030'         MSGN$,BEGIN,MSG1        ;ASCII MESSAGE CALL WITH COMMON HEADER

                                        ;VERIFY MODE 01  -- INPUT STROBE WILL SET THE INPUT DATA READY FLAG
004756' 012777  177777  173260  DI33$:  MOV     #-1,@DIR                ;CLEAR INPUT REG.
004764' 005077  173260                  CLR     @SBR                    ;CLEAR STIM. REG.
004770' 012777  000002  173242          MOV     #BIT1,@ICSR             ;CLEAR INPUT DATA READY FLAG
004776' 005077  173226                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
005002' 012777  052525  173224          MOV     #52525,@DOR             ;WRITE TO THE OUTPUT DATA REG.
005010' 012767  000202  173064          MOV     #BIT7!BIT1,ACSR ;LOAD EXPECTED
005016' 017767  173216  173060          MOV     @ICSR,ASTAT             ;READ STATUS
005024' 026767  173052  173052          CMP     ACSR,ASTAT              ;COMPARE
005032' 001403                          BEQ     DI34                    ;BR IF SAME
                                        ;*****************************************************************
005034' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;MODE 01  -- EXT, STROBE FAILED TO SET INPUT DATA READY
                                        ;*****************************************************************

                                        ;VERIFY MODE 10  -- INPUT STROBE WILL NOT SET INPUT DATA READY FLAG
005042' 012777  177777  173174  DI34$:  MOV     #-1,@DIR                ;CLEAR INPUT REG.
005050' 005077  173174                  CLR     @SBR                    ;CLEAR STIM. REG.
005054' 012777  000004  173156          MOV     #BIT2,@ICSR             ;CLEAR INPUT DATA READY FLAG
005062' 005077  173142                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
005066' 012777  070707  173140          MOV     #70707,@DOR             ;WRITE TO THE OUTPUT DATA REG.
005074' 012767  000004  173000          MOV     #BIT2,ACSR              ;LOAD EXPECTED
005102' 017767  173132  172774          MOV     @ICSR,ASTAT             ;READ STATUS
005110' 026767  172766  172766          CMP     ACSR,ASTAT              ;COMPARE
005116' 001403                          BEQ     DI35                    ;BR IF SAME
                                        ;*****************************************************************
005120' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;MODE 10  -- EXT, STROBE SET INPUT DATA READY FLAG IN ER
                                        ;*****************************************************************
```

```
                                        ;VERIFY INPUT REPLY SETS OUTPUT DONE FLAG
005126' 005077  173102          DI35:   CLR     @DOR                    ;CLEAR OUTPUT DATA
005132' 012767  000200  172742          MOV     #BIT7,ACSR              ;LOAD EXPECTED
005140' 005077  173064                  CLR     @OCSR                   ;CLEAR OUTPUT DONE FLAG
005144' 012777  004200  173066          MOV     #BITEXT!BIT7,@ICSR      ;SET INPUT READY FLAG
005152' 005077  173062                  CLR     @ICSR                   ;CLEAR INPUT READY FLAG<GEN, INPUT REPLY>
005156' 017767  173046  172720          MOV     @OCSR,ASTAT             ;READ OUTPUT STATUS
005164' 026767  172712  172712          CMP     ACSR,ASTAT              ;COMPARE
005172' 001403                          BEQ     DI36                    ;BR IF SAME
                                        ;*************************************************************
005174' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INPUT REPLY FAILED TO SET OUTPUT DONE FLAG
                                        ;*************************************************************
                                        ;VERIFY THE MNCDO - WRAPAROUND - MNCDI DATA PATH
005202' 012777  177777  173034  DI36:   MOV     #-1,@DIR                ;CLEAR INPUT REG.
005210' 005077  173014                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
005214' 005077  173020                  CLR     @ICSR                   ;CLEAR INPUT STATUS
005220' 012767  000001  173044          MOV     #BIT0,TEMP              ;LOAD EXPECTED
005226' 016767  173040  172646  1$:     MOV     TEMP,ACSR               ;LOAD TYPEOUT EXPECTED
005234' 016777  172642  172772          MOV     ACSR,@DOR               ;LOAD OUTPUT DATA REG.
005242' 017767  172776  172634          MOV     @DIR,ASTAT              ;READ INPUT DATA REGISTER
005250' 026767  172626  172626          CMP     ACSR,ASTAT              ;COMPARE
005256' 001403                          BEQ     2$                      ;BR IF SAME
                                        ;*************************************************************
005260' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INPUT DATA PATH ERROR
                                        ;*************************************************************
005266'                         2$:
005266' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
005272' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005276' 006367  172770                  ASL     TEMP                    ;TRY NEXT BIT
005302' 001351                          BNE     1$                      ;BR IF MORE BITS
                                        ;VERIFY THE MNCDO - WRAPAROUND - MNCDI INVERTED DATA PATH
005304' 012777  177777  172732  DI37:   MOV     #-1,@DIR                ;CLEAR INPUT REG.
005312' 005077  172712                  CLR     @OCSR                   ;CLEAR OUTPUT STATUS
005316' 012777  000060  172714          MOV     #BIT5!BIT4,@ICSR        ;LOAD INPUT STATUS <INVERT DATA>
005324' 012767  000001  172740          MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
005332' 016777  172734  172674  1$:     MOV     TEMP,@DOR               ;LOAD OUTPUT DATA REG.
005340' 017767  172700  172536          MOV     @DIR,ASTAT              ;READ INPUT DATA REGISTER
005346' 016767  172720  172526          MOV     TEMP,ACSR               ;GET THE BIT
005354' 005167  172522                  COM     ACSR                    ;INVERT EXPECTED INPUT DATA
005360' 026767  172516  172516          CMP     ACSR,ASTAT              ;COMPARE
005366' 001403                          BEQ     2$                      ;BR IF SAME
                                        ;*************************************************************
005370' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INVERTED INPUT DATA PATH ERROR
                                        ;*************************************************************
005376'                         2$:
005376' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
005402' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005406' 006367  172660                  ASL     TEMP                    ;TRY NEXT BIT
005412' 001347                          BNE     1$                      ;BR IF MORE BITS
```



```
                                        ;VERIFY IN 10 MODE THAT SBR AND INPUT  BITS SET INPUT READY
                                ;       INPUT DATA READY FLAG
                                ;               LOAD A FLOATING 1 ACROSS THE SBR
                                ;               VERIFY THAT ONLY THE CORRECT BIT SET DATA READY
005414' 005077  172614          DI40:   CLR     @DOR
005420' 012767  000001  172644          MOV     #BIT0,TEMP              ;LOAD INITIAL BIT
005426' 005077  172602          1$:     CLR     @DOR                    ;CLEAR OUTPUT BITS
005432' 005077  172612                  CLR     @SBR                    ;CLEAR SBR REG.
005436' 012777  000004  172574          MOV     #BIT2,@ICSR             ;CLEAR INPUT READY AND SET MODE 10
005444' 012777  177777  172572          MOV     #-1,@DIR                ;CLEAR INPUT REG.
005452' 016777  172614  172570          MOV     TEMP,@SBR               ;LOAD SBR REG.
005460' 042777  000200  172552          BIC     #BIT7,@ICSR             ;CLEAR INPUT READY BIT
005466' 016777  172600  172540          MOV     TEMP,@DOR               ;LOAD OUTPUT REG.
005474' 012767  100204  172400          MOV     #BIT15!BIT7!BIT2,ACSR   ;LOAD EXPECTED STATUS
005502' 017767  172532  172374          MOV     @ICSR,ASTAT             ;READ STATUS
005510' 026767  172366  172366          CMP     ACSR,ASTAT              ;COMPARE
005516' 001403                          BEQ     2$                      ;BR IF SAME
                                        ;*************************************************************
005520' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INPUT DATA READY FLAG FAILED
                                        ;*************************************************************
                                                                ;TO SET IN MODE 10 <STIMILUS MODE>

                                        ;NOW LOAD ALL BITS EXCEPT THE FLOATING BIT AND ENSURE INPUT DATA READY DOES NOT SET
005526' 016767  172540  172540  2$:     MOV     TEMP,TEMP1              ;COPY EXPECTED
005534' 005167  172534                  COM     TEMP1                   ;USE REVERSE PATTERN
005540' 005077  172470                  CLR     @DOR                    ;CLEAR OUTPUT REG.
005544' 012777  177777  172472          MOV     #-1,@DIR                ;CLEAR INPUT REG.
005552' 042777  100200  172460          BIC     #BIT15!BIT7,@ICSR       ;CLEAR INPUT DATA READY
005560' 012767  000004  172314          MOV     #BIT2,ACSR              ;LOAD EXPECTED
005566' 016777  172502  172440          MOV     TEMP1,@DOR              ;LOAD ALL OTHER DATA BITS
005574' 017767  172440  172302          MOV     @ICSR,ASTAT             ;READ INPUT STATUS
005602' 026767  172274  172274          CMP     ACSR,ASTAT              ;COMPARE
005610' 001403                          BEQ     3$                      ;BR IF SAME
                                        ;*************************************************************
005612' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;INPUT DATA READY FLAG SET IN ERROR
                                        ;*************************************************************
                                                                ;UNEXPECTED SBR BIT SET INPUT DATA READY
005620'                         3$:
005620' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
005624' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005630' 006367  172436                  ASL     TEMP                    ;TRY NEXT BIT
005634' 001274                          BNE     1$                      ;BR IF MORE BITS
```

```
1068                                        ;TEST THE TRANSITION ENABLE AND TRANSITION DETECTION
1069  005636' 005077  172372       DI41$:  CLR     @DOR
1070  005642' 012777  177777  172374        MOV     #-1,@DIR                ;CLEAR INPUT REGISTER
1071  005650' 012767  010000  172414        MOV     #BIT12,TEMP             ;LOAD INITIAL TRANSITION BIT
1072  005656' 012777  000404  172354        MOV     #BIT8!BIT2,@ICSR        ;SET STIM. CLEAR READY, ENABLE TRANS.
1073  005664' 012767  100604  172210        MOV     #BIT15!BIT8!BIT7!BIT2,ACSR      ;LOAD EXPECTED STATUS
1074  005672' 042777  100200  172340  1$:   BIC     #BIT15!BIT7,@ICSR       ;CLEAR READY
1075  005700' 016777  172366  172342        MOV     TEMP,@SBR               ;LOAD STIMULUS REG.
1076  005706' 016777  172360  172320        MOV     TEMP,@DOR               ;LOAD INPUT REG. <VIA OUTPUT REG.>
1077  005714' 017767  172320  172162        MOV     @ICSR,ASTAT             ;READ INPUT STATUS
1078  005722' 026767  172154  172154        CMP     ACSR,ASTAT              ;COMPARE
1079  005730' 001403                        BEQ     2$                      ;BR IF SAME
1080                                        ;*************************************************************
1081  005732' 104405  000000' 000000        HRDER$,BEGIN,NULL       ;TRANSITION ENABLE OR TRANSITION TO A ONE FAILED
1082                                        ;*************************************************************
1083
1084                                  ;NOW REMOVE THE TRANSITION DATA BIT (THIS SHOULD CAUSE THE INPUT READY FLAG TO SET AGAIN
1085  005740' 012777  177777  172276  2$:   MOV     #-1,@DIR                ;CLEAR INPUT REG
1086  005746' 042777  100200  172264        BIC     #BIT15!BIT7,@ICSR       ;CLEAR INPUT READY FLAG
1087  005754' 046777  172312  172252        BIC     TEMP,@DOR               ;REMOVE THE INPUT DATA
1088                                                        ;THIS SHOULD CAUSE THE TRANSITION TO A ZERO
1089  005762' 017767  172252  172114        MOV     @ICSR,ASTAT             ;READ INPUT STATUS
1090  005770' 026767  172106  172106        CMP     ACSR,ASTAT              ;COMPARE
1091  005776' 001403                        BEQ     3$                      ;BR IF SAME
1092                                        ;*************************************************************
1093  006000' 104405  000000' 000000        HRDER$,BEGIN,NULL       ;TRANSITION TO A ZERO FAILED
1094                                        ;*************************************************************
1095
1096  006006'                       3$:
1097  006006' 104407  000000'              BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
1098  006012' 104407  000000'              BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
1099  006016' 006367  172250              ASL     TEMP                    ;TRY ANOTHER BIT ?
1100  006022' 001323                      BNE     1$                      ;BR IF YES
1101
1102  006024' 000167  172370              JMP     CONT2                   ;TRY NEXT UNIT
1103
1104  006030' 006034'               MSG1$:  PRI1                    ;POINTER TO TEXT
1105  006032' 177777                        -1                      ;TERMINATOR
1106  006034' 053045  051105  043111  PRI1$:  .ASCIZ  \%VERIFY MNCDO WRAP-AROUND CONNECTION TO MNCDI\
1107  006042' 020131  047115  042103
1108  006050' 020117  051127  050101
1109  006056' 040455  047522  047125
1110  006064' 020104  047503  047116
1111  006072' 041505  044524  047117
1112  006100' 052040  020117  047115
1113  006106' 042103  000111
1114                                  .EVEN
1115          000001                  .END
```



```
ACSR    000102R          198#  305*  306   308   316*  321*  322*  323   325   338*  341   347*  350
                         350*  361   364   369   371   381*  382   385   390   392   402*  403   406
                         411   413   423*  424   427   432   434   444*  445   448   453   455   465*
                         466   469   474   476   486*  487   490   495   497   507*  508   511   516
                         518   528*  529   532   537   539   549*  550   553   558   560   571*  574
                         583*  585   593*  596   602*  606   617*  619   628*  633   640*  646   653*
                         656   662*  668   675*  677   686*  687*  693   709*  711   761*  762   764
                         772*  777*  778*  779   781   794*  797   803*  806   816*  817   820   825
                         827   837*  838   841   846   848   858*  859   862   867   869   879*  883
                         891*  901*  934*  936   949*  951   963*  965   972*  977   987*  988   990
                        1007*  1008*  1009  1037*  1039  1052*  1056  1073*  1078  1090
ADDR    000006R          164#  252
ADDR22= 001000           211#
ASB     000106R          202#
ASTAT   000104R          200#  307*  308   324*  325   340*  341   349*  350   362*  363*  364   370*
                         371   383*  384*  385   391*  392   404*  405*  406   412*  413   425*  426*
                         427   433*  434   446*  447*  448   454*  455   467*  468*  469   475*  476
                         488*  489*  490   496*  497   509*  510*  511   517*  518   530*  531*  532
                         538*  539   551*  552*  553   559*  560   572*  573*  574   584*  585   595*
                         596   605*  606   618*  619   632*  633   645*  646   655*  656   663*  668
                         676*  677   692*  693   710*  711   763*  764   780*  781   796*  797   805*
                         806   818*  819*  820   826*  827   839*  840*  841   847*  848   860*  861*
                         862   868*  869   882*  883   893*  903*  935*  936   950*  951   964*  965
                         976*  977   989*  990   1006*  1009  1038*  1039  1055*  1056  1077*  1078  1089*
                        1090
AWAS    000110R          203#
BEGIN   000000R          161#  295   311   314   315   328   331   332   344   353   356   357   367
                         374   377   378   388   395   398   399   409   416   419   420   430   437
                         440   441   451   458   461   462   472   479   482   483   493   500   503
                         504   514   521   524   525   535   542   545   546   556   563   566   567
                         577   588   599   609   622   625   626   636   649   659   666   671   680
                         696   699   700   714   726   730   745   748   767   770   771   784   787
                         788   800   809   812   813   823   830   833   834   844   851   854   855
                         865   872   875   876   886   896   906   917   921   939   941   954   968
                         980   993   996   997   1012  1015  1016  1042  1059  1063  1064  1081  1093
                        1097  1098
BITDAT= 010000           241#  654   661   674   685
BITEXT= 004000           242#  594   603   616   630   631   642   643   706   708   721   740   741
                         974
BIT0  = 000001           211#  240   245   305   320   684   761   776   881   890   900   912   986
                        1004  1026
BIT1  = 000002           211#  360   628   629   640   641   691   705   720   739   946   949
BIT10 = 002000           211#
BIT11 = 004000           211#  242
BIT12 = 010000           211#  241   528   705   1071
BIT13 = 020000           211#
BIT14 = 040000           211#  549   739
BIT15 = 100000           211#  628   631   643   740   741   1037  1051  1073  1074  1086
BIT2  = 000004           211#  381   571   615   617   960   963   1030  1037  1052  1072  1073
BIT3  = 000010           211#  402   858
BIT4  = 000020           211#  423   571   661   685   705   837   1003
BIT5  = 000040           211#  444   661   685   690   705   707   1003
BIT6  = 000100           211#  465   720   816   913
BIT7  = 000200           211#  363   384   405   426   447   468   489   510   531   552   573   593
                         594   628   630   631   640   642   643   706   708   721   740   741   819
                         840   861   879   934   949   972   974   1034  1037  1051  1073  1074  1086
```

```
BIT8  = 000400          211#    293     486    1072    1073
BIT9  = 001000          211#    507
BREAK$= 104407          211#    314     315     331     332     356     357     377     378     398     399     419     420
                        440     441     461     462     482     483     503     504     524     525     545     546     566
                        567     625     626     699     700     770     771     787     788     812     813     833     834
                        854     855     875     876     996     997    1015    1016    1063    1064    1097    1098
BR1     000012R         166#    719     738     910
BR2     000013R         167#
BTOD$ = 104421          211#
CDATA$= 104412          211#
CONFIG  000056R         186#
CONT1   000410R         269#    294
CONT2   000420R         272#    754    1102
CSRA    000100R         196#    299*    756*
DATCK$= 104411          211#
DATER$= 104404          211#
DIDINS  000262R         234#    719*    732     733*
DIDINV  000260R         233#    718*    732*
DIEINS  000266R         236#    265     738*    750     751*
DIEINV  000264R         235#    737*    750*
DIR     000244R         225#    301     614*    655     663     676     688     689*    692     710     929*    944*    958*
                        983*    989    1001*    1006    1031*   1050*   1070*   1085*
DIR1    000246R         226#
DI1     000542R         303#
DI10    001470R         442#
DI11    001600R         463#
DI12    001710R         484#
DI13    002020R         505#
DI14    002130R         526#
DI15    002240R         547#
DI16    002350R         569#
DI17    002422R         575     581#
DI2     000620R         318#
DI20    002466R         586     592#
DI21    002532R         597     602#
DI22    002574R         607     613#
DI23    002664R         628#
DI24    002740R         634     640#
DI25    003020R         647     653#
DI26    003164R         678     684#
DI27    003302R         705#
DI3     000710R         335#
DI30    003364R         712     718#
DI31    003454R         737#
DI32    004666R         929#
DI33    004756R         937     944#
DI34    005042R         952     958#
DI35    005126R         966     971#
DI36    005202R         978     983#
DI37    005304R        1001#
DI4     001030R         358#
DI40    005414R        1025#
DI41    005636R        1069#
DI5     001140R         379#
DI6     001250R         400#
DI7     001360R         421#
```



```
DODIN$  000256R         231#    260     910*    923     924*
DODINV  000254R         230#    255     909*    923*
DOR     000234R         219#    758     762*    763     779*    780     793*    796     802*    804*    805     880*    889*
                        899*    902*    911*    933*    948*    962*    971*    988*   1005*   1025*   1028*   1035*   1049*
                       1054*   1069*   1076*   1087*
DOR1    000236R         220#    795*
DO0     003564R         753     756#
DO1     003602R         759#
DO10    004470R         884     889#
DO11    004530R         894     899#
DO12    004570R         904     909#
DO2     003660R         774#
DO3     003750R         791#
DO4     004070R         814#
DO5     004200R         835#
DO6     004310R         856#
DO7     004420R         878#
DVID1   000014R         168#    269
ENDIT$= 104413          211#    295
END$  = 104410          211#
ERRTYP  000106R         201#
EXIT$ = 104400          211#    726     745     917
GETPA$= 104415          211#
GWBUF$= 104414          211#
HRDCNT  000044R         181#
HRDER$= 104405          211#    311     328     344     353     367     374     388     395     409     416     430     437
                        451     458     472     479     493     500     514     521     535     542     556     563     577
                        588     599     609     622     636     649     659     666     671     680     696     714     767
                        784     800     809     823     830     844     851     865     872     886     896     906     939
                        954     968     980     993    1012    1042    1059    1081    1093
HRDPA$  000050R         183#
ICONT   000036R         178#
ICOUNT  000040R         179#
ICSR    000240R         223#    250     299     300     361*    362     369*    370     382*    383     390*    391     403*
                        404     411*    412     424*    425     432*    433     445*    446     453*    454     466*    467
                        474*    475     487*    488     495*    496     508*    509     516*    517     529*    530     537*
                        538     550*    551     558*    559     569*    572     581*    582*    584     592*    594*    595
                        603*    604*    605     615*    616*    618     629*    630*    631*    632     641*    642*    643*
                        645     654*    661*    674*    685*    690*    691*    705*    706*    707*    708*    720*    721*
                        728*    739*    740*    741*    746*    931*    935     946*    950     960*    964     974*    975*
                        985*   1003*   1030*   1034*   1038    1051*   1055    1072*   1074*   1077    1086*   1089
ICSR1   000242R         224#    570*    644*
IDNUM   000122R         208#
INIT    000030R         175#
INPUT   000520R         270     299#
INTR    000120R         207#
MAP22$= 104416          211#
MODNAM  000000R         162#
MODSP   000224R         176     209#
MSGN$ = 104403          211#    941
MSGS$ = 104402          211#
MSG$  = 104401          211#
MSG1    006030R         941    1104#
NULL  = 000000          211#    311     328     344     353     367     374     388     395     409     416     430     437
                        451     458     472     479     493     500     514     521     535     542     556     563     577
                        588     599     609     622     636     649     659     666     671     680     696     714     767
```

```
                        784     800     809     823     830     844     851     865     872     886     896     906     939
                        954     968     980     993    1012    1042    1059    1081    1093
OCSR    000230R         216#    246     272     756     757     817*    818     825*    826     838*    839     846*    847
                        859*    860     867*    868     878*    882     892*    893     903     913*    919*    925*    932*
                        947*    961*    973*    976     984*   1002*
OCSR1   000232R         217#    881*    890*    900*    912*
OPEN  = 000000          163     169     170     171     172     189     190     191     192     193     194     195     196
                        198     200     202     203     205     206     207     211#
OTOA$ = 104420          211#
PASCNT  000034R         177#
PIRQ$ = 000004          211#    730     748     921
POPSP = 005726          211#
POPSP2= 022626          211#
PRI1    006034R        1104    1106#
PRTY  = 000000          211#
PRTY0 = 000000          167     211#
PRTY1 = 000040          211#
PRTY2 = 000100          211#
PRTY3 = 000140          211#
PRTY4 = 000200          166     211#
PRTY5 = 000240          211#
PRTY6 = 000300          211#
PRTY7 = 000340          211#
PS    = 177776          211#
PSW   = 177776          211#
PUSH  = 005746          211#
PUSH2 = 024646          211#
RAND$ = 104417          211#
RANNUM  000054R         185#
RESTRT  000306R         204     245#    297
RES1    000056R         187#
RES2    000060R         188#
RSTRT   000112R         204#
SBADR   000102R         197#
SBR     000250R         227#    302     306*    307     323*    324     337*    340     346*    348*    349     613*    930*
                        945*    959*   1029*   1033*   1075*
SBR1    000252R         228#    339*
SOFCNT  000042R         180#
SOFER$= 104406          211#
SOFPAS  000046R         182#
SPOINT  000032R         176#
SPSIZ = 000040            1#    209
SRMINE  000270R         237#    243*    752
SR1     000016R         169#    243
SR2     000020R         170#
SR3     000022R         171#
SR4     000024R         172#
START   000300R         175     243#
STAT    000026R         174#
SVR0    000062R         189#
SVR1    000064R         190#
SVR2    000066R         191#
SVR3    000070R         192#
SVR4    000072R         193#
SVR5    000074R         194#
SVR6    000076R         195#
```



```
SYSCNT  000052R         184#
TEMP    000272R         238#    320*    321     333*    684*    686     689     701*    776*    777     789*    986*    987
                        998*   1004*   1005    1007    1017*   1026*   1033    1035    1047    1065*   1071*   1075    1076
                       1087    1099*
TEMP1   000274R         239#    1047*   1048*   1054
TEMP2   000276R         240#    245*    269     292*    293     752
TRPDFD= 000022          211#
VECTOR  000010R         165#    262
WASADR  000104R         199#
WDFR    000116R         206#
WDTO    000114R         205#
XFLAG   000005R         163#
$BASE1  000224R         212#    247
$VECT1  000226R         213#    257


. ABS.  000000      000
        006112      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XMNBA0,XMNBA0/SOL/CRF:SYM=DDXCOM,XMNBA0
 RUN-TIME:  2  4  .5 SECONDS
 RUN-TIME RATIO: 185/8=21.5
 CORE USED:  7K  (13 PAGES)
```

```
1
2
3

        .REM !
```

IDENTIFICATION

PRODUCT CODE: AC-F416A-MC

PRODUCT NAME: CXMNAA0 MNCAD (A/D) MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

MNA IS AN IOMOD THAT EXERCISES THE MNCAD (A/D) ANALOG MODULE. THIS MODULE REQUIRES ONLY AN ANALOG GROUND ON CHANNEL ZERO IN ORDER TO BE RUN, HOWEVER; WITH SPECIAL SETUP, MORE OPTIONS CAN BE CHOSEN. ONE OPTION IS THE USE OF THE MNCKW (REAL TIME CLOCK) DEVICE. THIS OPTION ALLOWS EXERCISING THE MNCAD ASYNCHRONOUSLY WITH THE LSI-11 CPU, THAT IS, MNCAD CONVERSIONS WILL BE STARTED AT RANDOM TIMES TO ALLOW FOR MAXIMUM BUS NOISE DURING THE CONVERSION. IF THIS OPTION IS SELECTED, YOU MUST DESELECT MNC FROM A DEC/X11 RUN AND ONLY ONE MNCAD (A/D) CAN BE RUN.

WITH NORMAL OPERATION, RMS NOISE AND PEAK NOISE ARE SAMPLED ON CHANNEL ZERO AND COMPARED AGAINST A LIMIT. WITH THE SECOND OPERATION OPTION, MORE CHANNELS MAY BE SPECIFIED TO RUN THE NOISE TESTS ON. THE THIRD OPTION ALLOWS FOR SAMPLING OF ONE TO ALL THE CHANNELS OF THE MNCAD. A CHECK IS MADE TO SEE THAT THE INPUT VOLTAGE REMAINS STABLE WITHIN AN ALLOWED TOLERANCE. LOCATIONS WITHIN THIS MODULE ARE PROVIDED TO CHANGE ANY LIMIT, OR TO FORCE TYPEOUT OF ANY VALUE. UP TO FOUR MNCAD (A/D)'S CAN BE EXERCISED WITH THIS MODULE. WHEN SR1 BITS 1 OR 2 ARE SET AND ADDITIONAL MNCAD'S ARE ENABLED, THE SAME NUMBER OF CHANNELS ON EACH MNCAD WILL BE TESTED. IF SR1 BIT 0 IS SET, ONLY ONE MNCAD CAN BE BE TESTED.

2.0 REQUIREMENTS

HARDWARE: ONE MNCAD (A/D)
ONE MNCKW (CLOCK) <OPTIONAL>

STORAGE: MNA REQUIRES:
DECIMAL WORDS: 901
OCTAL WORDS: 1605
OCTAL BYTES: 3412

3.0 PASS DEFINITION

ONE PASS OF THE MNA MODULE CONSISTS OF GENERATING 8244(DECIMAL) INTERRUPTS (CONVERSIONS).

4.0 EXECUTION TIME

ONE PASS OF THE MNA MODULE RUNNING ALONE WITH SR1 = 0, 1 OR 2, WILL TAKE APPROXIMATELY ONE MINUTE. THE TIME FOR A PASS WITH SR1 = 4 WILL DEPEND UPON THE NUMBER OF CHANNELS SPECIFIED IN LOCATION "NLSTCH".

5.0 CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 171000, VECTOR: 400, BR1: 4

DEVCNT: 1, SR1: 0

REQUIRED PARAMETERS:

NONE IF SR1=0

IF SR1 BITS 0, 1 OR 2 = 1 THEN SEE OPERATION OPTIONS
IF SR1 BIT 0 IS SET, ONLY 1 MNCAD WILL BE TESTED.

6.0 DEVICE/OPTION SETUP

AN ANALOG GROUND MUST BE PUT ON CH. 0
(BY SET CH0 FRONT PANEL SWITCH TO "TEST").

SR1 BIT0=1 USE THE MNCKW OPTION FOR A TO D STARTS
(THE MNCKW <MNC> MODULE MUST BE DESELECTED)

SR1 BIT1=1 ALL CHANNELS SPECIFIED MUST HAVE
AN ANALOG GROUND.

SR1 BIT2=1 SELECT ADDITIONAL CHANNELS FOR NOISE TESTING.

7.0 MODULE OPERATION

1. (STAPT) (RESTRT) INITLIZE POINTERS AND TYPE-OUT VALUES.

2. (LOOP) BIT EXERCISE CSR

3. (ADPMS1) PREFORM RMS NOISE CHECK ON SPECIFIED CHANNEL. WE FIRST USE SAR TO FIND THE DAC VALUE THAT PRODUCES A 16/84 SPLIT FOR THE LEFT BOUNDARY OF NOISE. THEN WE USE SAR TO FIND THE DAC VALUE THAT PRODUCES A 84/16 SPLIT FOR THE RIGHT VALUE. WE THEN SUBTRACT THE TWO DAC VALUES AND WE HAVE A VALUE FOR THE 68% AREA OF NOISE (RMS). IT IS THEN COMARED AGAINST THE ALLOWED LIMIT TO SEE IF EXCESSIVE NOISE IS ON THE CHANNEL.

4. (ADPK1) PREFORM PEAK NOISE CHECK ON SPECIFIED CHANNEL. WE FIRST USE SAR TO FIND THE DAC VALUE THAT PRODUCES A .6% SPLIT FOR THE LEFT BOUNDARY OF NOISE. THEN WE USE SAR TO FIND THE DAC VALUE THAT PRODUCES A .6% SPLIT FOR THE RIGHT BOUNDARY. WE THEN SUBTRACT THE TWO DAC VALUES AND WE HAVE A VALUE OF 98% AREA OF NOISE (PEAK). IT IS THEN COMPARED AGAINST THE ALLOWED LIMIT TO SEE IF EXCESSIVE NOISE IS ON THE CHANNEL.

5. IF MULTIPLE CHANNELS ARE SELECTED FOR NOISE TESTING TEST NEXT CHANNEL, IF SINGULAR RETEST CHAN. 0.

6. IF MULTI-CHANNEL SAMPLING IS SELECTED, TAKE SAMPLES ON EACH CHANNEL SPECIFIED, AND COMPARE THE AVERAGE OF THE SAMPLES AGAINST THE OLD AVERAGE FOR THE CHANNEL. IF THE DIFFERENCE IS GREATER THAN THE TOLERANCE, REPORT THE DATA ON THAT CHANNEL.

7. REPORT END PASS IF LAST UNIT, BRANCH TO "LOOP" IF MORE UNITS.

8. (SAP) SAR IS A SUCCESSIVE APPROXIMATION ROUTINE. IT IS USED TO FIND A DAC VALUE THAT PRODUCES A DESIRED SPLIT. IT DOES THIS BY TRYING A DAC VALUE AND TAKING 512 CONVERSIONS ON THE A/D. IF THE AMOUNT OF THE SAMPLES IS LOWER THEN SPECIFIED IT INCREASES THE DAC VALUE, IF THE AMOUNT OF SAMPLES IS HIGHER THEN SPECIFIED, IT DECREASES THE DAC VALUE. IF THE END DAC VALUE IS EITHER 000 OR 377 WE HAVE A "WARPAROUND" ERROR. THIS OCCURS WHEN WE ARE UNABLE TO ADJUST THE DAC TO PRODUCE A DESIRED SPLIT, AND INDICATES EXCESSIVE NOISE ON A CHANNEL.

9. (RANDY) THIS IS A RANDOM NUMBER GENERATOR. IF THE MNCKW CLOCK OPTION IS SELECTED WE GET THE NUMBER THAT WE PUT INTO THE CLOCK PRESET REGISTER FROM THIS ROUTINE.

8.0 OPERATION OPTIONS

1. VALID SR1 VALUES

SR1 BIT	ENABLE/DISABLE	FUNCTION
0	0	INHIBIT USE OF MNCKW OPTION.
0	1	ENABLE USE OF MNCKW OPTION. NOTE: IF ENABLED, YOU MUST DESELECT "MNC" FROM DEC/X11 RUN AND ONLY 1 MNCAD WILL BE RUN.
1	0	INHIBIT SAMPLING ADDITIONAL CHANNELS FOR STABLE INPUT.
1	1	ENABLE SAMPLING CHANNEL ZERO THROUGH CHANNEL SPECIFIED BY CLSTCH FOR STABLE INPUT (+-) TOLERANCE SPECIFIED BY OFFALL
2	0	USE CHANNEL ZERO ONLY FOR NOISE TESTING.
2	1	USE CHANNEL ZERO THROUGH THE CHANNEL SPECIFIED IN NLSTCH FOR NOISE TESTING.

2. THE FOLLOWING ARE LOCATIONS WITHIN THIS MODULE THAT ENABLE THE USER TO CHANGE LIMITS AND SPECIFY CHANNELS.

LOCATION	FUNCTION
ARMLIM	SPECIFIES MAXIMUM LIMIT FOR RMS NOISE. MAY BE CHANGED TO ZERO TO FORCE TYPEOUT OF RMS NOISE ON A CHANNEL RUNNING IN A SYSTEM ENVIRONMENT.
APKLIM	SPECIFIES MAXIMUM LIMIT FOR PEAK NOISE. MAY BE CHANGED TO ZERO TO FORCE TYPEOUT OF PEAK NOISE ON A CHANNEL RUNNING IN A SYSTEM ENVIRONMENT.
CLSTCH	IF SR1 BIT1=1, USED TO SPECIFY END CHANNEL FOR SAMPLING STABLE INPUT.
OFFALL	IF SR1 BIT1=1, USED TO SPECIFY TOLERANCE OF STABLE CHANNEL. PRESET BY MODULE TO "000002".
NLSTCH	IF SR1 BIT2=1, USED TO SPECIFY LAST CHANNEL FOR NOISE TESTING.

9.0 NON-STANDARD PRINTOUTS

1. IF A CHANNEL HAS EXCESSIVE RMS NOISE, IT REPORTS IT IN AN ERROR CALL AND A MSGN CALL:

   (EXAMPLE)

   ```
   MNCAD (A/D) UNIT #0  ON CHAN 00
   A/D RMS NOISE = 0.52 LSB (LIMIT = .20 LSB)
   ```

2. IF A CHANNEL HAS EXCESSIVE PEAK NOISE, IT REPORTS IT IN AN ERROR CALL AND A MSGN CALL:

   (EXAMPLE)

   ```
   MNCAD (A/D) UNIT #0   ON CHAN 00
   A/D PEAK NOISE = 2.57 LSB (LIMIT = 2.00 LSB)
   ```

3. IF THERE IS AN EXCESSIVE AMOUNT OF NOISE SO THAT THE DAC CANNOT BE ADJUSTED, IT REPORTS IT IN AN ERROR CALL AND A MSGN CALL:

   (EXAMPLE)

   ```
   MNCAD (A/D) UNIT #0
   A/D PEAK WRAPAROUND ERROR ON CHAN  00
   ```

4. IF A CHANNEL IS FOUND TO BE UNSTABLE IN STABLE INPUT SAMPLEING, IT REPORTS IT IN A MSGN CALL:

   (EXAMPLE)

   ```
   A/D ERROR ON CHAN 14; OLD VALUE = 4066  NEW VALUE = 4000
   !
   ```

```
                        .TITLE  MNAA DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION  6       23-MAY-78
                                .LIST   BIN
                                ;*********************************************************************
000000'                         BEGIN:
000000' 047115  040501      040 MODNAM: .ASCII  /MNAA / ;MODULE NAME.
000005'    000                  XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF XBUFF USAGE
000006' 171000                  ADDR:   171000+0                ;1ST DEVICE ADDR.
000010' 000400                  VECTOR: 400+0                   ;1ST DEVICE VECTOR.
000012'    300                  BR1:    .BYTE   PRTY6+0         ;1ST BR LEVEL.
000013'    000                  BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;*********************************************************************
000026' 140000                  STAT:   140000                  ;STATUS WORD.
000030' 000274'                 INIT:   START                   ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                       ;PASS COUNTER.
000036' 000005                  ICONT:  5                       ;# OF ITERATIONS PER PASS=5
000040' 000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                         ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000102'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000104'                         WASADR:                         ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000106'                         ERRTYP:                         ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000112' 000274'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
000122' 000000                  IDNUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
000224'                         MODSP:
                                ;*********************************************************************
```



```
                                ;*OPTIONAL USER SUPPLIED VALUES
000224' 000000                  CLSTCH: .WORD   0               ;USER SUPPLIED LAST SAMPLED CH
000226' 000002                  OFFALL: .WORD   2               ;USER SUPPLIED TOLERANCE FOR SAMPLE
000230' 000000                  NLSTCH: .WORD   0               ;USER SUPPLIED LAST NOISE CH.
000232' 000062                  ARMLIM: 50.                     ;RMS NOISE LIMIT
000234' 000310                  APKLIM: 200.                    ;A/D PEAK NOISE LIMIT.
                                ;*
                                ;*REGISTER AND VECTOR ADDRESS
                                ;*
000236' 171000                  ADSR:   .WORD   171000          ;A/D CSR ADDRESS
000240'                         DAC:
000240' 171002                  ADBR:   .WORD   171002          ;DAC (WRITE ONLY) AND BUFFER REG (READ ONLY)

                                ;THE FOLLOWING ARE CLOCK ADDRESSES IF A MNCKW EXISTS.

000242' 171020                  ASP:    .WORD   171020          ;CLOCK A CSR.
000244' 171022                  ABR:    .WORD   171022          ;CLOCK A PRESET BUFFER.

                                ;*
                                ;*FLAGS, COUNTERS AND OTHER REGISTERS
                                ;*

000246' 000000                  WHO:    .WORD   0               ;0=RMS NOISE, 1=PEAK NOISE
000250' 000000                  INTFLG: .WORD   0               ;USED IN LOGIC TEST TO INDICATE AN A/D INTR.
000252' 000000                  FRED:   .WORD   0               ;USED TO HOLD CURRENT DAC VALUE.
000254' 000000                  EDGE:   .WORD   0               ;HOLD CURRENT EDGE VALUE.
000256' 000000                  TMP:    .WORD   0               ;TEMP WORKING AREA.
000260' 000000                  ARMX:   .WORD   0               ;USED TO HOLD RMS NOISE VALUE.
000262' 000000                  APKX:   .WORD   0               ;USED TO HOLD A/D PEAK NOISE VALUE.
000264' 000000                  NCCH:   .WORD   0               ;CURRENT NOISE CHAN.
000266' 000000                  CCH:    .WORD   0               ;CURRENT SAMPLE CHAN.
000270' 000001                  TEMP:   BIT0                    ;CURRENT UNIT BIT
000272' 000400                  VECTAD: 400                     ;A/D VECTOR
```

```
                                        ;*
                                        ;*START-UP AND INITILIZE SECTION
                                        ;*

000274'                         RESTRT:
000274'                         START:

000274' 012767  000001  177766  LOOP1:  MOV     #1,TEMP         ;LOAD UNIT BIT
000302' 016700  177500          MOV     ADDR,R0         ;GET BASE ADDR OF A/D.
000306' 010067  177724          MOV     R0,ADSR         ;FIX A/D CSR ADDR.
000312' 062700  000002          ADD     #2,R0           ;BUFFER REG=CSR+2.
000316' 010067  177716          MOV     R0,ADBR         ;FIX BUFFER REG ADDR.
000322' 016767  177462  177742  MOV     VECTOR,VECTAD   ;LOAD VECTOR VALUE
000330' 005067  177730          CLR     NCCH            ;CLEAR 1ST CHAN. FOR NOISE.

                                        ;*************************************************************
                                        ;CONVERT ARMLIM TO ASCII AND
                                        ;STORE AT DECIM
000334' 104421  000000' 000232'         BTOD$,BEGIN,ARMLIM,DECIM
000342' 002600'
                                        ;*************************************************************

000344' 116767  002232  002451          MOVB    DECIM+2,P7      ;NOW WE WILL PUT IT
000352' 116767  002225  002445          MOVB    DECIM+3,P7+2    ;INTO THE ASCII
000360' 116767  002220  002440          MOVB    DECIM+4,P7+3    ;MESSAGE THAT WE TYPE OUT.

                                        ;*************************************************************
                                        ;CONVERT APKLIM TO ASCII AND
                                        ;STORE AT DECIM
000366' 104421  000000' 000234'         BTOD$,BEGIN,APKLIM,DECIM
000374' 002600'
                                        ;*************************************************************

000376' 116767  002200  002431          MOVB    DECIM+2,P8      ;NOW WE WILL PUT IT
000404' 116767  002173  002425          MOVB    DECIM+3,P8+2    ;INTO THE ASCII MESSAGE
000412' 116767  002166  002420          MOVB    DECIM+4,P8+3    ;THAT WE TYPE OUT.

                                ;LOOP TO HERE IF MULTIPLE MNCAD'S
                                ;       FIX THE UNIT # FOR THE ERROR TYPEOUT
000420' 016700  177644          LOOP:   MOV     TEMP,R0         ;GET CURRENT UNIT #
000424' 005001                  CLR     R1              ;INIT VALUE
000426' 006200                  1$:     ASR     R0              ;MOVE IT
000430' 001402                  BEQ     2$              ;BR IF LAST
000432' 005201                  INC     R1              ;UPDATE COUNT
000434' 000774                  BR      1$
000436' 062701  000060          2$:     ADD     #60,R1          ;MAKE ASCII VALUE
000442' 110167  002442          MOVB    R1,P10U         ;SAVE IN MESSAGE
```



```
                                        ;*
                                        ;*LOGIC TEST #1
                                        ;*IN THIS TEST WE WILL SEE IF THE A/D RESPONDS TO ITS
                                        ;*ADDR. IF NOT, A DEC/X11 "SYS ERROR..." WILL OCCUR
                                        ;*

000446' 005777  177564          LOG1:   TST     @ADSR           ;ADDRESS THE A/D

                                        ;*
                                        ;*LOGIC TEST #2
                                        ;*IN THIS TEST WE WILL SEE IF THE CLOCK RESPONDS TO ITS ADDR, ONLY
                                        ;*IF THE CLOCK OPTION IS SELECTED BY SR1.

000452'                         LOG2:
000452' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000456' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000462' 032767  000001  177326          BIT     #BIT0,SR1       ;IS CLOCK OPTION SELECTED?
000470' 001402                  BEQ     LOG3            ;BR IF NOT TO NEXT TEST.

000472' 005777  177544                  TST     @ASR            ;CLOCK WAS SELECTED, WILL IT RESPOND?

                                        ;*
                                        ;*LOGIC TEST #3
                                        ;*IN THIS TEST WE WILL SEE IF THE A/D WILL INTERRUPT.
                                        ;*
000476' 005067  177546          LOG3:   CLR     INTFLG          ;CLEAR A/D DID INTR. FLAG.
000502' 012777  000566' 177562          MOV     #1$,@VECTAD     ;SET UP VECTOR ADDR.
000510' 012777  000101  177520          MOV     #101,@ADSR      ;START A CONVERSION, SHOULD INTERRUPT.
                                        ; BEFORE BREAK TIME IS OVER.

000516' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000522' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.

000526' 005767  177516                  TST     INTFLG          ;DID THE A/D INTERRUPT?
000532' 001024                  BNE     LOG4            ;IF SO, NEXT TEST
000534' 016767  177476  177336          MOV     ADSR,CSRA       ;SET ADDR. OF AD CSR FOR ERROR TYPEOUT.
000542' 017767  177470  177332          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF CSR.
000550' 005077  177462                  CLR     @ADSR           ;STOP A/D
                                        ;*************************************************************
000554' 104405  000000' 000000          HRDEP$,BEGIN,NULL       ;A/D FAILED TO INTERRUPT
                                        ;*************************************************************
000562' 104410  000000'                 END$,BEGIN              ;DROP MODULE ON THIS FAILURE

                                        ;*A/D SHOULD INTR. TO HERE.
000566' 005077  177444          1$:     CLR     @ADSR           ;STOP A/D.
000572' 005777  177442                  TST     @ADBR           ;CLEAR CSR BIT07.
000576' 005267  177446                  INC     INTFLG          ;INDICATE THAT IT DID INTR.
000602' 000002                  RTI                     ;EXIT INTR.
```

```
                                        ;*LOGIC TEST #4
                                        ;*THIS TEST WILL ONLY BE DONE IF THE CLOCK OPTION IS SELECTED
                                        ;*IN THIS TEST WE WILL SEE IF THE OVERFLOW OF CLOCK 1 WILL
                                        ;*TRIGGER A CONVERSION IN THE A/D
                                        ;*
000604' 032767  000001  177204  LOG4:   BIT     #BIT0,SR1       ;IS THE CLOCK OPTION SELECTED?
000612' 001446                          BEQ     LOG5            ;IF NOT, GOTO NEXT TEST.

000614' 012777  177777  177422          MOV     #177777,@ABR    ;PRESET CLOCK FOR ALL ONES.
000622' 012777  000040  177406          MOV     #BIT5,@ADSR     ;SET OVERFLOW ENABLE IN A/D.
000630' 012777  000011  177404          MOV     #11,@ASR        ;START CLOCK, 1MHZ, GO.
000636' 005000                          CLR     R0              ;SET FOR DELAY LOOP.

000640'                         1$:
000640' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000644' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000650' 105777  177366                  TSTB    @ASR            ;IS THE CLOCK OVERFLOW SET?
000654' 100402                          BMI     2$              ;YES-EXIT LOOP.
000656' 105200                          INCB    R0              ;NO,IS DELAY EXCEEDED?
000660' 001367                          BNE     1$              ;NO-CONTINUE DELAY.

000662'                         2$:
000662' 017767  177354  177214          MOV     @ASR,ASTAT      ;RECORD CONTENTS OF CLOCK CSR.
000670' 005077  177346                  CLR     @ASR            ;CLEAR THE CLOCK.

000674' 105777  177336                  TSTB    @ADSR           ;IS DONE FLAG SET?
000700' 100413                          BMI     LOG5            ;IF YES NEXT TEST.
                                                                ;IF NOT, ERROR
000702' 016767  177330  177170          MOV     ADSP,CSRA       ;RECORD A/D CSR ADDR.
000710' 017767  177322  177164          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR.
                                        ;*****************************************************************
000716' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;CLOCK OVERFLOW FAILED TO TRIGGER A/D CONVERSION
                                        ;*****************************************************************


                                                                ;FOR THIS ERROR YOU MIGHT CHECK
                                                                ;TO SEE IF A OVERFLOW IS WIRED TO
                                                                ;A/D INPUT.
                                                                ;DROP THIS MODULE - FATAL ERROR.
000724' 104410  000000'                 END$,BEGI               ;CAN'T CONTINUE IF OVERFLOW DOESN'T TRIGGER THE CONVERSI
```



```
                                        ;*
                                        ;*LOGIC TEST #5
                                        ;*IN THIS TEST WE CHECK THAT WHEN CHANNEL 0 IS CONVERTED
                                        ;*WITH THE MAINTENANCE BIT SET THE RESULT IS 0

000730'                         LOG5:
000730' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000734' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000740' 005777  177274                  TST     @ADBR           ;FALSE READ OF A/D BUFFER
000744' 012777  001014' 177320          MOV     #2$,@VECTAD     ;SET UP INTERRUPT VECTOR
000752' 012777  000105  177256          MOV     #105,@ADSP      ;START A/D WITH MAINTENANCE SET
000760' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
000764' 005777  177250          1$:     TST     @ADBR           ;TEST FOR ALL 0'S RESULT
                                                                ;NOTE: INTERRUPT SERVICE WILL
                                                                ;BEGIN USING HERE AFTER A PIRQ.
000770' 001414                          BEQ     LOG6            ;IF RESULT IS ALL 0'S, GOTO NEXT TEST
000772' 016767  177240  177100          MOV     ADSR,CSRA       ;RECORD A/D CSR ADDRESS
001000' 017767  177232  177074          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR
                                        ;*****************************************************************
001006' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;FAILED TO GET ALL 0'S RESULT FROM CONVERSION
                                        ;*****************************************************************


                                        ;*A/D INTERRUPTS TO HERE
001014'                         2$:
                                        ;---------------------------------------------------------------
001014' 000004  000000' 000764'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;---------------------------------------------------------------


                                        ;*
                                        ;*LOGIC TEST #6
                                        ;*IN THIS TEST WE CHECK THAT WHEN CHANNEL 1 IS CONVERTED
                                        ;*WITH THE MAINTENANCE BIT SET THE RESULT IS 7777

001022' 012777  001074' 177242  LOG6:   MOV     #2$,@VECTAD     ;SET UP INTERRUPT VECTOR
001030' 012777  000505  177200          MOV     #505,@ADSP      ;START A/D WITH MAINTENANCE SET
001036' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
001042' 022777  007777  177170  1$:     CMP     #7777,@ADBR     ;TEST FOR ALL 1'S RESULT
001050' 001414                          BEQ     LOG7            ;IF RESULT IS ALL 1'S, GOTO NEXT TEST
001052' 016767  177160  177020          MOV     ADSP,CSRA       ;RECORD A/D CSR ADDRESS
001060' 017767  177152  177014          MOV     @ADSR,ACSR      ;RECORD CONTENTS OF A/D CSR
                                        ;*****************************************************************
001066' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;FAILED TO GET ALL 1'S RESULT FROM CONVERSION
                                        ;*****************************************************************

                                        ;*A/D INTERRUPTS TO HERE
001074'                         2$:
                                        ;---------------------------------------------------------------
001074' 000004  000000' 001042'         PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
                                        ;---------------------------------------------------------------
```

```
                                        ;*
                                        ;*LOGIC TEST #7
                                        ;*IN THIS TEST WE WILL SAMPLE ALL CHANNELS SELECTED
                                        ;*FOR TEST AND STORE AWAY THEIR RESULTS.

001102'                         LOG7:
001102' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001106' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001112' 012777  000200  177120          MOV     #200,@ADBR      ;LOAD CORRECTION DAC
001120' 005777  177114                  TST     @ADBR           ;FALSE READ OF A/D BUFFER.
001124' 005077  177106                  CLR     @ADSR           ;CLEAR A/D'S CSR
001130' 005067  177132                  CLR     CCH             ;START ON CH. 0.
001134' 012777  001262' 177130          MOV     #4$,@VECTAD     ;SET UP INTR. VECTOR.

001142' 012700  177770          1$:     MOV     #-8.,R0         ;SET TO DO 8 CONVERSIONS.
001146' 005001                          CLR     R1              ;R1 WILL CONTAIN SUM OF CONVERSIONS.
001150' 016767  177112  177100          MOV     CCH,TMP         ;GET CH. NUMBER.
001156' 000367  177074                  SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001162' 052767  000101  177066          BIS     #PIT6!BIT0,TMP  ;ADD INTR. ENABLE AND GO.
001170' 016777  177062  177040  2$:     MOV     TMP,@ADSR       ;START A/D.
001176' 104400  000000'                 EXIT$,REGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001202'                         3$:
001202' 067701  177032                  ADD     @ADBR,R1        ;SUM THIS RESULT.
                                                                ;NOTE INTR SERVICE WILL
                                                                ;BEING USE HERE AFTER A PIRQ
001206' 005200                          INC     R0              ;DONE 8 CONVERSIONS?
001210' 100767                          BMI     2$              ;NO = DO ANOTHER ONE.

001212' 006201                          ASR     R1              ;NOW WE MUST
001214' 006201                          ASR     R1              ;CALCULATE THE AVERAGE
001216' 006201                          ASR     R1              ;OF THE SAMPLES THAT
001220' 005501                          ADC     R1              ;WE JUST TOOK.
001222' 016702  177040                  MOV     CCH,R2          ;PICK UP CH. NUMBER.
001226' 006302                          ASL     R2              ;USE IT AS AN OFFSET.
001230' 010162  003162'                 MOV     R1,RECSAM(2)    ;STORE THIS AVERAGE.

001234' 005267  177026                  INC     CCH             ;UPDATE CH. NUMBER.
001240' 026767  177022  176762          CMP     CCH,NLSTCH      ;DONE ALL NOISE CHANNELS?
001246' 003735                          BLE     1$              ;NO=DO NEXT ONE.
001250' 026767  177012  176746          CMP     CCH,CLSTCH      ;DONE ALL STABLE CHS?
001256' 003731                          BLE     1$              ;NO =DO NEXT ONE.
001260' 000403                          BR      REST0           ;YES=GOTO NOISE TESTS.


                                        ;*A/D INTERRUPTS TO HERE

001262'                         4$:
                                        ;-----------------------------------------------------------------
001262' 000004  000000' 001202'         PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3$ AND RTI
                                        ;-----------------------------------------------------------------
```



```
001270' 005077  176742  REST0: CLR     @ADSR           ;CLEAR A/D'S CSR.
001274' 016700  176772         MOV     VECTAD,R0       ;SET VECTOR AND PSW
001300' 012720  002262'        MOV     #WAINT,(R0)+    ;LOAD INTR. ADDR. INTO VECTOR ADDR.
001304' 116710  176502         MOVB    PR1,(R0)        ;LOAD PRIORITY
001310' 005067  176750         CLR     NCCH            ;INIT THE CHANNEL #
                               ;CALCULATE A/D RMS NOISE USING VERNIER DAC.

001314' 012700  000657  ADRMS1: MOV     #431.,R0        ;R0 = 84.13% OF 512 CONVERSIONS
001320' 016767  176740  176730         MOV     NCCH,TMP        ;GET THE CHAN #
001326' 000367  176724         SWAB    TMP             ;PUT IN PROPER CSR POSITION.
001332' 052767  000100  176716         BIS     #100,TMP        ;ADD INTR. ENABLE.
001340' 016701  176720         MOV     NCCH,R1         ;PICK UP CH. NUMBER.
001344' 006301                 ASL     R1              ;FORM AN OFFSET.
001346' 016167  003162' 176700         MOV     RECSAM(R1),EDGE ;GET EDGE VALUE FOR THIS CH.
001354' 005267  176666         INC     WHO             ;INDICATE RMS NOISE TEST.
001360' 016777  176672  176650         MOV     TMP,@ADSR       ;CH,#0, AND INTERRUPT ENABLE
001366' 016701  176646         MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001372' 004767  000552         JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 16/84 SPLIT
001376' 016705  176650         MOV     FRED,R5         ;SAVE LEFT BOUNDARY
001402' 012700  000121         MOV     #81.,R0         ;R0 = 15.87% OF 512 CONVERSIONS
001406' 004767  000536         JSR     PC,SAR          ;GET DAC VALUE THAT PRODUCES 84/16 SPLIT
001412' 166705  176634         SUB     FRED,R5         ;R5 = BREADTH OF NOISE @ 68% AREA
001416' 010567  176636         MOV     R5,ARMX         ;SAVE FOR DAC NOISE CALCULATIONS
001422' 020567  176604         CMP     R5,ARMLIM       ;< OR = RMS LIMIT?
001426' 003410                 BLE     ADPK1           ;IF WITHIN LIMIT THEN CONTINUE AT ADRMS2
001430' 004767  001052         JSR     PC,ERCOM        ;GET ERROR PARAMETERS
                                                       ;ERROR PARAMETERS LOADED BY "ERCOM"
                                       ;*****************************************************************
001434' 104405  000000' 000000         HRDER$,REGIN,NULL       ;;RMS NOISE ERROR=SEE NEXT TYPEOUT
                                       ;*****************************************************************
001442' 104403  000000' 002606'        MSGN$,BEGIN,MSG1        ;ASCII MESSAGE CALL WITH COMMON HEADER
                               ;CALCULATE A/D PEAK NOISE USING VERNIER DAC.

001450' 012700  000775  ADPK1:  MOV     #509.,R0        ;FOR EAK OF 2 1/2 SIGMA
001454' 016767  176604  176574         MOV     NCCH,TMP        ;GET CHAN. NUMBER.
001462' 000367  176570         SWAB    TMP             ;PUT IN CORRECT CSR POSITION.
001466' 052767  000100  176562         BIS     #100,TMP        ;ADD INTR. ENABLE
001474' 005067  176546         CLR     WHO             ;INDICATE PEAK NOISE TEST.
001500' 016777  176552  176530         MOV     TMP,@ADSR       ;CH.AND INTERRUPT ENABLE, AND CH
001506' 016701  176526         MOV     DAC,R1          ;R1 = ADDRESS OF SAR DAC
001512' 004767  000432         JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% LOW COUNT
001516' 016705  176530         MOV     FRED,R5         ;R5 = LEFT BOUNDARY
001522' 012700  000003         MOV     #3,R0           ;CHANGE SPLIT FOR RIGHT BOUNDARY
001526' 004767  000416         JSR     PC,SAR          ;GET DAC VALUE THAT GIVES .6% HIGH COUNT
001532' 166705  176514         SUB     FRED,R5         ;R5 = A/D PEAK TO PEAK NOISE
001536' 010567  176520         MOV     R5,APKX         ;SAVE FOR DAC NOISE CALCULATIONS
001542' 020567  176466         CMP     R5,APKLIM       ;< OR = A/D PEAK NOISE LIMIT?
001546' 003410                 BLE     ENDP            ;BRANCH IF NO ERROR
001550' 004767  000732         JSR     PC,ERCOM        ;GET ERROR PARAMETERS
                                                       ;ERROR PARAMETERS LOADED BY "ERCOM"
                                       ;*****************************************************************
001554' 104405  000000' 000000         HRDER$,BEGIN,NULL       ;;PEAK NOISE ERROR=SEE NEXT TYPE OUT
                                       ;*****************************************************************
001562' 104403  000000' 002630'        MSGN$,BEGIN,MSG5        ;ASCII MESSAGE CALL WITH COMMON HEADER
```

```
001570' 032767  000004  176220  ENDP:   BIT     #BIT2,SR1       ;DOING MULTIPLE NOISE CHANNELS?
001576' 001411                          BEQ     3$              ;NO = BYPASS
001600' 000402                          BR      2$
001602' 000167  177506          1$:     JMP     ALRMS1          ;NO DO  AGAIN.
001606' 005267  176452          2$:     INC     NCCH            ;POINT TO NEXT CH.
001612' 026767  176446  176410          CMP     NCCH,NLSTCH     ;EXCEED LAST NOISE CH?
001620' 101770                          BLOS    1$              ;NO-TEST THIS CH.
                                ;BY NOW THE PROGRAM HAS RUN THE LOGIC AND NOISE TESTS ON ALL SELECTED CHANNELS ON THIS U
001622' 032767  000002  176166  3$:     BIT     #BIT1,SR1       ;ARE WE DOING VOLTAGE SAMPLING?
001630' 001510                          BEQ     ENDP            ;NO = END PASS!
001632' 005067  176430                  CLR     CCH             ;YES = START WITH CH 0.
                                ;VOLTAGE SAMPLING HAS BEEN ENABLED -- NOW TEST IF BEFORE REPORTING EOP
001636' 026767  176424  176360  4$:     CMP     CCH,CLSTCH      ;DONE ALL CHANNELS?
001644' 003102                          BGT     ENDP            ;YES = REPORT END PASS.
001646' 005003                          CLR     R3              ;R3 WILL HOLD SAMPLE.
001650' 012700  000010                  MOV     #8.,R0          ;SET TO TAKE 8 SAMPLES
001654' 012777  001704' 176410          MOV     #6$,@VECTAD     ;SET VECTOR FOR INTERRUPT
001662' 016701  176400                  MOV     CCH,R1          ;GET CH. NUMBER
001666' 000301                          SWAB    R1              ;FIX RIGHT POSITION IN CSR.
001670' 052701  000101                  BIS     #101,R1         ;ADD INTR, ENABLE AND GO.
001674' 010177  176336          5$:     MOV     R1,@ADSR        ;START A/D.
001700' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
001704'                         6$:
                                ;-----------------------------------------------------------------
001704' 000004  000000' 001712'         PIPQ$,BEGIN,7$          ; QUEUE UP TO CONTINUE AT 7$ AND RTI
                                ;-----------------------------------------------------------------
001712' 067703  176322          7$:     ADD     @ADBR,R3        ;ADD THIS SAMPLE TO TOTAL
001716' 005300                          DEC     R0              ;ALL DONE ALL SAMPLES?
001720' 001365                          BNE     5$              ;NO = DO NEXT ONE.

001722' 006203                          ASR     R3              ;YES NOW WE
001724' 006203                          ASR     R3              ;MUST DIVIDE BY
001726' 006203                          ASR     R3              ;8 TO GET THE
001730' 005503                          ADC     R3              ;SAMPLE AVERAGE.
001732' 016700  176330                  MOV     CCH,R0          ;NOW GET CH. NUMBER
001736' 006300                          ASL     R0              ;FORM AN OFFSET
001740' 010301                          MOV     R3,R1           ;SAVE NEW SAMPLE VALUE.
001742' 166003  003162'                 SUB     RECSAM(R0),R3   ;GET THE DIFFERENCE BETWEEN
                                                                ;AND THE NEW ONE WE JUST GOT
001746' 100001                          BPL     8$              ;IF POSITIVE WE'RE OK.
001750' 005403                          NEG     R3              ;OTHERWISE MAKE IT POSTIVE.
001752' 020367  176250          8$:     CMP     R3,OFFALL       ;IS IT WITHIN TOLERANCE?
001756' 003431                          BLE     10$             ;YES DO NEXT CH

                                ;***************************************************************
                                                                ;CONVERT CCH TO ASCII AND
                                                                ;STORE AT CHANN
001760' 104420  000000' 000266'         OTOA$,BEGIN,CCH,CHANN
001766' 003402'
                                ;***************************************************************
001770' 016067  003162' 001364          MOV     RECSAM(R0),OLDS ;GET OLD VALUE
                                ;***************************************************************
                                                                ;CONVERT OLDS TO ASCII AND
                                                                ;STORE AT OLDS
```



```
001776' 104420  000000' 003362'         OTOA$,BEGIN,OLDS,OLDS
002004' 003362'
                                ;***************************************************************
002006' 010167  001360                  MOV     R1,NEWS         ;GET NEW VALUE
                                ;***************************************************************
                                                                ;CONVERT NEWS TO ASCII AND
                                                                ;STORE AT NEWS
002012' 104420  000000' 003372'         OTOA$,BEGIN,NEWS,NEWS
002020' 003372'
                                ;***************************************************************
002022' 010160  003162'                 MOV     R1,RECSAM(R0)   ;PUT NEW INTO OLD.
                                ;***************************************************************
002026' 104405  000000' 000000          ERDER$,BEGIN,NULL       ;REPORT NEW VALUE EXCEEDS THE ALLOWED TOLERANCE
                                ;***************************************************************
002034' 104403  000000' 002672'         MSGN$,BEGIN,MSG12       ;ASCII MESSAGE CALL WITH COMMON HEADER

002042' 005267  176220          10$:    INC     CCH             ;LOOK AT NEXT CHAN.
002046' 000167  177564                  JMP     4$              ;GO TEST IT

002052' 032767  000001  175736  ENDP:   BIT     #BIT0,SR1       ;TEST IF MNCKW ENABLED
002060' 001025                          BNE     1$              ;BR IF YES <DO NOT DO MULTIPLE UNITS>
002062' 006367  176202                  ASL     TEMP            ;MOVE OVER
002066' 022767  000020  176174          CMP     #BIT4,TEMP      ;TEST IF MAX. UNIT
002074' 001417                          BEQ     1$              ;BR IF MAX
002076' 036767  176166  175710          BIT     TEMP,DVID1      ;TEST IF SELECTED
002104' 001413                          BEQ     1$              ;BR IF NOT
002106' 062767  000004  176122          ADD     #4,ADSR         ;UPDATE ADDRESS
002114' 062767  000004  176116          ADD     #4,ADBR
002122' 062767  000010  176142          ADD     #10,VECTAD      ;UPDATE VECTOR
002130' 000167  176264                  JMP     LOOP            ;LOOP AND TEST NEXT UNIT
002134' 005067  176124          1$:     CLR     NCCH            ;INIT THE CHANNEL #
002140' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                ;MONITOR SHALL TEST END OF PASS
002144' 000167  176124                  JMP     RESTRT

                                ;USING SUCCESSIVE APPROXIMATION AND VERNIER DAC DEFINED IN R1.

002150' 012702  000200          SAR:    MOV     #200,R2         ;R2 = MSB OF DAC
002154' 005011                          CLR     (R1)            ;GET RID OF 'ONES'
002156' 005067  176070                  CLR     FRED            ;START WITH ZERO DAC
002162' 060267  176064          BIT:    ADD     R2,FRED         ;TRY THIS BIT
002166' 016711  176060                  MOV     FRED,(R1)       ;LOAD DAC.
002172' 005003                          CLR     R3              ;INIT HIGH COUNT
002174' 012704  001000                  MOV     #512.,R4        ;R4 = # OF SAMPLES IN A BURST
002200'                         CONV:

002200' 032767  000001  175610          BIT     #BIT0,SR1       ;IS CLOCK SELECTED?
002206' 001004                          BNE     1$              ;YES GOTO HANDLER.

002210' 052777  000001  176020          BIS     #BIT0,@ADSR     ;NO = SET GO BIT IN A/D.
002216' 000420                          BR      2$              ;GOTO 2$

002220' 004767  000172          1$:     JSR     PC,RANDY        ;GET A RANDOM NUMBER.
```

```
 768  002224' 005077  176012            CLR     @ASR            ;MAKE SURE THE CLOCK'S CSR IS CLEAR.
 769  002230' 052767  177770  000242    BIS     #177770,RNA     ;MAKE SURE OF HIGH NUMBER.
 770  002236' 016777  000236  176000    MOV     RNA,@ABR        ;SET CLOCK PRESET REG.
 771  002244' 052777  000040  175764    BIS     #BIT5,@ADSR     ;SET OVERFLOW ENABLE.
 772  002252' 012777  000011  175762    MOV     #11,@ASR        ;START CLOCK
 773  002260'                         2$:
 774  002260' 104400                    EXIT$                   ;RETURN TO MONITOR.
 775  002262'                         WAINT:
 776                                     ;-------------------------------------------------------------
 777  002262' 000004  000000' 002270'   PIRQ$,BEGIN,1$         ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 778                                     ;-------------------------------------------------------------
 779  002270' 027767  175744  175756  1$: CMP     @ADPR,EDGE      ;> OR = EDGE?
 780  002276' 103401                    BLO     2$              ;BRANCH IF <EDGE
 781  002300' 005203                    INC     R3              ;COUNT IF > OR =EDGE
 782  002302' 005304                 2$: DEC     R4              ;COUNT THROUGH BURST
 783  002304' 001335                    BNE     CONV            ;BRANCH IF NOT DONE
 784  002306' 020300                    CMP     R3,R0           ;HIGH COUNT > % OF 512 CONVERSIONS?
 785  002310' 003402                    BLE     3$              ;BRANCH TO LAVE BIT IN
 786  002312' 160267  175734            SUB     R2,FRED         ;TAKE BIT OUT
 787  002316' 006202                 3$: ASR     R2              ;NEXT BIT
 788  002320' 001320                    BNE     BIT             ;AND GO RESOLVE IT IF NOT DONE
 789  002322' 005767  175724            TST     FRED            ;CHECK FOR ALL 'ZEROES'
 790  002326' 001405                    BEQ     WRPERR          ;BRANCH IF INVALID RESULT
 791  002330' 026727  175716  000377    CMP     FRED,#377       ;CHECK FOR ALL 'ONES'
 792  002336' 001401                    BEQ     WRPERR          ;BRANCH IF INVALID RESULT
 793  002340' 000207                    RTS     PC              ;RETURN TO ADRMS1 OR ADPK1.
 794                                 ;VOLTAGE NEEDED TO PRODUCE # OF HIGH COUNTS SPECIFIED IS OUT OF THE
 795                                 ;RANGE OF THE WRAPAROUND DAC.
 796                                 ;CLEAR INTERRUPTS, PRINT MESSAGE AND DROP MODULE.
 797
 798  002342' 005077  175670         WRPERR: CLR     @ADSR           ;STOP A/D
 799  002346' 004767  000134            JSR     PC,ERCOM        ;GET ERROR PARAMETERS
 800                                                             ;ERROR PARAMETERS LOADED BY "ERCOM"
 801                                     ;*************************************************************
 802  002352' 104405  000000' 000000    HRDER$,BEGIN,NULL       ;;NOISE TEST WRAPAROUND ERROR
 803                                     ;*************************************************************
 804  002360' 005767  175662            TST     WHO             ;SEE WHICH TEST WE WERE DOING.
 805  002364' 001004                    BNE     1$              ;WHO=1 THEN RMS NOISE TEST.
 806  002366' 104403  000000' 002652'   MSGN$,BEGIN,MSG11       ;ASCII MESSAGE CALL WITH COMMON HEADER
 807  002374' 000403                    BR      2$              ;GO AHEAD.
 808  002376'                         1$:
 809  002376' 104403  000000' 002716'   MSGN$,BEGIN,MSG13       ;ASCII MESSAGE CALL WITH COMMON HEADER
 810  002404' 005267  175440         2$: INC     HRDPAS          ;UPDATE THE ERROR COUNT.
 811  002410' 005726                    TST     (SP)+           ;RESTORE STACK FROM THE JSR PC THAT
 812                                                             ;GOT US TO "SAR".
 813  002412' 000167  177152            JMP     FLOP            ;GET NEXT DIRECTIVE.
```



```
 814                                 ;GENERATE A RANDOM NUMBER
 815
 816  002416' 066767  000060  000054  RANDY:  ADD     RNB,RNA         ;THESE FOLLOW SEQUENCE OF
 817  002424' 066767  000054  000046          ADD     RNC,RNA         ; INSTRUCTIONS GENERATE
 818  002432' 005567  000042                  ADC     RNA             ; A RANDOM NUMBER
 819  002436' 066767  000036  000036          ADD     RNA,RNB         ; TO BE USED
 820  002444' 066767  000034  000030          ADD     RNC,RNB         ; TO BE LOADED
 821  002452' 005567  000024                  ADC     RNB             ; INTO THE CLOCK'S PRESET
 822  002456' 066767  000016  000020          ADD     RNA,RNC         ; BUFFER, IF SELECTED FOR TEST.
 823  002464' 066767  000012  000012          ADD     RNB,RNC         ; ONLY RANA WILL BE USED.
 824  002472' 005567  000006                  ADC     RNC
 825  002476' 000207                          RTS     PC              ;RETURN TO "CONV"
 826  002500' 063241                  RNA:    063241                  ;RANDOM TO NUMBER A.
 827  002502' 142315                  RNB:    142315                  ;RANDOM NUMBER B.
 828  002504' 127623                  RNC:    127623                  ;RANDOM NUMBER C.
 829
 830                                 ;CONVERT NOISE RESULT TO DECIMAL
 831
 832  002506' 016767  175524  175364  ERCOM:  MOV     ADSR,CSRA       ;LOAD HEADER FOR ERRORN CALL
 833  002514' 017767  175516  175360          MOV     @ADSR,ACSR
 834  002522' 016767  175300  175354          MOV     STAT,ASTAT
 835  002530' 010567  000044                  MOV     R5,DECIM        ;STACK BINARY NOISE VALUE
 836                                     ;*************************************************************
 837                                                             ;CONVERT DECIM TO ASCII AND
 838                                                             ;STORE AT DECIM
 839  002534' 104421  000000' 002600'   BTOD$,BEGIN,DECIM,DECIM
 840  002542' 002600'
 841                                     ;*************************************************************
 842  002544' 116767  000032  000363          MOVB    DECIM+2,VALUE   ;MOVE CONVERTED VALUES TO ASCII BUFFER
 843  002552' 116767  000025  000357          MOVB    DECIM+3,VALUE+2 ;FOR TYPEOUT OF ERROR
 844  002560' 116767  000020  000352          MOVB    DECIM+4,VALUE+3 ; ON RETURN
 845                                     ;*************************************************************
 846                                                             ;CONVERT NCCH TO ASCII AND
 847                                                             ;STORE AT CHANN
 848  002566' 104420  000000' 000264'   OTOA$,BEGIN,NCCH,CHANN
 849  002574' 003402'
 850                                     ;*************************************************************
 851  002576' 000207                     RTS     PC              ; EXIT TO CALLER.
 852
 853  002600' 000003                 DECIM:  .BLKW   3
```

```
                                        ;ASCII MESSAGES AND POINTERS

002606' 003065'                 MSG1:   P10                     ;UNIT #
002610' 002744'                         P2                      ;ON CHAN
002612' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002614' 002736'                         P1                      ; % A/D
002616' 002775'                         P4                      ; RMS
002620' 003011'                         P6                      ; NOISE =
002622' 003135'                         VALUE                   ; (CONVERTED BELOW) X,XX LSB (LIMIT =
002624' 003023'                         P7                      ; 0,50 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
002626' 177777                          -1                      ; MESSAGE TERMINATOR.

002630' 003065'                 MSG5:   P10                     ;UNIT #
002632' 002744'                         P2                      ;ON CHAN
002634' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002636' 002736'                         P1                      ; % A/D
002640' 003003'                         P5                      ; PEAK
002642' 003011'                         P6                      ; NOISE =
002644' 003135'                         VALUE                   ; (CONVERTED VALUE) X,XX LSB (LIMIT =
002646' 003035'                         P8                      ; 2,00 LSB) "THIS VALUE WILL CHANGE IF OPERATOR CHANGES
002650' 177777                          -1                      ; MESSAGE TERMINATOR

002652' 003065'                 MSG11:  P10                     ;UNIT #
002654' 002736'                         P1                      ; % A/D
002656' 003003'                         P5                      ; PEAK
002660' 003113'                         P13                     ; WRAPAROUND
002662' 003127'                         P15                     ; ERROR
002664' 002744'                         P2                      ; ON CHAN
002666' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002670' 177777                          -1                      ; MESSAGE TERMINATOR

002672' 003065'                 MSG12:  P10                     ;UNIT #
002674' 002736'                         P1                      ; % A/D
002676' 003127'                         P15                     ; ERROR
002700' 002744'                         P2                      ; ON CHAN
002702' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002704' 002756'                         P3                      ; ;OLD VALUE =
002706' 003364'                         OLDS+2                  ; "A/D READING 4 DIGITS
002710' 003047'                         P9                      ; NEW VALUE =
002712' 003374'                         NEWS+2                  ; "NEW A/D READING 4 DIGITS
002714' 177777                          -1                      ; MESSAGE TERMINATOR.

002716' 003065'                 MSG13:  P10                     ;UNIT #
002720' 002736'                         P1                      ; % A/D
002722' 002775'                         P4                      ; RMS
002724' 003113'                         P13                     ; WRAPAROUND
002726' 003127'                         P15                     ; ERROR
002730' 002744'                         P2                      ; ON CHAN
002732' 003406'                         CHANN+4                 ; (CHAN NUMBER 2 DIGITS)
002734' 177777                          -1                      ; MESSAGE TERMINATOR.
```



```
002736' 040445  042057  000040  P1:     .ASCIZ  '%A/D '
002744' 047440  020116  044103  P2:     .ASCIZ  ' ON CHAN '
002752' 047101  000040
002756' 020073  046117  020104  P3:     .ASCIZ  '; OLD VALUE = '
002764' 040526  052514  020105
002772' 020075    000
002775'    122  051515  000040  P4:     .ASCIZ  'RMS '
003002'    000                          .BYTE
003003'    120  040505  020113  P5:     .ASCIZ  'PEAK '
003010'    000
003011'    116  044517  042523  P6:     .ASCIZ  'NOISE = '
003016' 036440  000040
003022'    000                          .BYTE
003023'    060  032456  020060  P7:     .ASCIZ  '0,50 LSB)'
003030' 051514  024502    000
003035'    062  030056  020060  P8:     .ASCIZ  '2,00 LSB)'
003042' 051514  024502    000

003047'    040  042516  020127  P9:     .ASCIZ  ' NEW VALUE = '
003054' 040526  052514  020105
003062' 020075    000
003065'    045  047115  040503  P10:    .ASCII  '%MNCAD (A/D) UNIT #'
003072' 020104  040450  042057
003100' 020051  047125  052111
003106' 021440

003110'    060    040     000  P10U:   .BYTE   60,40,0
003113'    127  040522  040520  P13:    .ASCIZ  'WRAPAROUND '
003120' 047522  047125  020104
003126'    000
003127'    105  051122  051117  F15:    .ASCIZ  'ERROR'
003134'    000

003135'    130  054056  020130  VALUE:  .ASCIZ  'X,XX LSB (LIMIT = '
003142' 051514  020102  046050
003150' 046511  052111  036440
003156' 000040
003160'    000                          .BYTE

          003162'                       .EVEN
003162' 000100                  RECSAM: .BLKW   64.              ;USED TO STORE A/D RESULTS ON UP TO 64. CHANS.

003362' 000003                  OLDS:   .BLKW   3                ;USED FOR STORE OF ASCII OF OLD SAMPLE VALUE.
003370' 000000                          .WORD   0
003372' 000003                  NEWS:   .BLKW   3                ;USED FOR STORE OF ASCII NEW SAMPLE VALUE.
003400' 000000                          .WORD   0
003402' 000003                  CHANN:  .BLKW   3                ;USED FOR STORAGE OF ASCII OF CHAN. NUMBER.
003410' 000000                          .WORD   0
          000001                        .END
```

```
ABR     000244R   350#  473*  770#
ACSR    000102R   320#  452*  493*  520*  542*  833#
ADBR    000240R   345#  379*  461   511   515   539   559*  560   573   683   742*  779
ADDR    000006R   286#  376
ADDR22= 001000    333#
ADPK1   001450R   626   635#
ADRMS1  001314R   609#  659
ADSR    000236R   343#  377*  422   443*  451   452   453*  460*  474*  489   492   493   513*
                  519   520   537*  541   542   561*  570*  602*  617*  640*  676*  741*  764*
                  771*  798*  832   833
APKLIM  000234R   339#  397   648
APKX    000262R   362#  647*
ARMLIM  000232R   338#  386   625
ARMX    000260R   361#  624*
ASB     000106R   324#
ASR     000242R   349#  435   475*  481   486   487*  768*  772*
ASTAT   000104R   322#  486*  834*
AWAS    000110R   325#
BEGIN   000000R   283#  386   397   430   431   446   447   455   457   479   480   495   502
                  509   510   514   522   528   538   544   549   557   558   571   599   630
                  632   653   655   677   680   704   712   720   727   729   746   777   802
                  806   809   839   848
BIT     002162R   755#  788
BIT0  = 000001    333#  365   432   470   569   734   761   764
BIT1  = 000002    333#  664
BIT10 = 002000    333#
BIT11 = 004000    333#
BIT12 = 010000    333#
BIT13 = 020000    333#
BIT14 = 040000    333#
BIT15 = 100000    333#
BIT2  = 000004    333#  656
BIT3  = 000010    333#
BIT4  = 000020    333#  737
BIT5  = 000040    333#  474   771
BIT6  = 000100    333#  569
BIT7  = 000200    333#
BIT8  = 000400    333#
BIT9  = 001000    333#
BREAK$= 104407    333#  430   431   446   447   479   480   509   510   557   558
BR1     000012R   288#  605
BR2     000013R   289#
BTOD$ = 104421    333#  386   397   839
CCH     000266R   364#  562*  567   583   587*  588   590   666*  668   673   691   704   731*
CDATA$= 104412    333#
CHANN   003402R   704   848   858   868   882   889   902   949#
CLSTCH  000224R   335#  590   668
CONFIG  000056R   308#
CONV    002200R   759#  783
CSRA    000100R   318#  451*  492*  519*  541*  832*
DAC     000240R   344#  618   641
DATCK$= 104411    333#
DATER$= 104404    333#
DECIM   002600R   386   390   391   392   397   401   402   403   835*  839   842   843   844
                  853#
DVID1   000014R   290#  739
```



```
EDGE    000254R   359#  615*  779
EENDP   002052R   665   669   734#
ENDIT$= 104413    333#  746
ENDP    001570R   649   656#  813
END$  = 104410    333#  457   502
ERCOM   002506R   627   650   799   832#
ERRTYP  000106R   323#
EXIT$ = 104400    333#  514   538   571   677   774
FRED    000252R   358#  620   623   643   646   754*  755*  756   786*  789   791
GETPA$= 104415    333#
GWBUF$= 104414    333#
HRDCNT  000044R   303#
HRDER$= 104405    333#  455   495   522   544   630   653   727   802
HRDPAS  000050R   305#  810*
ICONT   000036P   300#
ICOUNT  000040R   301#
IDNUM   000122R   330#
INIT    000030R   297#
INTFLG  000250R   357#  441*  449   462*
INTR    000120R   329#
LOG1    000446R   422#
LOG2    000452R   429#
LOG3    000476R   433   441#
LOG4    000604R   450   470#
LOG5    000730R   471   490   508#
LOG6    001022P   518   536#
LOG7    001102R   540   556#
LOOP    000420R   407#  744
LOOP1   000274R   375#
MAP22$= 104416    333#
MODNAM  000000R   284#
MODSP   000224R   298   331#
MSGN$ = 104403    333#  632   655   729   806   809
MSGS$ = 104402    333#
MSG$  = 104401    333#
MSG1    002606R   632   856#
MSG11   002652R   806   876#
MSG12   002672R   729   885#
MSG13   002716R   809   896#
MSG5    002630R   655   866#
NCCH    000264R   363#  381*  606*  610   613   636   660*  661   745*  848
NEWS    003372R   716*  720   893   947#
NLSTCH  000230R   337#  588   661
NULL  = 000000    333#  455   495   522   544   630   653   727   802
OFFALL  000226R   336#  698
OLD$    003362R   708*  712   891   945#
OPEN  = 000000    285   291   292   293   294   311   312   313   314   315   316   317   318
                  320   322   324   325   327   328   329   333#
OTOA$ = 104420    333#  704   712   720   848
PASCNT  000034R   299#
PIRQ$ = 000004    333#  528   549   599   680   777
POPSP = 005726    333#
POPSP2= 022626    333#
PRTY  = 000000    333#
PRTY0 = 000000    289   333#
PRTY1 = 000040    333#
```

```
PRTY2 = 000100          333#
PRTY3 = 000140          333#
PRTY4 = 000200          333#
PRTY5 = 000240          333#
PRTY6 = 000300          288     333#
PRTY7 = 000340          333#
PS    = 177776          333#
PSW   = 177776          333#
PUSH  = 005746          333#
PUSH2 = 024646          333#
P1      002736R         859     869     877     886     897     904#
P10     003065R         856     866     876     885     896     925#
P10U    003110R         414*    929#
P13     003113R         879     899     930#
P15     003127R         880     887     900     933#
P2      002744R         857     867     881     888     901     905#
P3      002756R         890     907#
P4      002775R         860     898     910#
P5      003003R         870     878     912#
P6      003011R         861     871     914#
P7      003023R         390*    391*    392*    863     917#
P8      003035R         401*    402*    403*    873     919#
P9      003047R         892     922#
RANDY   002416R         767     816#
RAND$ = 104417          333#
RANNUM  000054R         307#
RECSAM  003162R         585*    615     694     708     724*    943#
RESTRT  000274R         326     372#    748
RESTØ   001270R         592     602#
RES1    000056R         309#
RES2    000060R         310#
RNA     002500R         769*    770     816*    817*    818*    819     822     826#
RNB     002502R         816     819*    820*    821*    823     827#
RNC     002504R         817     820     822*    823*    824*    828#
RSTRT   000112R         326#
SAR     002150R         619     622     642     645     752#
SBADR   000102R         319#
SOFCNT  000042R         302#
SOFER$= 104406          333#
SOFPAS  000046R         304#
SPOINT  000032R         298#
SPSIZ = 000040            1#    331
SR1     000016R         291#    432     470     656     664     734     761
SR2     000020R         292#
SR3     000022R         293#
SR4     000024R         294#
START   000274R         297     373#
STAT    000026R         296#    834
SVRØ    000062R         311#
SVR1    000064R         312#
SVR2    000066R         313#
SVR3    000070R         314#
SVR4    000072R         315#
SVR5    000074R         316#
SVR6    000076R         317#
SYSCNT  000052R         306#
```



```
TEMP    000270R         365#    375*    407     736*    737     739
TMP     000256R         360#    567*    568*    569*    570     610*    611*    612*    617     636*    637*    638*    640
TRPDFD= 000022          333#
VALUE   003135R         842*    843*    844*    862     872     936#
VECTAD  000272R         366#    380*    442*    512*    536*    563*    603     672*    743*
VECTOR  000010R         287#    380
WAINT   002262R         604     775#
WASADR  000104R         321#
WDFR    000116R         328#
WDTO    000114R         327#
WHO     000246R         356#    616*    639*    804
WRPERR  002342R         790     792     798#
XFLAG   000005R         285#
.     = 003412R         853#    942#    943#    945#    947#    949#

. ABS.  000000      000
        003412      001


 ERRORS DETECTED:  Ø
 DEFAULT GLOBALS GENERATED:  Ø

 XMNAA0,XMNAA0/SOL/CRF:SYM=DDXCOM,XMNAA0
 RUN-TIME:  2  2  .3 SECONDS
 RUN-TIME RATIO: 147/5=26.8
 CORE USED:  7K  (13 PAGES)
```

```
.REM _
```

IDENTIFICATION

PRODUCT CODE: AC-F234A-MC

PRODUCT NAME: CXNCBA0 NCV-11A MODULE

PRODUCT DATE: FEB 1979

MAINTAINER: DEC/X11 SUPPORT GROUP

.PAGE

1. ABSTRACT

"NCB" IS AN "IOMODX" THAT EXERCISES ONE NCV-11A INTERFACE. THE NCV-11A INTERFACE DOES MEMORY INCREMENTS VIA NPR UNTIL A WORD OR BYTE REACHES MAXIMUM CAPACITY AND ATTEMPS TO OVERFLOW. AT THIS TIME AN INTERRUPT IS GENERATED AT BR LEVEL 6. THE INTERFACE ALSO DOES TRANSFERS OF DATA TO SERIAL LOCATIONS IN CORE VIA NPR. THIS MODE IS TERMINATED BY A WORD COUNT OVERFLOW AND CONSEQUENT INTERRUPT. THE RATE OF INCREMENT OR TRANSFER IS SET BY A CLOCK SIGNAL DEVELOPED ON THE NCV11 CONTROLLER.

2. REQUIREMENTS

HARDWARE: NCV-11A INTERFACE
STORAGE: NCB MODULE REQUIRES 1400. WORDS OF STORAGE

3. PASS DEFINITION

ONE PASS OF NCA MODULE CONSISTS OF EIGHTY ITERATIONS OF EACH BASIC TEST SEQUENCE, WHICH RESULTS IN:

200 PROGRAM INTERRUPTS, 87,000 NON-PROCESSOR REQUESTS.

4. EXECUTION TIME

NCB RUNNING ALONE ON PDP-11/34 TAKES APPROXIMATELY 60 SECONDS.

5. CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 172760, VECTOR: 370, BR1: 6, DEVCNT: N/A

REQUIRED PARAMETERS:

NONE

6. DEVICE/OPTION SETUP

NONE.

7. MODULE OPERATION

THE MODULE CONSISTS OF A LOGIC TEST, LIST MODE, AND MATRIX MODE DATA TRANSFERS. THE LOGIC TEST PROVIDES A QUICK TEST OF THE MAJOR REGISTER FOR OPERATION. THE LIST MODE TRANSFERS ARE FIRST EXECUTED IN MAINTENCE MODE AND AT FULL SPEED. THE MODULE WILL THEN COLLECT DATA IN MATRIX MODE AT FULL SPEED. THE SEQUENCE IS REPEATED UNTIL THE PASS COUNTER IS EXHAUSTED. UPON COMPLETION, AN END OF PASS IS REPORTED AND THE MODULE IS RESTARTED.

8. OPERATION OPTIONS

SR1 IS USED TO INHIBIT TESTING MODES OF OPERATION OF THE NCV11A. IF THE BITS ARE USED, THE INTERATION COUNT LOCATION "PASS" MUST BE MODIFIED TO EXTEND THE EXECUTION TIME BEFORE THE END OF PASS REPORT.

```
SR1 BIT0 = 1    INHIBIT MATRIX WORD INCREMENT MODE.
SR1 BIT1 = 1    INHIBIT LIST MODE.
```

9. NON STANDARD PRINTOUTS

NONE. ALL PRINTOUTS HAVE STANDARD MEANINGS AS REPRESENTED IN DEC/X11 DOCUMENTATION.

10. MODULE TEST ENVIROMENT

THE NCA MODULE IS KNOWN TO OPERATE UNDER THIS ENVIROMENT:

```
#1      PDP-11/34 CPU WITH 64K          #2      PDP-11/40 CPU WITH 28K
        TC11    2 DRIVES                        RK11-D  1 DRIVE
        TM11    1 DRIVE                         TM11    1 DRIVE
        VSV01   1 SCOPE                         NCV11A  1 UNIT
        NCV11A  1 UNIT                          KW11L   1 UNIT
        LP11    1 UNIT

#3      PDP-11/34 CPU WITH 32K
        RK11-D  1 DRIVE
        NCV11A  1 UNIT
        KW11L   1 UNIT
```

```
                                        .LIST   SEQ,BIN
                                        .NLIST  MC,MD,CND
                                        .LIST   ME
                                        .TITLE  NCBA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDNCOM  VERSION  6      23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************************
000000-                         BEGIN:
000000- 041516  040502  040     MODNAM: .ASCII  /NCBA / ;MODULE NAME.
000005- 000                     XFLAG:  .BYTE   0PEN                    ;USED TO KEEP TRACK OF WBUFF USAGE
000006- 172760                  ADDR:   172760+0                        ;1ST DEVICE ADDR.
000010- 000370                  VECTOR: 370+0                           ;1ST DEVICE VECTOR.
000012- 300                     BR1:    .BYTE   PRTY6+0                 ;1ST BR LEVEL.
000013- 000                     BR2:    .BYTE   PRTY0+0                 ;2ND BR LEVEL.
000014- 000001                  DVID1:  0+1                             ;DEVICE INDICATOR 1.
000016- 000000                  SR1:    OPEN                            ;SWITCH REGISTER 1
000020- 000000                  SR2:    OPEN                            ;SWITCH REGISTER 2
000022- 000000                  SR3:    OPEN                            ;SWITCH REGISTER 3
000024- 000000                  SR4:    OPEN                            ;SWITCH REGISTER 4
                                ;*****************************************************************************
000026- 150000                  STAT:   150000                          ;STAT"S WORD.
000030- 000310-                 INIT:   START                           ;MODULE START ADDR.
000032- 000252-                 SPOINT: MODSP                           ;MODULE STACK POINTER.
000034- 000000                  PASCNT: 0                               ;PASS COUNTER.
000036- 000120                  ICONT:  80.                             ;# OF ITERATIONS PER PASS=80.
000040- 000000                  ICOUNT: 0                               ;LOC TO COUNT ITERATIONS
000042- 000000                  SOFCNT: 0                               ;LOC TO SAVE TOTAL SOFT ERRORS
000044- 000000                  HRDCNT: 0                               ;LOC TO SAVE TOTAL HARD ERRORS
000046- 000000                  SOFPAS: 0                               ;LOC TO SAVE SOFT ERRORS PER PASS
000050- 000000                  HRDPAS: 0                               ;LOC TO SAVE HARD ERRORS PER PASS
000052- 000000                  SYSCNT: 0                               ;# OF SYS ERRORS ACCUMULATED
000054- 000000                  RANNUM: 0                               ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056-                         CONFIG:                                 ;RESERVED FOR MONITOR USE
000056- 000000                  RES1:   0                               ;RESERVED FOR MONITOR USE
000060- 000000                  RES2:   0                               ;RESERVED FOR MONITOR USE
000062- 000000                  SVR0:   OPEN                            ;LOC TO SAVE R0.
000064- 000000                  SVR1:   OPEN                            ;LOC TO SAVE R1.
000066- 000000                  SVR2:   OPEN                            ;LOC TO SAVE R2.
000070- 000000                  SVR3:   OPEN                            ;LOC TO SAVE R3.
000072- 000000                  SVR4:   OPEN                            ;LOC TO SAVE R4.
000074- 000000                  SVR5:   OPEN                            ;LOC TO SAVE R5.
000076- 000000                  SVR6:   OPEN                            ;LOC TO SAVE R6.
000100- 000000                  CSRA:   OPEN                            ;ADDR OF CURRENT CSR.
000102-                         SBADR:                                  ;ADDR OF GOOD DATA, OR
000102- 000000                  ACSR:   OPEN                            ;CONTENTS OF CSR.
000104-                         WBADR:                                  ;ADDR OF BAD DATA, OR
000104- 000000                  ASTAT:  OPEN                            ;STATUS REG CONTENTS.
000106-                         ERRTYP:                                 ;TYPE OF ERROR
000106- 000000                  ASB:    OPEN                            ;EXPECTED DATA.
000110- 000000                  AWAS:   OPEN                            ;ACTUAL DATA.
000112- 000310-                 RSTRT:  RESTRT                          ;RESTART ADDRESS AFTER END OF PASS
000114- 000000                  WDTO:   OPEN                            ;WORDS TO MEMORY PER ITERATION
000116- 000000                  WDFR:   OPEN                            ;WORDS FROM MEMORY PER ITERATION
000120- 000000                  INTR:   OPEN                            ;# OF INTERRUPTS PER ITERATION
000122- 000000                  IDNUM:  0                               ;MODULE IDENTIFICATION NUMBER=0
```



```
000124- 003500-                 RBUFVA: BUF0                            ;READ BUFFER VIRTUAL ADDRESS
000126- 000000                  RBUFPA: OPEN                            ;READ BUFFER PHYSICAL ADDRESS
000130- 000000                  RBUFEA: OPEN                            ;READ BUFFER EA BITS
000132- 000400                  RBUFSZ: 256.                            ;SIZE OF THE READ BUFFER
000134- 000000                  WBUFPA: OPEN                            ;WRITE BUFFER PHYSICAL ADDRESS
000136- 000000                  WBUFEA: OPEN                            ;WRITE BUFFER EA BITS
000140- 000000                  WBUFRQ: 0                               ;WRITE BUFFER SIZE REQUESTED
000142- 000000                  WBUFSZ: OPEN                            ;WRITE BUFFER SIZE AVAILABLE
000144- 000000                  CDERCT: OPEN                            ;CDATA/DATCK ERROR COUNT
000146- 000000                  CDWDCT: OPEN                            ;CDATA/DATCK WORD COUNT
000150- 000000                  FREE:   OPEN                            ;RESERVED FOR FUTURE USE
000252-                         MODSP:                                  ;*****************************************************************************
                                        $TX=0
                                        ABASE=0
                                        AVECT1=0
                                        TSTDMA=BIT3
                                        ENDDMA=BIT8
                                        TESTZ=BIT1
                                        TSTCON=BIT2
                                        TSTDMA=BIT3
                                        CLRALL=BIT11
000252- 000000                  CSR:    ABASE
000254- 000002                  OFF:    ABASE+2
000256- 000004                  WCR:    ABASE+4
000260- 000006                  BAR:    ABASE+6
000262- 000010                  SFR:    ABASE+10
000264- 000012                  ADM:    ABASE+12
000266- 000014                  JOY:    ABASE+14
000270- 000016                  BAR1:   ABASE+16
000271- 000001                  CSRHB:  ABASE+1
000274- 000011                  SFRHB:  ABASE+11
000276- 000000                  VECTA0: AVECT1
000300- 000002                  VECTA1: AVECT1+2
000302- 000004                  VECTB0: AVECT1+4
000304- 000006                  VECTB1: AVECT1+6
000306- 000000                  $TEMP:  0
000310-                         START:
000310-                         RESTRT:
000310- 012777  004000  177744  LOGIC:  MOV     #CLRALL,@SFR            ;CLEAR NCV11
000316- 012700  000252-                 MOV     #CSR,R0                 ;LOAD ADDRESS POINTER
000322- 016701  177460                  MOV     ADDR,R1                 ;GET BUS ADDRESS
000326- 010120                  1$:     MOV     R1,(R0)+                ;LOAD DEVICE ADDRESS
000330- 062701  000002                  ADD     #2,R1                   ;UPDATE VALUE
000334- 020027  000272-                 CMP     R0,#CSRHB               ;TEST IF DONE
000340- 001372                          BNE     1$
000342- 016710  177440                  MOV     ADDR,(R0)               ;LOAD BYTE ADDRESSES
000346- 005720                          INC     (R0)+
000350- 016710  177432                  MOV     ADDR,(R0)               ;LOAD BYTE OF SFR
000354- 062720  000011                  ADD     #11,(R0)+
000360- 010167  177424                  MOV     VECTOR,R1
000364- 010120                  2$:     MOV     R1,(R0)+                ;LOAD VECTORS
000366- 062701  000002                  ADD     #2,R1                   ;UPDATE
000372- 022700  000306-                 CMP     #VECTB1+2,R0
000376- 001372                          BNE     2$                      ;BR UNTIL DONE
000400- 016767  177402  177472          MOV     ADDR,CSRA               ;LOAD FOR TYPE OUT
```

```
                                ;*****************************************************
                                ;VERIFY THE 8 NCV11 BUS ADDRESSES RESPOND
                                ;*****************************************************
000406'                         TST1:
000406' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000412' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000416' 005777  177630                  TST     @CSR                    ;ADDRESS
000422' 005777  177626                  TST     @DFF                    ;
000426' 005777  177624                  TST     @WCR                    ;THE
000432' 005777  177622                  TST     @BAR                    ;
000436' 005777  177620                  TST     @SFR                    ;NCV11
000442' 005777  177616                  TST     @ADM                    ;
000446' 005777  177616                  TST     @BAR1                   ;ADDRESSES
                                ;*****************************************************
                                ;FLOAT A 1 ACROSS 10 BITS OF THE COMMAND/STATUS REG.
                                ;*****************************************************
000452'                         TST2:
000452' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000456' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000462' 012767  004000  177616          MOV     #BIT11,$TEMP            ;LOAD INITIAL REG. VALUE
000470' 016777  177612  177554  1$:     MOV     $TEMP,@CSR              ;LOAD CSR REG.
000476' 017767  177550  177400          MOV     @CSR,ASTAT              ;READ CSR
000504' 016767  177576  177370          MOV     $TEMP,ACSR              ;LOAD EXPECTED
000512' 052767  000200  177362          BIS     #BIT7,ACSR              ;FUDGE THE "READY" BIT
000520' 026767  177356  177356          CMP     ACSR,ASTAT              ;COMPARE THE VALUES
000526' 001403                          BEQ     2$                      ;;/BR IF SAME
                                        ;*****************************************************
000530' 104405  000000' 000000          HRDERS,BEGIN,NULL       ;UNEXPECTED VALUE IN THE CSR REGISTER
                                        ;*****************************************************
000536'                         2$:
000536' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000542' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000546' 006267  177534                  ASR     $TEMP                   ;TRY THE NEXT DATA BIT
000552' 022767  000001  177526          CMP     #1,$TEMP                ;TEST IF NOW BIT 0
000560' 001343                          BNE     1$                      ;BR IF NOT
                                ;*****************************************************
                                ;VERIFY THAT "CLEAR ALL" CLEARS THE CSR REGISTER
                                ;*****************************************************
000562'                         TST3:
000562' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000566' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000572' 012777  001776  177452          MOV     #1776,@CSR              ;LOAD CSR REG.
000600' 012777  004000  177454          MOV     #CLRALL,@SFR            ;GENERATE "CLR ALL L"
000606' 012767  000200  177266          MOV     #BIT7,ACSR              ;LOAD EXPECTED VALUE
000614' 017767  177432  177262          MOV     @CSR,ASTAT              ;READ THE CSR REG.
000622' 026767  177254  177254          CMP     ACSR,ASTAT              ;COMPARE VALUES
000630' 001403                          BEQ     4$4     ;;/BR IF SAME
                                        ;*****************************************************
000632' 104405  000000' 000000          HRDERS,BEGIN,NULL       ;"CLR ALL L" FAILED TO CLEAR CSR REG.
                                        ;*****************************************************
```



```
                                ;*****************************************************
                                ;VERIFY LOW BYTE OPERATION OF THE "CSR" REGISTER
                                ;*****************************************************
000640'                         TST4:
000640' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000644' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000650' 012777  003636  177374          MOV     #3636,@CSR              ;LOAD CSR REGISTER
000656' 012767  003600  177216          MOV     #3600,ACSR              ;LOAD EXPECTED VALUE
000664' 105077  177362                  CLRB    @CSR                    ;CLEAR LOW BYTE
000670' 017767  177356  177206          MOV     @CSR,ASTAT              ;READ STATUS REG.
000676' 026767  177200  177200          CMP     ACSR,ASTAT              ;COMPARE VALUES
000704' 001403                          BEQ     1$                      ;;/BR IF SAME
                                        ;*****************************************************
000706' 104405  000000' 000000          HRDERS,BEGIN,NULL       ;;CLEARING LOW BYTE OF THE CSR CHANGED THE HIGH BYTE
                                        ;*****************************************************
000714' 012777  003636  177330  1$:     MOV     #3636,@CSR              ;LOAD CSR REGISTER
000722' 012767  000236  177152          MOV     #236,ACSR               ;LOAD EXPECTED VALUE
000730' 105077  177336                  CLRB    @CSRHB                  ;CLEAR HIGH BYTE OF THE CSR
000734' 017767  177312  177142          MOV     @CSR,ASTAT              ;READ STATUS
000742' 026767  177134  177134          CMP     ACSR,ASTAT              ;COMPARE VALUES
000750' 001403                          BEQ     TST5    ;;/BR IS SAME
                                        ;*****************************************************
000752' 104405  000000' 000000          HRDERS,BEGIN,NULL       ;;CLEARING HIGH BYTE OF CSR CHANGED THE LOW BYTE
                                        ;*****************************************************
                                ;*****************************************************
                                ;FLOAT A 1 ACROSS 4 BITS OF THE SPECIAL FUNCTION REGISTER
                                ;*****************************************************
000760'                         TST5:
000760' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000764' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000770' 012767  000020  177310          MOV     #BIT4,$TEMP             ;LOAD INITIAL REG. VALUE
                                
000776' 016777  177304  177256  1$:     MOV     $TEMP,@SFR              ;LOAD SFR REG.
001004' 017767  177252  177072          MOV     @SFR,ASTAT              ;READ SFR
001012' 016767  177270  177062          MOV     $TEMP,ACSR              ;LOAD EXPECTED
001020' 026767  177056  177056          CMP     ACSR,ASTAT              ;COMPARE THE VALUES
001026' 001403                          BEQ     2$                      ;;/BR IF SAME
                                        ;*****************************************************
001030' 104405  000000' 000000          HRDERS,BEGIN,NULL       ;;UNEXPECTED VALUE IN THE SFR REGISTER
                                        ;*****************************************************
001036'                         2$:
001036' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001042' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001046' 006267  177334                  ASR     $TEMP                   ;TRY THE NEXT DATA BIT
001052' 022767  000001  177226          CMP     #1,$TEMP                ;TEST IF NOW BIT 0
001060' 001346                          BNE     1$                      ;BR IF NOT
```

```
                                        ;*****************************************************
                                        ;VERIFY THAT CLEARING HIGH BYTE OF SFR DOES NOT CLEAR LOW BYTE
                                        ;*****************************************************
001062'                                 ;ST6:
001062' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001064' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001066' 012777  000014  177162          MOV     #14,@SFR        ;LOAD THE S.F. REGISTER
001074' 012767  000014  176774          MOV     #14,ACSR        ;LOAD THE EXPECTED VALUE
001102' 105077  177162                  CLRB    @SFR+1          ;CLEAR HIGH BYTE OF S.F. REG.
001106' 017767  177144  176764          MOV     @SFR,ASTAT      ;READ THE REGISTER
001114' 026767  176756  176756          CMP     ACSR,ASTAT      ;COMPARE THE VALUES
001122' 001403                          BEQ     TST7            ;;/BR IF SAME
                                        ;*****************************************************
001124' 104405  000000' 000000          ERROR$,BEGIN,NULL       ;;CLEARING HIGH BYTE OF CSR REG. CHANGED THE LOW BYTE AL
                                        ;*****************************************************


                                        ;*****************************************************
                                        ;VERIFY THAT "CLEAR ALL" CLEARS THE SFR REGISTER
                                        ;*****************************************************
001136'                                 ;ST7:
001136' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001140' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001142' 012777  000016  177106          MOV     #16,@SFR        ;LOAD SFR REG.
001150' 052777  004000  177100          BIS     #CLRALL,@SFR    ;GENERATE "CLR ALL L"
001156' 005067  176714                  CLR     ACSR            ;LOAD EXPECTED VALUE
001162' 017767  177070  176710          MOV     @SFR,ASTAT      ;READ THE SFR REG.
001170' 001403                          BEQ     TST10           ;;/BR IF SAME
                                        ;*****************************************************
001172' 104405  000000' 000000          ERROR$,BEGIN,NULL       ;;"CLR ALL L" FAILED TO CLEAR SFR REG.
                                        ;*****************************************************


                                        ;*****************************************************
                                        ;FLOAT A 1 ACROSS THE WORD COUNT REGISTER
                                        ;*****************************************************
001204'                                 ;ST10:
001204' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001206' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001210' 012767  000001  176660          MOV     #BIT0,ACSR      ;LOAD EXPECTED VALUE

001216' 012777  004000  177032  1$:     MOV     #CLRALL,@SFR    ;RESET THE DEVICE
001224' 016777  176646  177020          MOV     ACSR,@WCR       ;LOAD WORD COUNT "A"
001232' 017767  177014  176640          MOV     @WCR,ASTAT      ;READ WORD COUNT
001240' 026767  176632  176632          CMP     ACSR,ASTAT      ;COMPARE VALUES
001246' 001403                          BEQ     2$              ;;/BR IF SAME
                                        ;*****************************************************
001250' 104405  000000' 000000          ERROR$,BEGIN,NULL       ;;;WORD COUNT REG. IN ERROR
                                        ;*****************************************************

001256'                                 2$:
001256' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001260' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001262' 006367  176604                  ASL     ACSR            ;CHANGE THE DATA
001266' 001351                          BNE     1$              ;BR IF MORE DATA TO LOAD
```



```
                                        ;*****************************************************
                                        ;FLOAT A 1 ACROSS THE BUS ADDRESS REGISTER
                                        ;*****************************************************
001300'                                 ;ST11:
001300' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001302' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001304' 012767  000001  176564          MOV     #BIT0,ACSR      ;LOAD EXPECTED VALUE

001312' 012777  004000  176736  1$:     MOV     #CLRALL,@SFR    ;RESET THE DEVICE
001320' 016777  176552  176726          MOV     ACSR,@BAR       ;LOAD BUS ADDRESS "A"
001326' 017767  176722  176544          MOV     @BAR,ASTAT      ;READ BUS ADDRESS
001334' 026767  176536  176536          CMP     ACSR,ASTAT      ;COMPARE VALUES
001342' 001403                          BEQ     2$              ;;/BR IF SAME
                                        ;*****************************************************
001344' 104405  000000' 000000          ERROR$,BEGIN,NULL       ;;BUS ADDRESS REG. IN ERROR
                                        ;*****************************************************

001352'                                 2$:
001352' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001354' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001356' 006367  176510                  ASL     ACSR            ;CHANGE THE DATA
001362' 001351                          BNE     1$              ;BR IF MORE DATA TO LOAD
                                        ;*****************************************************
                                        ;FLOAT A 1 ACROSS THE OFFSET REGISTER
                                        ;*****************************************************
001364'                                 ;ST12:
001364' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001366' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001370' 012767  000001  176470          MOV     #BIT0,ACSR      ;LOAD EXPECTED VALUE

001376' 012777  004000  176642  1$:     MOV     #CLRALL,@SFR    ;RESET THE DEVICE
001404' 016777  176456  176626          MOV     ACSR,@OFF       ;LOAD OFFSET "A"
001412' 017767  176622  176450          MOV     @OFF,ASTAT      ;READ OFFSET
001420' 026767  176442  176442          CMP     ACSR,ASTAT      ;COMPARE VALUES
001426' 001403                          BEQ     2$              ;;/BR IF SAME
                                        ;*****************************************************
001430' 104405  000000' 000000          ERROR$,BEGIN,NULL       ;;OFFSET REG. IN ERROR
                                        ;*****************************************************

001436'                                 2$:
001436' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001440' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001442' 006367  176414                  ASL     ACSR            ;CHANGE THE DATA
001446' 001351                          BNE     1$              ;BR IF MORE DATA TO LOAD
```

```
                                        ;*****************************************************
                                        ;VERIFY NO DUAL REGISTER SELECTION
                                        ;*****************************************************
001470-                                 1ST13:
001470- 104407  000000-                         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001474- 104407  000000-                         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
                                        ;LOAD DIFFERENT NUMBERS INTO THE  WCR, BAR AND OFF REGISTERS
001500- 012777  004000  176554                  MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
001506- 012777  011111  176540                  MOV     #11111,@OFF             ;LOAD OFFSET REGISTER
001514- 012777  022222  176534                  MOV     #22222,@WCR             ;LOAD W.C. REGISTER
001522- 012777  033333  176530                  MOV     #33333,@BAR             ;LOAD B.A. REGISTER
001530- 012777  000036  176524                  MOV     #36,@SFR                ;LOAD SPECIAL FUNCTION REGISTER
001536- 012777  007636  176506                  MOV     #7636,@CSR              ;LOAD COMMAND/STATUS REGISTER
                                        ;NOW READ EACH REGISTER AND CHECK THE VALUE
001544- 012767  011111  176330                  MOV     #11111,ACSR             ;LOAD EXPECTED VALUE
001552- 017767  176476  176324                  MOV     @OFF,ASTAT              ;READ A OFFSET REG.
001560- 026767  176316  176316                  CMP     ACSR,ASTAT              ;COMPARE
001566- 001403                                  BEQ     1$                      ;/BR IF SAME
                                                ;*************************************************************
001570- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;DUAL ADDRESS ERROR
                                                ;*************************************************************
001576- 012767  022222  176276          1$:     MOV     #22222,ACSR             ;LOAD EXPECTED VALUE
001604- 017767  176446  176272                  MOV     @WCR,ASTAT              ;READ W.C. REG.
001612- 026767  176264  176264                  CMP     ACSR,ASTAT              ;COMPARE
001620- 001403                                  BEQ     2$                      ;/BR IF SAME
                                                ;*************************************************************
001622- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;DUAL ADDRESS ERROR
                                                ;*************************************************************
001630- 012767  033333  176244          2$:     MOV     #33333,ACSR             ;LOAD EXPECTED VLAUE
001636- 017767  176416  176240                  MOV     @BAR,ASTAT              ;READ B.A. REG.
001644- 026767  176232  176232                  CMP     ACSR,ASTAT              ;COMPARE
001652- 001403                                  BEQ     3$                      ;/BR IF SAME
                                                ;*************************************************************
001654- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;DUAL ADDRESS ERROR
                                                ;*************************************************************
001662- 012767  000036  176212          3$:     MOV     #36,ACSR                ;LOAD EXPECTED VALUE
001670- 017767  176366  176206                  MOV     @SFR,ASTAT              ;READ SPECIAL FUNCTION REG
001676- 026767  176200  176200                  CMP     ACSR,ASTAT              ;COMPARE
001704- 001403                                  BEQ     4$                      ;/BR IF SAME
                                                ;*************************************************************
001706- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;DUAL ADDRESS ERROR
                                                ;*************************************************************
001714- 012767  007636  176160          4$:     MOV     #7636,ACSR              ;LOAD EXPECTED VALUE
001722- 017767  176324  176154                  MOV     @CSR,ASTAT              ;READ COMMAND/STATUS REGISTER
001730- 026767  176146  176146                  CMP     ACSR,ASTAT              ;COMPARE
001736- 001403                                  BEQ     5$                      ;/BR IF SAME
                                                ;*************************************************************
001740- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;DUAL ADDRESS ERROR
                                                ;*************************************************************
001746- 012777  004000  176306          5$:     MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
```



```
                                        ;*****************************************************
                                        ;VERIFY "CLR ALL" CLEARS THE EXTENDED OFFSET BITS
                                        ;*****************************************************
001754-                                 1ST14:
001754- 104407  000000-                         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001760- 104407  000000-                         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001764- 005067  176112                          CLR     ACSR                    ;CLEAR EXPECTED VALUE
001770- 012777  004000  176264                  MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
001776- 012777  000003  176250                  MOV     #3,@OFF                 ;LOAD EXTENDED OFFSET REGISTER
002004- 012777  004000  176250                  MOV     #CLRALL,@SFR            ;CLEAR THE EXTENDED OFFSET REG.'S
002012- 017167  176236  176064                  MOV     @OFF,ASTAT              ;READ EXTENDED OFFSET REG.
002020- 001403                                  BEQ     ?ST15                   ;/BR IF CLEARED
                                                ;*************************************************************
002022- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;CLEAR ALL FAILED TO CLEAR EXTENDED OFFSET REGISTER
                                                ;*************************************************************

                                        ;*****************************************************
                                        ;TEST THE "ACTIVE" FLOP CAN SET AND CLEAR
                                        ;*****************************************************
002030-                                 ?ST15:
002030- 104407  000000-                         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
002034- 104407  000000-                         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002040- 012777  004000  176214                  MOV     #CLRALL,@SFR                    ;CLEAR THE DEVICE
002046- 052777  000010  176206                  BIS     #TSTDMA,@SFR    ;SET THE "TEST DMA" TO PREVENT DATA TRANSFERS
002054- 052777  000022  176170                  BIS     #BIT4!BIT1,@CSR         ;SET "MATRIX MODE" AND "BYTE" MODE
                                        ;NOW SET THE "ACTIVE" FLOP AND VERIFY "INTERFACE IDLE" GOES LOW
002062- 012767  000022  176012                  MOV     #BIT4!BIT1,ACSR ;LOAD EXPECTED VALUE
002070- 052777  000001  176154                  BIS     #BIT0,@CSR              ;SET "ACTIVE" TO A 1
002076- 017767  176156  176000                  MOV     @CSR,ASTAT              ;READ STATUS
002104- 026767  176776  175772                  CMP     ACSR,ASTAT              ;COMPARE VALUES
002112- 001406                                  BEQ     1$                      ;/BR IF EXPECTED
002114- 052777  000400  176140                  BIS     #ENDDMA,@SFR            ;STOP DMA IF POSSIBLE
                                                ;*************************************************************
002122- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;"ACTIVE" FLOP FAILED TO SET
                                                ;*************************************************************
                                        ;POKE THE "END DMA" SIGNAL AND VERIFY "ACTIVE" CLEARS
002130- 052777  000400  176124          1$:     BIS     #ENDDMA,@SFR            ;SEND "END DMA" SIGNAL
002136- 012767  000222  175736                  MOV     #BIT7!BIT4!BIT1,ACSR    ;SET "INTERFACE IDLE" INTO EXPECTED
002144- 017167  176102  175732                  MOV     @CSR,ASTAT              ;READ STATUS
002152- 026767  175724  175724                  CMP     ACSR,ASTAT              ;COMPARE
002160- 001403                                  BEQ     2$                      ;/BR IF SAME
                                                ;*************************************************************
002162- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;"END DMA" AGAIN FAILED TO CLEAR "ACTIVE" FLOP
                                                ;*************************************************************
                                        ;POKE "ENABLE DMA" AND THEN ISSUE "CLR ALL" SIGNAL TO   ENSURE THE "ACTIVE"  FLOP CLEARS
002170- 012767  000200  175704          2$:     MOV     #BIT7,ACSR              ;LOAD EXPECTED
002176- 052777  000001  176046                  BIS     #BIT0,@CSR              ;POKE "ENABLE DMA"
002204- 052777  004000  176050                  BIS     #CLRALL,@SFR            ;GENERATE "CLR ALL"
002212- 017767  176034  175664                  MOV     @CSR,ASTAT              ;READ STATUS
002220- 026767  175656  175656                  CMP     ACSR,ASTAT              ;COMPARE
002226- 001403                                  BEQ     ?ST16                   ;/BR IF SAME
                                                ;*************************************************************
002230- 104405  000000- 000000                  HRDER$,BEGIN,NULL       ;;"CLR ALL" FAILED TO CLEAR "ACTIVE AND PENDING" FLOPS
                                                ;*************************************************************
```

```
                                ;*************************************************
                                ;ENABLE A 512 WORD TRANSFER.....SECTION LIST MODE (MAINT MODE)
                                ;*************************************************
                                ;
002336'                         TST16:
002336' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
002342' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002346' 032767  000002  175542          BIT     #BIT1,SR1               ;TEST IF INHIBIT TEST IS SET
002354' 001146                          BNE     4$              ;;/BR IF SET
002356' 012777  004000  175776          MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
002364' 012700  003500'                 MOV     #BUF0,R0                ;LOAD BUFFER POINTER
002370' 012720  007070          4$:     MOV     #7070,(R0)+             ;PRESET THE BUFFER WITH DATA
002374' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
002400' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002404' 020027  004500'                 CMP     R0,#BUF1                ;TEST IF DONE
002410' 001367                          BNE     4$                      ;BR IF NOT
002412' 104415  000000' 000124'         GETPA$,BEGIN, RBUFVA            ;GET PHYSICAL ADDRESS FROM 16-BIT RBUFVA
002420' 016700  175604                  MOV     RBUFEA,R0               ;GET EXTENDED ADDRESS BITS
002424' 006200                          ASR     R0
002426' 006200                          ASR     R0
002430' 006200                          ASR     R0
002432' 006200                          ASR     R0
002434' 042700  177774                  BIC     #177774,R0              ;CLEAR OFF EXCESSIVE
002440' 010077  175610                  MOV     R0,@OFF                 ;LOAD EXTENDED BITS
002444' 012777  177000  175704          MOV     #-512.,@WCR             ;SET UP 512. WORD TRANSFER
002452' 016777  175550  175700          MOV     RBUFPA,@BAR             ;LOAD BUS ADDRESS FOR RESULT
002460' 012777  000014  175676          MOV     #TSTDMA!TSTCON,@SFR     ;ENABLE TEST CONTROL AND DMA FLOPS
002466' 052777  000001  175656          BIS     #BIT0,@CSR              ;ENABLE DEVICE
002474' 012767  001000  175704          MOV     #512.,STEMP             ;LOAD THE COUNTER
002502' 052777  000002  175652  3$:     BIS     #TEST2,@SFR     ;ENABLE INT. 2
002510' 042777  000004  175644          BIC     #TEST2,@SFR     ;DISABLE INT. 2
002516' 052777  010000  175636          BIS     #BIT12,@SFR             ;ALLOW 1 DMA TRANSFER
002424' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
002430' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002434' 005367  175646                  DEC     STEMP                   ;FINISHED ALL WORDS?
002440' 001360                          BNE     3$                      ;BR UNTILL DONE
                                ;THE TRANSFER IS NOW COMPLETE
002442' 005067  175434                  CLR     ACSR                    ;CLEAR EXPECTED
002446' 017767  175602  175430          MOV     @OFF,ASTAT              ;READ OFFSET REG.
002454' 042767  000003  175422          BIC     #3,ASTAT                ;CLEAR LOW 2 BITS
002462' 001403                          BEQ     2$              ;;/BR IF HIGH 14 BITS ARE CLEARED
                                ;*************************************************
002464' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;UNEXPECTED OFFSET REGISTER BIT SET
                                ;*************************************************
002472' 012777  004000  175562  2$:     MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
002500' 017767  175554  175376          MOV     @BAR,ASTAT              ;READ BUS ADDRESS
002506' 016767  175414  175366          MOV     RBUFPA,ACSR             ;GET PHYSICAL ADD
002514' 062767  002000  175360          ADD     #2000,ACSR              ;UPDATE TO THE END OF BUFFER
002522' 026767  175354  175354          CMP     ACSR,ASTAT              ;COMPARE VALUES
002530' 001403                          BEQ     1$                      ;;/BR IF SAME
                                ;*************************************************************
002532' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;INCORRECT BUS ADDRESS VALUE AFTER A 1 WORD TRANSFER
                                ;*************************************************************
002540' 017767  175512  175336  1$:     MOV     @WCR,ASTAT              ;READ W.C. REGISTER
002546' 012767  000000  175326          MOV     #0,ACSR         ;LOAD EXPECTED W.C. VALUE
002554' 026767  175322  175322          CMP     ACSR,ASTAT              ;COMPARE VALUES
002562' 001403                          BEQ     TST17   ;;/BR IF SAME
```



```
                                ;*************************************************************
002564' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;INCORRECT WORD COUNT REGISTER VALUE AFTER A 1 WORD TRA
                                ;*************************************************************
                                ;
                                ;*************************************************
                                ;ENABLE A 512 WORD TRANSFER      SECTION LIST MODE
                                ;*************************************************
                                ;
002572'                         TST17:
002572' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
002576' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002602' 032767  000002  175206          BIT     #BIT1,SR1               ;TEST IF INHIBIT IS SET
002610' 001153                          BNE     TST20   ;;/BR IF SET
002612' 012777  004000  175442          MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
002620' 012700  003500'                 MOV     #BUF0,R0                ;LOAD BUFFER POINTER
002624' 012720  007070          4$:     MOV     #7070,(R0)+             ;PRESET THE BUFFER WITH DATA
002630' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
002634' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002640' 020027  004500'                 CMP     R0,#BUF1                ;TEST IF DONE
002644' 001367                          BNE     4$                      ;BR IF NOT
002646' 104415  000000' 000124'         GETPA$,BEGIN, RBUFVA            ;GET PHYSICAL ADDRESS FROM 16-BIT RBUFVA
002654' 016700  175250                  MOV     RBUFEA,R0
002660' 006200                          ASR     R0
002662' 006200                          ASR     R0
002664' 006200                          ASR     R0
002666' 006200                          ASR     R0
002670' 042700  177774                  BIC     #177774,R0              ;CLEAR OFF THE EXCESS
002674' 010077  175354                  MOV     R0,@OFF                 ;LOAD EXTENDED ADDRESS BITS
002700' 012777  003750' 175370          MOV     #3$,@VECTAO             ;LOAD RETURN VECTOR
002706' 116777  175100  175364          MOVB    BR7,@VECTAI
002714' 012777  177600  175334          MOV     #-512.,@WCR             ;SET UP 512. WORD TRANSFER
002722' 016777  175200  175330          MOV     RBUFPA,@BAR             ;LOAD BUS ADDRESS FOR RESULT
002730' 012777  000006  175324          MOV     #TSTCON!TESTZ,@SFR      ;ENABLE TEST CONTROL
002736' 052777  000101  175306          BIS     #BIT6!BIT0,@CSR         ;ENABLE DEVICE
002744' 104400  000000'                 EXIT$,BEGIN             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.

002750' 017767  175300  175126  3$:     MOV     @OFF,ASTAT              ;READ OFFSET REG
002756' 042767  000003  175120          BIC     #3,ASTAT                ;MASK OFF LOW 3 BITS
002764' 052777  000400  175270          BIS     #ENDDMA,@SFR            ;STOP DMA
002772' 052777  004000  175262          BIS     #CLRALL,@SFR            ;CLEAR THE DEVICE
                                ;-------------------------------------------------------------
003000' 000004  000000' 003006'         PIRQ$,BEGIN,5$          ; QUEUE UP TO CONTINUE AT 5$ AND RTI
                                ;-------------------------------------------------------------
                                ;THE TRANSFER IS NOW COMPLETE
003006' 005067  175070          5$:     CLR     ACSR                    ;CLEAR EXPECTED
003012' 005767  175066                  TST     ASTAT                   ;CHECK OFFSER REG.
003016' 001403                          BEQ     2$              ;;/BR IF CLEARED
                                ;*************************************************************
003020' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;;UNEXPECTED HIGH 14 BITS OF OFFSET REGISTER BIT SET
                                ;*************************************************************
003026' 012777  004000  175226  2$:     MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
003034' 017767  175220  175042          MOV     @BAR,ASTAT              ;READ BUS ADDRESS
003042' 016767  175060  175032          MOV     RBUFPA,ACSR             ;GET PHYSICAL ADDRESS
003050' 062767  002000  175024          ADD     #2000,ACSR              ;POINT TO END OF BUFFER
003056' 026767  175020  175020          CMP     ACSR,ASTAT              ;COMPARE VALUES
003064' 001403                          BEQ     1$                      ;;/BR IF SAME
                                ;*************************************************************
```

```
                        ;HRDER$,BEGIN,NULL.....;;INCORRECT BUS ADDRESS VALUE AFTER A 512 WORD TRANSFER
                        ;*************************************************************
003066' 104405  000000' 000000          HRDER$,BEGIN,NULL.....;;INCORRECT BUS ADDRESS VALUE AFTER A 512 WORD TRANSFER
                        ;*************************************************************
003074' 017767  175156  175002  1$:     MOV     @WCR,ASTAT              ;READ W.C. REGISTER
003102' 012767  000000  174772          MOV     #0,ACSR         ;LOAD EXPECTED W.C. VALUE
003110' 026767  174766  174766          CMP     ACSR,ASTAT              ;COMPARE VALUES
003116' 001403                          BEQ     2$                      ;/BR IF SAME
                        ;*************************************************************
003120' 104405  000000' 000000          HRDER$,BEGIN,NULL.....;;INCORRECT WORD COUNT REGISTER VALUE AFTER A 512 WORD T
                        ;*************************************************************
003126' 016777  175146  175142  7$:     MOV     VECTA1,@VECTA0
003134' 005077  175140                  CLR     @VECTA1                 ;RESET VECTOR
```



```
                        ;*************************************************
                        ;MATRIX MODE DATA COLLECTOR
                        ;*************************************************
003140'                 ;ST20:
003140' 104407  000000'                 BREAK$,BEGIN                    ;TEMPORARY RETURN TO MONITOR....
003142' 104407  000000'                 BREAK$,BEGIN                    ;THEN CONTINUE AT NEXT INSTRUCTION.
003144' 032767  000001  174640          BIT     #BIT0,SR1               ;TEST IF INHINIT IS SET
003152' 001110                          BNE     ;ST21          ;;/BR IF SET
003154' 012700  003470'                 MOV     #RUFA,R0                ;LOAD BUFFER POINTER
003160' 012720  170300          1$:     MOV     #170300,(R0)+           ;CLEAR THE BUFFER
003164' 104407  000000'                 BREAK$,BEGIN                    ;TEMPORARY RETURN TO MONITOR....
003170' 104407  000000'                 BREAK$,BEGIN                    ;THEN CONTINUE AT NEXT INSTRUCTION.
003174' 022700  005500'                 CMP     #BUF2,R0                ;FINISHED ?
003200' 001367                          BNE     1$
                        ;
003202' 012777  004000  175046          MOV     #CLRALL,@SFR            ;CLEAR THE DEVICE
003210' 104415  000000' 000124'         GETPAS,BEGIN, RBUFVA                    ;GET PHYSICAL ADDRESS FROM 16-BIT RBUFVA
003216' 012777  003330  175052          MOV     #10$,@VECTB0            ;LOAD INTR. VECTOR
003224' 116777  174556  175046          MOVB    BR1,@VECTB1             ;LOAD BR LEVEL
003232' 016700  174664                  MOV     RBUFBA,R0               ;GET BUFFER POINTER
003236' 042700  000003                  BIC     #3,R0                   ;CLEAR EXT. MEMORY BITS
003242' 016701  174656                  MOV     RBUFEA,R1               ;GET EXTENDED MEMORY BITS
003246' 006201                          ASR     R1
003250' 006201                          ASR     R1
003252' 006201                          ASR     R1
003254' 006201                          ASR     R1
003256' 006201                          ASR     R1
003260' 042701  177774                  BIC     #177774,R1              ;CLEAR OFF EXCESS
003264' 060100                          ADD     R1,R0
003266' 010077  174760                  MOV     R0,@OFF                 ;LOAD COMBINED ADDRESS
003272' 012777  177766  174754          MOV     #-10.,@WCR              ;LOAD HIGH 16 BITS OF Z
003300' 012777  000000  174750          MOV     #0,@BAR                 ;LOAD LOW 16 BITS OF Z
003306' 012777  000062  174734          MOV     #BIT5122,@CSR           ;SET RES. AND MODE AND CELL INTR. ENABLE
003314' 052777  000002  174736          BIS     #TEST2,@SFR             ;SET Z PULSES
003322' 052777  000001  174720          BIS     #BIT0,@CSR              ;ENABLE THE DEVICE
003332' 104400  000000'                 EXIT$,BEGIN                             ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
                        ;
003336' 052777  000400  174716  10$:    BIS     #ENDDMA,@SFR            ;STOP TRANFERS
003344' 052777  004000  174710          BIS     #CLRALL,@SFR            ;CLEAR NCV11
                        ;-------------------------------------------------------------
003352' 000004  000000' 003360'         PIRQ$,BEGIN,11$         ; QUEUE UP TO CONTINUE AT 11$ AND RTI
                        ;-------------------------------------------------------------
003360' 016777  174720  174714  11$:    MOV     VECTB1,@VECTB0
003366' 005077  174712                  CLR     @VECTB1                 ;RESET THE VECTOR
003372' 000240                          NOP
003374' 000240                          NOP
003376' 000240                          NOP
```

```
                                ;*****************************************************
                                ;END OF PASS
                                ;*****************************************************
003400'                         TST21:
003400' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
003404' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
003410' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                ;MONITOR SHALL TEST END OF PASS
003414' 000167  174670          1$:     JMP     LOGIC                   ;RUN AGAIN

                                ;MODULE PATCH SPACE

003420' 000024                          .BLKW   20.

                                ;MODULE READ BUFFER SPACE

003470' 000240                  BUFA:   NOP             ;NOP'S MUST REMAIN <NOT A PATCH AREA>
003472' 000240                          NOP
003474' 000240                          NOP
003476' 000240                          NOP
003500' 000400                  BUF0:   .BLKW   256.
004500' 000400                  BUF1:   .BLKW   256.
005500' 000000                  BUF2:   0

000001                                  .END
```


CROSS REFERENCE TABLE -- USER SYMBOLS

```
ABASE = 000000
ACSR    000102R

ADDR    000006R
ADDR22= 001000
ADW     000264R
ASB     000106R
ASTAT   000104R

AVECT1= 000000
AWAS    000110R
BAR     000260R
BAR1    000270R
BEGIN   000000R

BIT0  = 000001
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BREAK$= 104407

BR1     000012R
BR2     000013R
BTOD$ = 104421
BUFA    003470R
BUF0    003500R
BUF1    004500R
BUF2    005500R
CDAREA= 104412
CDERCT  000144R
CDWDCT  000146R

210# 218  219  220  221  222  223  224  225  226  227
245* 246* 247  248  255* 297  303* 317  318* 321  336* 337  355*
358  372* 385* 388  390  399* 407* 410  412  421* 429* 432  434
443* 459* 461  466* 468  473* 475  480* 482  487* 489  501* 521*
524  532* 534  540* 544  585* 594* 595* 596  602* 603  648* 656*
657* 658  664* 665
151# 237  242  244  251
209# 263
263#
188#
187# 273* 276  296* 297  311* 312  320* 321  335* 337  357* 358
373* 389* 390  411* 412  433* 434  460* 461  467* 469  474* 475
401* 482  488* 489  505* 523* 524  533* 534  543* 544  586* 587*
593* 596  601* 603  640* 641* 649  655* 658  663* 665
211# 228  229  230  231
190#
221# 261  410* 411  455*  474  573*  593  635*  655  702*
225# 264
148# 256  257  269  270  279  282  283  291  292  300  306  307
315  324  330  331  340  343  344  352  353  361  368  369  376
383  384  393  397  398  405  406  415  419  420  427  428  437
441  442  450  464  471  478  485  492  499  500  508  515
516  528  537  547  553  554  560  561  564  580  581  590  599
606  613  614  620  621  624  638  645  652  661  668  676  677
682  683  688  706  711  723  724  725
286# 385  407  420  522  541  575  637  678  705
209# 214  519  521  532  555  615
209#
209# 217  271
209# 579
209#
209#
209#
209# 215
209# 212  216
209# 519  521  532
209# 384
209# 778
209# 637
209# 275  295  532  540
209# 213
209#
209# 256  257  269  270  282  283  291  292  306  307  330  331
343  344  352  353  368  369  383  384  397  398  405  406  419
427  428  441  442  449  450  499  500  515  516  553  554
560  561  580  581  613  614  620  621  676  677  682  683  723
724
153# 633  690
154#
209#
680  735#
196  558  618  739#
562  622  740#
684  741#
209#
209#
209#
```

```
CLRALL= 004000           217#  235   294   371   387   409   431   452   494   502   504   517   542
                         557   592   617   643   654   687   709
CONFIG  000056R          173#
CSR     000252R          218#  236   258   272*  273   293*  296   308*  310*  311   317*  320   457*
                         498   518*  522*  523   533   541*  543   575*  637*  703*  705*
CSRA    000100R          193#  251*
CSRHB   000272R          226#  240   319*
DATCKS= 104411           209#
DATERS= 104404           209#
DVID1   000014R          155#
ENDDMA= 000400           213#  526   531   642   708
ENDITS= 104413           209#  725
ENDS    104410           209#
ERRTYP  000106R          188#
EXITS = 104400           209#  638   706
FREE    000158R          192#
GETPAS= 104415           209#  564   624   688
GWBUFS= 104414           209#
HRDCNT  000034R          168#
HRDERS= 104405           209#  279   300   315   324   340   361   376   393   415   437   464   471
                         478   485   492   508   528   537   547   590   599   606   652   661   668
HRDPAS  000050R          170#
ICONT   000036R          165#
ICOUNT  000040R          166#
IDMOM   000122R          195#
IMODX.= 000060           207#
INIT    000030R          162#
INTR    000120R          194#
JOY     000266R          224#
LOGIC   000310R          235#  727
MAP22S= 104416           209#
MODRAM  000000R          142#
MODSP   000252R          207#
MSGMS = 104403           209#
MSGSS = 104402           209#
MSGS  = 104401           209#
NULL  = 000000           209#  279   300   315   324   340   361   376   393   415   437   464   471
                         478   485   492   508   528   537   547   590   599   606   652   661   668
OFF     000254R          219#  259   432*  433   453*  460   503*  505   571*  586   631*  640   700*
OPEN  = 000000           150   156   157   158   159   176   177   178   179   180   181   182   183
                         185   187   189   190   192   193   194   197   198   200   201   203   204
                         205   206   209#
OTOAS = 104420           209#
PASCNT  000034R          164#
PIRQS = 000004           209#  645   711
POPSP = 005726           209#
POPSP2= 022626           209#
PRTY  = 000000           209#
PRTY0 = 000000           154   209#
PRTY1 = 000040           209#
PRTY2 = 000100           209#
PRTY3 = 000140           209#
PRTY4 = 000200           209#
PRTY5 = 000240           209#
PRTY6 = 000300           153   209#
PRTY7 = 000340           209#
```



```
PS    = 177776           209#
PSW   = 177776           209#
PUSH  = 005746           209#
PUSH2 = 024646           209#
RAMDS = 104417           209#
RANMOM  000054R          172#
RBUFEA  000130R          198#  565   625   693
RBUFPA  000126R          197#  573   594   635   656   691
RBUFSZ  000132R          199#
RBUFVA  000124R          196#  564   624   688
RESTRT  000310R          191   234#
RES1    000056R          174#
RES2    000060R          175#
RSTRT   000112R          191#
SBADR   000162R          194#
SFR     000262R          222#  235*  262   294*  334*  335   354*  357   370*  371*  373   387*  409*
                         441*  452*  456*  481   484*  502*  504*  517*  518*  526*  531*  542*  557*
                         574*  577*  578*  579*  592*  617*  636*  642*  643*  654*  687*  704*  708*
                         709*
SFRHB   000274R          227#  356*
SOFCNT  000042R          167#
SOFERS= 104406           209#
SOFPAS  000046R          169#
SPOINT  000032R          163#
SPSIZ = 000040             1#  207
SR1     000016R          156#  555   615   678
SR2     000020R          157#
SR3     000022R          158#
SR4     000024R          159#
START   000310R          167   233#
STAT    000026R          161#
SVR0    000062R          177#
SVR1    000064R          177#
SVR2    000066R          178#
SVR3    000070R          179#
SVR4    000072R          180#
SVR5    000074R          181#
SVR6    000076R          182#
SYSCNT  000052R          171#
TESTZ = 000002           214#  577   578   636   704
TRPDFD= 000022           209#
TSTCON= 000004           215#  574   636
TSTDMA= 000010           212#  216#  518   574
TST1    000406R          255#
TST10   001204R          374   382#
TST11   001300R          404#
TST12   001374R          426#
TST13   001470R          448#
TST14   001754R          498#
TST15   002030R          502   514#
TST16   002236R          545   552#
TST17   002572R          556   604   612#
TST2    000452R          268#
TST20   003140R          616   675#
TST21   003400R          679   722#
TST3    000562R          290#
```

```
TST4    000640R          298     305#
TST5    000760R          322     329#
TST6    001062R          351#
TST7    001136R          359     367#
VECTA0  000276R          228#    632*    670*
VECTA1  000300R          229#    633*    670     671*
VECTB0  000302R          230#    689*    713*
VECTB1  000304R          231#    249     690*    713     714*
VECTOR  000010R          152#    246
WASADR  000104R          186#
WBUFEA  000136R          201#
WBUFPA  000134R          200#
WBUFRQ  000140R          202#
WBUFSZ  000142R          203#
WCR     000256R          220#    260     388*    389     454*    467     572*    601     634*    663     701*
WDFR    000116R          193#
WDTO    000114R          192#
XFLAG   000005R          150#
XTEMP   000306R          232#    271*    272     274     284*    285     332*    334     336     345*    346     576*    582*
$TX     = 000021         209#    252#    265#    287#    298     302#    322     326#    348#    359     364#    374     379#
                         401#    423#    445#    495#    506     511#    545     549#    556     604     609#    616     672#
                         679     719#
.       = 005502R        731#    739#    740#

. ABS.  000000      000
        005502      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XNCBA0,XNCBA0/SOL/CRF:SYM=DDXCOM,XNCBA0
 RUN-TIME:  2  3  .4 SECONDS
 RUN-TIME RATIO: 33/5=5.7
 CORE USED:  7K  (13 PAGES)
```

1

.REM_

IDENTIFICATION

PRODUCT CODE: AC-F805C-MC

PRODUCT NAME: CXDMDC0 DMP/DMV11 DECX MSTR MOD

PRODUCT DATE: AUGUST 1981

MAINTAINER: DIAGNOSTIC ENGINEERING CC:38P

AUTHORS: DAVID HOFFMAN
CHRIS BRIENEN

1. ABSTRACT

THERE ARE 2 DEC/X11 IOMODX MODULES WRITTEN FOR THE DMP/DMV11. THESE ARE DMD AND DME. TOGETHER, THESE 2 MODULES CAN OPERATE UP TO 16 (DEC.) DMP/DMV11 DEVICES IN POINT-TO-POINT LINKS, OR A SINGLE DEVICE CONFIGURED AS A MULTIPOINT CONTROL STATION COMMUNICATING WITH UP TO 32 (DEC.) TRIBUTARIES, OR UP TO 16 (DEC.) DEVICES CONFIGURED AS MULTIPOINT TRIBUTARIES ON THE SAME PDP-11 UNIBUS (16/16 W/DMV11 QBUS). THE BASIC OPERATION IS TO TRANSMIT, RECEIVE, AND CHECK 32(DEC.) DATA MESSAGES OF 1024 (DEC.) BYTES EACH, ON A GIVEN PHYSICAL LINK. BY DEFAULT, THIS WOULD INVOLVE A SINGLE PDP-11 SYSTEM WITH 1 OR MORE DEVICES OPERATED IN INTERNAL OR EXTERNAL LOOPBACK MODE. HOWEVER, BY OPERATOR SELECTION OF NON-DEFAULT MODES, ACTUAL POINT-TO-POINT OR MULTIPOINT OPERATION IS POSSIBLE. IN ONE SUCH MODE, THE MODULE CAN DRIVE THE OPERATION OF A CONTROL STATION WHICH SENDS AND RECEIVES BACK MESSAGES FROM A SPECIFIED LIST OF TRIBUTARIES, ON THE SAME OR DIFFERENT PDP-11 SYSTEMS. IN ANOTHER SUCH MODE, THE MODULE CAN DRIVE THE OPERATION OF A GROUP OF TRIBUTARIES WHICH RESIDE ON A SINGLE SYSTEM AND REFLECT BACK MESSAGES SENT TO THEM BY A CONTROL STATION , ON THE SAME OR DIFFERENT SYSTEM.

DMD IS THE MASTER MODULE WHICH CAN OPERATE UP TO 16 (DEC.) DEVICES IN LOOPED-BACK OR POINT-TO-POINT MASTER MODES, OR A SINGLE DEVICE IN MULTIPOINT CONTROL MODE.
DME IS THE SLAVE MODULE WHICH WHICH CAN OPERATE UP TO 16 (DEC.) DEVICES IN POINT-TO-POINT SLAVE OR MULTIPOINT TRIBUTARY MODES.
A MASTER MODULE CAN BE SELF-SUFFICIENT (IF LOOPBACK IS USED) OR IT CAN COMMUNICATE WITH A SLAVE MODULE(S) ON THE SAME OR ANOTHER PROCESSOR. THE REST OF THIS DOCUMENT WILL DESCRIBE THE USE AND OPERATION OF THE MASTER MODULE, DMD.

IT IS REQUIRED THAT THE OPERATOR CONFIGURE THE DEC/X11 EXERCISER WITH A SEPARATE COPY OF MODULE DMD FOR EACH GROUP OF LOOPED-BACK DEVICES, EACH CONTROL STATION DEVICE, OR EACH GROUP OF POINT-TO-POINT DEVICES. THE ACTUAL OPERATING MODE OF EACH COPY IS SELECTED BY THE SETTINGS OF SW1-SW4 (SOFTWARE SWITCHES) FOR THAT COPY. FOR MULTI-PROCESSOR CONFIGURATIONS, A DEC/X11 EXERCISER MUST BE CONFIGURED AND RUN ON EACH PDP-11 PROCESSOR, SIMULTANEOUSLY.

FOR ACTUAL LINK MODE OPERATION, NOTE THE FOLLOWING :
DO NOT ALLOW THE MODULE TO RELOCATE; USE A NON-RELOCATING MONITOR OR USE THE RUN LOCK COMMAND (RUNL).
AFTER THE MASTER MODULE (DMD) HAS BEEN STARTED ON ONE PROCESSOR, THE OPERATOR IS ALLOWED AT LEAST 5 MINUTES TO START THE SLAVE MODULE(S) (DME) ON THE OTHER PROCESSOR(S), BEFORE GETTING A PROTOCOL TRANSMIT THRESHOLD ERROR ON THE MASTER MODULE PROCESSOR.

THE M8203 LINE UNIT CONTAINS HARDWARE SWITCHES WHICH CAN DEFINE THE PHYSICAL LINK AND DMP11 OPERATING MODE, AND A TRIBUTARY ADDRESS (IF MULTIPOINT). EVERY EFFORT IS MADE TO OPERATE WITH THESE SWITCH VALUES, UNLESS THEY MUST BE OVERRIDDEN BY THE PROGRAM. FOR EXAMPLE, WITH LOOPBACK SELECTED, A TRIBUTARY COULD ONLY BE RUN AS A CONTROL STATION, AND A HALF-DUPLEX LINK COULD ONLY BE RUN AS A FULL DUPLEX LINK. IN ANY CASE, THE PROGRAM DOES NOT REQUIRE

OPERATOR INTERVENTION TO CHANGE HARDWARE SWITCH SETTINGS OR CONFIGURE CABLES, ETC.

2. REQUIREMENTS

HARDWARE: 1 TO 16 (DEC.) DMP/DMV11'S AND ASSOCIATED CABLES AND CONNECTORS.
STORAGE: DMD REQUIRES ABOUT 2K WORDS OF STORAGE

3. PASS DEFINITION

ONE PASS OF THE DMD MODULE CONSISTS OF TRANSMITTING AND RECEIVING A TOTAL OF 32,768 (DEC.) 8-BIT CHARACTERS, PER DMP/DMV11.

4. EXECUTION TIME

DMD RUNNING ALONE ON A PDP11/40 PROCESSOR TAKES APPROXIMATELY 1 MINUTE TO COMPLETE ONE PASS, AT 9600 BAUD. THIS TIME INCREASES ONLY SLIGHTLY AS MORE MASTER DEVICES ARE ADDED, BUT IS INVERSELY PROPORTIONAL TO THE BAUD RATE.

5. CONFIGURATION PARAMETERS

DEFAULT PARAMETERS :

DEVADR: 160170, VECTOR:300, BR1:5, BR2: NOT USED , DEVCNT:1

SOFTWARE SWITCH REGISTER OPTIONS :

THE ALLOWABLE OCTAL VALUES OF SR4 ARE:
SR4=000000 : IF TESTING DMP11
SR4=000001 : IF TESTING DMV11
SR4=000002 : IF TESTING DMV11 (AND Q22 SOFTWARE MODE IS DESIRED)

THE ALLOWABLE OCTAL VALUES OF SR1 ARE: 000000, 000001, AND 000002.
A DESCRIPTION OF SR1-SR3 (AND THEIR MEANING) ARE GIVEN BELOW:

FOR SR1 = 000000 :
ALL UNITS SELECTED IN DVID1 WILL BE RUN IN POINT-TO-POINT, FULL-DUPLEX MODE, WITH INTERNAL OR EXTERNAL LOOPBACK ON ALL DEVICES.
WHEN SR1 = 000000, SR2 HAS THE FOLLOWING MEANING :
- SR2 = 000000 IF INTERNAL LOOPBACK SHOULD BE PROVIDED BY PROGRAM, USING TTL-LEVEL LOOPBACK ON THE LINE UNIT. THIS IS THE DEFAULT MODE OF OPERATION.
- SR2 = 000001 IF EXTERNAL LOOPBACK WILL BE PROVIDED BY THE OPERATOR VIA H3254,5 TEST CONNECTORS ON EACH DEVICE. FOR DMV11'S, INTERFACE TYPE IS SELECTED BY HARDWARE. FOR THE DMP11, THE OPERATOR MUST SPECIFY THE DESIRED MODEM INTERFACE TO BE SELECTED FOR TESTING, USING SR3 :
FOR SR3 = 000000, V.35 WILL BE RUN BY DEFAULT.
FOR SR3 = 000010, INTEGRAL MODEM WILL BE RUN.

FOR SR3 = 000020, V.35 WILL BE RUN.
FOR SR3 = 000100, RS449/RS232C/RS423 WILL BE RUN.
FOR SR3 = 000200, RS422 WILL BE RUN.
THE SELECTED INTERFACE WILL BE RUN AT THE BAUD RATE SELECTED IN THE LINE UNIT BAUD RATE SWITCHES.
- SR2 = 000002 IF EXTERNAL LOOPBACK OTHER THAN H3254,5 IS PROVIDED (FOR EXAMPLE, CABLE LOOPBACK). (NOTE THAT MODEM LOOPBACK IS NOT SUPPORTED IN THIS MODULE).
THE HEADER ENTRY DVC MUST BE IN THE RANGE 1-20 (OCTAL).

FOR SR1 = 000001 :
ALL UNITS SELECTED IN DVID1 WILL BE RUN IN POINT-TO-POINT, FULL OR HALF-DUPLEX MODE (DEPENDING ON LINE UNIT SWITCH SETTINGS) WITHOUT LOOPBACK. THE MODULE COMMUNICATES WITH SLAVE MODULE(S) ON THE SAME AND/OR OTHER PDP-11 SYSTEM(S).
WHEN SR1 = 000001 : SR2 AND SR3 ARE UNUSED, AND THE HEADER ENTRY DVC MUST BE IN THE RANGE 1-20 (OCTAL).
****
NOTE:: WHEN IN POINT-TO-POINT OPERATION DO NOT ALLOW RELOCATION OF THE MONITOR. USE EITHER A MONITOR THAT DOES NOT ALLOW RELOCATION OR USE THE RUN LOCK COMMAND (RUNL).
****

FOR SR1 = 000002 :
ONLY 1 UNIT MUST BE SELECTED IN DVID1 (DVID1 = 000001), AND IT WILL BE RUN AS A MULTI-POINT CONTROL STATION, IN FULL OR HALF-DUPLEX MODE (DEPENDING ON LINE UNIT SWITCH SETTINGS) WITHOUT LOOPBACK. THE MODULE COMMUNICATES WITH SLAVE MODULE(S) ON THE SAME AND/OR OTHER PDP-11 SYSTEM(S).
WHEN SR1 = 000002, THE FOLLOWING MEANING IS GIVEN TO SR2 AND SR3 :
- SR2 = THE TOTAL NUMBER OF TRIBUTARIES (OCTAL) ON THIS MULTIPOINT LINK. THE ALLOWABLE RANGE IS 000001-000040(DMP; DMV = 000020).
- SR3 = THE STARTING TRIBUTARY ADDRESS (OCTAL). THE ALLOWABLE RANGE IS 000001-000377. THE PROGRAM WILL USE THIS STARTING TRIB ADRS TO COMPUTE THE OTHER TRIB ADDRESSES ON THE MULTIPOINT LINK, AND THE ADDRESSES CAN "WRAPAROUND" 377 TO 001, IF NECESSARY.
THE HEADER ENTRY DVC MUST = 1.
****
NOTE:: WHEN IN MULTIPOINT OPERATION DO NOT ALLOW RELOCATION OF THE MONITOR. USE EITHER A MONITOR THAT DOES NOT ALLOW RELOCATION OR USE THE RUN LOCK COMMAND (RUNL).
****

6. DEVICE/OPTION SETUP

IF EXTERNAL LOOPBACK IS DESIRED (SR1 = 000000 AND SR2 = 000001 OR 2), TEST CONNECTOR(S) MUST BE INSTALLED BY THE OPERATOR ON ALL DEVICES, PRIOR TO STARTING THE EXERCISER.

7. DMD MODULE OPERATION

TEST SEQUENCE:

A. INITIALIZE ALL MASTER DEVICES
B. LOAD INPUT AND OUTPUT INTERRUPT VECTORS ON ALL MASTERS
C. PERFORM MODE DEFINITION ON ALL MASTERS
D. VERIFY OPERATION OF INPUT AND OUTPUT INTERRUPTS ON ALL MASTERS
E. START THE DDCMP PROTOCOL ON ALL MASTERS, TO ALL SLAVES, AND SCAN EACH LOGICAL LINK UNTIL IT ENTERS THE RUNNING STATE.
F. EXIT TO MONITOR WITH INTERRUPTS ENABLED
G. INPUT INTERRUPT SERVICE :
   1)IF LOGICAL LINK WAS JUST STARTED, INPUT A RCV BUFFER
   2)IF RCV BUFFER WAS PREVIOUSLY INPUT, INPUT A TRANSMIT BUFFER
   3)IF TRANSMIT AND RCV BUFFERS HAVE BEEN INPUT ON ALL LOGICAL LINKS ON THIS DEVICE, DISABLE INPUT INTRPTS ON THIS DEVICE.
H. OUTPUT INTERRUPT SERVICE :
   1)IF TRANSMIT BUFFER WAS JUST RETURNED, CHECK FOR CORRECT BA, EA, TRANSMIT CHAR COUNT BITS.
   2)IF RCV BUFFER WAS JUST RETURNED, CHECK FOR CORRECT BA, EA, RCV CHAR COUNT BITS, AND DO DATA CHECK ON RCV BUFFER.
   3)IF ALL LOGICAL LINKS HAVE TRANSMITTED AND RECEIVED CURRENT DATA, SELECT A NEW BUFFER FULL OF DATA.
   4)IF ALL LOGICAL LINKS HAVE DONE REQUIRED NUMBER OF WRITE, READ, CHECK SEQUENCES, REPORT AN END OF PASS. THEN, RESTART THE PROTOCOL ON ALL MASTERS TO ALL SLAVES.
   5)REPORT ANY ERRORS PROVIDED BY ANY MASTER DEVICE(S). ALSO, AT THE END OF EACH PASS, THE PROGRAM WILL REPORT ANY SOFT (DATA OR HEADER CRC) ERRORS THAT HAVE BEEN DETECTED DURING THE PASS.
1. REPEAT G AND H

8. OPERATION OPTIONS

A. LOCATION DVID1 (DMD 14) MAY BE CHANGED TO SELECT ANY COMBINATION OF DEVICES BIT0=DEV0, BIT1=DEV1 .....BIT15=DEV15.

   NOTE: IF DVID1 IS INITIALLY = 0, DMD WILL BE DROPPED FROM TEST.

9. ERROR REPORTS

ALL ERROR REPORTS HAVE STANDARD FORMATS AS DESCRIBED IN THE DEC/X11 USER'S GUIDE. IN EACH ERROR PRINTOUT, THE SAME INFORMATION IS ALWAYS REPORTED, ALONG WITH THE ERROR TYPE AND PC, WHICH UNIQUELY IDENTIFY THE ERROR. THE FOLLOWING IS AN EXAMPLE PRINTOUT :

```
DMDA0  PA 00046706  APC 004462  PASS #00001  SOFT ERR #00001
CSRA: 160210  CSRC: 100000  ASTAT: 000602  ERRTYP: 000001
```



```
100000  000602  001005  000112  NNNNNN
```

ERRTYP = 1 INDICATES A DATA ERROR (AS DESCRIBED IN THE DEC/X11 USER'S GUIDE), WHICH IS A SOFT ERROR. CSRA IS THE DEVICE CSR ADDRESS, CSRC AND ASTAT ARE THE CONTENTS OF THE FIRST TWO CSR REGISTERS (SEL0 AND SEL2). THE FOUR (5 IF DMV) FIELDS OF NUMBERS ON THE THIRD LINE ARE THE CONTENTS OF ALL FOUR CSR REGISTERS (SEL0,2,4,6 (AND 10 IF DMV)). WHENEVER ERRTYP = 0 IS REPORTED (UNDEFINED ERROR) THE USER MUST REFER TO THE DIAGNOSTIC LISTING FOR THE ACTUAL EXPLANATION OF THE ERROR. THIS IS RECOMMENDED FOR ALL ERRORS, BECAUSE THE DEC/X11 ERROR TYPE DEFINITIONS ARE QUITE GENERAL.

```
                                        .TITLE  DMDC DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6      23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 046504  041504     040  MODNAM: .ASCII  /DMDC / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 160170                  ADDR:   160170+0                ;1ST DEVICE ADDR.
000010' 000300                  VECTOR: 300+0                   ;1ST DEVICE VECTOR.
000012'     240                 BR1:    .BYTE   PRTY5+0         ;1ST BR LEVEL.
000013'     240                 BR2:    .BYTE   PRTY5+0         ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;*****************************************************************************
000026' 150000                  STAT:   150000                  ;STATUS WORD.
000030' 000464'                 INIT:   START                   ;MODULE START ADDR.
000032' 000252'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                       ;PASS COUNTER.
000036' 000040                  ICONT:  32.                     ;# OF ITERATIONS PER PASS=32.
000040' 000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                         ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000102'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000104'                         WASADR:                         ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000106'                         ERRTYP:                         ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000112' 002550'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
000122' 000000                  IDNUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
000124' 007312'                 RBUFVA: BUFIN                   ;READ BUFFER VIRTUAL ADDRESS
000126' 000000                  RBUFPA: OPEN                    ;READ BUFFER PHYSICAL ADDRESS
000130' 000000                  RBUFEA: OPEN                    ;READ BUFFER EA BITS
```



```
000132' 001000                  RBUFSZ: 512.                    ;SIZE OF THE READ BUFFER
000134' 000000                  WBUFPA: OPEN                    ;WRITE BUFFER PHYSICAL ADDRESS
000136' 000000                  WBUFEA: OPEN                    ;WRITE BUFFER EA BITS
000140' 001000                  WBUFRQ: 512.                    ;WRITE BUFFER SIZE REQUESTED
000142' 000000                  WBUFSZ: OPEN                    ;WRITE BUFFER SIZE AVAILABLE
000144' 000000                  CDERCT: OPEN                    ;CDATA/DATCK ERROR COUNT
000146' 000000                  CDWDCT: OPEN                    ;CDATA/DATCK WORD COUNT
000150' 000000                  FREE:   OPEN                    ;RESERVED FOR FUTURE USE
000252'                         MODSP:
                                ;*****************************************************************************
```

```
                        ;BIT DEFINITIONS
100000                  BIT15=100000
040000                  BIT14=40000
020000                  BIT13=20000
010000                  BIT12=10000
004000                  BIT11=4000
002000                  BIT10=2000
001000                  BIT9=1000
000400                  BIT8=400
000200                  BIT7=200
000100                  BIT6=100
000040                  BIT5=40
000020                  BIT4=20
000010                  BIT3=10
000004                  BIT2=4
000002                  BIT1=2
000001                  BIT0=1


000305                  GOODIN  = 305           ;CODE LEFT IN BSEL6 AFTER INIT, IF NO ERRORS

021344                  RDLU16  = 021344        ;M8207 INSTRUCTION TO MOVE LU REG 16 INTO BSEL4

121000                  SWPBOT  = 121000        ;DMV-11 ADDRESS OF "SWPBOT" SWITCHES
000200                  MRDY    = 200           ;DMV-11 BIT INDICATING M-LOOP IS READY FOR NEXT COMMAND
000001                  REDLOC  = 1             ;DMV-11 M-LOOP COMMAND TO READ INTERNAL MEMORY
000006                  TTLOOP  = 6             ;DMV-11 M-LOOP COMMAND TO SETUP 56K BPS/INT LOOPBACK

                        ;*****************************************************************************
                        ;* SWITCH REGISTER 1 (SR1) BIT DEFINITIONS
                        ;*****************************************************************************
000001                  NONLUP  = BIT0  ;LOOPBACK MODE IF BIT = 0

                        ;*****************************************************************************
                        ;* SWITCH REGISTER 4 (SR4) BIT DEFINITIONS
                        ;*****************************************************************************
000001                  DMVBIT  = BIT0  ;UNIT IS DMV IF BIT = 1
000002                  MODE22  = BIT1  ;UNIT IS IN Q22 MODE IF BIT = 1

                        ;*****************************************************************************
                        ;* DEC/X11 ERROR CODES
                        ;*****************************************************************************
000000                  NOTDFN  = 0     ;ERROR NOT DEFINED
000001                  DATERR  = 1     ;DATA ERROR
000007                  SELERR  = 7     ;SELECTION ERROR
000011                  ILGINT  = 11    ;ILLEGAL INTERRUPT OCCURRED, OR DONE NOT SET
000023                  NOINTR  = 23    ;DEVICE FAILED TO INTERRUPT
000034                  NOINIT  = 34    ;DEVICE WILL NOT INITIALIZE
000036                  BADRED  = 36    ;UNABLE TO EXECUTE READ FUNCTION
000037                  BADWRT  = 37    ;UNABLE TO EXECUTE WRITE FUNCTION


                        ;*****************************************************************************
```



```
                        ;* OFFSETS FOR DMP/DMV11 CSR ADDRESSES
                        ;*****************************************************************************
000000                  SEL0    = 0
000000                  BSEL0   = 0
000001                  BSEL1   = 1
000002                  SEL2    = 2
000002                  BSEL2   = 2
000003                  BSEL3   = 3
000004                  SEL4    = 4
000004                  BSEL4   = 4
000005                  BSEL5   = 5
000006                  SEL6    = 6
000006                  BSEL6   = 6
000007                  BSEL7   = 7
000010                  SEL10   = 10
000010                  BSEL10  = 10
000011                  BSEL11  = 11


                        ;*****************************************************************************
                        ;* INPUT COMMAND TYPE CODES - BSEL2
                        ;*****************************************************************************
000002                  MODDEF  = 2
000001                  CTLIN   = 1
000004                  BACCIT  = 4
000000                  BACCIR  = 0


                        ;*****************************************************************************
                        ;* OUTPUT COMMAND TYPE CODES - BSEL2
                        ;*****************************************************************************
000001                  CTLOUT  = 1
000004                  BACCOT  = 4
000000                  BACCOR  = 0
000003                  BACORU  = 3
000006                  BACOTS  = 6
000007                  BACOTN  = 7
000002                  INFOUT  = 2


                        ;*****************************************************************************
                        ;* BIT MASKS FOR FIELDS IN COMMANDS
                        ;*****************************************************************************
000370                  CMDMSK  = 370           ;MASK FOR COMMAND TYPE IN BSEL2
000370                  MODMSK  = 370           ;MASK FOR MODE FIELD IN BSEL6 FOR MODE DEF'N CMND
000340                  REQMSK  = 340           ;MASK FOR REQUEST KEY IN BSEL6 FOR CTL IN CMND
000340                  RTNMSK  = 340           ;MASK FOR RETURN KEY IN BSEL6 FOR INFO OUT CMND
140000                  CCMSK   = 140000        ;MASK FOR CHARACTER COUNT IN SEL6
000077                  EAMSK   = 077           ;MASK FOR BUS ADRS EA BITS IN BSEL7
000361                  MODESW  = 361           ;MASK FOR DMP11 MODE SWITCHES IN IBUS REG 16


                        ;*****************************************************************************
```

```
                        ;* COMMAND CONTROL, INTERRUPT ENABLE BITS - BSEL0
                        ;*****************************************************************************
000200                  RQI     = BIT7
000020                  IED     = BIT4
000001                  IEI     = BIT0


                        ;*****************************************************************************
                        ;* COMMAND CONTROL BITS - BSEL2
                        ;*****************************************************************************
000200                  RDYO    = BIT7
000020                  RDYI    = BIT4
000010                  Q22BIT  = BIT3


                        ;*****************************************************************************
                        ;* MICROPROCESSOR CONTROL COMMAND - BSEL1
                        ;*****************************************************************************
000200                  RUN     = BIT7
000100                  MCLR    = BIT6
000020                  STEPLU  = BIT4
000010                  LULOOP  = BIT3
000004                  ROMO    = BIT2
000002                  ROMI    = BIT1
000001                  STEPMP  = BIT0


                        ;*****************************************************************************
                        ;* MODE DEFINITION COMMAND - BSEL6
                        ;*****************************************************************************
000000                  HDPPDC  = 0
000001                  FDPPDC  = 1
000002                  HDPP    = 2
000003                  FDPP    = 3
000004                  HDCS    = 4
000005                  FDCS    = 5
000006                  HDTS    = 6
000007                  FDTS    = 7


                        ;*****************************************************************************
                        ;* CONTROL IN COMMAND - BSEL6
                        ;*****************************************************************************
000200                  WRITSS  = BIT7
000100                  RDCTSS  = BIT6
000040                  REDTSS  = BIT5
000001                  ESTRIB  = 01
000002                  KILLTR  = 02
000003                  ISTART  = 03
000004                  MAINT   = 04
000005                  HALTST  = 05
000020                  REDMDM  = 20
000021                  WRIMDM  = 21
000022                  INTFDG  = 22
```



```
000023                  SELMDM  = 23
000112                  DEROTB  = 112
000113                  DERINB  = 113


                        ;*****************************************************************************
                        ;* CONTROL IN COMMAND - BSEL7
                        ;*****************************************************************************
000200                  DISPIG  = BIT7
000100                  ENBPIG  = BIT6
000040                  LATPS   = BIT5
000020                  ULATPS  = BIT4
000002                  DISCMP  = BIT1
000001                  ENACMP  = BIT0


                        ;*****************************************************************************
                        ;* BUFFER ADDRESS / CHARACTER COUNT IN/OUT COMMAND - SEL4
                        ;*****************************************************************************
100000                  BA15    = BIT15
040000                  BA14    = BIT14
020000                  BA13    = BIT13
010000                  BA12    = BIT12
004000                  BA11    = BIT11
002000                  BA10    = BIT10
001000                  BA9     = BIT9
000400                  BA8     = BIT8
000200                  BA7     = BIT7
000100                  BA6     = BIT6
000040                  BA5     = BIT5
000020                  BA4     = BIT4
000010                  BA3     = BIT3
000004                  BA2     = BIT2
000002                  BA1     = BIT1
000001                  BA0     = BIT0


                        ;*****************************************************************************
                        ;* BUFFER ADDRESS / CHARACTER COUNT IN/OUT - SEL6
                        ;*****************************************************************************
100000                  BA17    = BIT15
040000                  BA16    = BIT14
020000                  CC13    = BIT13
010000                  CC12    = BIT12
004000                  CC11    = BIT11
002000                  CC10    = BIT10
001000                  CC9     = BIT9
000400                  CC8     = BIT8
000200                  CC7     = BIT7
000100                  CC6     = BIT6
000040                  CC5     = BIT5
000020                  CC4     = BIT4
000010                  CC3     = BIT3
```

```
000004          CC2     = BIT2
000002          CC1     = BIT1
000001          CC0     = BIT0


                ;*****************************************************************************
                ;* CONTROL OUT COMMAND - BSEL6
                ;*****************************************************************************
000002          RTHRER  = 2
000004          TTHRER  = 4
000006          STHRER  = 6
000010          STARUN  = 10
000012          MNTRUN  = 12
000014          MNTHLT  = 14
000020          RINGDT  = 20
000022          DEADTR  = 22
000024          RUNST   = 24
000026          BABTRB  = 26
000030          STRTRB  = 30
000300          BUFTSM  = 300
000302          NONEXM  = 302
000304          DISCON  = 304
000306          QUOVRN  = 306


                ;*****************************************************************************
                ;* INFORMATION OUT COMMAND - BSEL6
                ;*****************************************************************************
000100          RDCTSS  = BIT6
000040          REDTSS  = BIT5
000000          NORET   = 00
000010          RETMST  = 10
000020          BUFRTC  = 20
000012          EROTB   = 12
000013          ERINB   = 13


                ;;*****************************************************************************
                ;* DMP OBUS REG 10 - TRANSMITTER BUFFER
                ;;*****************************************************************************
000200          TX7     = BIT7
000100          TX6     = BIT6
000040          TX5     = BIT5
000020          TX4     = BIT4
000010          TX3     = BIT3
000004          TX2     = BIT2
000002          TX1     = BIT1
000001          TX0     = BIT0

                ;;*****************************************************************************
                ;* DMP OBUS REG 11
                ;;*****************************************************************************
000200          OC      = BIT7
```



```
000010          GOAH    = BIT3
000004          ABORT   = BIT2
000002          EOM     = BIT1
000001          SOM     = BIT0

                ;;*****************************************************************************
                ;* DMP OBUS REG 12
                ;;*****************************************************************************
000200          IC      = BIT7
000100          BPOLL   = BIT6
000040          LULP    = BIT5

                ;;*****************************************************************************
                ;* DMP OBUS REG 13
                ;;*****************************************************************************
000200          POLL    = BIT7
000100          DTR     = BIT6
000040          SELFR   = BIT5
000020          HDX     = BIT4
000010          MAINT1  = BIT3
000004          MAINT2  = BIT2
000002          SELSBY  = BIT1

                ;;*****************************************************************************
                ;* DMP OBUS REG 14
                ;;*****************************************************************************
000100          TXEN    = BIT6
000040          DISSI   = BIT5
000020          RDAX    = BIT4
000010          WAX     = BIT3
000004          ENAX    = BIT2
000002          AX2     = BIT1
000001          AX1     = BIT0

                ;;*****************************************************************************
                ;* DMP OBUS REG 17
                ;;*****************************************************************************
000200          CRC2    = BIT7
000100          CRC1    = BIT6
000040          IDLE    = BIT5
000020          SECA    = BIT4
000010          STRIP   = BIT3
000004          RDALL   = BIT2
000002          IERR    = BIT1
000001          DDCMP   = BIT0

                ;;*****************************************************************************
                ;* DMP IBUS REG 10 - RECEIVER BUFFER
                ;;*****************************************************************************
000200          RX7     = BIT7
000100          RX6     = BIT6
000040          RX5     = BIT5
000020          RX4     = BIT4
000010          RX3     = BIT3
000004          RX2     = BIT2
000002          RX1     = BIT1
```

```
                000001          RXO     = BIT0
                                ;;******************************************************************************
                                ;* DMP IBUS REG 12
                                ;;******************************************************************************
                000200          IC      = BIT7
                000100          IACT    = BIT6
                000040          LULP    = BIT5
                000020          IRDY    = BIT4
                000010          OVRR    = BIT3
                000004          RAB     = BIT2
                000002          EBLK    = BIT1
                000001          BCC     = BIT0
                                ;;******************************************************************************
                                ;* DMP IBUS REG 13
                                ;;******************************************************************************
                000200          RING    = BIT7
                000100          DTR     = BIT6
                000040          RTS     = BIT5
                000020          HDX     = BIT4
                000010          MODR    = BIT3
                000004          CS      = BIT2
                000002          STBY    = BIT1
                000001          CARR    = BIT0
                                ;;******************************************************************************
                                ;* DMP IBUS REG 14
                                ;;******************************************************************************
                000200          READY   = BIT7
                000100          TXEN    = BIT6
                000040          DISSI   = BIT5
                000020          RDAX    = BIT4
                000010          WAX     = BIT3
                000004          ENAX    = BIT2
                000002          AX2     = BIT1
                000001          AX1     = BIT0
                                ;;******************************************************************************
                                ;* DMP IBUS REG 16 -- DMV "SWPBOT" (W/BITS RE-ARRANGED)
                                ;;******************************************************************************
                000020          MODSW0  = BIT4
                000040          MODSW1  = BIT5
                000100          MODSW2  = BIT6
                000200          ENABSW  = BIT7
                                ;;******************************************************************************
                                ;* DMP IBUS REG 17
                                ;;******************************************************************************
                000200          SIGR    = BIT7
                000100          SIGQ    = BIT6
                000040          TXDATA  = BIT5
                000020          OCUR    = BIT4
                000010          ICIR    = BIT3
                000004          TESTMD  = BIT2
                000002          MCLK    = BIT1
```



```
                000001          DDCMP   = BIT0
                                ;;******************************************************************************
                                ;* DMP AX3-15 - USYRT REG 6
                                ;;******************************************************************************
                000200          1422    = BIT7
                000100          XYZ     = BIT6
                000040          C32BCC  = BIT5
                000020          V35     = BIT4
                000010          INTGRL  = BIT3
                000004          C32ENB  = BIT2
                000002          OP      = BIT1
                000001          TEST    = BIT0
                                ;******************************************************************************
                                ;* INTERRUPT FLAG BIT DEFINITIONS IN "FLAGS"
                                ;******************************************************************************
                000001          ININT   = BIT0          ; =1 IF INPUT INTRPT OCCURRED
                000002          OUTINT  = BIT1          ; =1 IF OUTPUT INTRPT OCCURRED
                000004          NONQI   = BIT2          ; =1 FOR NON-QUEUED (REQUEST-AND-WAIT) INPUT OPERATION
                000010          NONQO   = BIT3          ; =1 FOR NON-QUEUED (REQUEST-AND-WAIT) OUTPUT OPERATION
                                ;******************************************************************************
                                ;* INTERRUPT STATUS WORD (INTABL) BIT DEFINITIONS
                                ;******************************************************************************
                000001          MDFDON  = BIT0          ;MODE DEFINITION DONE
                000002          HLTDON  = BIT1          ;HALT DONE
                000004          ISTDON  = BIT2          ;ISTART DONE
                000010          BIRDON  = BIT3          ;BACCIR DONE
                000020          BITDON  = BIT4          ;BACCIT DONE
                000040          BORDON  = BIT5          ;BACCOR DONE
                000100          BOTDON  = BIT6          ;BACCOT DONE
                000200          OERDON  = BIT7          ;DATA ERRORS OUTBOUND HAVE BEEN REQUESTED
                000400          IERDON  = BIT8          ;DATA ERRORS INBOUND HAVE BEEN REQUESTED
                100000          OWAIT   = BIT15         ;WAITING FOR OUTPUT ON LOG LINK
                                ;******************************************************************************
                                ;* ERROR FLAG BIT DEFINITIONS IN "ERRORS"
                                ;******************************************************************************
                000001          INTIMO  = BIT0          ;INPUT INTERRUPT TIMED-OUT
                000002          OUTIMO  = BIT1          ;OUTPUT INTERRUPT TIMED-OUT
                000004          BADINI  = BIT2          ;MASTER CLEAR FAILED ON DEVICE

000252' 000000  WBUFEX: .WORD   0       ;ADJUSTED WBUFEA
000254' 000000  RBUFEX: .WORD   0       ;ADJUSTED RBUFEA

000256' 000000  N.DEVS: .WORD   0       ;NUMBER OF DEVICES PRESENT
```

```
000260' 000000          TOTAL:  .WORD   0       ;NUMBER OF DMP11'S CURRENTLY ACTIVE
000262' 000000          COUNT:  .WORD   0       ;ITERATION COUNT FOR MODULE
000264' 000000          SAVBF:  .WORD   0       ;RCV ISR TEMPORARY STORAGE
000266' 000000          SELECT: .WORD   0       ;SOFTWARE BIT MAP OF ACTIVE DEVICES
000270' 000000          DEVPTR: .WORD   0       ;DEVICE SELECTION POINTER
000272' 000000          LURG16: .WORD   0       ;STORAGE FOR LU IBUS REG 16
000274' 000000          CSEL0:  .WORD   0       ;STORAGE FOR INPUT COMMAND TO BE PERFORMED
000276' 000000          CSEL2:  .WORD   0
000300' 000000          CSEL4:  .WORD   0
000302' 000000          CSEL6:  .WORD   0
000304' 000000          CSEL10: .WORD   0
000306' 000000          RSEL0:  .WORD   0       ;STORAGE FOR CSR'S RETURNED ON OUTPUT CMND
000310' 000000          RSEL2:  .WORD   0
000312' 000000          RSEL4:  .WORD   0
000314' 000000          RSEL6:  .WORD   0
000316' 000000          RSEL10: .WORD   0
000320' 000000          FLAGS:  .WORD   0       ;INPUT AND OUTPUT INTERRUPT FLAG BITS
000322' 000000          ERRORS: .WORD   0       ;ERROR FLAG BITS
000324' 000000          INTIMR: .WORD   0       ;INPUT INTERRUPT TIMER
000326' 000000          OUTIMR: .WORD   0       ;OUTPUT INTERRUPT TIMER
000330' 000000          LLKCNT: .WORD   0       ;COUNT OF LOGICAL LINKS
                                                ;   IF P-TO-P, THIS IS SAME AS NO. OF DEVICES.
                                                ;   IF MULTIPOINT, THIS IS SAME AS NO. OF TRIBUTARIES.
000332' 000000          ISTCNT: .WORD   0       ;COUNT OF ISTART ATTEMPTS ON LOGICAL LINK
000334' 000000          SCRACH: .WORD   0       ;MISCELLANEOUS STORAGE LOCATION
000336' 000000          TRBADR: .WORD   0       ;CURRENT TRIB ADDRESS
000340' 000000          TRBMAX: .WORD   0       ;MAXIMUM TRIB ADDRESS
000342' 000000          LNKDON: .WORD   0       ;NO. OF LOG. LNKS DONE WITH MSG
000344' 000000          OUTDON: .WORD   0       ;OUTPUT COUNT OF LOGICAL LINKS DONE WITH MSG
                                                ;   PER ITERATION
000346' 000000          WATCNT: .WORD   0       ;MULTIDROP CNT OF LNKS AWAITING OUTPUT

                        ;*****************************************************************************
                        ;* INTABLE - INTERRUPT STATUS TABLE - 32 (MAX.) 1-WORD ENTRIES, 1 ENTRY
                        ;*  PER LOGICAL LINK.
                        ;*
                        ;*  IF POINT-TO-POINT OR LPBK MODE, UP TO 16 ENTRIES CAN BE USED, WHERE AN ENTRY
                        ;*     CONSISTS OF 16 INTERRUPT STATUS FLAG BITS.
                        ;*  IF MULTIPOINT MODE, UP TO 32 ENTRIES CAN BE USED, WHERE AN ENTRY CONSISTS OF
                        ;*     16 INTERRUPT STATUS FLAG BITS.
                        ;*  IN ANY CASE, THE NO. OF OCCUPIED ENTRIES = THE NO. OF LOGICAL LINKS, KEPT IN
                        ;*     LLKCNT.
                        ;*****************************************************************************
000350' 000040          INTABL: .BLKW   32.


                        ; REGISTER ADDRESS TABLE FOR ERROR REPORTS
000450' 160170          REGTBL: .WORD   160170
000452' 160172                  .WORD   160172
000454' 160174                  .WORD   160174
000456' 160176                  .WORD   160176
000460' 160200                  .WORD   160200
000462' 177777                  .WORD   177777
```



```
                        ;-----------------------------------------------------------------------------
                        ; THIS IS START OF MODULE
                        ;-----------------------------------------------------------------------------
000464' 016767  177324  177574  START:  MOV     DVID1,SELECT    ;GET ACTIVE DEVICES
000472' 001004                  BNE     SETUP           ;BR IF ANY ARE SELECTED
000474' 004767  005400          DRPMOD: JSR     PC,DISABL       ;DISABLE INTERRUPTS ON ALL SELECTED DEVICES
000500' 104410  000000'                 END$,BEGIN              ;;DROP THE DEVICE MODULE

                        ;-----------------------------------------------------------------------------
                        ;SETUP VECTORS FOR ACTIVE DEVICES, AND INITIALIZE THEM
                        ;-----------------------------------------------------------------------------
000504' 005067  177546          SETUP:  CLR     N.DEVS          ;CLEAR NO. OF DEVICES
000510' 005067  177606          CLR     ERRORS          ;CLEAR ERROR FLAGS
000514' 012767  000010  177576  MOV     #NONQO,FLAGS    ;SET FLAG FOR NON-QUEUED OUTPUT INTRPTS
000522' 004767  005330          JSR     PC,CLRCMD       ;CLEAR COMMAND STORAGE AREA
000526' 012704  000350'         MOV     #INTABL,R4      ;INIT POINTER TO INTRPT STATUS TABLE
000532' 005024          1$:     CLR     (R4)+           ;CLEAR A TABLE ENTRY
000534' 020427  000450'         CMP     R4,#INTABL+64.  ;SEE IF ALL ENTRIES CLEARED YET
000540' 103774          BLO     1$              ;BR IF NOT YET
000542' 016702  177242          MOV     VECTOR,R2       ;GET INITIAL VECTOR
000546' 016700  177234          MOV     ADDR,R0         ;GET INITIAL ADDRESS
000552' 012703  006352'         MOV     #IISR0,R3       ;INIT ISR POINTER
000556' 012767  000001  177504  MOV     #BIT0,DEVPTR    ;INIT SELECTION POINTER
000564' 005067  177540          CLR     LLKCNT          ;INIT LOGICAL LINK CNT
                        ;FIND A SELECTED DEVICE
000570' 036767  177474  177470  2$:     BIT     DEVPTR,SELECT   ;SEE IF THIS DEVICE IS SELECTED
000576' 001021          BNE     5$              ;BR IF SELECTED
000600' 006367  177464          3$:     ASL     DEVPTR          ;SHIFT SELECTION POINTER
000604' 001002          BNE     4$              ;BR IF NOT ALL DEVICES SCANNED YET
000606' 000167  000662          JMP     STRTUP          ;ALL DEVICES SCANNED - START LINKS
000612' 062703  000036  4$:     ADD     #36,R3          ;POP ISR POINTER
000616' 062702  000010          ADD     #10,R2          ;POP VECTOR
000622' 005767  177176          TST     SR4             ; IS THIS A DMV ?
000626' 001402          BEQ     25$
000630' 062700  000010          ADD     #10,R0          ;IF YES: POP ADDRESS (20)
000634' 062700  000010  25$:    ADD     #10,R0          ;IF NO: POP DMP11 ADDRESS (10)
000640' 000753          BR      2$              ;CONTINUE
                        ; MASTER CLEAR THIS DEVICE
000642' 112760  000100  000001  5$:     MOVB    #MCLR,BSEL1(R0) ;SET MASTER CLEAR
000650' 005767  177150          TST     SR4             ; IS THIS A DMV ?
000654' 001011          BNE     23$             ; IF YES: SKIP LULOOP/RUN STUFF
000656' 112760  000200  000001          MOVB    #RUN,BSEL1(R0)  ;SET RUN BIT
000664' 142760  000200  000001          BICB    #RUN,BSEL1(R0)  ;CLEAR RUN
000672' 152760  000010  000001          BISB    #LULOOP,BSEL1(R0) ;SET LULOOP
000700' 004767  004664          23$:    JSR     PC,READ16       ;READ LU REG 16 FOR SWITCHES
000704' 004767  004500          JSR     PC,INIDEV       ;MASTER CLEAR DEVICE
000710' 132767  000004  177404  BITB    #BADINI,ERRORS  ;SEE IF INIT WAS BAD
000716' 001071          BNE     8$              ;IF INIT BAD, GO DROP MODULE
                        ; LOAD INPUT AND OUTPUT VECTORS FOR THIS DEVICE
000720' 010312          MOV     R3,(R2)         ;LOAD INPUT ISR VECTOR
000722' 116762  177064  000002          MOVB    BR1,2(R2)       ;LOAD PRIORITY
000730' 105062  000003          CLRB    3(R2)           ;CLEAR HI BYTE OF PROGRAM STATUS WORD
000734' 010063  000034          MOV     R0,34(R3)       ;LOAD DMP11 ADDRESS
000740' 010362  000004          MOV     R3,4(R2)        ;LOAD OUTPUT ISR VECTOR
```

```
 901  000744' 062762  000016  000004          ADD     #16,4(R2)       ;POINT TO OUTPUT ISR
 902  000752' 116762  177034  000006          MOVB    BR1,6(R2)       ;LOAD PRIORITY
 903  000760' 105062  000007                  CLRB    7(R2)           ;CLEAR HI BYTE OF STATUS WORD
 904                                  ;IF DMV/INTERNAL-LOOP SELECTED THEN SETUP DEVICE FOR IT HERE
 905  000764' 005767  177034                  TST     SR4             ;IF DMV RUNNING...
 906  000770' 001412                          BEQ     24$             ; (BR IF NOT)
 907  000772' 026727  177020  000000          CMP     SR1,#0          ;AND WITH LOOPBACK...
 908  001000' 001006                          BNE     24$             ; (BR IF NOT)
 909  001002' 026727  177012  000000          CMP     SR2,#0          ;AND WITH INTERNAL LOOPBACK...
 910  001010' 001002                          BNE     24$             ; (BR IF NOT)
 911  001012' 004767  004172                  JSR     PC,DMVLUP       ; THEN SET TTLOOP W/56K BPS INTERNAL
 912                                  ;DEFINE THE MODE OF THIS DEVICE (FDX, HDX, P-P, M-P CTL)
 913  001016' 112767  000002  177252  24$:    MOVB    #MODDEF,CSEL2   ;SET MODE DEFINITION COMMAND
 914  001024' 152760  000010  000001          BISB    #LULOOP,BSEL1(R0) ;SET LULOOP TO FORCE MODE
 915  001032' 126727  176760  000000          CMPB    SR1,#0          ;SEE IF GRP OF LOOPED-BACK DEVICES
 916  001040' 001025                          BNE     9$              ;BR IF NOT
 917  001042' 112767  000003  177232  6$:     MOVB    #FDPP,CSEL6     ;SET FDX P-TO-P MODE
 918  001050' 004767  004634          7$:     JSR     PC,INCMND       ;PERFORM MODE DEF'N CMND
 919  001054' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
 920  001060' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
 921  001064' 142760  000010  000001          BICB    #LULOOP,BSEL1(R0) ;CLEAR LU LOOP
 922  001072' 032767  000001  177222          BIT     #INTIMO,ERRORS  ;SEE IF INPUT INTRPT TIMED-OUT
 923  001100' 001464                          BEQ     16$             ;BR IF NO TIME-OUT
 924  001102' 142760  000200  000001  8$:     BICB    #RUN,BSEL1(R0)  ;CLEAR RUN ON THIS DEVICE
 925  001110' 000167  177360                  JMP     DRPMOD          ;GO DROP MODULE
 926  001114' 126727  176676  000001  9$:     CMPB    SR1,#1          ;SEE IF GRP OF P-TO-P DEVICES
 927  001122' 001015                          BNE     12$             ;BR IF NOT
 928  001124' 132767  000200  177140          BITB    #ENABSW,LURG16  ;SEE IF LU SWITCHES ENABLED
 929  001132' 001004                          BNE     11$             ;BR IF YES
 930  001134' 112767  000002  177140  10$:    MOVB    #HDPP,CSEL6     ;SET HALF-DPX P-TO-P MODE
 931  001142' 000742                          BR      7$              ;GO SET MODE
 932  001144' 132767  000020  177120  11$:    BITB    #MODSW0,LURG16  ;SEE IF HDX IN SWITCHES
 933  001152' 001770                          BEQ     10$             ;BR IF HDX
 934  001154' 000732                          BR      6$
 935  001156' 126727  176634  000002  12$:    CMPB    SR1,#2          ;SEE IF MULTI-POINT CTL STATION
 936  001164' 001020                          BNE     15$             ;BR IF NOT
 937  001166' 132767  000200  177076          BITB    #ENABSW,LURG16  ;SEE IF LU SWITCHES ENABLED
 938  001174' 001004                          BNE     14$             ;BR IF YES
 939  001176' 112767  000004  177076  13$:    MOVB    #HDCS,CSEL6     ;SET HDX CTL MODE
 940  001204' 000721                          BR      7$              ;GO SET MODE
 941  001206' 132767  000020  177056  14$:    BITB    #MODSW0,LURG16  ;SEE IF HDX IN SWITCHES
 942  001214' 001770                          BEQ     13$             ;BR IF HDX
 943  001216' 112767  000005  177056          MOVB    #FDCS,CSEL6     ;SET FDX CTL MODE
 944  001224' 000711                          BR      7$              ;GO SET MODE
 945  001226' 004767  004710          15$:    JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
 946  001232' 012767  000007  176646          MOV     #SELERR,ERRTYP  ;SET CODE FOR SELECTION ERROR
 947                                          ;*****************************************************************
 948  001240' 104405  000000' 000000          HRDER$,BEGIN,NULL       ;OPERATOR SPEC'D INVALID SR VALUE
 949                                          ;*****************************************************************
 950  001246' 000167  177222                  JMP     DRPMOD          ;GO DROP THE MODULE
 951                                  ; IF H3254,5 LOOPBACK, SELECT MODEM INTERFACE DESIRED BY OPERATOR
 952  001252' 005767  176546  16$:    TST     SR4             ; IS THIS A DMV ?
 953  001256' 001054                          BNE     20$             ; IF YES: SKIP INTERFACE SETUP.
 954  001260' 126727  176532  000000          CMPB    SR1,#0          ;SEE IF LOOPBACK MODE
 955  001266' 001050                          BNE     20$             ;BR IF NOT
 956  001270' 026727  176524  000001          CMP     SR2,#1          ;SEE IF H3254,5 USED
```



```
 957  001276' 001044                          BNE     20$             ;BR IF NOT
 958  001300' 012767  000333  176772          MOV     #I422!XYZ!V35!INTGRL!OP!TEST,CSEL4 ;INIT MODEM INTERFACE BITS
 959  001306' 016701  176510                  MOV     SR3,R1          ;GET DESIRED MODEM SELECTION
 960  001312' 001004                          BNE     17$             ;BR IF NOT DEFAULT
 961  001314' 042767  000020  176756          BIC     #V35,CSEL4      ;SELECT V.35 BY DEFAULT
 962  001322' 000416                          BR      19$
 963  001324' 020127  000200          17$:    CMP     R1,#I422        ;SEE IF 422 DESIRED
 964  001330' 001411                          BEQ     18$             ;BR IF YES
 965  001332' 020127  000100                  CMP     R1,#XYZ         ;SEE IF XYZ DESIRED
 966  001336' 001406                          BEQ     18$             ;BR IF YES
 967  001340' 020127  000020                  CMP     R1,#V35         ;SEE IF V.35 DESIRED
 968  001344' 001403                          BEQ     18$             ;BR IF YES
 969  001346' 020127  000010                  CMP     R1,#INTGRL      ;SEE IF INTEGRAL MODEM DESIRED
 970  001352' 001325                          BNE     15$             ;BR IF NOT, TO REPORT ILLEGAL SR3 VALUE
 971  001354' 040167  176720          18$:    BIC     R1,CSEL4        ;SELECT THE INTERFACE IN WORD
 972  001360' 012767  000001  176710  19$:    MOV     #CTLIN,CSEL2    ;SET CTL IN CMND
 973  001366' 012767  000023  176706          MOV     #SELMDM,CSEL6   ;CODE FOR SELECT MODEM INTERFACE
 974  001374' 004767  004310                  JSR     PC,INCMND       ;SELECT THE INTERFACE IN REG AX3-15
 975  001400' 032767  000001  176714          BIT     #INTIMO,ERRORS  ;SEE IF INPUT CMND TIMED-OUT
 976  001406' 001235                          BNE     8$              ;BR IF YES, TO DROP MODULE
 977  001410' 005267  176642          20$:    INC     N.DEVS          ;UPDATE THE NO. OF DEVICES TO RUN
 978  001414' 126727  176376  000000          CMPB    SR1,#0          ;SEE IF RUNNING WITH LOOPBACK
 979  001422' 001007                          BNE     21$             ;BR IF NOT
 980  001424' 026727  176370  000000          CMP     SR2,#0          ;SEE IF INTERNAL LOOPBACK DESIRED
 981  001432' 001003                          BNE     21$             ;BR IF NOT
 982  001434' 152760  000010  000001          BISB    #LULOOP,BSEL1(R0) ;SET LINE UNIT LOOP
 983                                  ;COMPUTE NO. OF LOGICAL LINKS IN LLKCNT
 984  001442' 126727  176350  000002  21$:    CMPB    SR1,#2          ;SEE IF MULTIPOINT
 985  001450' 001406                          BEQ     22$             ;BR IF MULTIPOINT
 986  001452' 105060  000000                  CLRB    BSEL0(R0)       ;DISABLE INTRPTS FROM THIS DEVICE
 987  001456' 005267  176646                  INC     LLKCNT          ;INCR NO. OF LOGICAL LINKS
 988  001462' 000167  177112                  JMP     3$              ;CONTINUE SETUP
 989  001466' 016767  176326  176634  22$:    MOV     SR2,LLKCNT      ;STORE NO. OF LOGICAL LINKS
 990
 991                                  ;-------------------------------------------------------------------------
 992                                  ; ATTEMPT TO START THE PROTOCOL ON EACH LOGICAL LINK, AND IF A LOGICAL LINK
 993                                  ; CAN'T BE STARTED, DROP THE DEVICE WHICH CONTROLS THAT LINK, FROM TESTING
 994                                  ;-------------------------------------------------------------------------
 995  001474'                        STRTUP:
 996  001474' 012767  000001  176634          MOV     #1,TRBADR       ;SET P-TO-P TRIB ADRS = 1
 997  001502' 126727  176310  000002          CMPB    SR1,#2          ;SEE IF MULTIPOINT
 998  001510' 001003                          BNE     1$              ;BR IF NOT MULTIPOINT
 999  001512' 016767  176304  176616          MOV     SR3,TRBADR      ;INIT MULTIPOINT TRIB ADRS
1000  001520' 016705  176604          1$:     MOV     LLKCNT,R5       ;INIT LOG LNK COUNTER
1001  001524' 016700  176256                  MOV     ADDR,R0         ;INIT DEVICE ADRS
1002  001530' 012767  000001  176532          MOV     #BIT0,DEVPTR    ;INIT SELECTION POINTER
1003                                  ;FIND A SELECTED DEVICE TO START
1004  001536' 036767  176526  176522  2$:     BIT     DEVPTR,SELECT   ;SEE IF THIS DEVICE IS SELECTED
1005  001544' 001015                          BNE     22$             ;BR IF SELECTED
1006  001546' 006367  176516          20$:    ASL     DEVPTR          ;SHIFT SELECTION POINTER
1007  001552' 001002                          BNE     21$             ;BR TO KEEP SCANNING
1008  001554' 000167  000706                  JMP     18$             ;ALL DEVICES SCANNED
1009  001560' 005767  176240          21$:    TST     SR4             ; IS THIS A DMV ?
1010  001564' 001402                          BEQ     25$
1011  001566' 062700  000010                  ADD     #10,R0          ;IF YES: INCREMENT DEVICE ADDRESS (20)
1012  001572' 062700  000010          25$:    ADD     #10,R0          ;IF NO: INCREMENT DMP11 ADDRESS (10)
```

```
1013  001576' 000757                        BR      2$                      ;CONTINUE
1014                                ;SET UP FOR CONTROL IN COMMAND
1015  001600' 112767  000001  176470  22$:   MOVB    #CTLIN,CSEL2            ;SET CONTROL IN CMND
1016  001606' 116767  176524  176463        MOVB    TRBADR,CSEL2+1          ;SET TRIB ADRS
1017  001614' 005067  176460               CLR     CSEL4
1018  001620' 005067  176456               CLR     CSEL6
1019                                ;ESTABLISH A LOGICAL LINK
1020  001624' 112767  000001  176450        MOVB    #ESTRIB,CSEL6           ;SET CODE FOR ESTABLISH TRIB CMND
1021  001632' 042767  000002  176460        BIC     #OUTINT,FLAGS           ;CLEAR OUTPUT INTRPT FLAG
1022  001640' 004767  004044                JSR     PC,INCMND               ;ESTABLISH THIS TRIBUTARY
1023                                ;CHANGE THE SELECTION INTERVAL TO 5 SECONDS
1024  001644' 005767  176154                TST     SR4                     ; IS THIS A DMV ?
1025  001650' 001404                        BEQ     23$
1026  001652' 012767  000764  176420        MOV     #500.,CSEL4             ;IF YES: CONSTANT = 500.
1027  001660' 000403                        BR      24$
1028  001662' 012767  011610  176410  23$:   MOV     #5000.,CSEL4            ;IF NO: CONSTANT = 5000.
1029  001670' 012767  000236  176404  24$:   MOV     #WRITSS+36,CSEL6 ;WRITE TSS ADDRESS 36 (SEL INTERVAL).
1030  001676' 042767  000002  176414        BIC     #OUTINT,FLAGS           ;CLEAR OUTPUT INTRPT FLAG.
1031  001704' 004767  004000                JSR     PC,INCMND               ;CHANGE THE INTERVAL.
1032                                ;ISTART THE LOGICAL LINK
1033  001710' 005067  176364                CLR     CSEL4                   ;CLEAR INTERVAL VALUE.
1034  001714' 012767  000010  176410        MOV     #8.,ISTCNT              ;INIT ISTART COUNTER FOR APPROX 5 MINUTES
1035                                                                        ;   STARTUP (EACH COUNT = 35. SEC.)
1036  001722' 112767  000003  176352  3$:   MOVB    #ISTART,CSEL6           ;SET CODE FOR ISTART REQUEST
1037  001730' 042767  000002  176362        BIC     #OUTINT,FLAGS           ;CLEAR OUTPUT INTRPT FLAG
1038  001736' 004767  003746                JSR     PC,INCMND               ;PERFORM ISTART CMND
1039  001742' 012767  177777  176356        MOV     #-1,OUTIMR              ;INIT OUTPUT INTERRUPT TIMER
1040  001750'                           4$:
1041  001750' 104407  000000'              BREAK$,BEGIN                    ;TEMPORARY RETURN TO MONITOR....
1042  001754' 104407  000000'              BREAK$,BEGIN                    ;THEN CONTINUE AT NEXT INSTRUCTION.
1043  001760' 132767  000002  176332        BITB    #OUTINT,FLAGS           ;SEE IF OUTPUT INTRPT SERVICED YET
1044  001766' 001015                        BNE     5$                      ;BR IF YES
1045  001770' 005367  176332                DEC     OUTIMR                  ;DECREMENT TIMER
1046  001774' 001365                        BNE     4$                      ;BR IF OUTPUT INTRPT DIDN'T TIME-OUT YET
1047  001776' 004767  004140                JSR     PC,GETERR               ;LOAD ERROR INFORMATION FOR PRINTOUT
1048  002002' 012767  000023  176076        MOV     #NOINTR,ERRTYP          ;CODE FOR DEVICE FAILED TO INTRPT
1049                                        ;*****************************************************************
1050  002010' 104405  000000' 000450'      HRDER$,BEGIN,REGTBL             ;OUTPUT INTERRUPT TIMED-OUT
1051                                        ;*****************************************************************
1052  002016' 000167  000360                JMP     14$                     ;GO DROP MODULE
1053  002022' 142767  000370  176260  5$:   BICB    #CMDMSK,RSEL2           ;MASK FOR OUTPUT COMMAND FIELD
1054  002030' 126727  176254  000001        CMPB    RSEL2,#CTLOUT           ;CHK FOR CTL OUT CMND RETURNED
1055  002036' 001412                        BEQ     6$                      ;BR IF YES
1056  002040' 004767  004076                JSR     PC,GETERR               ;LOAD ERROR INFORMATION FOR PRINTOUT
1057  002044' 012767  000000  176034        MOV     #NOTDFN,ERRTYP          ;CODE FOR ERROR NOT DEFINED (MUST INTERPRET
1058                                                                        ;   CSR CONTENTS)
1059                                        ;*****************************************************************
1060  002052' 104405  000000' 000450'      HRDER$,BEGIN,REGTBL             ;UNEXPECTED OUTPUT CMND
1061                                        ;*****************************************************************
1062  002060' 000167  000316                JMP     14$                     ;GO DROP MODULE
1063                                ;CHECK FOR LOGICAL LINK RUNNING YET
1064  002064' 126727  176224  000024  6$:   CMPB    RSEL6,#RUNSI            ;SEE IF PROTOCOL RUNNING ON THIS LOG LINK YET
1065  002072' 001002                        BNE     7$                      ;BR IF NOT
1066  002074' 000167  000312                JMP     15$                     ;GO SELECT NEXT LOGICAL LINK
1067                                ;CHECK FOR TRANSMIT OR SELECT THRESHOLD ERROR
1068  002100' 126727  176210  000004  7$:   CMPB    RSEL6,#TTHRER           ;SEE IF TRANSMIT THRESHOLD ERROR (TRIB NOT RESPONDING)
```



```
1069  002106' 001416                        BEQ     8$                      ;BR IF YES
1070  002110' 126727  176200  000006        CMPB    RSEL6,#STHRER           ;SEE IF SELECT THRESHOLD ERROR
1071  002116' 001412                        BEQ     8$                      ;BR IF YES
1072  002120' 004767  004016                JSR     PC,GETERR               ;LOAD ERROR INFORMATION FOR PRINTOUT
1073  002124' 012767  000000  175754        MOV     #NOTDFN,ERRTYP          ;CODE FOR ERROR NOT DEFINED (MUST INTERPRET
1074                                                                        ;   CSR CONTENTS)
1075                                        ;*****************************************************************
1076  002132' 104405  000000' 000450'      HRDER$,BEGIN,REGTBL             ;UNEXPECTED CTL OUT CMND
1077                                        ;*****************************************************************
1078  002140' 000167  000236                JMP     14$                     ;GO DROP MODULE
1079  002144' 005367  176162        8$:     DEC     ISTCNT                  ;DECREMENT ISTART COUNTER
1080  002150' 001012                        BNE     9$                      ;BR IF COUNT NOT OVERFLOWED YET
1081  002152' 004767  003764                JSR     PC,GETERR               ;LOAD ERROR INFORMATION FOR PRINTOUT
1082  002156' 012767  000000  175722        MOV     #NOTDFN,ERRTYP          ;CODE FOR ERROR NOT DEFINED (MUST INTERPRET
1083                                                                        ;   CSR CONTENTS)
1084                                        ;*****************************************************************
1085  002164' 104405  000000' 000450'      HRDER$,BEGIN,REGTBL             ;TIMED-OUT ATTEMPTING TO START LINK
1086                                        ;*****************************************************************
1087  002172' 000167  000204                JMP     14$                     ;GO DROP MODULE
1088                                ;HALT PROTOCOL PRIOR TO ISTART RETRY
1089  002176' 112767  000005  176076  9$:   MOVB    #HALTST,CSEL6           ;SET CODE FOR HALT REQUEST
1090  002204' 042767  000002  176106        BIC     #OUTINT,FLAGS           ;CLEAR OUTPUT INTRPT FLAG
1091  002212' 004767  003472                JSR     PC,INCMND               ;PERFORM HALT CMND TO RE-INIT LINK STATUS
1092  002216' 012767  020000  176102        MOV     #20000,OUTIMR           ;INIT OUTPUT INTERRUPT TIMER
1093  002224'                           10$:
1094  002224' 104407  000000'              BREAK$,BEGIN                    ;TEMPORARY RETURN TO MONITOR....
1095  002230' 104407  000000'              BREAK$,BEGIN                    ;THEN CONTINUE AT NEXT INSTRUCTION.
1096  002234' 032767  000002  176056        BIT     #OUTINT,FLAGS           ;SEE IF OUTPUT INTRPT SERVICED YET
1097  002242' 001015                        BNE     11$                     ;BR IF YES
1098  002244' 005367  176056                DEC     OUTIMR                  ;DECREMENT TIMER
1099  002250' 001365                        BNE     10$                     ;BR IF OUTPUT INTRPT DIDN'T TIME-OUT YET
1100  002252' 004767  003664                JSR     PC,GETERR               ;LOAD ERROR INFORMATION FOR PRINTOUT
1101  002256' 012767  000023  175622        MOV     #NOINTR,ERRTYP          ;CODE FOR DEVICE FAILED TO INTRPT
1102                                        ;*****************************************************************
1103  002264' 104405  000000' 000450'      HRDER$,BEGIN,REGTBL             ;OUTPUT INTERRUPT TIMED-OUT
1104                                        ;*****************************************************************
1105  002272' 000167  000104                JMP     14$                     ;GO DROP MODULE
1106  002276' 142767  000370  176004  11$:   BICB    #CMDMSK,RSEL2           ;MASK FOR OUT CMND RETURNED
1107                                ;CHECK FOR INFO OUT WITH "BUFFERS RETURNED" CODE
1108  002304' 126727  176000  000002        CMPB    RSEL2,#INFOUT           ;CHK FOR INFO OUT CMND RETURNED
1109  002312' 001022                        BNE     13$                     ;BR IF NOT
1110  002314' 142767  000340  175772        BICB    #RTNMSK,RSEL6           ;MASK FOR RETURN KEY
1111  002322' 126727  175766  000020        CMPB    RSEL6,#BUFRTC           ;CHK FOR BUFFER RETURN COMPLETE
1112  002330' 001002                        BNE     12$                     ;BR IF NOT BUFFER RETURN COMPLETE
1113  002332' 000167  177364                JMP     3$                      ;GO TRY TO ISTART AGAIN
1114  002336' 004767  003600        12$:    JSR     PC,GETERR               ;LOAD ERROR INFORMATION FOR PRINTOUT
1115  002342' 012767  000000  175536        MOV     #NOTDFN,ERRTYP          ;CODE FOR ERROR NOT DEFINED (MUST INTERPRET
1116                                                                        ;   CSR CONTENTS)
1117                                        ;*****************************************************************
1118  002350' 104405  000000' 000450'      HRDER$,BEGIN,REGTBL             ;UNEXPECTED INFO OUT CMND
1119                                        ;*****************************************************************
1120  002356' 000411                        BR      14$                     ;GO DROP MODULE
1121  002360' 042767  000002  175732  13$:   BIC     #OUTINT,FLAGS           ;CLEAR OUTPUT INTRPT FLAG
1122  002366' 012767  020000  175732        MOV     #20000,OUTIMR           ;INIT OUTPUT INTRPT TIMER
1123  002374' 105060  000002                CLRB    BSEL2(R0)               ;RELEASE THE PORT AGAIN
1124  002400' 000711                        BR      10$                     ;KEEP LOOKING FOR INFO OUT CMND
```

```
 1125  002402' 005060  000000          14$:    CLR     SEL0(R0)        ;CLEAR RUN, DISABLE INTRPTS ON THIS DEVICE
 1126  002406' 000167  176062                  JMP     DRPMOD          ;GO DROP MODULE
 1127                                  ;SELECT NEXT LOGICAL LINK TO START
 1128  002412' 105060  000002          15$:    CLRB    BSEL2(R0)       ;CLEAR BSEL2 (INCLUDING RDYO)
 1129  002416' 126727  175374  000002          CMPB    SR1,#2          ;SEE IF MULTIPOINT
 1130  002424' 001010                          BNE     16$             ;BR IF NOT MULTIPOINT
 1131  002426' 005267  175704                  INC     TRBADR          ;INCREMENT TRIB ADRS
 1132  002432' 105767  175700                  TSTB    TRBADR          ;SEE IF TRIB ADRS OVERFLOW
 1133  002436' 001007                          BNE     17$             ;BR IF NOT
 1134  002440' 005267  175672                  INC     TRBADR          ;IF TRIB ADRS = 0, MAKE IT 1
 1135                                                                  ;    TRIB ADRS = 0 IS ILLEGAL
 1136  002444' 000404                          BR      17$             ;PROCEED
 1137  002446' 105060  000000          16$:    CLRB    BSEL0(R0)       ;DISABLE INTRPTS FROM THIS DEVICE
 1138  002452' 000167  177070                  JMP     20$             ;GO START NEXT LOGICAL LINK
 1139  002456' 005305                  17$:    DEC     R5              ;DECREMENT LOG LNK COUNTER
 1140  002460' 001402                          BEQ     18$             ;BR IF NO MORE LNKS TO DO
 1141  002462' 000167  177112                  JMP     22$             ;GO START NEXT LOGICAL LINK
 1142                                  ;COMPUTE MAXIMUM TRIBUTARY ADDRESS
 1143  002466' 012767  000001  175642  18$:    MOV     #1,TRBADR       ;SET POINT-TO-POINT TRIB ADRS = 1
 1144  002474' 126727  175316  000002          CMPB    SR1,#2          ;SEE IF MULTIPOINT
 1145  002502' 001021                          BNE     19$             ;BR IF NOT MULTIPOINT
 1146  002504' 016767  175312  175624          MOV     SR3,TRBADR      ;INIT MULTIPOINT TRIB ADRS
 1147  002512' 016767  175620  175620          MOV     TRBADR,TRBMAX   ;COMPUTE MAX TRIB NO IN TRBMAX
 1148  002520' 005367  175614                  DEC     TRBMAX
 1149  002524' 066767  175270  175606          ADD     SR2,TRBMAX
 1150  002532' 026727  175602  000400          CMP     TRBMAX,#400     ;ADJUST FOR TRIB ADRS OVERFLOW
 1151  002540' 002402                          BLT     19$
 1152  002542' 005267  175572                  INC     TRBMAX          ;TRIB ADRS = 0 IS ILLEGAL
 1153  002546' 000411                  19$:    BR      REENTR          ;GO BEGIN TESTING ON ALL LOGICAL LINKS
 1154                                  ;-------------------------------------------------------------------------------
 1155                                  ; RESTART ADDRESS FOR A PASS ON ALL DEVICES
 1156                                  ;-------------------------------------------------------------------------------
 1157  002550' 012705  000040          RESTRT: MOV     #32.,R5         ;INIT INTRPT STATUS TABLE ENTRY COUNTER
 1158  002554' 012704  000350'                 MOV     #INTABL,R4      ;INIT TABLE POINTER
 1159  002560' 012724  000001          1$:     MOV     #MDFDON,(R4)+   ;SET MODE DEF'N DONE FLAG IN STATUS BYTE
 1160  002564' 005305                          DEC     R5              ;DECR TABLE ENTRY COUNTER
 1161  002566' 001374                          BNE     1$              ;BR IF MORE TO DO
 1162  002570' 000410                          BR      ENABLE
 1163
 1164                                  ;-------------------------------------------------------------------------------
 1165                                  ; REENTRY ADDRESS FOR NEXT ITERATION ON ALL DEVICES
 1166                                  ;-------------------------------------------------------------------------------
 1167  002572' 012705  000040          REENTR: MOV     #32.,R5         ;INIT INTRPT STATUS TABLE ENTRY COUNTER
 1168  002576' 012704  000350'                 MOV     #INTABL,R4      ;INIT TABLE POINTER
 1169  002602' 012724  000607          1$:     MOV     #MDFDON!HLTDON!ISTDON!OERDON!IERDON,(R4)+ ;INIT FLAGS FOR ITERATION
 1170  002606' 005305                          DEC     R5              ;DECREMENT COUNTER
 1171  002610' 001374                          BNE     1$              ;BR IF MORE TO DO
 1172
 1173                                  ;-------------------------------------------------------------------------------
 1174                                  ;GET A WRITE BUFFER, ENABLE INTERRUPTS, REQUEST INPUTS ON ALL DEVICES
 1175                                  ;-------------------------------------------------------------------------------
 1176  002612' 012767  000001  175516  ENABLE: MOV     #1,TRBADR       ;SET POINT-TO-POINT TRIB ADRS = 1
 1177  002620' 005067  175520                  CLR     OUTDON          ;INIT OUTPUT DONE COUNT TO 0
 1178  002624' 005067  175516                  CLR     WATCNT          ;CLEAR NO. OF MULTIDROP LNKS AWAITING OUTPUT
 1179  002630' 126727  175162  000002          CMPB    SR1,#2          ;SEE IF MULTIPOINT
 1180  002636' 001005                          BNE     1$              ;BR IF NOT MULTIPOINT
```



```
 1181  002640' 016767  175156  175470          MOV     SR3,TRBADR      ;INIT MULTIPOINT TRIB ADRS
 1182  002646' 005367  175464                  DEC     TRBADR          ;GET TRIB ADRS - 1
 1183
 1184  002652'                          1$:
 1185  002652' 104414  000000'                GWBUF$, BEGIN           ;GET WRITE BUFFER INFORMATION
 1186  002656' 016767  175254  175366          MOV     WBUFEA,WBUFEX
 1187  002664' 006367  175362                  ASL     WBUFEX          ;GET TX BUFFER EA BITS INTO BITS 6,7
 1188  002670' 006367  175356                  ASL     WBUFEX
 1189  002674' 032767  000002  175122          BIT     #MODE22,SR4     ;* DMV IN Q22 MODE ??
 1190  002702' 001406                          BEQ     5$              ;* NO: DONE ADJUSTING...
 1191  002704' 006367  175342                  ASL     WBUFEX          ;* YES: ADJUST FOR Q22 MODE
 1192  002710' 006367  175336                  ASL     WBUFEX          ;*
 1193  002714' 000367  175332                  SWAB    WBUFEX          ;*
 1194
 1195  002720'                          5$:
 1196  002720' 104415  000000' 000124'        GETPA$,BEGIN, RBUFVA            ;GET PHYSICAL ADDRESS FROM 16-BIT RBUFVA
 1197  002726' 016767  175176  175320          MOV     RBUFEA,RBUFEX
 1198  002734' 006367  175314                  ASL     RBUFEX          ;GET RCV BUF EA BITS INTO BITS 6,7
 1199  002740' 006367  175310                  ASL     RBUFEX
 1200  002744' 032767  000002  175052          BIT     #MODE22,SR4     ;* DMV IN Q22 MODE ??
 1201  002752' 001406                          BEQ     6$              ;* NO: DONE ADJUSTING...
 1202  002754' 006367  175274                  ASL     RBUFEX          ;* YES: ADJUST FOR Q22 MODE
 1203  002760' 006367  175270                  ASL     RBUFEX          ;*
 1204  002764' 000367  175264                  SWAB    RBUFEX          ;*
 1205
 1206  002770' 016700  175012          6$:     MOV     ADDR,R0         ;INIT DEVICE ADDRESS
 1207  002774' 012767  000001  175266          MOV     #BIT0,DEVPTR    ;INIT SELECTION POINTER
 1208  003002' 042767  000010  175310          BIC     #NONQO,FLAGS    ;CLEAR FLAG TO GET QUEUED OUTPUT INTRPTS AGAIN
 1209  003010' 036767  175254  175250  2$:     BIT     DEVPTR,SELECT   ;SEE IF THIS DEVICE IS SELECTED
 1210  003016' 001403                          BEQ     3$              ;BR IF NOT SELECTED
 1211  003020' 112760  000221  000000          MOVB    #RQI!IEI!IEO,BSEL0(R0) ;ENABLE INTRPTS, SET RQI ON THIS DEVICE
 1212  003026' 005767  174772          3$:     TST     SR4             ; IS THIS A DMV ?
 1213  003032' 001402                          BEQ     4$
 1214  003034' 062700  000010                  ADD     #10,R0          ;IF YES: INCREMENT DEVICE ADDRESS (20)
 1215  003040' 062700  000010          4$:     ADD     #10,R0          ;IF NO: INCREMENT DMP11 ADDRESS (10)
 1216  003044' 006367  175220                  ASL     DEVPTR          ;SHIFT SELECTION POINTER
 1217  003050' 001357                          BNE     2$              ;BR IF NOT ALL ENABLED YET
 1218  003052'                          RTNMON:
 1219  003052' 104400  000000'                EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
 1220
 1221
 1222
```

```
                                        ;*****************************************************************************
                                        ;* INISR - THIS IS THE INPUT INTERRUPT SERVICE ROUTINE.
                                        ;*      WHEN THE INPUT INTERRUPT OCCURS, THE PROGRAM LOADS THE APPROPRIATE
                                        ;*      INPUT COMMAND INTO THE DEVICE CSR'S, AND MONITORS THE PROGRESS OF THE
                                        ;*      MODULE ON THE CURRENT ITERATION.
                                        ;*****************************************************************************
003056'                                 INISR:
003056' 142760  000021  000000          BICB    #IEI!IEO,BSEL0(R0) ;DISABLE INTERRUPTS
003064' 032767  000004  175226          BIT     #NONQI,FLAGS    ;SEE IF NON-QUEUED INPUT OPERATION REQUESTED
003072' 001421                          BEQ     1$              ;BR FOR QUEUED OPERATION
003074' 116760  175177  000003          MOVB    CSEL2+1,BSEL3(R0) ;LOAD BSEL3
003102' 016760  175172  000004          MOV     CSEL4,SEL4(R0)  ;LOAD SEL4
003110' 016760  175166  000006          MOV     CSEL6,SEL6(R0)  ;LOAD SEL6
003116' 152767  000001  175174          BISB    #ININT,FLAGS    ;SET INPUT INTRPT FLAG
003124' 142760  000200  000000          BICB    #RQI,BSEL0(R0)  ;CLEAR RQI
003132' 000167  000640                  JMP     15$
003136' 126727  174654  000002  1$:     CMPB    SR1,#2          ;SEE IF MULTIPOINT
003144' 001406                          BEQ     2$              ;BR IF MULTIPOINT
003146' 010004                          MOV     R0,R4           ;GET DEVICE ADDRESS
003150' 166704  174632                  SUB     ADDR,R4         ;COMPUTE INTRPT STATUS TABLE INDEX
003154' 006204                          ASR     R4
003156' 006204                          ASR     R4
003160' 000431                          BR      5$
003162' 005267  175150          2$:     INC     TRBADR          ;INCR TRIB ADRS
003166' 105767  175144                  TSTB    TRBADR          ;CHK TRIB ADRS
003172' 001002                          BNE     3$              ;IF TRIB ADRS = 0, MAKE IT 1
003174' 005267  175136                  INC     TRBADR          ;(TRIB ADRS = 0 IS ILLEGAL)
003200' 026767  175132  175132  3$:     CMP     TRBADR,TRBMAX   ;SEE IF MAX TRIB ADRS EXCEEDED
003206' 003403                          BLE     4$              ;BR IF NOT
003210' 016767  174606  175120          MOV     SR3,TRBADR      ;RE-INIT TRIB ADRS
003216' 016704  175114          4$:     MOV     TRBADR,R4
003222' 166704  174574                  SUB     SR3,R4          ;COMPUTE INTRPT STATUS TABLE INDEX
003226' 006304                          ASL     R4
003230' 126767  175102  174564          CMPB    TRBADR,SR3      ;SEE IF TRIB ADRS HAS OVERFLOWED
003236' 103002                          BHIS    5$              ;BR IF NOT
003240' 162704  000002                  SUB     #2,R4           ;ADJUST TABLE INDEX TO SKIP TRIB 0
003244' 032764  100000  000350' 5$:     BIT     #GWAIT,INTABL(R4) ;SEE IF LOG LINK AWAITING OUTPUT
003252' 001051                          BNE     7$              ;BR IF YES, TO RETURN
003254' 032764  000020  000350'         BIT     #BITDON,INTABL(R4) ;SEE IF BACCIT JUST COMPLETED
003262' 001045                          BNE     7$              ;BR IF YES, TO RETURN
003264' 032764  000010  000350'         BIT     #BIRDON,INTABL(R4) ;SEE IF BACCIR JUST COMPLETED
003272' 001140                          BNE     11$             ;BR IF YES, TO ISSUE BACCIT
003274' 032764  000004  000350'         BIT     #ISTDON,INTABL(R4) ;SEE IF ISTART JUST COMPLETED
003302' 001077                          BNE     10$             ;BR IF YES, TO ISSUE BACCIR
003304' 032764  000002  000350'         BIT     #HLTDON,INTABL(R4) ;SEE IF HALT JUST COMPLETED
003312' 001053                          BNE     9$              ;BR IF YES, TO ISSUE ISTART
003314' 032764  000400  000350'         BIT     #IERDON,INTABL(R4) ;SEE IF DATA ERRORS INBOUND REQUESTED
003322' 001027                          BNE     8$              ;BR IF YES, TO ISSUE HALT
003324' 032764  000200  000350'         BIT     #OERDON,INTABL(R4) ;SEE IF DATA ERRORS OUTBOUND REQUESTED
003332' 001177                          BNE     13$             ;BR IF YES, TO REQUEST DATA ERRORS INBOUND
003334' 032764  000001  000350'         BIT     #MDFDON,INTABL(R4) ;SEE IF MODE DEFINITION JUST COMPLETED
```



```
003342' 001153                          BNE     12$             ;BR IF YES, TO REQUEST DATA ERRORS OUTBOUND
003344' 004767  002572                  JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
003350' 012767  000011  174530          MOV     #ILGINT,ERRTYP  ;CODE FOR ILLEGAL INTRPT OCCURRED
                                        ;*****************************************************************
003356' 104405  000000' 000450'         HRDERR$,BEGIN,REGTBL    ;UNSOLICITED INPUT INTERRUPT OCCURRED
                                        ;*****************************************************************
003364' 142760  000201  000000  6$:     BICB    #RQI!IEI,BSEL0(R0)      ;UNSOLICITED INPUT INTRPT - CLR RQI,IEI
003372' 000167  177454                  JMP     RTNMON          ;RETURN TO DEC/X11 MONITOR TO AWAIT INTERRUPT
003376' 000167  000402          7$:     JMP CHKMOR              ;GO SCAN FOR INPUT ON NEXT LOGICAL LINK
                                        ;ISSUE HALT STATE CTL IN
003402' 112767  000001  174666  8$:     MOVB    #CTLIN,CSEL2    ;SET UP CONTROL IN COMMAND
003410' 005060  000004                  CLR     SEL4(R0)        ;CLEAR SEL4
003414' 005060  000006                  CLR     SEL6(R0)        ;CLEAR SEL6
003420' 112760  000005  000006          MOVB    #HALTST,BSEL6(R0) ;SET HALT STATE
003426' 052764  000002  000350'         BIS     #HLTDON,INTABL(R4) ;SET HALT DONE IN STATUS WORD
003434' 004767  002616                  JSR     PC,CLRRQI       ;GO SEE IF RQI SHOULD BE CLEARED
003440' 000553                          BR      14$
                                        ;ISSUE ISTART CTL IN
003442' 112767  000001  174626  9$:     MOVB    #CTLIN,CSEL2    ;SET UP CTL IN COMMAND
003450' 005060  000004                  CLR     SEL4(R0)        ;CLEAR SEL4
003454' 005060  000006                  CLR     SEL6(R0)        ;CLEAR SEL6
003460' 112760  000003  000006          MOVB    #ISTART,BSEL6(R0) ;SET ISTART STATE
003466' 052764  000004  000350'         BIS     #ISTDON,INTABL(R4) ;SET ISTART DONE IN STATUS WORD
003474' 004767  002556                  JSR     PC,CLRRQI       ;GO SEE IF RQI SHOULD BE CLEARED
003500' 000533                          BR      14$
                                        ;ISSUE BACCIR
003502' 112767  000000  174566  10$:    MOVB    #BACCIR,CSEL2   ;SET UP BACCIR COMMAND
003510' 016760  174412  000004          MOV     RBUFPA,SEL4(R0) ;LOAD RCV BUF ADRS LO BITS
003516' 012760  002000  000006          MOV     #1024.,SEL6(R0) ;SET RCV CHAR COUNT = 1024.
003524' 156760  174524  000007          BISB    RBUFEX,BSEL7(R0) ;LOAD RCV BUF EA BITS
003532' 032767  000002  174264          BIT     #MODE22,SR4     ;* IS THIS A DMV IN Q22 MODE
003540' 001411                          BEQ     16$             ;* IF YES:
003542' 052767  000010  174526          BIS     #Q22BIT,CSEL2   ;*  SET Q22 MODE BIT
003550' 012760  002000  000010          MOV     #1024.,SEL10(R0);*  SET RX CHAR CNT=1024.
003556' 016760  174472  000006          MOV     RBUFEX,SEL6(R0) ;*  LOAD RCV BUF EA BITS
                                                                ;* IF NO: LEAVE ALONE.
003564' 052764  000010  000350' 16$:    BIS     #BIRDON,INTABL(R4) ;SET BACCIR DONE IN STATUS WORD
003572' 000476                          BR      14$
                                        ;ISSUE BACCIT
003574' 112767  000004  174474  11$:    MOVB    #BACCIT,CSEL2   ;SET UP BACCIT COMMAND
003602' 016760  174326  000004          MOV     XBUFPA,SEL4(R0) ;LOAD TX BUFFER ADDRESS LO BITS
003610' 012760  002000  000006          MOV     #1024.,SEL6(R0) ;SET TX CHAR CNT = 1024.
003616' 156760  174430  000007          BISB    XBUFEX,BSEL7(R0) ;LOAD TX BUF EA BITS
003624' 032767  000002  174172          BIT     #MODE22,SR4     ;* IS THIS A DMV IN Q22 MODE
003632' 001411                          BEQ     17$             ;* IF YES:
003634' 052767  000010  174434          BIS     #Q22BIT,CSEL2   ;*  SET Q22 MODE BIT
003642' 012760  002000  000010          MOV     #1024.,SEL10(R0);*  SET RX CHAR CNT=1024.
003650' 016760  174376  000006          MOV     XBUFEX,SEL6(R0) ;*  LOAD RCV BUF EA BITS
003656'                                 17$:                    ;* IF NO: LEAVE ALONE.
003656' 052764  000020  000350'         BIS     #BITDON,INTABL(R4) ;SET BACCIT DONE IN STATUS WORD
003664' 004767  002366                  JSR     PC,CLRRQI       ;GO SEE IF RQI SHOULD BE CLEARED
003670' 000437                          BR      14$
                                        ;REQUEST DATA ERRORS OUTBOUND
003672' 112767  000001  174376  12$:    MOVB    #CTLIN,CSEL2    ;SET UP CTL IN CMND
003700' 005060  000004                  CLR     SEL4(R0)        ;CLEAR SEL4
003704' 005060  000006                  CLR     SEL6(R0)        ;CLEAR SEL6
```

```
1335  003710' 112760  000112  000006          MOVB    #RDCTSS!DEROTB,BSEL6(R0) ;SET REQUEST TO READ DATA ERRORS OUTBOUND
1336  003716' 052764  000200  000350'         BIS     #DERDON,INTABL(R4) ;SET REQUEST DONE FLAG
1337  003724' 004767  002326                  JSR     PC,CLRRQI       ;GO SEE IF RQI SHOULD BE CLEARED
1338  003730' 000417                          BR      14$
1339                                  ;REQUEST DATA ERRORS INBOUND
1340  003732' 112767  000001  174336  13$:    MOVB    #CTLIN,CSEL2    ;SET UP CTL IN CMND
1341  003740' 005060  000004                  CLR     SEL4(R0)        ;CLEAR SEL4
1342  003744' 005060  000006                  CLR     SEL6(R0)        ;CLEAR SEL6
1343  003750' 112760  000113  000006          MOVB    #RDCTSS!DERINB,BSEL6(R0) ;SET REQUEST TO READ DATA ERRORS INBOUND
1344  003756' 052764  000400  000350'         BIS     #IERDON,INTABL(R4) ;SET REQUEST DONE FLAG
1345  003764' 004767  002266                  JSR     PC,CLRRQI       ;GO SEE IF RQI SHOULD BE CLEARED
1346                                  ;LOAD TRIB ADRS AND COMMAND INTO CSR'S
1347  003770' 116760  174342  000003  14$:    MOVB    TRBADR,BSEL3(R0) ;LOAD TRIB ADRS
1348  003776' 116760  174274  000002  15$:    MOVB    CSEL2,BSEL2(R0) ;LOAD COMMAND, CLEAR RDYI
1349
1350  004004' 132760  000020  000002  CHKMOR: BITB    #RDYI,BSEL2(R0) ;SEE IF RDYI SET AGAIN
1351  004012' 001402                          BEQ     1$              ;BR IF RDYI NOT SET AGAIN
1352  004014' 000167  177036                  JMP     INISR           ;GO HANDLE NEXT INPUT
1353  004020' 132760  000200  000002  1$:     BITB    #RDYO,BSEL2(R0) ;SEE IF AN OUTPUT IS PENDING
1354  004026' 001402                          BEQ     RTNINT          ;BR IF NOT
1355  004030' 000167  000012                  JMP     OUTISR          ;GO HANDLE OUTPUT COMMAND
1356  004034' 152760  000021  000000  RTNINT: BISB    #IEI!IEO,BSEL0(R0) ;SET IEI AND IEO AGAIN
1357  004042' 000167  177004                  JMP     RTNMON          ;RETURN TO DEC/X11 TO AWAIT INTERRUPT
1358
1359
1360
1361
1362
1363                                  ;*****************************************************************************
1364                                  ;* OUTISR - THIS IS THE OUTPUT INTERRUPT SERVICE ROUTINE. WHEN THE OUTPUT
1365                                  ;*       INTERRUPT OCCURS, THIS ROUTINE READS THE DEVICE CSR'S,
1366                                  ;*       CHECKS THE OUTPUT, AND MONITORS THE PROGRESS OF THE
1367                                  ;*       MODULE ON THE CURRENT ITERATION. IF SUFFICIENT ITERATIONS HAVE BEEN
1368                                  ;*       PERFORMED, AN END OF PASS IS INDICATED, AND A NEW PASS IS BEGUN.
1369                                  ;*****************************************************************************
1370  004046'                         OUTISR:
1371  004046' 142760  000021  000000          BICB    #IEI!IEO,BSEL0(R0) ;CLEAR IEI, IEO
1372  004054' 032767  000010  174236          BIT     #NONQO,FLAGS    ;SEE IF NON-QUEUED OUTPUT INTRPT MODE
1373  004062' 001427                          BEQ     1$              ;BR IF NOT
1374  004064' 016067  000000  174214          MOV     SEL0(R0),RSEL0  ;READ AND STORE CSR'S
1375  004072' 016067  000002  174210          MOV     SEL2(R0),RSEL2
1376  004100' 016067  000004  174204          MOV     SEL4(R0),RSEL4
1377  004106' 016067  000006  174200          MOV     SEL6(R0),RSEL6
1378  004114' 005767  173704                  TST     SR4             ; IS THIS A DMV?
1379  004120' 001403                          BEQ     26$             ; IF NO: SKIP READ (DMP HAS NO SEL10!)
1380  004122' 016067  000010  174166          MOV     SEL10(R0),RSEL10; IF YES: READ SEL10
1381  004130' 052767  000002  174162  26$:    BIS     #OUTINT,FLAGS   ;SET OUTPUT INTERRUPT FLAG
1382  004136' 000167  177672                  JMP     RTNINT          ;GO RETURN TO DEC/X11 MONITOR
1383  004142' 126727  173650  000002  1$:     CMPB    SR1,#2          ;SEE IF MULTIPOINT
1384  004150' 001406                          BEQ     2$              ;BR IF MULTIPOINT
1385  004152' 010004                          MOV     R0,R4           ;GET DEVICE ADDRESS
1386  004154' 166704  173626                  SUB     ADDR,R4         ;COMPUTE INTRPT STATUS TABLE INDEX
1387  004160' 006204                          ASR     R4
1388  004162' 006204                          ASR     R4
1389  004164' 000415                          BR      4$              ;PROCEED
1390  004166' 116004  000003          2$:     MOVB    BSEL3(R0),R4    ;GET TRIB ADRS WHICH WAS RETURNED
```



```
1391  004172' 042704  177400                  BIC     #177400,R4      ;CLEAR UNUSED BITS
1392  004176' 120467  173620                  CMPB    R4,SR3          ;COMPUTE INTRPT STATUS TABLE INDEX
1393  004202' 103003                          BHIS    3$
1394  004204' 062704  000400                  ADD     #400,R4         ;ADJUST FOR TRIB ADRS OVERFLOW
1395  004210' 005304                          DEC     R4
1396  004212' 166704  173604          3$:     SUB     SR3,R4
1397  004216' 006304                          ASL     R4
1398  004220' 116001  000002          4$:     MOVB    BSEL2(R0),R1    ;GET BSEL2 CONTENTS
1399  004224' 142701  000370                  BICB    #CMDMSK,R1      ;MASK OFF ALL BUT CMND BITS
1400  004230' 120127  000004                  CMPB    R1,#BACCOT      ;SEE IF BACCOT CMND JUST COMPLETED
1401  004234' 001432                          BEQ     7$              ;BR IF YES
1402  004236' 120127  000000                  CMPB    R1,#BACCOR      ;SEE IF BACCOR CMND JUST COMPLETED
1403  004242' 001523                          BEQ     11$             ;BR IF YES
1404  004244' 120127  000001                  CMPB    R1,#CTLOUT      ;SEE IF CONTROL OUT CMND JUST COMPLETED
1405  004250' 001002                          BNE     5$              ;BR IF NOT
1406  004252' 000167  000522                  JMP     18$             ;GO HANDLE CONTROL OUT COMMAND
1407  004256' 120127  000002          5$:     CMPB    R1,#INFOUT      ;SEE IF INFORMATION OUT CMND JUST COMPLETED
1408  004262' 001002                          BNE     6$              ;BR IF NOT
1409  004264' 000167  000550                  JMP     20$             ;GO HANDLE INFO OUT COMMAND
1410  004270' 004767  001646          6$:     JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
1411  004274' 012767  000011  173604          MOV     #ILGINT,ERRTYP  ;SET CODE FOR ILLEGAL INTRPT OCCURRED
1412                                          ;*****************************************************************
1413  004302' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;UNSOLICITED OUTPUT INTRPT OCCURRED
1414                                          ;*****************************************************************
1415  004310' 142760  000020  000000          BICB    #IEO,BSEL0(R0)  ;UNSOLICITED OUTPUT INTRPT - CLEAR IEO
1416  004316' 000167  176530                  JMP     RTNMON          ;RETURN TO DEC/X11 MONITOR TO AWAIT INTRPT
1417                                  ;HANDLE BACCOT
1418  004322' 026067  000004  173604  7$:     CMP     SEL4(R0),#BUFPA ;CHK FOR CORRECT TX BUFFER LO BITS RETURNED
1419  004330' 001411                          BEQ     8$              ;BR IF CORRECT
1420  004332' 004767  001604                  JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
1421  004336' 012767  000037  173542          MOV     #BADWRT,ERRTYP  ;CODE FOR UNABLE TO EXEC WRITE FUNCTION
1422                                          ;*****************************************************************
1423  004344' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;INCORRECT TX BA BITS RETURNED
1424                                          ;*****************************************************************
1425  004352' 000453                          BR      10$
1426  004354' 116001  000007          8$:     MOVB    BSEL7(R0),R1    ;GET BSEL7 CONTENTS
1427  004360' 142701  000077                  BICB    #EAMSK,R1       ;MASK FOR BUS ADRS EA BITS
1428  004364' 032767  000002  173432          BIT     #MODE22,SR4     ;* IF THIS IS A DMV IN Q22 MODE...
1429  004372' 001402                          BEQ     30$             ;* THEN GET EXTENDED ADDRESS (EA)
1430  004374' 116001  000006                  MOVB    BSEL6(R0),R1    ;* BITS FROM A DIFFERENT CSR.
1431  004400' 120167  173646          30$:    CMPB    R1,WBUFEX       ;CHK FOR CORRECT EA BITS RETURNED
1432  004404' 001411                          BEQ     9$              ;BR IF CORRECT
1433  004406' 004767  001530                  JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
1434  004412' 012767  000037  173466          MOV     #BADWRT,ERRTYP  ;CODE FOR UNABLE TO EXEC WRITE FUNCTION
1435                                          ;*****************************************************************
1436  004420' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;INCORRECT TX EA BITS RETURNED
1437                                          ;*****************************************************************
1438  004426' 000425                          BR      10$
1439  004430' 016001  000006          9$:     MOV     SEL6(R0),R1     ;GET SEL6 CONTENTS
1440  004434' 032767  000002  173362          BIT     #MODE22,SR4     ;* IF THIS IS A DMV IN Q22 MODE...
1441  004442' 001402                          BEQ     27$             ;* THEN GET CHARACTER COUNT
1442  004444' 016001  000010                  MOV     SEL10(R0),R1    ;* BITS FROM A DIFFERENT CSR.
1443  004450' 042701  140000          27$:    BIC     #CCMSK,R1       ;MASK FOR CHAR COUNT BITS
1444  004454' 020127  002000                  CMP     R1,#1024.       ;CHK FOR CORRECT CHAR CNT RETURNED
1445  004460' 001410                          BEQ     10$             ;BR IF CORRECT
1446  004462' 004767  001454                  JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
```

```
 1447  004466' 012767  000037  173412          MOV     #BADWRT,ERRTYP  ;CODE FOR UNABLE TO EXEC WRITE FUNCTION
 1448                                          ;****************************************************************
 1449  004474' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;INCORRECT TX CHAR CNT BITS RETURNED
 1450                                          ;****************************************************************
 1451  004502' 052764  000100  000350' 10$:    BIS     #BOTDON,INTABL(R4) ;SET BACCOT DONE IN STATUS BYTE
 1452  004510' 000505                         BR      17$
 1453                                  ;HANDLE BACCOR
 1454  004512' 026067  000004  173406 11$:     CMP     SEL4(R0),RBUFPA ;CHK FOR CORRECT RCV BUFFER LO BITS RETURNED
 1455  004520' 001411                         BEQ     12$             ;BR IF CORRECT
 1456  004522' 004767  001414                 JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
 1457  004526' 012767  000036  173352         MOV     #BADRED,ERRTYP  ;CODE FOR UNABLE TO EXEC READ FUNCTION
 1458                                          ;****************************************************************
 1459  004534' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;INCORRECT RCV BA BITS RETURNED
 1460                                          ;****************************************************************
 1461  004542' 000463                         BR      16$
 1462  004544' 116001  000007          12$:    MOVB    BSEL7(R0),R1    ;GET BSEL7 CONTENTS
 1463  004550' 142701  000077                 BICB    #EAMSK,R1       ;MASK FOR BUS ADRS EA BITS
 1464  004554' 032767  000002  173242         BIT     #MODE22,SR4     ;* IF THIS IS A DMV IN Q22 MODE...
 1465  004562' 001402                         BEQ     28$             ;* THEN GET EXTENDED ADDRESS (EA)
 1466  004564' 116001  000006                 MOVB    BSEL6(R0),R1    ;* BITS FROM A DIFFERENT CSR.
 1467  004570' 120167  173460          28$:    CMPB    R1,RBUFEX       ;CHK FOR CORRECT EA BITS RETURNED
 1468  004574' 001411                         BEQ     13$             ;BR IF CORRECT
 1469  004576' 004767  001340                 JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
 1470  004602' 012767  000036  173276         MOV     #BADRED,ERRTYP  ;CODE FOR UNABLE TO EXEC READ FUNCTION
 1471                                          ;****************************************************************
 1472  004610' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;INCORRECT RCV EA BITS RETURNED
 1473                                          ;****************************************************************
 1474  004616' 000435                         BR      16$
 1475  004620' 016001  000006          13$:    MOV     SEL6(R0),R1     ;GET SEL6 CONTENTS
 1476  004624' 032767  000002  173172         BIT     #MODE22,SR4     ;* IF THIS IS A DMV IN Q22 MODE...
 1477  004632' 001402                         BEQ     29$             ;* THEN GET CHARACTER COUNT
 1478  004634' 016001  000010                 MOV     SEL10(R0),R1    ;* BITS FROM A DIFFERENT CSR.
 1479  004640' 042701  140000          29$:    BIC     #CCMSK,R1       ;MASK FOR CHAR COUNT BITS
 1480  004644' 020127  002000                 CMP     R1,#1024.       ;CHK FOR CORRECT CHAR CNT RETURNED
 1481  004650' 001411                         BEQ     14$             ;BR IF CORRECT
 1482  004652' 004767  001264                 JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
 1483  004656' 012767  000036  173222         MOV     #BADRED,ERRTYP  ;CODE FOR UNABLE TO EXEC READ FUNCTION
 1484                                          ;****************************************************************
 1485  004664' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;INCORRECT RCV CHAR CNT BITS RETURNED
 1486                                          ;****************************************************************
 1487  004672' 000407                         BR      16$
 1488  004674' 010067  173200          14$:    MOV     R0,CSRA         ;STORE CSR ADDRESS FOR POSSIBLE ERROR REPORT
 1489  004700' 104412  000000' 000126'         CDATA$,BEGIN,RBUFPA     ; REQUEST FOR MONITOR TO CHECK DATA
 1490  004706' 004712'                         15$                     ; IF ERROR, RETURN AT TAG 15$
 1491  004710' 000400                         BR      16$
 1492  004712'                          15$:
 1493  004712' 016700  173162          16$:    MOV     CSRA,R0         ;RESTORE CSRA (JUST IN CASE.)
 1494  004716' 052764  000040  000350'         BIS     #BORDON,INTABL(R4) ;SET BACCOR DONE IN STATUS WORD
 1495  004724' 032764  000100  000350' 17$:    BIT     #BOTDON,INTABL(R4) ;SEE IF BACCOT DONE
 1496  004732' 001522                         BEQ     25$             ;BR IF NOT
 1497  004734' 032764  000040  000350'         BIT     #BORDON,INTABL(R4) ;SEE IF BACCOR DONE
 1498  004742' 001516                         BEQ     25$             ;BR IF NOT
 1499  004744' 005267  173374                 INC     OUTDON          ;INCR CNT OF LOG LNKS DONE WITH MSG
 1500  004750' 026767  173370  173352         CMP     OUTDON,LLKCNT   ;SEE IF ALL LOG LNKS DONE YET
 1501  004756' 002510                         BLT     25$             ;BR IF ALL NOT DONE
 1502  004760' 105060  000002                 CLRB    BSEL2(R0)       ;CLEAR BSEL2 (INCLUDING RDYO)
```



```
 1503  004764' 004767  001110                 JSR     PC,DISABL       ;DISABLE INTRPTS ON ALL DEVICES
 1504  004770' 104413  000000'                ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
 1505                                                                  ;MONITOR SHALL TEST END OF PASS
 1506  004774' 000167  175572                 JMP     REENTR          ;GO START NEW ITERATION
 1507                                  ;HANDLE CTL OUT
 1508  005000' 126027  000006  000024 18$:     CMPB    BSEL6(R0),#RUNST ;SEE IF PROTOCOL JUST ENTERED THE RUN STATE
 1509  005006' 001003                         BNE     19$             ;BR IF NOT
 1510  005010' 004767  001304                 JSR     PC,SETRQI       ;GO SET RQI
 1511  005014' 000471                         BR      25$             ;PROCEED
 1512  005016' 004767  001120          19$:    JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
 1513  005022' 012767  000000  173056         MOV     #NOTDFN,ERRTYP  ;CODE FOR ERROR NOT DEFINED (MUST INTERPRET
 1514                                                                  ;   CSR CONTENTS)
 1515                                          ;****************************************************************
 1516  005030' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;UNEXPECTED CTL OUT CMND
 1517                                          ;****************************************************************
 1518  005036' 000460                         BR      25$
 1519                                  ;HANDLE INFO OUT
 1520  005040' 116001  000006          20$:    MOVB    BSEL6(R0),R1    ;GET BSEL6 CODE RETURNED
 1521  005044' 132701  000100                 BITB    #RDCTSS,R1      ;SEE IF RESPONSE TO READ ERROR REQUEST
 1522  005050' 001433                         BEQ     23$             ;BR IF NOT
 1523  005052' 142701  000340                 BICB    #RTNMSK,R1      ;MASK FOR THE TSS NUMBER
 1524  005056' 120127  000012                 CMPB    R1,#EROTB       ;SEE IF DATA ERRORS OUTBOUND BEING RETURNED
 1525  005062' 001403                         BEQ     21$             ;BR IF YES
 1526  005064' 120127  000013                 CMPB    R1,#ERINB       ;SEE IF DATA ERRORS INBOUND BEING RETURNED
 1527  005070' 001033                         BNE     24$             ;BR IF NOT
 1528  005072' 116001  000004          21$:    MOVB    BSEL4(R0),R1    ;GET BSEL4
 1529  005076' 105701                         TSTB    R1              ;SEE IF ANY DATA ERRORS RETURNED
 1530  005100' 001414                         BEQ     22$             ;BR IF NOT
 1531  005102' 004767  001034                 JSR     PC,GETERR       ;LOAD ERROR INFO FOR PRINTOUT
 1532  005106' 012767  000001  172772         MOV     #DATERR,ERRTYP  ;CODE FOR DATA ERROR
 1533  005114' 010046                         MOV     R0,-(SP)        ;SAVE R0
 1534  005116' 010446                         MOV     R4,-(SP)        ;SAVE R4
 1535                                          ;****************************************************************
 1536  005120' 104406  000000' 000450'         SOFER$,BEGIN,REGTBL     ;DATA ERROR
 1537                                          ;****************************************************************
 1538  005126' 012604                         MOV     (SP)+,R4        ;RESTORE R4
 1539  005130' 012600                         MOV     (SP)+,R0        ;RESTORE R0
 1540  005132' 004767  001162          22$:    JSR     PC,SETRQI       ;ENABLE INTERRUPTS ON THIS DEVICE
 1541  005136' 000420                         BR      25$             ;PROCEED
 1542  005140' 142701  000340          23$:    BICB    #RTNMSK,R1      ;MASK FOR RETURN CODE
 1543  005144' 120127  000020                 CMPB    R1,#BUFRTC      ;CHECK FOR BUFFER RETURN COMPLETE
 1544  005150' 001003                         BNE     24$             ;BR IF NOT
 1545  005152' 004767  001142                 JSR     PC,SETRQI       ;ENABLE INPUTS ON THIS DEVICE
 1546  005156' 000410                         BR      25$             ;PROCEED
 1547  005160' 004767  000756          24$:    JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
 1548  005164' 012767  000000  172714         MOV     #NOTDFN,ERRTYP  ;CODE FOR ERROR NOT DEFINED (MUST INTERPRET
 1549                                                                  ;   CSR CONTENTS)
 1550                                          ;****************************************************************
 1551  005172' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;UNEXPECTED INFO OUT CMND
 1552                                          ;****************************************************************
 1553  005200' 105060  000002          25$:    CLRB    BSEL2(R0)       ;CLEAR BSEL2 (INCLUDING RDYO)
 1554  005204' 000167  176574                 JMP     CHKMOR          ;GO SEE IF RDYI OR RDYO STILL SET,
 1555                                                                  ;   AND RETURN TO DEC/X11 MONITOR IF NOT
 1556
 1557
 1558                                  ;*******************************************************************************
```

```
                                        ;* DMVLUP - THIS SUBROUTINE ENTERS M-LOOP, SETS TTL LOOPBACK AND TIMER 1 FOR
                                        ;*      56K BPS, AND EXITS MLOOP. (DMV ONLY)
                                        ;*****************************************************************************
005210' 112760  000301  000001  DMVLUP: MOVB    #301,BSEL1(R0)  ;ENTER MAINTENANCE LOOP
005216'                         1$:
005216' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
005222' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005226' 132760  000200  000002          BITB    #MRDY,BSEL2(R0) ;MAKE SURE MLOOP COMPLETE
005234' 001770                          BEQ     1$
005236' 012760  000006  000002          MOV     #TTLOOP,BSEL2(R0) ;ISSUE TTLOOP CONFIGURE COMMAND
005244' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
005250' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005254' 000240                          NOP
005256' 000240                          NOP
005260' 000207                          RTS     PC              ;RETURN


                                        ;*****************************************************************************
                                        ;* DMVRD - THIS SUBROUTINE FORCES THE DMV WHOSE ADRS IS PASSED IN R0 ON ENTRY
                                        ;*      TO ENTER THE M-LOOP AND READ THE INTERNAL DMV LOCATION PASSED IN THE
                                        ;*      WORD FOLLOWING THE CALL.
                                        ;*****************************************************************************
005262' 112760  000301  000001  DMVRD:  MOVB    #301,BSEL1(R0)  ;ENTER MAINTENANCE LOOP
005270'                         1$:
005270' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
005274' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005300' 132760  000200  000002          BITB    #MRDY,BSEL2(R0) ;MAKE SURE MLOOP COMPLETE
005306' 001770                          BEQ     1$
005310' 017660  000000  000004          MOV     @(SP),SEL4(R0)  ;PASS DMV INTERNAL ADDRESS
005316' 012760  000001  000002          MOV     #REDLOC,BSEL2(R0) ;ISSUE READ COMMAND
005324'                         2$:
005324' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
005330' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005334' 132760  000200  000002          BITB    #MRDY,BSEL2(R0) ;MAKE SURE MLOOP READ IS COMPLETE
005342' 001770                          BEQ     2$
005344' 062716  000002                  ADD     #2,(SP)         ;FIX UP RETURN PC
005350' 000207                          RTS     PC              ;RETURN


                                        ;*****************************************************************************
                                        ;* EXECUT - THIS SUBROUTINE FORCES THE MICROPROCESSOR WHOSE ADRS IS PASSED
                                        ;*      IN R0 ON ENTRY TO EXECUTE AN INSTRUCTION WHICH IS PASSED IN THE WORD
                                        ;*      FOLLOWING THE CALL (DMP ONLY).
                                        ;*****************************************************************************
005352' 152760  000006  000001  EXECUT: BISB    #ROMO!ROM1,BSEL1(R0)    ;SET ROMO, ROM1 BITS IN BSEL1
005360' 017660  000000  000006          MOV     @(SP),SEL6(R0)  ;PUT INSTRUCTION INTO SEL6
005366' 152760  000007  000001          BISB    #ROMO!ROM1!STEPMP,BSEL1(R0) ;SET ROMO, ROM1, STEPMP IN BSEL1
005374' 142760  000007  000001          BICB    #ROMO!ROM1!STEPMP,BSEL1(R0) ;CLEAR ROMO, ROM1, STEPMP IN BSEL1
005402' 062716  000002                  ADD     #2,(SP)         ;FIX UP RETURN PC
005406' 000207                          RTS     PC              ;RETURN


                                        ;*****************************************************************************
                                        ;* INIDEV - THIS SUBROUTINE ISSUES A MASTER CLEAR AND CLEARS THE CSR'S,
                                        ;*      FOR THE DEVICE WHOSE ADDRESS IS PASSED IN R0 ON ENTRY. (DMP AND DMV).
```



```
                                        ;*****************************************************************************
005410' 010146                  INIDEV: MOV     R1,-(SP)        ;SAVE R1
005412' 004767  000236                  JSR     PC,CLRCSR       ;CLEAR BSEL0, SEL2,4,6(,10)
005416' 112760  000100  000001          MOVB    #MCLR,BSEL1(R0) ;SET MASTER CLEAR BIT
005424' 005767  172374                  TST     SR4             ; IF DMV ...
005430' 001003                          BNE     6$              ; THEN DON'T SET RUN.
                                        ;-----------------------------------------------------------------------------
                                        ;;;     MOVB    #RUN,BSEL1(R0)  ;PATCH TO SET RUN BIT IF USING M8206
005432' 000240                          NOP                     ; 112760
005434' 000240                          NOP                     ;   200
005436' 000240                          NOP                     ;     1
                                        ;-----------------------------------------------------------------------------
005440' 042767  000004  172654  6$:     BIC     #BADINI,ERRORS  ;CLEAR BAD INIT ERROR FLAG
005446' 012701  020000                  MOV     #20000,R1       ;INITIALIZE INIT TIMER
005452' 132760  000200  000001  1$:     BITB    #RUN,BSEL1(R0)  ;SEE IF RUN IS SET AGAIN
005460' 001020                          BNE     3$              ;BR IF RUN IS SET
005462' 005301                          DEC     R1              ;DECR TIMER
005464' 001405                          BEQ     2$              ;BR IF INIT TIMED-OUT
005466' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
005472' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
005476' 000765                          BR      1$              ;GO CHECK AGAIN
005500' 004767  000436          2$:     JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
005504' 012767  000034  172374          MOV     #NOINIT,ERRTYP  ;CODE FOR DEVICE WILL NOT INIT
                                        ;*************************************************************
005512' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;TIMED-OUT WAITING FOR RUN TO SET
                                        ;*************************************************************
005520' 000414                          BR      4$
005522' 126027  000006  000305  3$:     CMPB    BSEL6(R0),#GOODIN ;SEE IF INIT COMPLETE CODE SET BY MICRO-CPU
005530' 001413                          BEQ     5$              ;BR IF YES
005532' 004767  000404                  JSR     PC,GETERR       ;LOAD ERROR INFORMATION FOR PRINTOUT
005536' 012767  000034  172342          MOV     #NOINIT,ERRTYP  ;CODE FOR DEVICE WILL NOT INIT
                                        ;*************************************************************
005544' 104405  000000' 000450'         HRDER$,BEGIN,REGTBL     ;INIT COMPLETE CODE NOT SET BY MICRO-CPU
                                        ;*************************************************************
005552' 052767  000004  172542  4$:     BIS     #BADINI,ERRORS  ;SET BAD INIT ERROR FLAG
005560' 004767  000070          5$:     JSR     PC,CLRCSR       ;CLEAR BSEL0, SEL2,4,6
005564' 012601                          MOV     (SP)+,R1        ;RESTORE R1
005566' 000207                          RTS     PC              ;RETURN


                                        ;*****************************************************************************
                                        ;* READ16 - THIS SUBROUTINE FORCES THE MICROPROCESSOR WHOSE ADDRESS IS
                                        ;*      PASSED IN R0 ON ENTRY, TO EXECUTE AN INSTRUCTION WHICH READS DMP LINE
                                        ;*      UNIT IBUS REG 16 INTO BSEL4, AND THEN IT PLACES THE CONTENTS INTO LOC.
                                        ;*      LURG16 (IF DMV: EXECUTE MLOOP, "SWPBOT" => LURG16, AND ADJUST).
                                        ;*****************************************************************************
005570' 005767  172230          READ16: TST     SR4             ;IS THIS A DMV ?
005574' 001007                          BNE     1$              ;BRANCH IF YES

005576' 004767  177550                  JSR     PC,EXECUT       ;DMP: EXECUTE MOVE INSTRUCTION
005602' 021344                          .WORD   RDLU16
005604' 116067  000004  172460          MOVB    BSEL4(R0),LURG16 ;GET REG 16 CONTENTS INTO LURG16
005612' 000415                          BR      2$

005614' 004767  177442          1$:     JSR     PC,DMVRD        ;DMV: READ "SWPBOT" SWITCHES
```

```
 1671  005620' 121000                          .WORD   SWPBOT
 1672  005622' 116067  000006  172442          MOVB    BSEL6(R0),LURG16 ;PUT IT IN LURG16
 1673  005630' 000241                          CLC                      ;ADJUST "SWPBOT" BITS
 1674  005632' 106267  172434                  ASRB    LURG16           ; TO LOOK LIKE THE DMP
 1675  005636' 103003                          BCC     2$               ;  REG 16 SWITCHES
 1676  005640' 152767  000200  172424          BISB    #ENABSW,LURG16
 1677
 1678  005646' 105067  172421          2$:     CLRB    LURG16+1         ;CLEAR HI BYTE
 1679  005652' 000207                          RTS     PC               ;RETURN
 1680
 1681
 1682
 1683
 1684                                  ;*************************************************************************
 1685                                  ;* CLRCSR - THIS SUBROUTINE CLEARS BSEL0, SEL2,4,6, FOR THE DEVICE WHOSE ADRS
 1686                                  ;*       IS PASSED IN R0 ON ENTRY (SEL10 TOO IF DMV11).
 1687                                  ;*************************************************************************
 1688  005654' 105060  000000          CLRCSR: CLRB    BSEL0(R0)        ;CLEAR BSEL0
 1689  005660' 005060  000002                  CLR     SEL2(R0)         ;CLEAR SEL2
 1690  005664' 005060  000004                  CLR     SEL4(R0)         ;CLEAR SEL4
 1691  005670' 005060  000006                  CLR     SEL6(R0)         ;CLEAR SEL6
 1692  005674' 005767  172124                  TST     SR4
 1693  005700' 001402                          BEQ     1$
 1694  005702' 005060  000010                  CLR     SEL10(R0)        ;IF DMV: CLEAR SEL10
 1695  005706' 000207                  1$:     RTS     PC               ;RETURN
 1696
 1697
 1698
 1699                                  ;*************************************************************************
 1700                                  ;* INCMND - THIS SUBROUTINE ISSUES AN INPUT COMMAND TO THE
 1701                                  ;*       DEVICE WHOSE ADDRESS IS PASSED IN R0, BY ENABLING INTERRUPTS
 1702                                  ;*       AND SETTING RQI. THEN, IT SETS NONQI IN FLAGS, FOR NON-QUEUED
 1703                                  ;*       (REQUEST-AND-WAIT) INPUT INTERRUPT OPERATION.
 1704                                  ;*       IF THE PROGRAM TIMES-OUT WAITING FOR COMMAND COMPLETION,
 1705                                  ;*       THE ERROR IS REPORTED, AND A RETURN IS MADE WITH THE INPUT INTERRUPT
 1706                                  ;*       TIME-OUT FLAG IN "ERRORS" SET TO 1.
 1707                                  ;*************************************************************************
 1708  005710' 042767  000001  172404  INCMND: BIC     #INTIMO,ERRORS   ;CLR INPUT INTRPT TIME-OUT FLAG
 1709  005716' 052767  000004  172374          BIS     #NONQI,FLAGS     ;SET BIT FOR NON-QUEUED INPUT INTRPT OPERATION
 1710  005724' 042767  000001  172366          BIC     #ININT,FLAGS     ;CLR INPUT INTRPT FLAG
 1711  005732' 005067  172350                  CLR     RSEL0            ;CLEAR OUTPUT CMND RETURN AREA
 1712  005736' 005067  172346                  CLR     RSEL2
 1713  005742' 005067  172344                  CLR     RSEL4
 1714  005746' 005067  172342                  CLR     RSEL6
 1715  005752' 005067  172340                  CLR     RSEL10
 1716  005756' 012767  030000  172340          MOV     #30000,INTIMR    ;INIT INPUT INTRPT TIMER
 1717  005764' 112760  000221  000000          MOVB    #RQI!IEI!IEO,BSEL0(R0) ;REQUEST INPUT, ENABLE INTERRUPTS
 1718  005772'                          1$:
 1719  005772' 104407  000000'                BREAK$,BEGIN             ;TEMPORARY RETURN TO MONITOR....
 1720  005776' 104407  000000'                BREAK$,BEGIN             ;THEN CONTINUE AT NEXT INSTRUCTION.
 1721  006002' 132767  000001  172310          BITB    #ININT,FLAGS     ;SEE IF INPUT INTRPT SERVICED YET
 1722  006010' 001016                          BNE     3$               ;BR IF YES
 1723  006012' 005367  172306                  DEC     INTIMR           ;DECREMENT TIMER
 1724  006016' 001365                          BNE     1$               ;BR IF INPUT INTERRUPT DIDN'T TIME-OUT YET
 1725  006020' 004767  000116          2$:     JSR     PC,GETERR        ;LOAD ERROR INFORMATION FOR PRINTOUT
 1726  006024' 012767  000023  172054          MOV     #NOINTR,ERRTYP   ;CODE FOR DEV FAILED TO INTRPT
```



```
 1727                                          ;*****************************************************************
 1728  006032' 104405  000000' 000450'        HRDER$,BEGIN,REGTBL      ;INPUT INTERRUPT TIMED-OUT
 1729                                          ;*****************************************************************
 1730  006040' 052767  000001  172254          BIS     #INTIMO,ERRORS   ;SET INPUT INTERRUPT TIME-OUT FLAG
 1731  006046' 042767  000004  172244  3$:     BIC     #NONQI,FLAGS     ;CLEAR NON-QUEUED BIT
 1732  006054' 000207                          RTS     PC       ;RETURN
 1733
 1734
 1735
 1736                                  ;*************************************************************************
 1737                                  ;* CLRCMD - THIS SUBROUTINE CLEARS THE COMMAND AREA : CSEL2,CSEL4,CSEL6
 1738                                  ;*************************************************************************
 1739  006056' 005067  172214          CLRCMD: CLR     CSEL2
 1740  006062' 005067  172212                  CLR     CSEL4
 1741  006066' 005067  172210                  CLR     CSEL6
 1742  006072' 005067  172206                  CLR     CSEL10
 1743  006076' 000207                          RTS     PC               ;RETURN
 1744
 1745
 1746
 1747                                  ;*************************************************************************
 1748                                  ;* DISABL - DISABLE INTERRUPTS ON ALL DEVICES
 1749                                  ;*************************************************************************
 1750  006100' 016700  171702          DISABL: MOV     ADDR,R0          ;INIT DEVICE ADDRESS
 1751  006104' 012767  000001  172156          MOV     #BIT0,DEVPTR     ;INIT SELECTION POINTER
 1752  006112' 036767  172152  172146  1$:     BIT     DEVPTR,SELECT    ;SEE IF THIS DEVICE IS SELECTED
 1753  006120' 001402                          BEQ     2$               ;BR IF NOT SELECTED
 1754  006122' 105060  000000                  CLRB    BSEL0(R0)        ;CLEAR RQI, IEI, IEO
 1755  006126' 062700  000010          2$:     ADD     #10,R0           ;INCR DEVICE ADDRESS
 1756  006132' 006367  172132                  ASL     DEVPTR           ;SHIFT SELECTION POINTER
 1757  006136' 001365                          BNE     1$               ;BR IF MORE TO SCAN
 1758  006140' 000207                          RTS     PC               ;RETURN
 1759
 1760
 1761
 1762                                  ;*************************************************************************
 1763                                  ;* GETERR - THIS SUBROUTINE LOADS THE CSR ADRS, SEL0 AND SEL2 CONTENTS INTO
 1764                                  ;*       CSRA, ACSR, AND ASTAT FOR ERROR PRINTOUT. THE DEVICE ADRS IS ASSUMED
 1765                                  ;*       TO BE IN R0 ON ENTRY. THEN, IT LOADS THE CSR ADDRESSES INTO REGTBL.
 1766                                  ;*************************************************************************
 1767  006142' 010067  171732          GETERR: MOV     R0,CSRA          ;LOAD CSR ADDRESS
 1768  006146' 016067  000000  171726          MOV     SEL0(R0),ACSR    ;LOAD SEL0 CONTENTS
 1769  006154' 016067  000002  171722          MOV     SEL2(R0),ASTAT   ;LOAD SEL2 CONTENTS
 1770  006162' 010067  172262                  MOV     R0,REGTBL        ;GET CSR ADDRESSES FOR DEVICE INTO REGTBL
 1771  006166' 010067  172260                  MOV     R0,REGTBL+2
 1772  006172' 062767  000002  172252          ADD     #2,REGTBL+2
 1773  006200' 010067  172250                  MOV     R0,REGTBL+4
 1774  006204' 062767  000004  172242          ADD     #4,REGTBL+4
 1775  006212' 010067  172240                  MOV     R0,REGTBL+6
 1776  006216' 062767  000006  172232          ADD     #6,REGTBL+6
 1777  006224' 005767  171574                  TST     SR4              ; IS THIS A DMV?
 1778  006230' 001004                          BNE     1$
 1779  006232' 012767  177777  172220          MOV     #177777,REGTBL+10        ; NO: DISABLE SEL10
 1780  006240' 000405                          BR      2$                       ;     AND GOTO RETURN
 1781  006242' 010067  172212          1$:     MOV     R0,REGTBL+10     ; YES: ENABLE SEL10 READ
 1782  006246' 062767  000010  172204          ADD     #10,REGTBL+10
```

```
1783  006254' 000207                  2$:     RTS     PC              ;RETURN
1784
1785
1786
1787
1788
1789                                  ;*****************************************************************************
1790                                  ;* CLRRQI - IN P-P MODE, THIS SUBROUTINE CLEARS RQI, DISABLING INPUT REQUESTS
1791                                  ;*       ON THAT LOG LINK. IF M-P MODE, IT INCREMENTS WATCNT, AND IF IT EQUALS
1792                                  ;*       LLKCNT, IT CLEARS RQI. IN EITHER CASE, IT SETS THE OWAIT BIT IN THE
1793                                  ;*       INTRPT STATUS WORD FOR THAT LINK. THE DEVICE ADRS IS ASSUMED TO BE IN
1794                                  ;*       R0 ON ENTRY, AND THE INTRPT STATUS TABLE INDEX IS ASSUMED TO BE IN R4.
1795                                  ;*****************************************************************************
1796  006256' 052764  100000  000350' CLRRQI: BIS     #OWAIT,INTABL(R4) ;SET OWAIT BIT FOR AWAITING OUTPUT
1797  006264' 126727  171526  000002          CMPB    SR1,#2          ;SEE IF MULTIDROP
1798  006272' 001006                          BNE     1$              ;BR IF NOT
1799  006274' 005267  172046                  INC     WATCNT          ;INCR CNT OF LNKS AWAITING OUTPUT
1800  006300' 026767  172042  172022          CMP     WATCNT,LLKCNT   ;SEE IF ALL LINKS WAITING
1801  006306' 002403                          BLT     2$              ;BR IF NOT
1802  006310' 142760  000200  000000  1$:     BICB    #RQI,BSEL0(R0)  ;CLEAR RQI ON DEVICE
1803  006316' 000207                  2$:     RTS     PC              ;RETURN
1804
1805
1806
1807
1808
1809                                  ;*****************************************************************************
1810                                  ;* SETRQI - IN P-P MODE, THIS SUBROUTINE SETS RQI, ENABLING INPUT REQUESTS
1811                                  ;*       ON THIS UNIT. IN M-P MODE, THE SUBROUTINE DECREMENTS WATCNT AND SETS
1812                                  ;*       RQI. IN EITHER CASE, IT CLEARS THE OWAIT BIT IN THE INTRPT STATUS
1813                                  ;*       WORD FOR THAT LINK. THE DEVICE ADDRESS IS ASSUMED TO BE IN R0 ON ENTRY
1814                                  ;*       AND THE INTRPT STATUS TABLE INDEX IS ASSUMED TO BE IN R4.
1815                                  ;*****************************************************************************
1816  006320' 042764  100000  000350' SETRQI: BIC     #OWAIT,INTABL(R4) ;CLEAR AWAITING OUTPUT BIT
1817  006326' 126727  171464  000002          CMPB    SR1,#2          ;SEE IF MULTIDROP
1818  006334' 001002                          BNE     1$              ;BR IF NOT
1819  006336' 005367  172004                  DEC     WATCNT          ;DECR CNT OF LNKS AWAITING OUTPUT
1820  006342' 152760  000200  000000  1$:     BISB    #RQI,BSEL0(R0)  ;SET RQI ON DEVICE
1821  006350' 000207                          RTS     PC              ;RETURN
1822
1823
1824
1825
1826
1827
1828                                  ;*****************************************************************************
1829                                  ;* SERVICE CODE FOR LINKING A PARTICULAR DEVICE TO
1830                                  ;* A COMMON INPUT OR OUTPUT INTERRUPT SERVICE ROUTINE.
1831                                  ;*****************************************************************************
1832
1833
1834  006352'                         IISR0:
1835                                          ;-------------------------------------------------------------
1836  006352' 000004  000000' 006360'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1837                                          ;-------------------------------------------------------------
1838  006360' 016700  000022          1$:    MOV     ADDR0,R0        ;PUT DEVICE 0 CSR ADDRESS IN R0
```



```
1839  006364' 000167  174466                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 0
1840
1841  006370'                         OISR0:
1842                                          ;-------------------------------------------------------------
1843  006370' 000004  000000' 006376'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1844                                          ;-------------------------------------------------------------
1845  006376' 016700  000004          1$:    MOV     ADDR0,R0        ;PUT DEVICE 0 CSR ADDRESS IN R0
1846  006402' 000167  175440                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 0
1847
1848  006406' 000000                  ADDR0:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1849
1850
1851
1852  006410'                         IISR1:
1853                                          ;-------------------------------------------------------------
1854  006410' 000004  000000' 006416'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1855                                          ;-------------------------------------------------------------
1856  006416' 016700  000022          1$:    MOV     ADDR1,R0        ;PUT DEVICE 1 CSR ADDRESS IN R0
1857  006422' 000167  174430                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 1
1858
1859  006426'                         OISR1:
1860                                          ;-------------------------------------------------------------
1861  006426' 000004  000000' 006434'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1862                                          ;-------------------------------------------------------------
1863  006434' 016700  000004          1$:    MOV     ADDR1,R0        ;PUT DEVICE 1 CSR ADDRESS IN R0
1864  006440' 000167  175402                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 1
1865
1866  006444' 000000                  ADDR1:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1867
1868
1869
1870  006446'                         IISR2:
1871                                          ;-------------------------------------------------------------
1872  006446' 000004  000000' 006454'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1873                                          ;-------------------------------------------------------------
1874  006454' 016700  000022          1$:    MOV     ADDR2,R0        ;PUT DEVICE 2 CSR ADDRESS IN R0
1875  006460' 000167  174372                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 2
1876
1877  006464'                         OISR2:
1878                                          ;-------------------------------------------------------------
1879  006464' 000004  000000' 006472'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1880                                          ;-------------------------------------------------------------
1881  006472' 016700  000004          1$:    MOV     ADDR2,R0        ;PUT DEVICE 2 CSR ADDRESS IN R0
1882  006476' 000167  175344                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 2
1883
1884  006502' 000000                  ADDR2:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1885
1886
1887
1888  006504'                         IISR3:
1889                                          ;-------------------------------------------------------------
1890  006504' 000004  000000' 006512'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1891                                          ;-------------------------------------------------------------
1892  006512' 016700  000022          1$:    MOV     ADDR3,R0        ;PUT DEVICE 3 CSR ADDRESS IN R0
1893  006516' 000167  174334                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 3
1894
```

```
1895  006522'                          OISR3:
1896                                    ;-------------------------------------------------------------
1897  006522' 000004  000000' 006530'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1898                                    ;-------------------------------------------------------------
1899  006530' 016700  000004  1$:      MOV     ADDR3,R0        ;PUT DEVICE 3 CSR ADDRESS IN R0
1900  006534' 000167  175306           JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 3
1901
1902  006540' 000000                   ADDR3:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1903
1904
1905
1906  006542'                          IISR4:
1907                                    ;-------------------------------------------------------------
1908  006542' 000004  000000' 006550'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1909                                    ;-------------------------------------------------------------
1910  006550' 016700  000022  1$:      MOV     ADDR4,R0        ;PUT DEVICE 4 CSR ADDRESS IN R0
1911  006554' 000167  174276           JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 4
1912
1913  006560'                          OISR4:
1914                                    ;-------------------------------------------------------------
1915  006560' 000004  000000' 006566'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1916                                    ;-------------------------------------------------------------
1917  006566' 016700  000004  1$:      MOV     ADDR4,R0        ;PUT DEVICE 4 CSR ADDRESS IN R0
1918  006572' 000167  175250           JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 4
1919
1920  006576' 000000                   ADDR4:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1921
1922
1923
1924  006600'                          IISR5:
1925                                    ;-------------------------------------------------------------
1926  006600' 000004  000000' 006606'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1927                                    ;-------------------------------------------------------------
1928  006606' 016700  000022  1$:      MOV     ADDR5,R0        ;PUT DEVICE 5 CSR ADDRESS IN R0
1929  006612' 000167  174240           JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 5
1930
1931  006616'                          OISR5:
1932                                    ;-------------------------------------------------------------
1933  006616' 000004  000000' 006624'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1934                                    ;-------------------------------------------------------------
1935  006624' 016700  000004  1$:      MOV     ADDR5,R0        ;PUT DEVICE 5 CSR ADDRESS IN R0
1936  006630' 000167  175212           JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 5
1937
1938  006634' 000000                   ADDR5:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1939
1940
1941
1942  006636'                          IISR6:
1943                                    ;-------------------------------------------------------------
1944  006636' 000004  000000' 006644'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1945                                    ;-------------------------------------------------------------
1946  006644' 016700  000022  1$:      MOV     ADDR6,R0        ;PUT DEVICE 6 CSR ADDRESS IN R0
1947  006650' 000167  174202           JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 6
1948
1949  006654'                          OISR6:
1950                                    ;-------------------------------------------------------------
```



```
1951  006654' 000004  000000' 006662'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1952                                    ;-------------------------------------------------------------
1953  006662' 016700  000004  1$:      MOV     ADDR6,R0        ;PUT DEVICE 6 CSR ADDRESS IN R0
1954  006666' 000167  175154           JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 6
1955
1956  006672' 000000                   ADDR6:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1957
1958
1959
1960  006674'                          IISR7:
1961                                    ;-------------------------------------------------------------
1962  006674' 000004  000000' 006702'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1963                                    ;-------------------------------------------------------------
1964  006702' 016700  000022  1$:      MOV     ADDR7,R0        ;PUT DEVICE 7 CSR ADDRESS IN R0
1965  006706' 000167  174144           JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 7
1966
1967  006712'                          OISR7:
1968                                    ;-------------------------------------------------------------
1969  006712' 000004  000000' 006720'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1970                                    ;-------------------------------------------------------------
1971  006720' 016700  000004  1$:      MOV     ADDR7,R0        ;PUT DEVICE 7 CSR ADDRESS IN R0
1972  006724' 000167  175116           JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 7
1973
1974  006730' 000000                   ADDR7:  .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1975
1976
1977
1978  006732'                          IISR10:
1979                                    ;-------------------------------------------------------------
1980  006732' 000004  000000' 006740'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1981                                    ;-------------------------------------------------------------
1982  006740' 016700  000022  1$:      MOV     ADDR10,R0       ;PUT DEVICE 10 CSR ADDRESS IN R0
1983  006744' 000167  174106           JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 10
1984
1985  006750'                          OISR10:
1986                                    ;-------------------------------------------------------------
1987  006750' 000004  000000' 006756'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1988                                    ;-------------------------------------------------------------
1989  006756' 016700  000004  1$:      MOV     ADDR10,R0       ;PUT DEVICE 10 CSR ADDRESS IN R0
1990  006762' 000167  175060           JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 10
1991
1992  006766' 000000                   ADDR10: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
1993
1994
1995
1996  006770'                          IISR11:
1997                                    ;-------------------------------------------------------------
1998  006770' 000004  000000' 006776'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
1999                                    ;-------------------------------------------------------------
2000  006776' 016700  000022  1$:      MOV     ADDR11,R0       ;PUT DEVICE 11 CSR ADDRESS IN R0
2001  007002' 000167  174050           JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 11
2002
2003  007006'                          OISR11:
2004                                    ;-------------------------------------------------------------
2005  007006' 000004  000000' 007014'  PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
2006                                    ;-------------------------------------------------------------
```

```
 2007  007014' 016700  000004          1$:     MOV     ADDR11,R0       ;PUT DEVICE 11 CSR ADDRESS IN R0
 2008  007020' 000167  175022                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 11
 2009
 2010  007024' 000000                  ADDR11: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
 2011
 2012
 2013
 2014  007026'                         IISR12:
 2015                                          ;---------------------------------------------------------------
 2016  007026' 000004  000000' 007034'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2017                                          ;---------------------------------------------------------------
 2018  007034' 016700  000022          1$:     MOV     ADDR12,R0       ;PUT DEVICE 12 CSR ADDRESS IN R0
 2019  007040' 000167  174012                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 12
 2020
 2021  007044'                         OISR12:
 2022                                          ;---------------------------------------------------------------
 2023  007044' 000004  000000' 007052'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2024                                          ;---------------------------------------------------------------
 2025  007052' 016700  000004          1$:     MOV     ADDR12,R0       ;PUT DEVICE 12 CSR ADDRESS IN R0
 2026  007056' 000167  174764                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 12
 2027
 2028  007062' 000000                  ADDR12: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
 2029
 2030
 2031
 2032  007064'                         IISR13:
 2033                                          ;---------------------------------------------------------------
 2034  007064' 000004  000000' 007072'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2035                                          ;---------------------------------------------------------------
 2036  007072' 016700  000022          1$:     MOV     ADDR13,R0       ;PUT DEVICE 13 CSR ADDRESS IN R0
 2037  007076' 000167  173754                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 13
 2038
 2039  007102'                         OISR13:
 2040                                          ;---------------------------------------------------------------
 2041  007102' 000004  000000' 007110'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2042                                          ;---------------------------------------------------------------
 2043  007110' 016700  000004          1$:     MOV     ADDR13,R0       ;PUT DEVICE 13 CSR ADDRESS IN R0
 2044  007114' 000167  174726                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 13
 2045
 2046  007120' 000000                  ADDR13: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
 2047
 2048
 2049
 2050  007122'                         IISR14:
 2051                                          ;---------------------------------------------------------------
 2052  007122' 000004  000000' 007130'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2053                                          ;---------------------------------------------------------------
 2054  007130' 016700  000022          1$:     MOV     ADDR14,R0       ;PUT DEVICE 14 CSR ADDRESS IN R0
 2055  007134' 000167  173716                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 14
 2056
 2057  007140'                         OISR14:
 2058                                          ;---------------------------------------------------------------
 2059  007140' 000004  000000' 007146'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2060                                          ;---------------------------------------------------------------
 2061  007146' 016700  000004          1$:     MOV     ADDR14,R0       ;PUT DEVICE 14 CSR ADDRESS IN R0
 2062  007152' 000167  174670                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 14
```



```
 2063
 2064  007156' 000000                  ADDR14: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
 2065
 2066
 2067
 2068  007160'                         IISR15:
 2069                                          ;---------------------------------------------------------------
 2070  007160' 000004  000000' 007166'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2071                                          ;---------------------------------------------------------------
 2072  007166' 016700  000022          1$:     MOV     ADDR15,R0       ;PUT DEVICE 15 CSR ADDRESS IN R0
 2073  007172' 000167  173660                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 15
 2074
 2075  007176'                         OISR15:
 2076                                          ;---------------------------------------------------------------
 2077  007176' 000004  000000' 007204'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2078                                          ;---------------------------------------------------------------
 2079  007204' 016700  000004          1$:     MOV     ADDR15,R0       ;PUT DEVICE 15 CSR ADDRESS IN R0
 2080  007210' 000167  174632                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 15
 2081
 2082  007214' 000000                  ADDR15: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
 2083
 2084
 2085
 2086  007216'                         IISR16:
 2087                                          ;---------------------------------------------------------------
 2088  007216' 000004  000000' 007224'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2089                                          ;---------------------------------------------------------------
 2090  007224' 016700  000022          1$:     MOV     ADDR16,R0       ;PUT DEVICE 16 CSR ADDRESS IN R0
 2091  007230' 000167  173622                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 16
 2092
 2093  007234'                         OISR16:
 2094                                          ;---------------------------------------------------------------
 2095  007234' 000004  000000' 007242'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2096                                          ;---------------------------------------------------------------
 2097  007242' 016700  000004          1$:     MOV     ADDR16,R0       ;PUT DEVICE 16 CSR ADDRESS IN R0
 2098  007246' 000167  174574                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 16
 2099
 2100  007252' 000000                  ADDR16: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
 2101
 2102
 2103
 2104  007254'                         IISR17:
 2105                                          ;---------------------------------------------------------------
 2106  007254' 000004  000000' 007262'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2107                                          ;---------------------------------------------------------------
 2108  007262' 016700  000022          1$:     MOV     ADDR17,R0       ;PUT DEVICE 17 CSR ADDRESS IN R0
 2109  007266' 000167  173564                  JMP     INISR           ;GO SERVICE INPUT INTERRUPT ON DEVICE 17
 2110
 2111  007272'                         OISR17:
 2112                                          ;---------------------------------------------------------------
 2113  007272' 000004  000000' 007300'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RTI
 2114                                          ;---------------------------------------------------------------
 2115  007300' 016700  000004          1$:     MOV     ADDR17,R0       ;PUT DEVICE 17 CSR ADDRESS IN R0
 2116  007304' 000167  174536                  JMP     OUTISR          ;GO SERVICE OUTPUT INTERRUPT ON DEVICE 17
 2117
 2118  007310' 000000                  ADDR17: .WORD   0               ;DEVICE CSR ADDRESS STORED HERE
```

```
 2119
 2120
 2121
```



```
 2122
 2123
 2124
 2125
 2126
 2127  007312' 002000                BUFIN:  .BLKB   1024.   ;INPUT BUFFER (1024 BYTES)
 2128
 2129
 2130
 2131
 2132
 2133                                ;************* PATCH AREA FOR DEBUG **************
 2134  011312' 000036                PATCH:  .BLKW   30.
 2135                                ;************************************************
 2136
 2137          000001                        .END
```

```
ABORT = 000004          622#
ACSR    000102R         317#  1768#
ADDR    000006R         283#   867    1001    1206    1247    1386    1750
ADDR0   006406R        1838   1845    1848#
ADDR1   006444R        1856   1863    1866#
ADDR10  006766R        1982   1989    1992#
ADDR11  007024R        2000   2007    2010#
ADDR12  007062R        2018   2025    2028#
ADDR13  007120R        2036   2043    2046#
ADDR14  007156R        2054   2061    2064#
ADDR15  007214R        2072   2079    2082#
ADDR16  007252R        2090   2097    2100#
ADDR17  007310R        2108   2115    2118#
ADDR2   006502R        1874   1881    1884#
ADDR22= 001000          341#
ADDR3   006540R        1892   1899    1902#
ADDR4   006576R        1910   1917    1920#
ADDR5   006634R        1928   1935    1938#
ADDR6   006672R        1946   1953    1956#
ADDR7   006730R        1964   1971    1974#
ASB     000106R         321#
ASTAT   000104R         319#  1769*
AWAS    000110R         322#
AX1   = 000001          653#   713#
AX2   = 000002          652#   712#
BABTRB= 000026          583#
BACCIR= 000000          422#  1305
BACCIT= 000004          421#  1318
BACCOR= 000000          431#  1402
BACCOT= 000004          430#  1400
BACORU= 000003          432#
BACOTN= 000007          434#
BACOTS= 000006          433#
BADINI= 000004          781#   893    1627    1649
BADRED= 000036          391#  1457    1470    1483
BADWRT= 000037          392#  1421    1434    1447
BA0   = 000001          545#
BA1   = 000002          544#
BA10  = 002000          535#
BA11  = 004000          534#
BA12  = 010000          533#
BA13  = 020000          532#
BA14  = 040000          531#
BA15  = 100000          530#
BA16  = 040000          553#
BA17  = 100000          552#
BA2   = 000004          543#
BA3   = 000010          542#
BA4   = 000020          541#
BA5   = 000040          540#
BA6   = 000100          539#
BA7   = 000200          538#
BA8   = 000400          537#
BA9   = 001000          536#
BCC   = 000001          689#
BEGIN   000000R         280#   853    919     920     948    1041    1042    1050    1060    1076    1085    1094    1095
```



```
                       1103   1118   1185   1196   1219   1283   1413   1423   1436   1449   1459   1472   1485
                       1489   1504   1516   1536   1551   1564   1565   1569   1570   1583   1584   1590   1591
                       1633   1634   1639   1647   1719   1720   1728   1836   1843   1854   1861   1872   1879
                       1890   1897   1908   1915   1926   1933   1944   1951   1962   1969   1980   1987   1998
                       2005   2016   2023   2034   2041   2052   2059   2070   2077   2088   2095   2106   2113
BIRDON= 000010          766#  1268   1315
BITDON= 000020          767#  1266   1328
BIT0  = 000001          341#   358#   374    379    457    477    523    545    567    615    624    653    665
                        677    689    701    713    733    747    753    763    779    869   1002   1207   1751
BIT1  = 000002          341#   357#   380    476    522    544    566    614    623    642    652    664    676
                        688    700    712    732    746    754    764    780
BIT10 = 002000          341#   348#   535    557
BIT11 = 004000          341#   347#   534    556
BIT12 = 010000          341#   346#   533    555
BIT13 = 020000          341#   345#   532    554
BIT14 = 040000          341#   344#   531    553
BIT15 = 100000          341#   343#   530    552    772
BIT2  = 000004          341#   356#   475    543    565    613    622    641    651    663    675    687    699
                        711    731    745    755    765    781
BIT3  = 000010          341#   355#   465    474    542    564    612    621    640    650    662    674    686
                        698    710    730    744    756    766
BIT4  = 000020          341#   354#   456    464    473    521    541    563    611    639    649    661    673
                        685    697    709    718    729    743    767
BIT5  = 000040          341#   353#   500    520    540    562    596    610    631    638    648    660    672
                        684    696    708    719    728    742    768
BIT6  = 000100          341#   352#   472    499    519    539    561    595    609    630    637    647    659
                        671    683    695    707    720    727    741    769
BIT7  = 000200          341#   351#   455    463    471    498    518    538    560    608    620    629    636
                        658    670    682    694    706    721    726    740    770
BIT8  = 000400          341#   350#   537    559    771
BIT9  = 001000          341#   349#   536    558
BORDON= 000040          768#  1494   1497
BOTDON= 000100          769#  1451   1495
BPULL = 000100          630#
BREAKS= 104407          341#   919    920    1041   1042   1094   1095   1564   1565   1569   1570   1583   1584
                       1590   1591   1633   1634   1719   1720
BR1     000012R         285#   897    902
BR2     000013R         286#
BSEL0 = 000000          400#   986*  1137*  1211*  1235*  1242*  1285*  1356*  1371*  1415*  1688*  1717*  1754*
                       1802*  1820*
BSEL1 = 000001          401#   885*   888*   889*   890*   914*   921*   924*   982*  1562*  1581*  1603*  1605*
                       1606*  1618*  1629
BSEL10= 000010          412#
BSEL11= 000011          413#
BSEL2 = 000002          403#  1123*  1128*  1348*  1350   1353   1398   1502*  1553*  1566   1568*  1585   1588*
                       1592
BSEL3 = 000003          404#  1238*  1347*  1390
BSEL4 = 000004          406#  1528   1667
BSEL5 = 000005          407#
BSEL6 = 000006          409#  1292*  1300*  1335*  1343*  1430   1466   1508   1520   1642   1672
BSEL7 = 000007          410#  1308*  1321*  1426   1462
BTOD$ = 104421          341#
BUFIN   007312R         328   2127#
BUFRTC= 000020          599#  1111   1543
BUFTSM= 000300          585#
CARR  = 000001          701#
```

CXDMDC.P11 25-MAR-81 08:25 CROSS REFERENCE TABLE -- USER SYMBOLS

```
CCMSK = 140000          446#  1443  1479
CC0   = 000001          567#
CC1   = 000002          566#
CC10  = 002000          557#
CC11  = 004000          556#
CC12  = 010000          555#
CC13  = 020000          554#
CC2   = 000004          565#
CC3   = 000010          564#
CC4   = 000020          563#
CC5   = 000040          562#
CC6   = 000100          561#
CC7   = 000200          560#
CC8   = 000400          559#
CC9   = 001000          558#
CDATA$= 104412          341#  1489
CDERCT  000144R         336#
CDWDCT  000146R         337#
CHKMOR  004004R        1287   1350#  1554
CLRCMD  006056R         861   1739#
CLRCSR  005654R        1617   1650   1688#
CLRRQI  006256R        1294   1302   1329   1337   1345   1796#
CMDMSK= 000370          442#  1053   1106   1399
CONFIG  000056R         305#
COUNT   000262R         790#
CRC1  = 000100          659#
CRC2  = 000200          658#
CS    = 000004          699#
CSEL0   000274R         795#
CSEL10  000304R         799#  1742#
CSEL2   000276R         796#   913#   972#  1015#  1016#  1238  1289#  1297#  1305#  1311#  1318#  1324#  1332#
                       1340#  1348   1739#
CSEL4   000300R         797#   958#   961#  1017#  1026#  1028#  1033#  1239  1740#
CSEL6   000302R         798#   917#   930#   939#   943#   973#  1018#  1020#  1029#  1036#  1089#  1240  1741#
CSRA    000100R         315#  1488#  1493   1767#
CTLIN = 000001          420#   972   1015   1289   1297   1332  1340
CTLOUT= 000001          429#  1054   1404
C32BCC= 000040          742#
C32ENB= 000004          745#
DATCK$= 104411          341#
DATERR= 000001          386#  1532
DATER$= 104404          341#
DDCMP = 000001          665#   733#
DEADTR= 000022          581#
DERINB= 000113          511#  1343
DEROTB= 000112          510#  1335
DEVPTR  000270R         793#   869#   872   874#  1002#  1004  1006#  1207#  1209  1216#  1751#  1752  1756#
DISABL  006100R         852   1503   1750#
DISCMP= 000002          522#
DISCON= 000304          587#
DISPIG= 000200          518#
DISSI = 000040          648#   708#
DMVBIT= 000001          379#
DMVLUP  005210R         911   1562#
DMVRD   005262R        1581#  1670
DRPMOD  000474R         852#   925   950   1126
```


CXDMDC.P11 25-MAR-81 08:25 CROSS REFERENCE TABLE -- USER SYMBOLS

```
DTR   = 000100          637#   695#
DVID1   000014R         287#   850
EAMSK = 000077          447#  1427   1463
EBLK  = 000002          688#
ENABLE  002612R        1162   1176#
ENABS$= 000200          721#   928    937   1676
ENACMP= 000001          523#
ENAX  = 000004          651#   711#
ENBPIG= 000100          519#
ENDIT$= 104413          341#  1504
ENDS  = 104410          341#   853
EOM   = 000002          623#
ERINB = 000013          601#  1526
EROTB = 000012          600#  1524
ERRORS  000322R         806#   859#   893    922    975   1627#  1649#  1708#  1730#
ERRTYP  000106R         320#   946#  1048#  1057#  1073#  1082#  1101#  1115#  1281#  1411#  1421#  1434#  1447#
                       1457#  1470#  1483#  1513#  1532#  1548#  1637#  1645#  1726#
ESTRIH= 000001          501#  1020
EXECUT  005352R        1603#  1665
EXIT$ = 104400          341#  1219
FDCS  = 000005          489#   943
FDPP  = 000003          487#   917
FDPPDC= 000001          485#
FDTS  = 000007          491#
FLAGS   000320R         805#   860#  1021#  1030#  1037#  1043  1090#  1096  1121#  1208#  1236  1241#  1372
                       1381#  1709#  1710#  1721   1731#
FREE    000150R         338#
GETERR  006142R         945   1047   1056   1072   1081   1100  1114  1280  1410  1420  1433  1446  1456
                       1469   1482   1512   1531   1547   1636  1644  1725  1767#
GETPA$= 104415          341#  1196
GOAH  = 000010          621#
GOODIN= 000305          362#  1642
GWBUF$= 104414          341#  1185
HALT$= 000005           505#  1089   1292
HDCS  = 000004          488#   939
HDPP  = 000002          486#   930
HDPPDC= 000000          484#
HDTS  = 000006          490#
HDX   = 000020          639#   697#
HLTDON= 000002          764#  1169   1272   1293
HRDCNT  000044R         300#
HRDER$= 104405          341#   948   1050   1060   1076  1085  1103  1118  1283  1413  1423  1436  1449
                       1459   1472   1485   1516   1551  1639  1647  1728
HRDPAS  000050R         302#
IACT  = 000100          683#
IC    = 000200          629#   682#
ICIR  = 000010          730#
ICONT   000036R         297#
ICOUNT  000040R         298#
IDLE  = 000040          660#
IDNUM   000122R         327#
IEI   = 000001          457#  1211   1235   1285   1356  1371  1717
IEO   = 000020          456#  1211   1235   1356   1371  1415  1717
IERDON= 000400          771#  1169   1274   1344
IERR  = 000002          664#
IISR0   006352R         868   1834#
```

```
IISR1   006410R          1852#
IISR10  006732R          1978#
IISR11  006770R          1996#
IISR12  007026R          2014#
IISR13  007064R          2032#
IISR14  007122R          2050#
IISR15  007160R          2068#
IISR16  007216R          2086#
IISR17  007254R          2104#
IISR2   006446R          1870#
IISR3   006504R          1888#
IISR4   006542R          1906#
IISR5   006600R          1924#
IISR6   006636R          1942#
IISR7   006674R          1960#
ILGINT= 000011            388#  1281  1411
IMODX.= 000000            339#  1186
INCMND  005710R           918   974   1022  1031  1038  1091  1708#
INFOUT= 000002            435#  1108  1407
INIDEV  005410R           892   1616#
ININT = 000001            753#  1241  1710  1721
INISR   003056R          1234#  1352  1839  1857  1875  1893  1911  1929  1947  1965  1983  2001  2019
                         2037   2055  2073  2091  2109
INIT    000030R           294#
INTABL  000350R           832#   862   864   1158  1168  1264  1266  1268  1270  1272  1274  1276  1278
                         1293*  1301*  1315*  1328*  1336*  1344*  1451*  1494*  1495  1497  1796*  1816*
INTFDG= 000022            508#
INTGRL= 000010            744#   958   969
INTIMO= 000001            779#   922   975  1708  1730
INTIMR  000324R           807#  1716*  1723*
INTR    000120R           326#
IRDY  = 000020            685#
ISTART= 000003            503#  1036  1300
ISTCNT  000332R           812#  1034*  1079*
ISTDON= 000004            765#  1169  1270  1301
I422  = 000200            740#   958   963
KILLTR= 000002            502#
LATPS = 000040            520#
LLKCNT  000330R           809#   870*  987*  989*  1000  1500  1800
LNKDON  000342R           816#
LULOOP= 000010            474#   890   914   921   982
LULP  = 000040            631#   684#
LURG16  000272R           794#   928   932   937   941  1667*  1672*  1674*  1676*  1678*
MAINT = 000004            504#
MAINT1= 000010            640#
MAINT2= 000004            641#
MAP22$= 104416            341#
MCLK  = 000002            732#
MCLR  = 000100            472#   885  1618
MDFDON= 000001            763#  1159  1169  1278
MNTHLT= 000014            579#
MNTRUN= 000012            578#
MODDEF= 000002            419#   913
MODES$= 000361            448#
MODE22= 000002            380#  1189  1200  1309  1322  1428  1440  1464  1476
MODMSK= 000370            443#
```



```
MODNAM  000000R           281#
MODR  = 000010            698#
MODSP   000252R           295    339#
MODSW0= 000020            718#   932   941
MODSW1= 000040            719#
MODSW2= 000100            720#
MRDY  = 000200            367#  1566  1585  1592
MSGN$ = 104403            341#
MSGS$ = 104402            341#
MSG$  = 104401            341#
NOINIT= 000034            390#  1637  1645
NOINTR= 000023            389#  1048  1101  1726
NONEXM= 000302            586#
NONLUP= 000001            374#
NONQI = 000004            755#  1236  1709  1731
NONQO = 000010            756#   860  1208  1372
NORET = 000000            597#
NOTDFN= 000000            385#  1057  1073  1082  1115  1513  1548
NULL  = 000000            341#   948
N.DEVS  000256R           788#   858*  977*
OC    = 000200            620#
OCUR  = 000020            729#
OERDON= 000200            770#  1169  1276  1336
OISR0   006370R          1841#
OISR1   006426R          1859#
OISR10  006750R          1985#
OISR11  007006R          2003#
OISR12  007044R          2021#
OISR13  007102R          2039#
OISR14  007140R          2057#
OISR15  007176R          2075#
OISR16  007234R          2093#
OISR17  007272R          2111#
OISR2   006464R          1877#
OISR3   006522R          1895#
OISR4   006560R          1913#
OISR5   006616R          1931#
OISR6   006654R          1949#
OISR7   006712R          1967#
OP    = 000002            746#   958
OPEN  = 000000            282    288   289   290   291   308   309   310   311   312   313   314   315
                          317    319   321   322   324   325   326   329   330   332   333   335   336
                          337    338   341#
OTOA$ = 104420            341#
OUTDON  000344R           817#  1177*  1499*  1500
OUTIMO= 000002            780#
OUTIMR  000326R           808#  1039*  1045*  1092*  1098*  1122*
OUTINT= 000002            754#  1021   1030   1037   1043   1090   1096   1121   1381
OUTISR  004046R          1355  1370#  1846   1864   1882   1900   1918   1936   1954   1972   1990   2008   2026
                         2044   2062   2080   2098   2116
OVRR  = 000010            686#
OWAIT = 100000            772#  1264  1796  1816
PASCNT  000034R           296#
PATCH   011312R          2134#
PIRQ$ = 000004            341#  1836  1843  1854  1861  1872  1879  1890  1897  1908  1915  1926  1933
                         1944   1951  1962  1969  1980  1987  1998  2005  2016  2023  2034  2041  2052
```

CXDMDC.P11 25-MAR-81 08:25 CROSS REFERENCE TABLE -- USER SYMBOLS

```
                        2059    2070    2077    2088    2095    2106    2113
POLL  = 000200           636#
PUPSP = 005726           341#
PUPSP2= 022626           341#
PRTY  = 000000           341#
PRTY0 = 000000           341#
PRTY1 = 000040           341#
PRTY2 = 000100           341#
PRTY3 = 000140           341#
PRTY4 = 000200           341#
PRTY5 = 000240           285     286     341#
PRTY6 = 000300           341#
PRTY7 = 000340           341#
PS    = 177776           341#
PSW   = 177776           341#
PUSH  = 005746           341#
PUSH2 = 024646           341#
PWRFLG= 000002           341#
QUOVRN= 000306           588#
Q22BIT= 000010           465#    1311    1324
RAB   = 000004           687#
RAND$ = 104417           341#
RANNUM  000054R          304#
RBUFEA  000130R          330#    1197
RBUFEX  000254R          786#    1197*   1198*   1199*   1202*   1203*   1204*   1308    1313    1467
RBUFPA  000126R          329#    1306    1454    1489
RBUFSZ  000132R          331#
RBUFVA  000124R          328#    1196
RDALL = 000004           663#
RDAX  = 000020           649#     709#
RDCTSS= 000100           499#     595#   1335    1343    1521
RDLU16= 021344           364#    1666
RDY1  = 000020           464#    1350
RDY0  = 000200           463#    1353
READY = 000200           706#
READ16  005570R          891     1662#
REDLOC= 000001           368#    1588
REDMDM= 000020           506#
REDTSS= 000040           500#     596#
REENTR  002572R         1153    1167#   1506
REGTBL  000450R          837#    1050    1060    1076    1085    1103    1118    1283    1413    1423    1436    1449    1459
                        1472    1485    1516    1536    1551    1639    1647    1728    1770*   1771*   1772*   1773*   1774*
                        1775*   1776*   1779*   1781*   1782*
REQMSK= 000340           444#
RESTRT  002550R          323     1157#
RES1    000056R          306#
RES2    000060R          307#
RETMST= 000010           598#
RING  = 000200           694#
RINGDT= 000020           580#
RQMI  = 000002           476#    1603    1605    1606
RQMO  = 000004           475#    1603    1605    1606
RQI   = 000200           455#    1211    1242    1285    1717    1802    1820
RSEL0   000306R          800#    1374*   1711*
RSEL10  000316R          804#    1380*   1715*
RSEL2   000310R          801#    1053*   1054    1106*   1108    1375*   1712*
```


CXDMDC.P11 25-MAR-81 08:25 CROSS REFERENCE TABLE -- USER SYMBOLS

```
RSEL4   000312R          802#    1376*   1713*
RSEL6   000314R          803#    1064    1068    1070    1110*   1111    1377*   1714*
RSTRT   000112R          323#
RTHRER= 000002           574#
RTNINI  004034R         1354    1356#   1382
RTNMON  003052R         1218#    1286    1357    1416
RTNMSK= 000340           445#    1110    1523    1542
RTS   = 000040           696#
RUN   = 000200           471#     888     889     924    1629
RUNST = 000024           582#    1064    1508
RX0   = 000001           677#
RX1   = 000002           676#
RX2   = 000004           675#
RX3   = 000010           674#
RX4   = 000020           673#
RX5   = 000040           672#
RX6   = 000100           671#
RX7   = 000200           670#
SAVBF   000264R          791#
SBADR   000102R          316#
SCRACH  000334R          813#
SECA  = 000020           661#
SELECT  000266R          792#     850*    872    1004    1209    1752
SELERR= 000007           387#     946
SELFR = 000040           638#
SELMDM= 000023           509#     973
SELSHY= 000002           642#
SEL0  = 000000           399#    1125*   1374    1768
SEL10 = 000010           411#    1312*   1325*   1380    1442    1478    1694*
SEL2  = 000002           402#    1375    1689*   1769
SEL4  = 000004           405#    1239*   1290*   1298*   1306*   1319*   1333*   1341*   1376    1418    1454    1587*   1690*
SEL6  = 000006           408#    1240*   1291*   1299*   1307*   1313*   1320*   1326*   1334*   1342*   1377    1439    1475
                        1604*   1691*
SETRQI  006320R         1510    1540    1545    1816#
SETUP   000504R          851     858#
SIGQ  = 000100           727#
SIGR  = 000200           726#
SOFCNT  000042R          299#
SOFERS= 104406           341#    1536
SOFPAS  000046R          301#
SOM   = 000001           624#
SPOINT  000032R          295#
SPSIZ = 000040             1#    339
SR1     000016R          288#    907     915     926     935     954     978     984     997    1129    1144    1179    1244
                        1383    1797    1817
SR2     000020R          289#    909     956     980     989    1149
SR3     000022R          290#    959     999    1146    1181    1257    1259    1261    1392    1396
SR4     000024R          291#    879     886     905     952    1009    1024    1189    1200    1212    1309    1322    1378
                        1428    1440    1464    1476    1619    1662    1692    1777
START   000464R          294     850#
STARUN= 000010           577#
STAT    000026R          293#
STBY  = 000002           700#
STEPLU= 000020           473#
STEPMP= 000001           477#    1605    1606
STHRER= 000006           576#    1070
```

```
STRIP = 000010           662#
STRTRB= 000030           584#
STRTUP  001474R          876     995#
SVR0    000062R          308#
SVR1    000064R          309#
SVR2    000066R          310#
SVR3    000070R          311#
SVR4    000072R          312#
SVR5    000074R          313#
SVR6    000076R          314#
SWPBUF= 121000           366#   1671
SYSCNT  000052R          303#
TEST  = 000001           747#    958
TESTMD= 000004           731#
TOTAL   000260R          789#
TRBADR  000336R          814#   996#   999#  1016  1131#  1132  1134#  1143#  1146#  1147  1176#  1181#  1182#
                        1251#  1252  1254#  1255  1257#  1258  1261  1347
TRBMAX  000340R          815#  1147#  1148#  1149#  1150  1152#  1255
TRPDFD= 000022           341#
TTHRER= 000004           575#  1068
TTLOOP= 000006           369#  1568
TXDATA= 000040           728#
TXEN  = 000100           647#   707#
TX0   = 000001           615#
TX1   = 000002           614#
TX2   = 000004           613#
TX3   = 000010           612#
TX4   = 000020           611#
TX5   = 000040           610#
TX6   = 000100           609#
TX7   = 000200           608#
ULATPS= 000020           521#
VECTOR  000010R          284#   866
V35   = 000020           743#   958    961    967
WASADR  000104R          318#
WATCNT  000346R          819#  1178#  1799#  1800  1819#
WAX   = 000010           650#   710#
WBUFEA  000136R          333#  1186
WBUFEX  000252R          785#  1186#  1187#  1188#  1191#  1192#  1193#  1321  1326  1431
WBUFPA  000134R          332#  1319   1418
WBUFRQ  000140R          334#
WBUFSZ  000142R          335#
WDFR    000116R          325#
WDTO    000114R          324#
WRIMDM= 000021           507#
WRITSS= 000200           498#  1029
XFLAG   000056R          282#
XYZ   = 000100           741#   958    965
.     = 011406R          832#  2127#  2134#
```



```
BKMOD       1#
BREAK       1#    919   1040   1093   1563   1569   1582   1589   1633   1718
BTOD        1#
CKDATA      1#   1489
DATACK      1#
DATERR      1#
DFSEVN      1#    341
DSEVNT      1#    341
END         1#
ENDIT       1#   1504
ENDMOD      1#    853
EQUATS      1#    341
EXIT        1#   1218
GETPA       1#   1195
GWBUFF      1#   1184
HRDER       1#    947   1049   1059   1075   1084   1102   1117   1282   1412   1422   1435   1448   1458   1471
         1484   1515   1550   1638   1646   1727
IOMOD       1#
IOMODP      1#
IOMODR      1#
IOMODX      1#    276
ISRS     1833#  1834   1852   1870   1888   1906   1924   1942   1960   1978   1996   2014   2032   2050   2068
         2086   2104
MAP22       1#
MODULE      1#    276
MSG         1#
MSGN        1#
MSGS        1#
NBKMOD      1#
OTOA        1#
PIRQ        1#   1834   1841   1852   1859   1870   1877   1888   1895   1906   1913   1924   1931   1942   1949
         1960   1967   1978   1985   1996   2003   2014   2021   2032   2039   2050   2057   2068   2075   2086
         2093   2104   2111
RAND        1#
SBKMOD      1#
SOFER       1#   1535


. ABS.  000000    000
        011406    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XDMDC0,CXDMDC.SEQ/CRF/SOL/NL:TOC=DDXCOM.P11,CXDMDC.P11
 RUN-TIME: 2 5 .8 SECONDS
 RUN-TIME RATIO: 79/9=8.7
 CORE USED:  10K  (19 PAGES)
```

.REM !

IDENTIFICATION

PRODUCT CODE: AC-F0729-MC

PRODUCT NAME: CXBMHB0 M9312 MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

THIS MODULE PERFORMS CHECKSUM VERIFICATION OF THE M9312 BOOTSTRAP TERMINATOR. IT COMPARES THE CHECKSUM FOUND IN THE LAST LOCATION OF THE ROMS TO ONE IT CALCULATES FROM ALL OTHER LOCATIONS OTHER THAN THE EXCEPTION LOCATION. LOCATION SR1 IS USED TO SELECT WHICH ROMS TO TEST.

2.0 REQUIREMENTS

HARDWARE: ANY PDP-11 PROCESSOR WITH A M9312 BOOTSTRAP TERMINATOR AND AT LEAST ONE ROM PLUGGED INTO THE TERMINATOR.

STORAGE:: BMH REQUIRES:
1. DECIMAL WORDS: 237
2. OCTAL WORDS: 0355
3. OCTAL BYTES: 732

3.0 PASS DEFINITION

ONE PASS CONSTSTS OF DOING A CHECKSUM ON EACH ROM 30 (8) TIMES.

4.0 EXECUTION TIME

BMH TAKES APPROXIMATELY 35 SECONDS TO COMPLETE A PASS WHEN RUNNING ALONE.

5.0 CONFIGURATION REQUIREMENTS

SET THE CORRESPONDING BITS IN SR1 TO A"1" FOR THE DESIRED ROMS:

BIT 0 = 1 - DIAG. ROM
BIT 1 = 1 - BOOT ROM IN E-35 (173000-173177)
BIT 2 = 1 - BOOT ROM IN E-33 (173200-173377)
BIT 3 = 1 - BOOT ROM IN E-34 (173400-173577)
BIT 4 = 1 - BOOT ROM IN E-32 (173600-173777)

6.0 DEVICE/OPTION SETUP

NONE

7.0 MODULE OPERATION

THIS MODULE FIRST CHECKS FOR SR1 RO BE NON-ZERO. IF IT IS ZERO THE MODULE TYPES THE MESSAGE:

NO ROMS SELECTED SR 1 = 0

AND THEN NOTIFIES THE MONITOR TO DROP IT. IF SR1 IS NON-ZERO THE CONTROL ROUTINE THEN CALLS THE "GETROM" SUBROUTINE TO LOCATE THE FIRST ROM. IF THE FIRST ROM IS NOT SELECTED THE "GETROM" SUBROUTINE WILL RETURN WITH LOCATION "FIRSTA"=0. IF THE FIRST ROM IS SELECTED "GETROM" WILL RETURN WITH "FIRSTA" CONTAINING THE FIRST ADDRESS TO BE SUMMED, LOCATION "EXCADR" CONTAINING THE EXCEPTION ADDRESS, LOCATION "LASTA" CONTAINING THE LAST ADDRESS TO BE SUMMED AND LOCATION "GOODA" CONTAINING THE ADDRESS OF THE ROMS CHECKSUM.

THE CONTROL ROUTINE THEN CALLS THE SUB-ROUTINE "CHECKR" TO CHECK THE ROM. IF LOCATION "FIRSTA" EQUALS ZERO "CHECKR" WILL JUST RETURN TO THE CONTROL ROUTINE. BUT IF "FIRSTA" IS NON-ZERO "CHECKR" WILL CALL THE ROUTINE "CALSUM". "CALSUM" CALCULATES THE CRC16 OF THE ROM FROM "FIRSTA" TO "LASTA" WITH THE EXCEPTION OF "EXCADR". "CALSUM" RETURNS WITH THE CALCULATED CHECKSUM IN LOCATION "BAD". THE ROUTINE "CHECKR" THEN COMPARES THE GOOD CHECKSUM FROM THE BOARD WITH THE "BAD" SUM AND IF THEY DON'T COMPARE REPORTS AN ERROR:

CHECKSUM ERROR ON M9312 BOOTSTRAP.

THE ROUTINE "CHECKR" THEN RETURNS TO THE CONTROL ROUTINE AND THE PROCESS IS REPEATED FOR EACH ROM.

8.0 OPERATING OPTIONS

NONE

9.0 NON-STANDARD PRINTOUTS

NONE

```
000000'  BKMOD   <BMHB >,0,0,0,0,0,30,164
000000'           MODULE  40020,BMHB ,0,0,0,0,0,30,164
                  .TITLE  BMHB DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION   6      23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 046502  041110      040 MODNAM: .ASCII  /BMHB / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN    ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000000                  ADDR:   0+0             ;1ST DEVICE ADDR.
000010' 000000                  VECTOR: 0+0             ;1ST DEVICE VECTOR.
000012'     000                 BR1:    .BYTE   PRTY0+0 ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0 ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1             ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN            ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN            ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN            ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN            ;SWITCH REGISTER 4
                                ;*****************************************************************************
000026' 040020                  STAT:   40020           ;STATUS WORD.
000030' 000252'                 INIT:   START           ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP           ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0               ;PASS COUNTER.
000036' 000030                  ICONT:  30              ;# OF ITERATIONS PER PASS=30
000040' 000000                  ICOUNT: 0               ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0               ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0               ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0               ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0               ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0               ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0               ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                 ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0               ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0               ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN            ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN            ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN            ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN            ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN            ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN            ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN            ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN            ;ADDR OF CURRENT CSR.
000102'                         SBADR:                  ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN            ;CONTENTS OF CSR.
000104'                         WASADR:                 ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN            ;STATUS REG CONTENTS.
000106'                         ERRTYP:                 ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN            ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN            ;ACTUAL DATA.
000112' 000257'                 RSTRT:  RESTRT          ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN            ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN            ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN            ;# OF INTERRUPTS PER ITERATION
000122' 000164                  IDNUM:  164             ;MODULE IDENTIFICATION NUMBER=164
        000040                          .REPT   SPSIZ   ;MODULE STACK STARTS HERE.
                                        .NLIST
```



```
                                        .WORD   0
                                        .LIST
                                        .ENDR
        000224'                 MODSP:
                                ;*****************************************************************************
  200
  201   000224' 000000                  FIRSTA: 0
  202   000226' 000000                  EXCADR: 0
  203   000230' 000000                  LASTA:  0
  204   000232' 000000                  CRDMP:  0
  205   000234' 000000                  GOODA:  0
  206   000236' 000000                  BAD:    0
  207   000240' 000000                  COUNT:  0
  208   000242' 000236'                 BADADR: #BAD    ;+ADDRESS POINTER FOR ERRORM CALL
  209   000244' 000234'                 CRCER1: #GOODA  ;+ADDRESS OF EXPECTED CHECKSUM
  210   000246' 000242'                         #BADADR ;+ADDRESS OF ACTUAL   CHECKSUM
  211   000250' 177777                  -1              ;+MESSAGE TERMINATOR
```

```
                                ;
000252'                 START:
000252' 005767  177540  RESTRT: TST     SR1             ;ARE ANY ROMS TO BE CHECKED
000256' 001005                  BNE     2$              ;IF YES DO TEST
000260' 104403  000000' 000622'         MSGSWS,BEGIN,MSG1       ;ASCII MESSAGE CALL WITH COMMON HEADER
000266' 104410  000000'                 ENDS,BEGIN              ;
000272' 012767  000001  177732  2$:     MOV     #1,     CROMP   ;INITIALIZE CURRENT ROM POINTER
000300' 022767  000040  177724  3$:     CMP     #40,    CROMP   ;HAVE ALL ROMS BEEN TESTED
000306' 001405                  BEQ     4$              ;IF YES WE'RE DONE
000310' 004767  000014                  JSR     PC,     GETROM  ;GO GET ROM ADDRESSES
000314' 004767  000146                  JSR     PC,     CHECKR  ;GO TEST ROM
000320' 000767                  BR      3$              ;CONTINUE TESTING
000322'                         4$:
000322' 104413  000000'                 ENDITS,BEGIN            ;SIGNAL END OF ITERATION
                                                        ;MONITOR SHALL TEST END OF PASS
000326' 000761                  BR      2$

000330' 005067  177670  GETROM: CLR     FIRSTA          ;CLEAR FIRST CHECKSUM ADDRESS
000334' 036767  177672  177454  BIT     CROMP,  SR1     ;IS CURRENT ROM TO BE TESTED
000342' 001446                  BEQ     4$              ;IF NO THEN JUST EXIT
000344' 016700  177662          MOV     CROMP,  R0      ;GET CURRENT ROM POINTER TO WORK ON
000350' 000241                  CLC                     ;CARRY BIT TO BE USED AS INDICATOR
000352' 006000                  ROR     R0              ;IF C-BIT NOT SET THEN BOOT ROM
000354' 103014                  BCC     1$              ;BEING TEST SO GO SET UP FOR BOOT ROM
000356' 012767  165000  177640          MOV     #165000,FIRSTA  ;IF DIAG. ROM SET UP ITS FIRST ADDRESS
000364' 005067  177636          CLR     EXCADR          ;NO EXCEPTION ADDRESS IN DIAG.ROM
000370' 012767  165775  177632          MOV     #165775,LASTA   ;SET UP LAST ADDRESS
000376' 012767  165776  177630          MOV     #165776, GOODA  ;SET UP CHECKSUM ADDRESS OF DIAG. ROM
000404' 000425                  BR      4$              ;DIAG. ROM SET UP SO GO EXIT
000406' 012701  173000          1$:     MOV     #173000,R1      ;GET FIRST BOOT ROM ADDRESS FOR WORKING
000412' 006000                  2$:     ROR     R0              ;IF C-BIT SETS WE'VE FOUND OUR ROM
000414' 103403                  BCS     3$              ;SO GO SET UP FOR ADDRESS FOR IT
000416' 062701  000200          ADD     #200,   R1      ;IF C-BIT CLEAR UPDATE TO NEXT ROM
000422' 000773                  BR      2$              ;AND GO CHECK IF ITS THE ONE
000424' 010167  177574          3$:     MOV     R1,     FIRSTA  ;SET UP FIRST ADDRESS
000430' 062701  000024          ADD     #24,    R1      ;CALCULATE EXCEPTION ADDRESS
000434' 010167  177566          MOV     R1,     EXCADR  ;AND STORE IT
000440' 062701  000151          ADD     #151,   R1      ;CALCULATE LAST ADDRESS
000444' 010167  177560          MOV     R1,     LASTA   ;AND STORE IT
000450' 062701  000001          ADD     #1,     R1      ;CALCULATE GOOD CHECKSUM ADDRESS
000454' 010167  177554          MOV     R1,     GOODA   ;AND STORE IT
000460' 006167  177546          4$:     ROL     CROMP           ;UPDATE CROMP TO NEXT ROM
000464' 000207                  RTS     PC              ;RETURN

000466' 005767  177532  CHECKR: TST     FIRSTA          ;IS FIRST ADDRESS ZERO
000472' 001416                  BEQ     1$              ;IF YES DON'T TEST JUST EXIT
000474' 004767  000032          JSR     PC,     CALSUM  ;IF NOT GO CALCULATE CHECKSUM
000500' 027767  177530  177530  CMP     @GOODA, BAD     ;DO BOTH CHECKSUMS COMPARE
000506' 001410                  BEQ     1$              ;IF YES EXIT
000510' 104403  000000' 000626'         MSGSWS,BEGIN,MSG2       ;ASCII MESSAGE CALL WITH COMMON HEADER
000516' 005067  177364                  CLR     ERRTYP          ;
                                ;*****************************************************************
000522' 104405  000000' 000244'         HRDERS,BEGIN,CRCERR     ;CHECKSUM ERROR
                                ;*****************************************************************
```



```
000530' 000207          1$: RTS PC                      ;RETURN


000532' 016700  177466  CALSUM: MOV     FIRSTA, R0      ;GET STARTING ADDRESS
000536' 005005                  CLR     R5              ;CLEAR CRC WORD
000540' 012003          LOOP:   MOV     (R0)+,  R3      ;GET BYTE
000542' 012702  000020          MOV     #16.,   R2      ;SET BIT COUNT
000546' 000241          CRCLOP: CLC                     ;CLEAR CARRY
000550' 006005                  ROR     R5              ;ROTATE CRC WORD
000552' 006003                  ROR     R3              ;ROTATE BYTE
000554' 102006                  BVC     1$
000556' 012701  120001          MOV     #120001,R1
000562' 040501                  BIC     R5,     R1
000564' 042705  120001          BIC     #120001,R5
000570' 050105                  BIS     R1,     R5
000572' 005302          1$:     DEC     R2              ;GONE THROUGH HERE 16 TIMES
000574' 003364                  BGT     CRCLOP          ;IF NO DO IT AGAIN
000576' 020067  177424          CMP     R0,     EXCADR  ;IS NEXT ADDR. AN EXCEPTION ADDR.
000602' 001001                  BNE     2$              ;NO CONTINUE
000604' 005720                  TST     (R0)+           ;YES ADD 2 TO ADDR. TO SKIP IT
000606' 020067  177416  2$:     CMP     R0,     LASTA   ;HAVE WE SUM THE WHOLE ROM
000612' 101752                  BLOS    LOOP            ;IF NO CONTINUE WITH NEXT BYTE
000614' 010567  177416          MOV     R5,     BAD     ;IF YES STORE CALCULATED CHECKSUM
000620' 000207                  RTS     PC              ;AND EXIT


000622' 000632'         MSG1:   NOROMS
000624' 177777                  177777
000626' 000664'         MSG2:   CRCERR
000630' 177777                  177777

000632' 020045  047516  051040  NOROMS: .ASCIZ  "% NO ROMS SELECTED SR1=0%"
000640' 046517  020123  042523
000646' 042514  052103  042105
000654' 051440  030522  030075
000662' 000045
000664' 020045  044103  041505  CRCERR: .ASCIZ  "% CHECKSUM ERROR ON M9312 BOOTSTRAP%"
000672' 051513  046525  042440
000700' 051122  051117  047440
000706' 020116  034515  030463
000714' 020062  047502  052117
000722' 052123  040522  022520
000730'     000
000732'                 .EVEN
000001                  .END
```

```
ACSR    000102R          182#
ADDR    000006R          148#
ADDR22= 001000           200#
ASB     000106R          186#
ASTAT   000104R          184#
AWAS    000110R          187#
BAD     000236R          206#    208     261     290*
BADADR  000242R          208#    210
BEGIN   000000R          145#    216     217     225     263     266
BIT0    =  000001        200#
BIT1    =  000002        200#
BIT10   =  002000        200#
BIT11   =  004000        200#
BIT12   =  010000        200#
BIT13   =  020000        200#
BIT14   =  040000        200#
BIT15   =  100000        200#
BIT2    =  000004        200#
BIT3    =  000010        200#
BIT4    =  000020        200#
BIT5    =  000040        200#
BIT6    =  000100        200#
BIT7    =  000200        200#
BIT8    =  000400        200#
BIT9    =  001000        200#
BREAKS= 104407           200#
BR1     000012R          150#
BR2     000013R          151#
BTODS = 104421           200#
CALSUM  000532R          260     271#
CDATAS= 104412           200#
CHECKR  000466R          222     258#
CONFIG  000058R          170#
COUNT   000240R          207#
CRCERR  000664R          296     304#
CRCERI  000244R          209#    266
CRCLOP  000545R          275#    284
CROMP   000232R          204#    218*    219     231     233     254*
CSRA    000100R          180#
DATCKS= 104411           200#
DATERS= 104404           200#
DVID1   000014R          152#
EMDITS= 104413           200#    225
ENDS    = 104410         200#    217
ERRTYP  000106R          185#    264*
EXCADR  000226R          202#    238*    249*    285
EXITS = 104400           200#
FIRSTA  000224R          201#    230*    237*    247*    258     271
GETPAS= 104415           200#
GETROM  000330R          221     230#
GOODA   000234R          205#    209     240*    253*    261
GWBUFS= 104414           200#
HRDCNT  000044R          165#
HRDERS= 104405           200#    266
HRDPAS  000050R          167#
ICONT   000036R          162#
```



```
ICOUNT  000040R          163#
IDNUM   000122R          192#
INIT    000030R          159#
INTR    000120R          191#
LASTA   000230R          203#    239*    251*    288
LOOP    000540R          273#    289
MAP22S= 104416           200#
MDNAM   000000R          146#
MODSP   000224R          160     198#
MSGMS = 104403           200#    216     263
MSGSS = 104402           200#
MSGS  = 104401           200#
MSG1    000622R          216     294#
MSG2    000626R          263     296#
NOROMS  000632R          294     299#
NULL  = 000000           200#
OPEN  = 000000           147     153     154     155     156     173     174     175     176     177     178     179     180
                         182     184     186     187     189     190     191     200#
OTOAS = 104420           200#
PASCNT  000034R          161#
PIRQS = 000004           200#
POPSP = 005726           200#
POPSP2= 022626           200#
PRTY  = 000000           200#
PRTY0 = 000000           150     151     200#
PRTY1 = 000040           200#
PRTY2 = 000100           200#
PRTY3 = 000140           200#
PRTY4 = 000200           200#
PRTY5 = 000240           200#
PRTY6 = 000300           200#
PRTY7 = 000340           200#
PS    = 177776           200#
PSW   = 177776           200#
PUSH  = 005746           200#
PUSH2 = 024646           200#
RANDS = 104417           200#
RANNUM  000054R          169#
RESTRT  000252R          188     214#
RES1    000056R          171#
RES2    000060R          172#
RSTRT   000112R          188#
SMADR   000102R          181#
SOFCNT  000042R          164#
SOFERS= 104406           200#
SOFPAS  000046R          166#
SPOINT  000032R          160#
SPSIZ = 000040           161#    193
SR1     000016R          153#    214     231
SR2     000020R          154#
SR3     000022R          155#
SR4     000024R          156#
START   000252R          159     213#
STAT    000026R          158#
SVR0    000062R          173#
SVR1    000064R          174#
```

```
SVR2     000066R          175#
SVR3     000070R          176#
SVR4     000072R          177#
SVR5     000074R          178#
SVR6     000076R          179#
SYSCNT   000052R          168#
TRPDFD=  000022           200#
VECTOR   000010R          149#
WASADR   000104R          183#
WDFR     000116R          190#
WDTO     000114R          189#
XFLAG    000005R          147#
.      = 000732R          311#

. ABS.   000000     000
         000732     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 X8MHB0,X8MHB0/SOL/CRF:SYM=DDXCOM,X8MHB0
 RUN-TIME:  1  1.2 SECONDS
 RUN-TIME RATIO: 24/2=8.6
 CORE USED:  7K  (13 PAGES)
```

```
000000                          .REPT   0
```

IDENTIFICATION

PRODUCT CODE: AC-F084B-MC

PRODUCT NAME: CXPLAB0 PCL11 MODULE

PRODUCT DATE: FEB 1979

MAINTAINER: DAVE WIENS, CSS KANATA

```
                        1.      ABSTRACT
                                --------

                                PLA IS AN IOMODX ROUTINE THAT EXERCISES 1 (ONE) PCL11
                                ATTACHED TO THE PDP-11 SYSTEM UNIBUS. IT EXERCISES PCL11
                                BY CAUSING THE TRANSMITTER IN THE UNIT TO TRANSMIT A FULL
                                SILO (64. WORDS) TO THE RECEIVER IN THE SAME UNIT. DATA
                                IS OBTAINED FOR THE TRANSMISSION VIA NPR AND READ BACK
                                VIA NPR. DATA IS CHECKED IN CORE BY THE DEC/X11 MONITOR.
                                ALL ERRORS ARE REPORTED ON THE CONSOLE PRINT DEVICE.

                        2.      REQUIREMENTS
                                ------------

                                HARDWARE:       PDP-11 WITH PCL11 ON THE UNIBUS
                                                TDM BUS CABLE DETACHED FROM UNIT
                                                UNDER TEST.

                                STORAGE:: PLA REQUIRES:
                                                1. DECIMAL WORDS: 810
                                                2. OCTAL WORDS: 1452
                                                3. OCTAL BYTES: 3124


                                OTHER:          THIS MODULE IS MEANT TO BE CONFIGURED
                                                AND RUN WITH DEC/X11 ONLY.

                        3.      PASS DEFINITION
                                ---------------

                                ONE PASS OF THE PLA MODULE CONSISTS OF 3072 CYCLES
                                OF THE BASIC TEST SEQUENCE: TRANSMIT 64 WORDS TO THE
                                RECEIVER, CHECK RCV'D WORDS FOR DATA ERRORS, TRAP ALL
                                HARDWARE ERRORS. ONE PASS TAKES APPROXIMATELY 30 SECONDS.

                        4.      CONFIGURATION REQUIREMENTS
                                --------------------------

                                DEFAULT PARAMETERS:
                                        DEVICE ADDRESS: 164200
                                        VECTOR:         170
                                        PRIORITY (BR1)  5

                                REQUIRED PARAMETERS:
                                        SR1:            MUST CONTAIN PCL11 RCVR
                                                        TDM-BUS ADDRESS.
```

```
                        5.      DEVICE / OPTION SETUP
                                ---------------------

                        SINCE PCL11 IS AN INTER-PROCESSOR COMMUNICATIONS DEVICE,
                        MAKE CERTAIN THAT THE UNIT IS DISCONNECTED FROM THE TDM BUS
                        TO OTHER PROCESSORS.
                          ALSO, IT IS VITAL THAT THE RECEIVERS TDM-BUS ADDRESS BE
                        KNOWN SO THAT IT MAY BE ENTERED INTO "SR1" AT CONFIGURE TIME.

                        7.      MODULE OPERATION
                                ----------------

                        TEST SEQUENCE:

                        A.  SET UP DEVICE REGISTER ADDRESSES AND MODULE VARIABLES
                        B.  PREFORM VALIDITY CHECK ON HARDWARE TO INSURE THAT THE
                            MODULE WILL NOT GET "HUNG UP"
                        C.  GET WRITE AND READ INFORMATION FOR NPR
                        D.  ENABLE XMTR AND RCVR NPR - START TRANSMISSION.
                        E.  CHECK FOR XMTR & RCVR ERRORS - REPORT ANY AND RETRY
                            UP TO RETRY LIMIT. MODULE GETS DROPPED IF 10 (OCTAL)
                            ERRORS OCCUR IN ONE CYCLE AND SR1 IS POSITIVE (BIT 15 CLEAR).
                        F.  IF END OF PASS, REPORT AND GO TO C

                        7.      OPERATION OPTIONS
                                -----------------

                        SR1 BIT15 SET (1)
                                IF ERROR RETRY LIMIT IS EXCEEDED, RESET RETRY
                                LIMIT AND CONTINUE.
                        SR1 BIT15 CLEAR (0)
                                IF ERROR RETRY LIMIT IS EXCEEDED, A HARD ERROR
                                IS ASSUMED AND THE MODULE IS DROPPED.

                                        NOTE
                                        ----

                                THE RETRY COUNT IS CLEARED EVERY CYCLE UNLESS
                                ERRORS OCCUR.

                        SR1 BITS<7:0>
                                MUST CONTAIN THE CORRECT RECEIVER TDM BUS ADDRESS.
                                (COMMONLY CALLED RECEIVER NUMBER)
                                THIS IS AN OCTAL NUMBER BETWEEN 1 AND 37.
```

```
                        8.      ERROR MESSAGES
                                --------------

                                A.  THE STANDARD DEC/X11 ERROR ROUTINES ARE USED IN
                                    THIS MODULE.
                                B.  FURTHER PRINTOUTS ARE SELF- EXPLANITORY.
                                C.  ERROR MESSAGES DUMP THE CONTENTS OF THE DEVICE
                                    REGISTERS IN THE FOLLOWING ORDER:
                                RECEIVER:

                                RCR   RSR   RDDB  RDBC  RDBA  RDCRC

                                TRANSMITTER:

                                TCR   TSR   TSDB  TSBC  TSBA  TMMR  TSCRC
```

```
                                        .ENDR
                                ;PCL11 - DEC/X11 SYSTEM EXERCISER MODULE
                                
000000'                         IOMODX  <PLAB >,164200,170,5,0,0,3072.,0,BUFIN,64.,64.
000000'                                 MODULE  150000,PLAB ,164200,170,5,0,0,3072.,0,BUFIN,64.,64.
                                        .TITLE  PLAB DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6      23-MAY-78
                                        .LIST   BIN
                                ;******************************************************************************
000000'                         BEGIN:
000000' 046120  041101      040 MODNAM: .ASCII  /PLAB / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 164200                  ADDR:   164200+0                ;1ST DEVICE ADDR.
000010' 000170                  VECTOR: 170+0                   ;1ST DEVICE VECTOR.
000012'     240                 BR1:    .BYTE   PRTY5+0         ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;******************************************************************************
000026' 150000                  STAT:   150000                  ;STATUS WORD.
000030' 000252'                 INIT:   START                   ;MODULE START ADDR.
000032' 000252'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                       ;PASS COUNTER.
000036' 006000                  ICONT:  3072.                   ;# OF ITERATIONS PER PASS=3072.
000040' 000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                         ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000102'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000104'                         WASADR:                         ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000106'                         ERRTYP:                         ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000112' 000270'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
```



```
000122' 000000                  IDNUM:  0                       ;MODULE IDENTIFICATION NUMBER=0
000124' 002230'                 RBUFVA: BUFIN                   ;READ BUFFER VIRTUAL ADDRESS
000126' 000000                  RBUFPA: OPEN                    ;READ BUFFER PHYSICAL ADDRESS
000130' 000000                  RBUFEA: OPEN                    ;READ BUFFER EA BITS
000132' 000100                  RBUFSZ: 64.                     ;SIZE OF THE READ BUFFER
000134' 000000                  WBUFPA: OPEN                    ;WRITE BUFFER PHYSICAL ADDRESS
000136' 000000                  WBUFEA: OPEN                    ;WRITE BUFFER EA BITS
000140' 000100                  WBUFRQ: 64.                     ;WRITE BUFFER SIZE REQUESTED
000142' 000000                  WBUFSZ: OPEN                    ;WRITE BUFFER SIZE AVAILABLE
000144' 000000                  CDERCT: OPEN                    ;CDATA/DATCK ERROR COUNT
000146' 000000                  CDWDCT: OPEN                    ;CDATA/DATCK WORD COUNT
000150' 000000                  FREE:   OPEN                    ;RESERVED FOR FUTURE USE
          000040                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000252'                         MODSP:
                                ;******************************************************************************
                                ;SOME EXTRA DEFINITIONS:

                                ;TRANSMIT START FUNCTION:

          060101                TXMSTR  =       60101

                                ;RECEIVE START FUNCTION:

          060001                RCVSTR  =       60001
```

```
000252' 004767  001126          START:  JSR     PC,SETUP                ;GENERATE DEVICE ADDRESSES
000256' 004767  001322                  JSR     PC,TSTDRP               ;SET MASTER, DROP MODULE?
000262' 105767  001734                  TSTB    DEVICE                  ;IF SET, DROP MODULE
000266' 001045                          BNE     FINI
000270'                         RESTRT:
000270' 104415  000000' 000124'         GETPA$,BEGIN, RBUFVA            ;GET PHYSICAL ADDRESS FROM 16-BIT RBUFVA
000276' 016767  177630  001702          MOV     RBUFSZ,RCBC             ;GET RCVR BUFF SIZE
000304' 006367  001676                  ASL     RCBC                    ;DOUBLE IT FOR BYTE COUNT
000310' 005467  001672                  NEG     RCBC                    ;NEGATE BYTE COUNT
000314'                         BCGW:
000314' 104414  000000'                 GWBUF$, BEGIN           ;GET WRITE BUFFER INFORMATION
000320' 016767  177616  001662          MOV     #BUFSZ,TXBC             ;GET ALLOCATED BUFFER SIZE
000326' 006367  001656                  ASL     TXBC                    ;DOUBLE IT FOR BYTE COUNT
000332' 005467  001652                  NEG     TXBC                    ;NEGATE BYTE COUNT
000336' 105067  001662          BCOK:   CLRB    ERFLG                   ;CLEAR ERRORS FLAG
000342' 004767  000122                  JSR     PC,TRNSFR               ;SEND SOME DATA & RCV IT
000346' 105767  001652                  TSTB    ERFLG                   ;IF ERRORS, RETRY UP TO 10 TIMES
000352' 001023                          BNE     RETRY
000354' 004767  001006                  JSR     PC,ERSUB2               ;LOAD ERROR INFORMATION
000360' 104412  000000' 000126'         CDATA$,BEGIN,RBUFPA     ; REQUEST FOR MONITOR TO CHECK DATA
000366' 000370'                         .+2                     ; IF ERROR, CONTINUE
000370' 105067  001627                  CLRB    TRY                     ;REFRESH RETRY COUNT
000374'                         PASS:
000374' 104413  000000'                 ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                ;MONITOR SHALL TEST END OF PASS
000400' 000745                          BR      BCGW
000402' 052777  000002  002042  FINI:   BIS     #BIT1,@RCR              ;CLEAR RCVR HDWARE
000410' 052777  000002  002012          BIS     #BIT1,@TCR              ;CLEAR XMTR HDWARE
000416' 104410  000000'                 END$,BEGIN              ;
```



```
000422'                         RETRY:                                  ;ERROR RETRY RTN ENTRY POINT
000422' 105267  001575                  INCB    TRY                     ;COUNT THE RETRYS
000426' 122767  000010  001567          CMPB    #10,TRY                 ;LIMIT EXCEEDED?
000434' 001340                          BNE     BCOK                    ;NO,GO TRY AGAIN
000436' 104403  000000' 002474'         MSGN$,BEGIN,EXCED       ;ASCII MESSAGE CALL WITH COMMON HEADER
000444' 012777  000002  002000          MOV     #BIT1,@RCR              ;CLEAR RCVR HDWARE
000452' 012777  000002  001750          MOV     #BIT1,@TCR              ;CLEAR XMTR HDWARE
000460' 005767  177332                  TST     SR1                     ;DROP THE MODULE?
000464' 100724                          BMI     BCOK                    ;NOT IF SR1 BIT15=1
000466' 000745                          BR      FINI                    ;YES,IF SR1 BIT15=0

                                ;PCL DRIVE SUBROUTINE
                                ;CALLED BY: JSR PC,TRNSFR

                                ;TRANSFER 64 WORDS TO XMTR VIA NPR
                                ;TRANSMIT IT TO THE RECEIVER VIA TDM BUS
                                ;TRANSFER RCV' DATA TO MODULE (BUFIN) VIA NPR
                                ;WAIT FOR COMPLETION FROM XMTR & RCVR

000470'                         TRNSFR:                                 ;DRIVER SUBROUTINE ENTRY
000470' 012777  000002  001754          MOV     #BIT1,@RCR              ;CLR RCVR HDWARE
000476' 012777  000002  001724          MOV     #BIT1,@TCR              ;CLR XMTR HDWARE
000504' 016777  001476  001746          MOV     RCBC,@RDBC              ;LOAD RCVR BYTE COUNT
000512' 016777  001472  001716          MOV     TXBC,@TSBC              ;LOAD XMTR BYTE COUNT
000520' 016777  177402  001734          MOV     RBUFPA,@RDBA            ;LOAD RCVR NPR DEST ADDRESS
000526' 016767  177376  001462          MOV     RBUFEA,RFUNCT           ;AND GET EXT ADDR BITS
000534' 016777  177374  001676          MOV     WBUFPA,@TSBA            ;LOAD XMTR NPR SOURCE ADDRESS
000542' 016767  177370  001450          MOV     WBUFEA,XFUNCT           ;AND GET EXT ADDR BITS
000550' 116767  177242  001443          MOVB    SR1,XFUNCT+1            ;GET RCVR ADDRESS FOR XMTR
000556' 052767  060001  001432          BIS     #RCVSTR,RFUNCT          ;LOAD RCV FUNCTION AND GO
000564' 052767  060101  001426          BIS     #TXMSTR,XFUNCT          ;LOAD XMTR FUNCTION AND GO
000572' 012777  000620' 001672          MOV     #TXINT,@TXVECT          ;SET XMTR INTR ENTRY POINT
000600' 016777  001412  001644          MOV     RFUNCT,@RCR             ;CONNECT RCVR FIRST
000606' 016777  001406  001614          MOV     XFUNCT,@TCR             ;NOW CONNECT XMTR
000614' 104400  000000'                 EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
```

```
                                        ;TRANSMITTER INTERRUPT ENTRY POINT
                                        ;
000620'                                 TXINT:                          ;XMTR INTR ENTRY POINT
000620' 042777  000100  001602          BIC     #BIT6,@TCR              ;CLEAR XMTR INTR ENABLE
                                        ;---------------------------------------------------------------
000626' 000004  000000' 000634'         PIRQ$,BEGIN,11$         ; QUEUE UP TO CONTINUE AT 11$ AND RTI
                                        ;---------------------------------------------------------------
                                        ;
000634' 004767  000056          11$:    JSR     PC,TXERS                ;GO CHECK FOR XMTR ERRORS
000640' 105767  001362                  TSTB    TXMERS                  ;IF ANY ERRORS, SET ERFLG
000644' 001015                          BNE     24$
000646' 004767  000314          12$:    JSR     PC,RCERS                ;GO CHECK FOR RCVR ERRORS
000652' 105767  001347                  TSTB    RCVERS                  ;IF ERRORS, SET ERFLG
000656' 001001                          BNE     23$
000660' 000207                  13$:    RTS     PC                      ;RETURN
                                        ;
000662' 105067  001337          23$:    CLRB    RCVERS                  ;CLR RCVR ERR FLG
000666' 112767  177777  001330          MOVB    #-1,ERFLG               ;SET ERROR FLAG
000674' 000167  177760                  JMP     13$
000700' 105067  001322          24$:    CLRB    TXMERS                  ;CLR XMTR ERR FLG
000704' 112767  177777  001312          MOVB    #-1,ERFLG               ;SET ERROR FLAG
000712' 000167  177730                  JMP     12$
```



```
                                        ;CHECK FOR TRANSMITTER ERRORS AND REPORT, IF ANY
                                        ;
000716'                                 TXERS:                          ;ENTRY POINT FOR XMTR ERR CHK
000716' 112767  177777  001302          MOVB    #-1,TXMERS              ;SET XMTR EROR FLAG
000724' 032777  100000  001500          BIT     #BIT15,@TSR             ;HARDWARE ERROR?
000732' 001430                          BEQ     10$                     ;NO, CHECK FURTHER
000734' 032777  004000  001470          BIT     #BIT11,@TSR             ;WAS IT MASTER DOWN?
000742' 001412                          BEQ     2$                      ;NO, REPORT WHAT IT WAS
000744' 104403  000000' 002500'         MSGN$,BEGIN,MSTDWN      ;ASCII MESSAGE CALL WITH COMMON HEADER
000752' 004767  000626                  JSR     PC,TSTDRP               ;GO SEE IF WE CAN KEEP MODULE
000756' 105767  001240                  TSTB    DEVICE                  ;OK?
000762' 001474                          BEQ     1XRRTN                  ;YES, TRY AGAIN
000764' 000167  177412                  JMP     FIN1                    ;NO, DROP THE MODULE BECAUSE
                                                                        ;THE HARDWARE IS NON-RUNABLE
000770' 004767  000354          2$:     JSR     PC,ERSUB1               ;LOAD ERROR INFORMATION
000774' 012767  000020  177104          MOV     #20,ERRTYP              ;UNKNOWN XMTR ERROR
                                        ;***************************************************************
001002' 104405  000000' 002430'         HRDER$,BEGIN,TABLE1     ;XMTR HARDWARE ERROR
                                        ;***************************************************************
001010' 000167  000140                  JMP     TXRRTN                  ;TRY AGAIN
001014' 032777  000020  001410  10$:    BIT     #BIT4,@TSR              ;WAS RCVR BUSY OR NOT THERE?
001022' 001415                          BEQ     3$                      ;NO, CONTINUE CHECK
001024' 104403  000000' 002504'         MSGN$,BEGIN,WRCAD       ;ASCII MESSAGE CALL WITH COMMON HEADER
001032' 004767  000312                  JSR     PC,ERSUB1               ;GET ERROR INFORMATION
001036' 012767  000007  177042          MOV     #7,ERRTYP               ;SELECTION ERROR
                                        ;***************************************************************
001044' 104406  000000' 002430'         SOFER$,BEGIN,TABLE1     ;RCVR BUSY OR WRONG ADDR
                                        ;***************************************************************
001052' 000167  000076                  JMP     TXRRTN                  ;TRY AGAIN
001056' 132777  000004  001364  3$:     BITB    #BIT2,@TMMRH            ;IS NOW MASTER CAUSING INTERRUPT?
001064' 001410                          BEQ     4$                      ;NO, CONTINUE CHECK
001066' 142777  000004  001354          BICB    #BIT2,@TMMRH            ;YES,CLEAR IT
001074' 104403  000000' 002510'         MSGN$,BEGIN,NMSINT      ;ASCII MESSAGE CALL WITH COMMON HEADER
001102' 000167  000046                  JMP     TXRRTN                  ;TRY AGAIN
001106' 032777  000040  001316  4$:     BIT     #BIT5,@TSR              ;WAS MESSAGE REJECTED?
001114' 001405                          BEQ     5$                      ;NO, CONTINUE CHECK
001116' 104403  000000' 002514'         MSGN$,BEGIN,MRJCT       ;ASCII MESSAGE CALL WITH COMMON HEADER
001124' 000167  000024                  JMP     TXRRTN                  ;TRY AGAIN
001130' 032777  000200  001274  5$:     BIT     #BIT7,@TSR              ;SUCCESSFUL TRANSFER
001136' 001007                          BNE     TXOK                    ;YES, OK LEAVE
001140' 012767  000011  176740          MOV     #11,ERRTYP              ;ILLEGAL INTR OR DONE CLR
                                        ;***************************************************************
001146' 104406  000000' 002430'         SOFER$,BEGIN,TABLE1     ;UNKNOWN INTERRUPT
                                        ;***************************************************************
001154' 000207                  TXRRTN: RTS     PC                      ;RETURN TO CALLER
                                        ;
001156' 105067  001044          TXOK:   CLRB    TXMERS                  ;NO ERRORS, SKIP RETRY
001162' 000167  177766                  JMP     TXRRTN
```

```
                                        ;CHECK FOR RECEIVER ERRORS AND REPORT, IF ANY.

001166'                                 RCERS:                          ;ENTRY POINT FOR RCVR ERR CHK
001166' 112767  177777  001031          MOVB    #-1,RCVERS              ;SET RCVR ERROR FLAG
001174' 032777  100000  001252          BIT     #BIT15,@RSR             ;HARDWARE ERROR?
001202' 001412                          BEQ     10$                     ;NO,CHECK FURTHER
001204' 004767  000156                  JSR     PC,ERSUB2               ;LOAD ERROR INFORMATION
001210' 012767  000017  176670          MOV     #17,ERRTYP              ;UNKNOWN RECEIVER ERROR
                                        ;*****************************************************************
001216' 104405  000000' 002452'         HRDER$,BEGIN,TABLE2     ;RCVR HARDWARE ERROR
                                        ;*****************************************************************
001224' 000167  000106                  JMP     RCRRTN                  ;ERROR RETURN
001230' 032777  000400  001216  10$:    BIT     #BIT8,@RSR              ;DATA OUTPUT RDY SET?
001236' 001415                          BEQ     3$                      ;NO,CONTINUE CHECK
001240' 104403  000000' 002522'         MSGN$,BEGIN,ERDOPR      ;ASCII MESSAGE CALL WITH COMMON HEADER
001246' 004767  000114                  JSR     PC,ERSUB2               ;LOAD ERROR INFORMATION
001252' 012767  000044  176626          MOV     #44,ERRTYP              ;FLAG SHOULD NOT BE SET
                                        ;*****************************************************************
001260' 104406  000000' 002452'         SOFER$,BEGIN,TABLE2     ;DAT OUT RDY SET
                                        ;*****************************************************************
001266' 000167  000044                  JMP     RCRRTN                  ;ERROR RETURN
001272' 032777  000040  001154  3$:     BIT     #BIT5,@RSR              ;REJECT COMPLETED INTERRUPT?
001300' 001415                          BEQ     4$                      ;NO, CONTINUE CHECK
001302' 104403  000000' 002530'         MSGN$,BEGIN,MRJTD       ;ASCII MESSAGE CALL WITH COMMON HEADER
001310' 004767  000052                  JSR     PC,ERSUB2               ;LOAD ERROR INFORMATION
001314' 012767  000044  176564          MOV     #44,ERRTYP              ;FLAG SHOULD NOT BE SET
                                        ;*****************************************************************
001322' 104406  000000' 002452'         SOFER$,BEGIN,TABLE2     ;REJ COMPL SET
                                        ;*****************************************************************
001330' 000167  000002                  JMP     RCRRTN                  ;ERROR RETURN
001334' 000401                  4$:     BR      RCOK                    ;SHOULDN'T BE HERE, LEAVE.
001336' 000207                  RCRRTN: RTS     PC                      ;RETURN TO CALLER

001340' 105067  000661          RCOK:   CLRB    RCVERS                  ;NO ERROKS, SKIP RETRY
001344' 000167  177766                  JMP     RCRRTN
```



```
001350'                                 ERSUB1:                         ;RTN TO LD XMTR ERR INFO
001350' 016767  001056  176522                  MOV     TSR,CSRA        ;LOAD ADDR OF XMTR STATUS REG
001356' 017767  001050  176516                  MOV     @TSR,ACSR       ;LOAD CONTENTS OF XMTR STAT REG
001364' 000207                                  RTS     PC              ;RETURN

001366'                                 ERSUB2:                         ;ROUTINE TO LD RCVR ERR INFO
001366' 016767  001062  176504                  MOV     RSR,CSRA        ;LOAD ADDR OF RCVR STAT REG
001374' 017767  001054  176500                  MOV     @RSR,ACSR       ;LD CONTENTS OF RCVR STAT REG
001402' 000207                                  RTS     PC              ;RETURN


                                        ;ROUTINE TO GENERATE DEVICE ADDRSSES AND VECTORS

001404'                                 SETUP:                          ;SETUP ROUTINE ENTRY POINT
001404' 016700  176376                          MOV     ADDR,R0         ;GET DEVICE BASIC ADDRESS
001410' 010067  001014                          MOV     R0,ICR          ;ICR ADDRESS
001414' 005720                                  TST     (R0)+
001416' 010067  001010                          MOV     R0,TSR          ;TSR ADDRESS
001422' 005720                                  TST     (R0)+
001424' 010067  001004                          MOV     R0,TSDB         ;TSDB ADDRESS
001430' 005720                                  TST     (R0)+
001432' 010067  001000                          MOV     R0,TSBC         ;TSBC ADDRESS
001436' 005720                                  TST     (R0)+
001440' 010067  000774                          MOV     R0,TSBA         ;TSBA ADDRESS
001444' 005720                                  TST     (R0)+
001446' 010067  000770                          MOV     R0,TMMR         ;TMMR ADDRESS
001452' 005200                                  INC     R0
001454' 010067  000770                          MOV     R0,TMMRH        ;TMMR HIGH BYTE
001460' 005200                                  INC     R0
001462' 010067  000756                          MOV     R0,TSCRC        ;TSCRC ADDRESS
001466' 016700  176316                          MOV     VECTOR,R0       ;GET BASIC VECTOR
001472' 010067  000774                          MOV     R0,TXVECT       ;SAVE IT
001476' 012720  000620'                         MOV     #IXINT,(R0)+    ;SET INTR POINTER IN CASE.
001502' 116710  176304                          MOVB    BR1,(R0)        ;SET XMTR PRIORITY
001506' 005720                                  TST     (R0)+
001510' 010067  000754                          MOV     R0,RCVECT       ;SAVE RCVR VECTOR
001514' 062700  000002                          ADD     #2,R0
001520' 116710  176266                          MOVB    BR1,(R0)
```

```
447  001524' 016700  176256          MOV     ADDR,R0                 ;GET DEVICE BASIC ADDRESS
448  001530' 062700  000020          ADD     #20,R0                  ;ADD OFFSET FOR RCVR ADDRS
449  001534' 010067  000712          MOV     R0,RCR                  ;RCR ADDRESS
450  001540' 005720                  TST     (R0)+
451  001542' 010067  000706          MOV     R0,RSR                  ;RSR ADDRESS
452  001546' 005720                  TST     (R0)+
453  001550' 010067  000702          MOV     R0,RDDB                 ;RDDB ADDRESS
454  001554' 005720                  TST     (R0)+
455  001556' 010067  000676          MOV     R0,RDBC                 ;RDBC ADDRESS
456  001562' 005720                  TST     (R0)+
457  001564' 010067  000672          MOV     R0,RDBA                 ;RDBA ADDRESS
458  001570' 022020                  CMP     (R0)+,(R0)+
459  001572' 010067  000666          MOV     R0,RDCRC                ;RDCRC ADDRESS
460  001576' 105067  000421          CLRB    TRY                     ;CLEAR RETRY COUNTER
461  001602' 000207                  RTS     PC                      ;RETURN
```



```
462                                     ;THIS ROUTINE TESTS IF THE DEVICE IS WORTHY OF RUNNING WITH
463                                     ;DEC/X11 MODULE.
464                                     ;  IT CHECKS THAT MASTER IS SET, (IF NOT, THAT IT CAN BE SET)
465                                     ;THEN DOES A SIMPLE TEST TO SEE IF THE DEVICE CAN INTERRUPT
466                                     ;AND GO
467                                     ;IF THIS TEST FAILS, THE MODULE WILL BE DROPPED.
468
469  001604'                            TSTDRP:                                 ;DROP TEST ROUTINE ENTRY
470  001604' 052777  000002  000616              BIS     #BIT1,@TCR              ;CLEAR XMTR HDWARE
471  001612' 052777  000002  000632              BIS     #BIT1,@RCR              ;CLEAR RCVR HDWARE
472  001620' 105067  000376                      CLRB    DEVICE                  ;CLEAR HARD ERROR FLAG
473  001624' 105077  000620                      CLRB    @TMMRH                  ;CLEAR MASTER, AUTO ADDR ETC.
474  001630' 112777  000021  000612              MOVB    #21,@TMMRH              ;SET MASTER & AUTO ADDR
475  001636' 132777  000021  000604              BITB    #21,@TMMRH              ;ARE THEY SET?
476  001644' 001016                              BNE     1$                      ;YES, NEXT TEST
477  001646' 112767  177777  000346              MOVB    #-1,DEVICE              ;NO,SET DEVICE TO DROP MODULE
478  001654' 104403  000000' 002536'             MSGNS,BEGIN,MSHDER      ;ASCII MESSAGE CALL WITH COMMON HEADER
479  001662' 012767  000042  176216              MOV     #42,ERRTYP              ;ACTIVE BIT SHD BE SET.
480                                              ;*****************************************************************
481  001670' 104405  000000' 002430'             HRDERS,BEGIN,TABLE1     ;MASTER OR AUTO ADDR CLR
482                                              ;*****************************************************************
483  001676' 000167  000266                      JMP     TSTRTN                  ;EXIT
484  001702' 012777  120000  000520  1$:         MOV     #120000,@TCR            ;SET RIB & SND WD
485  001710' 012777  177777  000516              MOV     #-1,@TSDB               ;PUT A WORD INTO SILO
486  001716' 012767  177600  000270              MOV     #-200,CLK               ;SET UP TO WAIT A BIT
487  001724'                         2$:
488  001724' 104407  000000'                     BREAKS,BEGIN            ;TEMPORARY RETURN TO MONITOR....
489  001730' 104407  000000'                     BREAKS,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
490  001734' 032777  000020  000470              BIT     #BIT4,@TSR              ;TDM BUS BUSY SET?
491  001742' 001016                              BNE     3$                      ;YES, NEXT TEST
492  001744' 005267  000244                      INC     CLK                     ;NO, WAIT A WHILE
493  001750' 001365                              BNE     2$
494  001752' 112767  177777  000242              MOVB    #-1,DEVICE              ;SET DEVICE TO DROP MODULE
495  001760' 012767  000006  176120              MOV     #6,ERRTYP               ;DEVICE WON'T GO.
496                                              ;*****************************************************************
497  001766' 104405  000000' 002430'             HRDERS,BEGIN,TABLE1     ;XMTR WON'T GO
498                                              ;*****************************************************************
499  001774' 000167  000170                      JMP     TSTRTN                  ;EXIT
500  002000' 052777  010000  000424  3$:         BIS     #BIT12,@TSR             ;CAUSE FORCED TXM ERR
501  002006' 012767  177600  000200              MOV     #-200,CLK               ;SET UP TO WAIT A BIT
502  002014'                         4$:
503  002014' 104407  000000'                     BREAKS,BEGIN            ;TEMPORARY RETURN TO MONITOR....
504  002020' 104407  000000'                     BREAKS,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
505  002024' 032777  020000  000376              BIT     #BIT13,@TCR             ;IS SND WD CLEAR NOW?
506  002032' 001416                              BEQ     5$                      ;YES, XMTR LOOKS OK
507  002034' 005267  000154                      INC     CLK                     ;WAIT & GIVE IT A CHANCE
508  002040' 001365                              BNE     4$
509  002042' 112767  177777  000152              MOVB    #-1,DEVICE              ;SET DEVICE TO DROP MODULE
510  002050' 012767  000023  176030              MOV     #23,ERRTYP              ;DEVICE FAILED TO INTERRUPT
511                                              ;*****************************************************************
512  002056' 104405  000000' 002430'             HRDERS,BEGIN,TABLE1     ;XMTR WON'T INTERRUPT
513                                              ;*****************************************************************
514  002064' 000167  000100                      JMP     TSTRTN                  ;EXIT
```

```
   515  002070' 012777  000002  000332  5$:     MOV     #BIT1,@TCR              ;CLEAR XMTR HDWARE
   516  002076' 052777  020000  000346          BIS     #BIT13,@RCR             ;SET RCV WD IN RCVR
   517  002104' 052777  010000  000342          BIS     #BIT12,@RSR             ;CAUSE FAKE TXM ERR IN RCVR
   518  002112' 012767  177600  000074          MOV     #-200,CLK               ;SET UP TO WAIT A BIT
   519  002120'                                 6$:
   520  002120' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
   521  002124' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
   522  002130' 032777  020000  000314          BIT     #BIT13,@RCR             ;IS RCV WD CLEAR NOW?
   523  002136' 001414                          BEQ     TSTRTN                  ;YES, RCVR LOOKS GOOD TOO
   524  002140' 005267  000050                  INC     CLK                     ;NO,WAIT & GIVE IT A CHANCE
   525  002144' 001365                          BNE     6$
   526  002146' 112767  177777  000046          MOVB    #-1,DEVICE              ;SET DEVICE TO DROP MODULE
   527  002154' 012767  000023  175724          MOV     #23,ERRTYP              ;DEVICE FAILED TO INTERRUPT
   528                                          ;*************************************************************
   529  002162' 104405  000000' 002452'         HRDERR$,BEGIN,TABLE2    ;RCVR WON'T INTERRUPT
   530                                          ;*************************************************************
   531  002170' 052777  000002  000254  TSTRTN: BIS     #BIT1,@RCR              ;CLEAR RCVR HDWARE
   532  002176' 052777  000002  000224          BIS     #BIT1,@TCR              ;CLEAR XMTR HDWARE
   533  002204' 000207                          RTS     PC                      ;RETURN
```



```
   534                                  ;CONSTANT AND VARIABLE STORAGE
   535
   536  002206' 000000                  RCBC:   .WORD   0
   537  002210' 000000                  TXBC:   .WORD   0
   538  002212' 000000                  CNT:    .WORD   0
   539  002214' 000000                  CLK:    .WORD   0
   540  002216' 000000                  RFUNCT: .WORD   0
   541  002220' 000000                  XFUNCT: .WORD   0
   542  002222'    000                  DEVICE: .BYTE   0
   543  002223'    000                  TRY:    .BYTE   0
   544  002224'    000                  ERFLG:  .BYTE   0
   545  002225'    000                  RCVERS: .BYTE   0
   546  002226'    000                  TXMERS: .BYTE   0
   547         002230'                  .EVEN
   548  002230' 000100                  BUFIN:  .BLKW   64.
   549
   550  002430'                         TABLE1:
   551  002430' 000000                  TCR:    .WORD   0
   552  002432' 000000                  TSR:    .WORD   0
   553  002434' 000000                  TSDB:   .WORD   0
   554  002436' 000000                  TSBC:   .WORD   0
   555  002440' 000000                  TSBA:   .WORD   0
   556  002442' 000000                  TMMR:   .WORD   0
   557  002444' 000000                  TSCRC:  .WORD   0
   558  002446' 177777                          .WORD   177777
   559
   560  002450' 000000                  TMMRH:  .WORD   0
   561
   562  002452'                         TABLE2:
   563  002452' 000000                  RCR:    .WORD   0
   564  002454' 000000                  RSR:    .WORD   0
   565  002456' 000000                  RDDB:   .WORD   0
   566  002460' 000000                  RDBC:   .WORD   0
   567  002462' 000000                  RDBA:   .WORD   0
   568  002464' 000000                  RDCRC:  .WORD   0
   569  002466' 177777                          .WORD   177777
   570  002470' 000000                  RCVECT: .WORD   0
   571  002472' 000000                  IXVECT: .WORD   0
```

```
  572                                   ;ASCII MESSAGE STORAGE
  573
  574  002474' 002544'                  EXCED:  MSG1
  575  002476' 177777                           177777
  576  002500' 002612'                  MSTDWN: MSG2
  577  002502' 177777                           177777
  578  002504' 002644'                  WRCAD:  MSG3
  579  002506' 177777                           177777
  580  002510' 002707'                  NMSINT: MSG4
  581  002512' 177777                           177777
  582  002514' 003067'                  MRJCT:  MSG11
  583  002516' 002753'                          MSG5
  584  002520' 177777                           177777
  585  002522' 003077'                  ERDOPR: MSG12
  586  002524' 003012'                          MSG7
  587  002526' 177777                           177777
  588  002530' 003077'                  MRJTD:  MSG12
  589  002532' 002753'                          MSG5
  590  002534' 177777                           177777
  591  002536' 003107'                  MSHDER: MSG13
  592  002540' 003042'                          MSG8
  593  002542' 177777                           177777
```



```
  594  002544' 042445  051122  051117  MSG1:   .ASCIZ  '%ERROR RETRY FOR THIS CYCLE EXCEEDED%'
  595  002552' 051040  052105  054522
  596  002560' 043040  051117  052040
  597  002566' 044510  020123  054503
  598  002574' 046103  020105  054105
  599  002602' 042503  042105  042105
  600  002610' 000045
  601  002612' 050045  046103  046440  MSG2:   .ASCIZ  '%PCL MASTER WENT DOWN !!%'
  602  002620' 051501  042524  020122
  603  002626' 042527  052116  042040
  604  002634' 053517  020116  020441
  605  002642' 000045
  606  002644' 051045  053103  020122  MSG3:   .ASCIZ  '%RCVR BUSY, OR WRONG RCVR ADDRESS%'
  607  002652' 052502  054523  020054
  608  002660' 051117  053440  047522
  609  002666' 043516  051040  053103
  610  002674' 020122  042101  051104
  611  002702' 051505  022523     000
  612  002707'    045  040515  052123  MSG4:   .ASCIZ  '%MASTER HAS JUST SET ON THIS PCL11%'
  613  002714' 051105  044040  051501
  614  002722' 045040  051525  020124
  615  002730' 042523  020124  047117
  616  002736' 052040  044510  020123
  617  002744' 041520  030514  022461
  618  002752'    000
  619  002753'    040  042515  051523  MSG5:   .ASCIZ  ' MESSAGE WAS REJECTED BY RCVR%'
  620  002760' 043501  020105  040527
  621  002766' 020123  042522  042512
  622  002774' 052103  042105  041040
  623  003002' 020131  041522  051126
  624  003010' 000045
  625  003012' 042040  052101  020101  MSG7:   .ASCIZ  ' DATA OUTPUT READY SET%'
  626  003020' 052517  050124  052125
  627  003026' 051040  040505  054504
  628  003034' 051440  052105  000045
  629  003042' 040515  052123  051105  MSG8:   .ASCIZ  'MASTER WILL NOT SET%'
  630  003050' 053440  046111  020114
  631  003056' 047516  020124  042523
  632  003064' 022524     000
  633  003067'    045  054055  052115  MSG11:  .ASCIZ  '%-XMTR-'
  634  003074' 026522     000
  635  003077'    045  051055  053103  MSG12:  .ASCIZ  '%-RCVR-'
  636  003104' 026522     000
  637  003107'    045  040510  042122  MSG13:  .ASCIZ  '%HARD ERROR%'
  638  003114' 042440  051122  051117
  639  003122' 000045
  640          000001                          .END
```

```
ACSR    000102R          196#    411*    416#
ADDR    000006R          162#    423     447
ADDR22= 001000           225#
ASB     000106R          200#
ASTAT   000104R          198#
AWAS    000110R          201#
BCGm    000314R          245#    264
BCOK    000336R          251#    272     277
BEGIN   000000R          159#    240     246     257     262     268     273     303     309     334     343     348     352
                         358     362     368     383     388     392     397     401     478     481     488     489     497
                         503     504     512     520     521     529
BIT0  = 000001           225#
BIT1  = 000002           225#    266     267     274     275     289     290     470     471     515     531     532
BIT10 = 002000           225#
BIT11 = 004000           225#    332
BIT12 = 010000           225#    500     517
BIT13 = 020000           225#    505     516     522
BIT14 = 040000           225#
BIT15 = 100000           225#    330     378
BIT2  = 000004           225#    355     357
BIT3  = 000010           225#
BIT4  = 000020           225#    346     490
BIT5  = 000040           225#    360     395
BIT6  = 000100           225#    307
BIT7  = 000200           225#    364
BIT8  = 000400           225#    386
BIT9  = 001000           225#
BREAK$= 104407           225#    488     489     503     504     520     521
BR1     000012R          164#    442     446
BR2     000013R          165#
BTOD$ = 104421           225#
BUFIN   002230R          207     548#
CDATA$= 104412           225#    257
CDERCT  000144R          215#
CDmDCT  000146R          216#
CLK     002214R          486*    492*    501*    507*    518*    524*    539#
CNT     002212R          538#
CONFIG  000056R          184#
CSRA    000100R          194#    410*    415*
DATCK$= 104411           225#
DATER$= 104404           225#
DEVICE  002222R          236     336     472*    477*    494*    509*    526*    542#
DVID1   000014R          166#
ENDIT$= 104413           225#    262
ENDS  = 104410           225#    268
ERDOPR  002522R          388     585#
ERFLG   002224R          251#    253     321*    324*    544#
ERRTYP  000106R          199#    341*    350*    366*    381*    390*    399*    479*    495*    510*    527*
ERSUB1  001350R          340     349     409#
ERSUB2  001366R          255     380     389     398     414#
EXCED   002474R          273     574#
EXITS = 104400           225#    303
FINI    000402R          237     266#    278     338
FREE    000150R          217#
GETPA$= 104415           225#    240
GMBUF$= 104414           225#    246
```



```
HRDCNT  000044R          179#
HRDER$= 104405           225#    343     383     481     497     512     529
HRDPAS  000050R          181#
ICONT   000036R          176#
ICOUNT  000040R          177#
IDNUM   000122R          206#
IMODX.= 000000           218#    247
INIT    000030R          173#
INTR    000120R          205#
MAP22$= 104416           225#
MODNAM  000000R          160#
MODSP   000252R          174     223#
MRJCT   002514R          362     582#
MRJID   002530R          397     588#
MSGN$ = 104403           225#    273     334     348     358     362     388     397     478
MSGS$ = 104402           225#
MSG$  = 104401           225#
MSG1    002544R          574     594#
MSG11   003067R          582     633#
MSG12   003077R          585     588     635#
MSG13   003107R          591     637#
MSG2    002612R          576     601#
MSG3    002644R          578     606#
MSG4    002707R          580     612#
MSG5    002753R          583     589     619#
MSG7    003012R          586     625#
MSG8    003042R          592     629#
MSHDER  002536R          478     591#
MSTDWN  002500R          334     576#
NMSINT  002510R          358     580#
NULL  = 000000           225#
OPEN  = 000000           161     167     168     169     170     187     188     189     190     191     192     193     194
                         196     198     200     201     203     204     205     208     209     211     212     214     215
                         216     217     225#
OTOA$ = 104420           225#
PASCNT  000034R          175#
PASS    000374R          261#
PIRQ$ = 000004           225#    309
POPSP = 005726           225#
POPSP2= 022626           225#
PRTY  = 000000           225#
PRTY0 = 000000           165     225#
PRTY1 = 000040           225#
PRTY2 = 000100           225#
PRTY3 = 000140           225#
PRTY4 = 000200           225#
PRTY5 = 000240           164     225#
PRTY6 = 000300           225#
PRTY7 = 000340           225#
PS    = 177776           225#
PSw   = 177776           225#
PUSH  = 005746           225#
PUSH2 = 024646           225#
RAND$ = 104417           225#
RANNUM  000054R          183#
RBUFEA  000130R          209#    294
```

```
RBUFPA  000126R          208#    257     293
RBUFSZ  000132R          210#    241
RBUFVA  000124R          207#    240
RCBC    002206R          241*    242*    243*    291     536#
RCERS   001166R          315     376#
RCOK    001340R          404     407#
RCR     002452R          266*    274*    289*    301*    449*    471*    516*    522     531*    563#
RCRRTN  001336R          385     394     403     405#    408
RCVECT  002470R          444*    570#
RCVERS  002225R          316     320*    377*    407*    545#
RCVSTR= 060001           233#    298
RDBA    002462R          293*    457*    567#
RDBC    002460R          291*    455*    566#
RDCRC   002464R          459*    568#
RDDB    002456R          453*    565#
RESTRT  000270R          202     239#
RES1    000056R          185#
RES2    000060R          186#
RETRY   000422R          254     269#
RFUNCT  002216R          294*    298*    301     540#
RSR     002454R          378     386     395     415     416     451*    517*    564#
RSTRT   000112R          202#
SBADR   000102R          195#
SETUP   001404R          234     422#
SOFCNT  000042R          178#
SOFERS= 104406           225#    352     368     392     401
SOFPAS  000046R          180#
SPUINT  000032R          174#
SPSIZ = 000040             1#    218
SR1     000016R          167#    276     297
SR2     000020R          168#
SR3     000022R          169#
SR4     000024R          170#
START   000252R          173     234#
STAT    000026R          172#
SVR0    000062R          187#
SVR1    000064R          188#
SVR2    000066R          189#
SVR3    000070R          190#
SVR4    000072R          191#
SVR5    000074R          192#
SVR6    000076R          193#
SYSCNT  000052R          182#
TABLE1  002430R          343     352     368     481     497     512     550#
TABLE2  002452R          383     392     401     529     562#
TCR     002430R          267*    275*    290*    302*    307*    424*    470*    484*    505     515*    532*    551#
TMMR    002442R          434*    556#
TMMRH   002450R          355     357*    436*    473*    474*    475     560#
TRNSFR  000470R          252     288#
TRPOFD= 000022           225#
TRY     002223R          260*    270*    271     460*    543#
TSBA    002440R          295*    432*    555#
TSBC    002436R          292*    430*    554#
TSCRC   002444R          438*    557#
TSDB    002434R          428*    485*    553#
TSR     002432R          330     332     346     360     364     410     411     426*    490     500*    552#
```



```
TSTDRP  001604R          235     335     469#
TSTRTN  002170R          483     499     514     523     531#
TXBC    002210R          247*    248*    249*    292     537#
TXERS   000716R          312     328#
TXINT   000620R          300     306#    441
TXMERS  002226R          313     323*    329*    372*    546#
TXMSTR= 060101           229#    299
TXOK    001156R          365     372#
TXRRTN  001154R          337     345     354     359     363     370#    373
TXVECT  002472R          300*    440*    571#
VECTOR  000010R          163#    439
WASADR  000104R          197#
WBUFEA  000136R          212#    296
WBUFPA  000134R          211#    295
WBUFRQ  000140R          213#
WBUFSZ  000142R          214#    247
WDFR    000116R          204#
WDTO    000114R          203#
WRCAD   002504R          348     578#
XFLAG   000005R          161#
XFUNCT  002220R          296*    297*    299*    302     541#
.     = 003124R          258     547#    548#

. ABS.  000000     000
        003124     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XPLAB0,XPLAB0/SOL/CRF:SYM=DDXCOM,XPLAB0
 RUN-TIME: 1 2 .3 SECONDS
 RUN-TIME RATIO: 18/4=4.1
 CORE USED:  8K  (15 PAGES)
```

```
.REM  -
```

IDENTIFICATION

PRODUCT CODE: AC-F0698-MC

PRODUCT NAME: CX8MGB0 BM873-YJ MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

BMG IS A BACKGROUND MODULE THAT EXERCISES A SINGLE BM873-YJ BOOTSTRAP ROM OPTION. IT COMPARES THE CONTENTS OF EACH OF THE 256(10) LOCATIONS STORED IN THE ROM WITH THE CONTENTS OF A 256(10) WORD CORE MEMORY BUFFER TO VERIFY THAT EACH LOCATION IN THE ROM CAN BE UNIQUELY ADDRESSED AND CONTAINS THE CORRECT DATA. ALL ERRORS ARE REPORTED VIA THE CONSOLE DEVICE.

2.0 REQUIREMENTS

HARDWARE: A PDP11 COMPUTER WITH A BM873-YJ OPTION

STORAGE:: BMG REQUIRES:
1. DECIMAL WORDS: 368
2. OCTAL WORDS: 0560
3. OCTAL BYTES: 1340

3.0 PASS DEFINITION

THE INITIAL PASS CONSISTS OF EXECUTING THE BASIC TEST SEQUENCE ONE TIME BEFORE REPORTING END OF PASS.
SUBSEQUENT PASSES OF THE BMGB MODULE CONSISTS OF 100(8) ITERATIONS OF THE BASIC TEST SEQUENCE DESCRIBED IN PARA. 7 BELOW.

4.0 EXECUTION TIME

PASS TIME VARIES DEPENDENT UPON CPU TYPE AND THE CONFIGURATION BEING EXERCISED.

5.0 CONFIGURATION OPTIONS

DEFAULT PARAMETERS:

DVA: 173000

REQUIRED PARAMETERS:

NONE

6.0 DEVICE OPTION SETUP

NONE REQUIRED

7.0 MODULE OPERATION

TEST SEQUENCE:

1. R1 IS SET UP TO POINT TO THE FIRST WORD IN THE ROM
2. R2 IS SET UP TO POINT TO THE CORRESPONDING WORD IN THE CORE MEMORY BUFFER.

   THE ADDRESS IN R1 IS CHECKED FOR EQUALITY TO EITHER 173024 OR 173224 AND IF FOUND EQUAL GOES TO STEP (5) - IF NOT IT PROCEEDS WITH STEP (3). THESE TWO ADDRESSES ARE NOT CHECKED BECAUSE THEIR CONTENTS AS READ ON THE BUS WILL VARY DEPENDENT UPON WHICH PARTICULAR "LOAD" BUTTON HAD BEEN INITIALLY DEPRESSED TO LOAD THE PROGRAM.
3. R1 AND R2 ARE USED TO COMPARE A ROM WORD WITH ITS CORE IMAGE COUNTERPART. IF THE WORDS DON'T COMPARE A SUB-ROUTINE IS CALLED TO SET UP THE ERROR INFORMATION AND REPORT IT VIA A "DATER" CALL TO THE MONITOR.
4. STEP (3) IS REPEATED.
5. R1 AND R2 ARE UPDATED TO POINT TO THE NEXT WORD AND A TEST MADE ON R2 TO SEE IF 256(10) WORDS HAVE BEEN CHECKED. IF YES, GO TO STEP (6) IF NOT REPEAT (3) THRU (5).
6. A PASS COUNTER IS DECREMENTED AND TESTED TO SEE IF 100(8) ITERATIONS OF STEPS (1) THRU (5) HAVE OCCURRED - IF YES GO TO STEP (7) IF NOT REPEAT (1) THRU (5).
7. REPORT END OF PASS AND REPEAT (1) THRU (6).

8.0 OPERATOR OPTIONS

(NONE)

9.0 NON-STANDARD PRINTOUTS

(NONE)

```
000000'          BKMOD   <BMGB >,173000,0,0,0,0,100,156
000000'          MODULE  40020,BMGB ,173000,0,0,0,0,100,156
                 .TITLE  BMGB DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION  6      23-MAY-78
                                .LIST   BIN
                        ;*****************************************************************
000000'                 BEGIN:
000000' 046502  041107      040 MODNAM: .ASCII  /BMGB / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   0       ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 173000              ADDR:   173000+0        ;1ST DEVICE ADDR.
000010' 000000              VECTOR: 0+0             ;1ST DEVICE VECTOR.
000012'     000             BR1:    .BYTE   PRTY0+0 ;1ST BR LEVEL.
000013'     000             BR2:    .BYTE   PRTY0+0 ;2ND BR LEVEL.
000014' 000001              DVID1:  0+1             ;DEVICE INDICATOR 1.
000016' 000000              SR1:    OPEN            ;SWITCH REGISTER 1
000020' 000000              SR2:    OPEN            ;SWITCH REGISTER 2
000022' 000000              SR3:    OPEN            ;SWITCH REGISTER 3
000024' 000000              SR4:    OPEN            ;SWITCH REGISTER 4
                        ;*****************************************************************
000026' 040020              STAT:   40020           ;STATUS WORD.
000030' 000224'             INIT:   START           ;MODULE START ADDR.
000032' 000224'             SPOINT: MODSP           ;MODULE STACK POINTER.
000034' 000000              PASCNT: 0               ;PASS COUNTER.
000036' 000100              ICONT:  100             ;# OF ITERATIONS PER PASS=100
000040' 000000              ICOUNT: 0               ;LOC TO COUNT ITERATIONS
000042' 000000              SOFCNT: 0               ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000              HRDCNT: 0               ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000              SOFPAS: 0               ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000              HRDPAS: 0               ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000              SYSCNT: 0               ;# OF SYS ERRORS ACCUMULATED
000054' 000000              RANNUM: 0               ;HOLDS RANDOM # WHEN .RAND MACRO IS CALLED
000056'                     CONFIG:                 ;RESERVED FOR MONITOR USE
000056' 000000              RES1:   0               ;RESERVED FOR MONITOR USE
000060' 000000              RES2:   0               ;RESERVED FOR MONITOR USE
000062' 000000              SVR0:   OPEN            ;LOC TO SAVE R0.
000064' 000000              SVR1:   OPEN            ;LOC TO SAVE R1.
000066' 000000              SVR2:   OPEN            ;LOC TO SAVE R2.
000070' 000000              SVR3:   OPEN            ;LOC TO SAVE R3.
000072' 000000              SVR4:   OPEN            ;LOC TO SAVE R4.
000074' 000000              SVR5:   OPEN            ;LOC TO SAVE R5.
000076' 000000              SVR6:   OPEN            ;LOC TO SAVE R6.
000100' 000000              CSRA:   OPEN            ;ADDR OF CURRENT CSR.
000102'                     SRADR:                  ;ADDR OF GOOD DATA, OR
000102' 000000              ACSR:   OPEN            ;CONTENTS OF CSR.
000104'                     WASADR:                 ;ADDR OF BAD DATA, OR
000104' 000000              ASTAT:  OPEN            ;STATUS REG CONTENTS.
000106'                     ERRTYP:                 ;TYPE OF ERROR
000106' 000000              ASB:    OPEN            ;EXPECTED DATA.
000110' 000000              AWAS:   OPEN            ;ACTUAL DATA.
000112' 000230'             RSTRT:  RESTRT          ;RESTART ADDRESS AFTER END OF PASS
000114' 000000              WDTO:   OPEN            ;WORDS TO MEMORY PER ITERATION
000116' 000000              WDFR:   OPEN            ;WORDS FROM MEMORY PER ITERATION
000120' 000000              INTR:   OPEN            ;# OF INTERRUPTS PER ITERATION
000122' 000156              IDNUM:  156             ;MODULE IDENTIFICATION NUMBER=156
        000040                      .REPT   SPSIZ   ;MODULE STACK STARTS HERE.
                                    .NLIST
```



```
                                .WORD   0
                                .LIST
                                .ENDR
000224'                 MODSP:
                        ;*****************************************************************
```

```
                                ;THIS ROUTINE SETS UP AND REPORTS ALL DATA COMPARE ERRORS
                                ;256(10) WORD TABLE THAT STORES A CORE IMAGE OF THE CONTENTS OF THE ROM
```

```
311  000522' 032711                  032711  ;173164'          032711          BIT     #RXERR!RXDONE,(R1) ;DONE OR ERRO
312  000524' 100040                  100040  ;173166  100040
313  000526' 001775                  001775  ;173170'          001775          BEQ     50$                ;NO-- WAIT
314  000530' 100745                  100745  ;173172'          100745          BMI     RXRTRY             ;YES-- ERROR IN
315  000532' 012711                  012711  ;173174'          012711          MOV     #RXEMPT+RXGO,(R1) ;START EMPTY
316  000534' 000003                  000003  ;173176  000003
317                                          ;173200'                  60$:
318  000536' 132711                  132711  ;173200'          132711          BITB    #RXTREQ!RXDONE,(R1) ;READY FOR W
319  000540' 000240                  000240  ;173202  000240
320  000542' 001775                  001775  ;173204'          001775          BEQ     60$                ;NOT READY-- WAI
321  000544' 100147                  100147  ;173206'          100147          BPL     CLRPC              ;DONE-- GO TO LO
322  000546' 111324                  111324  ;173210'          111324          MOVB    (R3),(R4)+         ;NOT DONE-- GET
323  000550' 000772                  000772  ;173212'          000772          BR      60$                ;WAIT FOR NEXT B
324                                          ;173214'                  BUTON5:
325  000552' 005000                  005000  ;173214'          005000          CLR     R0                 ;HERE TO START W
326  000554' 000425                  000425  ;173216'          000425          BR      TCBOT0             ;BOOT FROM TAPE
327                                          ;173220'                  BUTON0:
328  000556' 000000                  000000  ;173220'          000000          HALT                       ;HALT NOW
329  000560' 000666                  000666  ;173222'          000666          BR      BUTON1             ;BUT LOOK AT SWR
330                                          ;173224'                  .FILLTO 424
331  000562' 173230                  173230  ;173224'          173230'         .WORD   BM873,PR7
332  000564' 000340                  000340  ;173226  000340
333                                          ;173230'                  BUTON4:
334  000566' 010037                  010037  ;173230'          010037          MOV     R0,R$SGSV+0        ;SAVE R0 IN 40
335  000570' 000040                  000040  ;173232  000040
336  000572' 012700                  012700  ;173234'          012700          MOV     #DTEDMP,R0         ;SET RET
337  000574' 173622                  173622  ;173236  173622
338                                          ;173240'                  REGSAV:
339  000576' 010037                  010037  ;173240'          010037          MOV     R0,R$SGSV+16       ;SAVE R0 AS PC
340  000600' 000056                  000056  ;173242  000056
341  000602' 012700                  012700  ;173244'          012700          MOV     #R$SGSV+16,R0      ;R0 NOW POINTS T
342  000604' 000056                  000056  ;173246  000056
343  000606' 010640                  010640  ;173250'          010640          MOV     SP,-(R0)           ;SAVE SP IN 54
344  000610' 010540                  010540  ;173252'          010540          MOV     R5,-(R0)           ;SAVE R5 IN 52
345  000612' 010440                  010440  ;173254'          010440          MOV     R4,-(R0)           ;SAVE R4 IN 50
346  000614' 010340                  010340  ;173256'          010340          MOV     R3,-(R0)           ;SAVE R3 IN 46
347  000616' 010240                  010240  ;173260'          010240          MOV     R2,-(R0)           ;SAVE R2 IN 44
348  000620' 010140                  010140  ;173262'          010140          MOV     R1,-(R0)           ;SAVE R1 IN 42
349  000622' 014000                  014000  ;173264'          014000          MOV     -(R0),R0           ;RESTORE R0 FROM
350  000624' 000177                  000177  ;173266'          000177          JMP     @R$SGSV+16         ;GO TO SAVED PC
351  000626' 004564                  004564  ;173270  004564
352                                          ;173272'                  TCBOT0:
353  000630' 005005                  005005  ;173272'          005005          CLR     R5                 ;SET SWR PARAMET
354                                          ;173274'                  TCBOOT:
355  000632' 012737                  012737  ;173274'          012737          MOV     #DLBOOT,@#4        ;IN CASE
356  000634' 173530                  173530  ;173276  173530  000004
357  000636' 000004                  000004  ;173300  000004
358  000640' 012706                  012706  ;173302'          012706          MOV     #R$STRY,SP         ;INIT RETRY COUN
359  000642' 000020                  000020  ;173304  000020
360  000644' 012701                  012701  ;173306'          012701          MOV     #TSSC11+TCCM,R1 ;POINT TO COMMAN
361  000646' 177342                  177342  ;173310  177342
362                                          ;173312'                  TCRTRY:
363  000650' 105011                  105011  ;173312'          105011          CLRB    (R1)               ;STOP ALL TAPE M
364  000652' 005705                  005705  ;173314'          005705          TST     R5                 ;INDEFINITE RETR
365  000654' 100402                  100402  ;173316'          100402          BMI     10$                ;YES-- TRY HARDE
366  000656' 005306                  005306  ;173320'          005306          DEC     SP                 ;NO-- DECREMENT
```



```
367  000660' 002426                  002426  ;173322'          002426          BLT     TCERLT             ;TOO MANY-- GIVE
368                                          ;173324'                  10$:
369  000662' 000005                  000005  ;173324'          000005          RESET                      ;CLEAR TC11
370  000664' 110061                  110061  ;173326'          110061          MOVB    R0,1(R1)           ;SELECT PROPER U
371  000666' 000001                  000001  ;173330  000001
372  000670' 052711                  052711  ;173332'          052711          BIS     #TCREV+TCRNUM+TCGO,(R1) ;START T
373  000672' 004003                  004003  ;173334  004003
374                                          ;173336'                  20$:
375  000674' 005711                  005711  ;173336'          005711          TST     (R1)               ;ERROR?
376  000676' 100376                  100376  ;173340'          100376          BPL     20$                ;NO-- WAIT FOR E
377  000700' 005761                  005761  ;173342'          005761          TST     TCST-TCCM(R1)      ;END-ZONE UP YET
378  000702' 177776                  177776  ;173344  177776
379  000704' 100361                  100361  ;173346'          100361          BPL     TCRTRY             ;NO-- MUST BE T
380  000706' 012761                  012761  ;173350'          012761          MOV     #-256.,TCWC-TCCM(R1) ;SET WORD
381  000710' 177400                  177400  ;173352  177400
382  000712' 000002                  000002  ;173354  000002
383  000714' 042711                  042711  ;173356'          042711          BIC     #TCREV,(R1)        ;SET FORWARD MOD
384  000716' 004000                  004000  ;173360  004000
385  000720' 112711                  112711  ;173362'          112711          MOVB    #TCREAD+TCGO,(R1) ;START READ, F
386  000722' 000005                  000005  ;173364  000005
387                                          ;173366'                  30$:
388  000724' 105711                  105711  ;173366'          105711          TSTB    (R1)               ;TRANSFER DONE?
389  000726' 100376                  100376  ;173370'          100376          BPL     30$                ;NO-- WAIT SOME
390  000730' 005711                  005711  ;173372'          005711          TST     (R1)               ;YES-- ERROR?
391  000732' 100746                  100746  ;173374'          100746          BMI     TCRTRY             ;YES-- RETRY
392  000734' 005007                  005007  ;173376'          005007          CLR     PC                 ;NO-- DONE-- GOT
393                                          ;173400'  .IIF DF R$$P04, RPERLT:
394                                          ;173400'  .IIF DF R$$X11, RXERLT:
395                                          ;173400'  .IIF DF T$$C11, TCERLT:
396                                          ;173400'  .IIF DF D$$L11, DLERLT:
397                                          ;173400'  HALTED:
398  000736' 000000                  000000  ;173400'          000000          HALT                       ;DIE
399  000740' 000776                  000776  ;173402'          000776          BR      HALTED             ;STAY DEAD
400                                          ;173404'
401  000742' 012706                  012706  ;173404'          012706          MOV     #R$STRY,SP         ;RETRY RETRY TIM
402  000744' 000020                  000020  ;173406  000020
403  000746' 012701                  012701  ;173410'          012701          MOV     #R$SP04+RPCS1,R1           ;ADDRESS
404  000750' 176700                  176700  ;173412  176700
405  000752' 012702                  012702  ;173414'          012702          MOV     #RPECCI,R2         ;SET ECC INHIBIT
406  000754' 004000                  004000  ;173416  004000
407                                          ;173420'                  RPRTRY:
408  000756' 005705                  005705  ;173420'          005705          TST     R5                 ;INFINITE RETRY?
409  000760' 100402                  100402  ;173422'          100402          BMI     RPRSET             ;YES-- TRY AGAIN
410  000762' 005306                  005306  ;173424'          005306          DEC     SP                 ;RETRY COUNT EXH
411  000764' 002764                  002764  ;173426'          002764          BLT     RPERLT             ;YES-- GIVE UP
412                                          ;173430'                  RPRSET:
413  000766' 000005                  000005  ;173430'          000005          RESET                      ;ZAP!!
414  000770' 110061                  110061  ;173432'          110061          MOVB    R0,RPCS2(R1)       ;SELECT PROPER U
415  000772' 000010                  000010  ;173434  000010
416  000774' 012711                  012711  ;173436'          012711          MOV     #RPPRST+RPGO,(R1) ;DO "READ-IN P
417  000776' 000021                  000021  ;173440  000021
418  001000' 005061                  005061  ;173442'          005061          CLR     RPDC(R1)           ;SET CYLINDER 0
419  001002' 000034                  000034  ;173444  000034
420  001004' 005061                  005061  ;173446'          005061          CLR     RPDA(R1)           ; TRACK 0, SECTO
421  001006' 000006                  000006  ;173450  000006
422  001010' 050261                  050261  ;173452'          050261          BIS     R2,RPOF(R1)        ;SET INHIBIT ECC
```

```
 423  001012' 000032                  000032  ;173454'  000032
 424  001014' 012761                  012761  ;173456'                012761          MOV     #-256.,RPWC(R1) ;SET UP WORD COU
 425  001016' 177400                  177400  ;173460'  177400
 426  001020' 000002                  000002  ;173462'  000002
 427  001022' 012711                  012711  ;173464'                012711          MOV     #RPREAD+RPGO,(R1) ;START READ FU
 428  001024' 000071                  000071  ;173466'  000071
 429                                          ;173470'                        20$:
 430  001026' 105711                  105711  ;173470'                105711          TSTB    (R1)            ;READY?
 431  001030' 100376                  100376  ;173472'                100376          BPL     20$             ;NO-- WAIT UNTIL
 432  001032' 032761                  032761  ;173474'                032761          BIT     #RPPER,RPER1(R1) ;FORMAT ERROR?
 433  001034' 000020                  000020  ;173476'  000020
 434  001036' 000014                  000014  ;173500'  000014
 435  001040' 001402                  001402  ;173502'                001402          BEQ     30$             ;NO-- TRY AGAIN
 436  001042' 052702                  052702  ;173504'                052702          BIS     #RPFM22,R2      ;YES-- TRY FOR 2
 437  001044' 010000                  010000  ;173506'  010000
 438                                          ;173510'                        30$:
 439  001046' 032711                  032711  ;173510'                032711          BIT     #RPTRE!RPMCPE,(R1) ;TRANSFER OR
 440  001050' 060000                  060000  ;173512'  060000
 441  001052' 001341                  001341  ;173514'                001341          BNE     RPRTRY          ;YES-- ERROR-- T
 442  001054' 032761                  032761  ;173516'                032761          BIT     #RPATA!RPERR,RPDS(R1) ;ATTN OR 0
 443  001056' 140000                  140000  ;173520'  140000
 444  001060' 000012                  000012  ;173522'  000012
 445  001062' 001335                  001335  ;173524'                001335          BNE     RPRTRY          ;YES-- ERROR-- T
 446                                          ;173526'                        CLRPC:
 447  001064' 005007                  005007  ;173526'                005007          CLR     PC              ;JMP 0
 448                                          ;173530'                        BUTON6:
 449                                          ;173530'                        DLBOOT:
 450  001066' 012701                  012701  ;173530'                012701          MOV     #D$SL11+DLRCSR,R1 ;GET DL11 EXTE
 451  001070' 177560                  177560  ;173532'  177560
 452  001072' 011706                  011706  ;173534'                011706          MOV     (PC),SP         ;SET TEMP STACK
 453  001074' 005004                  005004  ;173536'                005004          CLR     R4              ;RESET MEMORY AD
 454  001076' 012702                  012702  ;173540'                012702          MOV     #DLCHAR,R2      ;SET ADD
 455  001100' 173610                  173610  ;173542'  173610
 456                                          ;173544'                        10$:
 457                                          ;173544'                        CALL    (R2)            ;GET A CHARACTER
 458  001102' 004712                  004712  ;173544'                004712          JSR PC,(R2)
 459  001104' 124427                  124427  ;173546'                124427          CMPB    -(R4),#220      ;DLE?
 460  001106' 000220                  000220  ;173550'  000220
 461  001110' 001374                  001374  ;173552'                001374          BNE     10$             ;NO-- KEEP ON LO
 462                                          ;173554'                        CALL    -(R2)           ;GET TWO BYTES
 463  001112' 004742                  004742  ;173554'                004742          JSR PC,-(R2)
 464  001114' 014403                  014403  ;173556'                014403          MOV     -(R4),R3        ;GET BYTE COUNT
 465  001116' 042703                  042703  ;173560'                042703          BIC     #BIT15!BIT14,R3 ;CLEAR QSYNC AND
 466  001120' 140000                  140000  ;173562'  140000
 467                                          ;173564'                        CALL    (R2)            ;SKIP TWO MORE B
 468  001122' 004712                  004712  ;173564'                004712          JSR PC,(R2)
 469                                          ;173566'                        CALL    (R2)            ; AND TWO MORE
 470  001124' 004712                  004712  ;173566'                004712          JSR P ,(R2)
 471  001126' 005722                  005722  ;173570'                005722          TST     (R2)+           ;SKIP THE CALL (
 472                                          ;173572'                        CALL    (R2)            ; +1 MAKES 5 BYT
 473  001130' 004712                  004712  ;173572'                004712          JSR PC,(R2)
 474  001132' 005004                  005004  ;173574'                005004          CLR     R4              ;RESET BACK TO 0
 475                                          ;173576'                        20$:
 476                                          ;173576'                        CALL    (R2)            ;GET A CHARACTER
 477  001134' 004712                  004712  ;173576'                004712          JSR PC,(R2)
 478  001136' 005303                  005303  ;173600'                005303          DEC     R3              ;REDUCE COUNT
```



```
 479  001140' 003375                  003375  ;173602'                003375          BGT     20$             ;BACK IF MORE
 480  001142' 005007                  005007  ;173604'                005007          CLR     PC              ;ELSE GO TO LOAD
 481                                          ;173606'                        CALL    (PC)            ;GET A BYTE, THE
 482  001144' 004717                  004717  ;173606'                004717          JSR PC,(PC)
 483                                          ;173610'                        DLCHAR:
 484  001146' 105711                  105711  ;173610'                105711          TSTB    (R1)            ;READY WITH A CH
 485  001150' 100376                  100376  ;173612'                100376          BPL     DLCHAR          ;NO-- WAIT SOME
 486  001152' 116124                  116124  ;173614'                116124          MOVB    DLRBUF-DLRCSR(R1),(R4)+ ;YES-- S
 487  001154' 000002                  000002  ;173616'  000002
 488                                          ;173620'                        RETURN                  ;AND RETURN FROM
 489  001156' 000207                  000207  ;173620'                000207          RTS PC
 490                                          ;173622'                        DTEDMP:
 491  001160' 005005                  005005  ;173622'                005005          CLR     R5              ;POINT TO LOCATI
 492  001162' 012500                  012500  ;173624'                012500          MOV     (R5)+,R0        ;SAVE LOCATION 0
 493  001164' 012501                  012501  ;173626'                012501          MOV     (R5)+,R1        ; AND LOCATION 2
 494  001166' 011502                  011502  ;173630'                011502          MOV     (R5),R2         ;SAVE 4 IN R2
 495  001170' 012725                  012725  ;173632'                012725          MOV     #21$,(R5)+      ;SET VXM
 496  001172' 173646                  173646  ;173634'  173646
 497  001174' 011503                  011503  ;173636'                011503          MOV     (R5),R3         ;SAVE 6 IN R3
 498  001176' 005015                  005015  ;173640'                005015          CLR     (R5)            ;SET PS FOR TRAP
 499                                          ;173642'                        20$:
 500  001200' 012704                  012704  ;173642'                012704          MOV     #D$ST20+DLYCNT-DTESIZ,R4 ;POINT
 501  001202' 174340                  174340  ;173644'  174340
 502                                          ;173646'                        21$:
 503  001204' 012706                  012706  ;173646'                012706          MOV     #S$SDTE+10,SP   ;SET STACK TO S
 504  001206' 000140                  000140  ;173650'  000140
 505                                          ;173652'                        22$:
 506  001210' 062704                  062704  ;173652'                062704          ADD     #DTESIZ,R4      ;BUMP TO NEXT DT
 507  001212' 000040                  000040  ;173654'  000040
 508  001214' 105704                  105704  ;173656'                105704          TSTB    R4              ;IS THIS THE END
 509  001216' 100770                  100770  ;173660'                100770          BMI     20$             ;YES-- START ALL
 510  001220' 032764                  032764  ;173662'                032764          BIT     #TO11DB,STAT-DLYCNT(R4) ;DOORBEL
 511  001222' 004000                  004000  ;173664'  004000
 512  001224' 000034                  000034  ;173666'  000034
 513  001226' 001770                  001770  ;173670'                001770          BEQ     22$             ;NO-- TRY NEXT D
 514  001230' 026417                  026417  ;173672'                026417          CMP     TO108C-DLYCNT(R4),(PC) ;DOES THI
 515  001232' 000014                  000014  ;173674'  000014
 516  001234' 001365                  001365  ;173676'                001365          BNE     22$             ;NO-- TRY ANOTHE
 517  001236' 010315                  010315  ;173700'                010315          MOV     R3,(R5)         ;RESTORE LOCATIO
 518  001240' 010245                  010245  ;173702'                010245          MOV     R2,-(R5)        ; 4
 519  001242' 010145                  010145  ;173704'                010145          MOV     R1,-(R5)        ;RESTORE 2
 520  001244' 010045                  010045  ;173706'                010045          MOV     R0,-(R5)        ; AND 0
 521  001246' 012700                  012700  ;173710'                012700          MOV     #S$SDTE,R0      ;POINT TO SAVE S
 522  001250' 000130                  000130  ;173712'  000130
 523                                          ;173714'                        29$:
 524  001252' 012420                  012420  ;173714'                012420          MOV     (R4)+,(R0)+     ;SAVE A REGISTER
 525  001254' 022700                  022700  ;173716'                022700          CMP     #TO11DT-DLYCNT+S$SDTE,R0 ;FINISH
 526  001256' 000156                  000156  ;173720'  000156
 527  001260' 103374                  103374  ;173722'                103374          BHIS    29$             ;NO-- SAVE SOME
 528                                          ;173724'                        ADDR    DIAG2-<TO11DT+2>,R4 ;R4 POINTS T
 529  001262' 005724                  005724  ;173724'                005724          TST     (R4)+
 530  001264' 010401                  010401  ;173726'                010401          MOV     R4,R1           ; SO DOES R1
 531  001266' 012700                  012700  ;173730'                012700          MOV     #DRESET,R0      ;SETUP R0 FOR TO
 532  001270' 000100                  000100  ;173732'  000100
 533  001272' 010021                  010021  ;173734'                010021          MOV     R0,(R1)+        ;R1 POINTS TO ST
 534  001274' 005061                  005061  ;173736'                005061          CLR     DLYCNT-STAT(R1) ;SET DTE20 FOR M
```

```
   535  001276' 177744                    177744  ;1173740 177744
   536  001300' 005061                    005061  ;1173742'        005061          CLR     TO10AD-STAT(R1) ;START DUMPING -
   537  001302' 177764                    177764  ;1173744 177764
   538                                             ;1173746'                30$:
   539  001304' 032711                    032711  ;1173746'        032711          BIT     #TO11DB,(R1)    ;IS DOORBELL RIN
   540  001306' 004000                    004000  ;1173750 004000
   541  001310' 001775                    001775  ;1173752'        001775          BEQ     30$             ;NO-- WAIT FOR D
   542  001312' 010014                    010014  ;1173754'        010014          MOV     R0,(R4)         ;YES-- CLEAR DOO
   543  001314' 005061                    005061  ;1173756'        005061          CLR     TO11AD-STAT(R1) ;START INPUT TO
   544  001316' 177766                    177766  ;1173760 177766
   545  001320' 012761                    012761  ;1173762'        012761          MOV     #IFLOP!<<-256.>&BCOUNT>,TO11BC-S
   546  001322' 107400                    107400  ;1173764 107400
   547  001324' 177762                    177762  ;1173766 177762
   548                                             ;1173770'                40$:
   549  001326' 105711                    105711  ;1173770'        105711          TSTB    (R1)            ;TRANSFER COMPLE
   550  001330' 100376                    100376  ;1173772'        100376          BPL     40$             ;NO-- WAIT SOME
   551  001332' 005007                    005007  ;1173774'        005007          CLR     PC              ;GO TO LOADED CO
   552                                             ;1173776'                FYLLTO  1000
   553  001334' 000000                    000000  ;1173776'           000          .BYTE   0
   554                                             ;1173777'           000          .BYTE   0
   555  001336' 177777                TABEND: 177777
   556
   557          000001                            .END
```



```
ACSR     000102R          181#
ADDR     000006R          147#  200
ADDR22=  001000           199#
AGAIN    000230R          203#  221
ASB      000106R          185#  229*
ASTAT    000104R          183#
AWAS     000110R          186#  228*
BEGIN    000000R          144#  219   231
BIT0   = 000001           199#
BIT1   = 000002           199#
BIT10  = 002000           199#
BIT11  = 004000           199#
BIT12  = 010000           199#
BIT13  = 020000           199#
BIT14  = 040000           199#
BIT15  = 100000           199#
BIT2   = 000004           199#
BIT3   = 000010           199#
BIT4   = 000020           199#
BIT5   = 000040           199#
BIT6   = 000100           199#
BIT7   = 000200           199#
BIT8   = 000400           199#
BIT9   = 001000           199#
BMERR    000310R          211   214   226#
BMTAB    000336R          204   240#
BREAKS=  104407           199#
BP1      000012R          149#
BP2      000013R          150#
BTOOS  = 104421           199#
CDATAS=  104412           199#
CONFIG   000056R          168#
CSRA     000100R          179#
DATCKS=  104411           199#
DATERS=  104404           199#   231
DVID1    000014R          151#
ENDITS=  104413           199#   219
ENDS   = 104410           199#
ERRTYP   000106R          184#
EXITS  = 104400           199#
GETPAS=  104415           199#
GWBUFS=  104414           199#
HRDCNT   000044R          164#
HRDSRS=  104405           199#
HRDPAS   000050R          166#
ICONT    000036R          161#
ICONAT   000040R          162#
IDNUM    000122R          191#
IWLT     000030R          156#
IWTB     000120R          190#
MAP22S=  104416           199#
MODNAM   000000R          145#
MODSP    000224R          159   197#
MSGNS  = 104403           199#
MSGSS  = 104402           199#
MSGS   = 104401           199#
```

```
NULL   = 000000            199#
OPEN   = 000000            146     152     153     154     155     172     173     174     175     176     177     178     179
                           181     183     185     186     188     189     190     199#
QTOAS  = 104420            199#
PASCNT  000034R            160#
PIRQS  = 000004            199#
POPSP  = 005726            199#
POPSP2 = 022626            199#
PRTY   = 000000            199#
PRTY0  = 000000            149     150     199#
PRTY1  = 000040            199#
PRTY2  = 000100            199#
PRTY3  = 000140            199#
PRTY4  = 000200            199#
PRTY5  = 000240            199#
PRTY6  = 000300            199#
PRTY7  = 000340            199#
PS     = 177776            199#
PSW    = 177776            199#
PUSH   = 005746            199#
PUSH2  = 024646            199#
RAND$  = 104417            199#
RANNUM  000054R            168#
RESTRT  000230R            187     202#
RES1    000056R            170#
RES2    000060R            171#
RSTRT   000112R            187#
SBADR   000102R            180#    226*
SOFCNT  000042R            163#
SOFFRS= 104406             199#
SOFPAS  000046R            165#
SPDINT  000032R            159#
SPSIZ  = 000040            192     192#
SR1     000016R            152#
SR2     000020R            153#
SR3     000022R            154#
SR4     000024R            155#
START   000224R            158     200#
STAT    000026R            157#
SVR0    000062R            172#
SVR1    000064R            173#
SVR2    000066R            174#
SVR3    000070R            175#
SVR4    000072R            176#
SVR5    000074R            177#
SVR6    000076R            178#
SVSCNT  000052R            167#
TABEND  001336R            216     555#
TRPDEF= 000022             199#
VECTOR  000010R            148#
WASADR  000104R            182#    227*
WDFR    000116R            189#
WDTO    000114R            188#
XPLAG   000005R            146#
```



```
. ABS.   000000     000
         001340     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XBMGB0,XBMGB0/SOL/CRF:SYM=DOXCON,XBMGB0
 RUN-TIME:  1  1  .1 SECONDS
 RUN-TIME RATIO: 38/3=10.2
 CORE USED:  7K  (13 PAGES)
```

.REM

IDENTIFICATION

PRODUCT CODE: AC-F066B-MC

PRODUCT NAME: CXBMFB0 BM873-YB MODULE

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 SUPPORT GROUP

1.0 ABSTRACT

BMF IS A BACKGROUND MODULE THAT EXERCISES A SINGLE BM873-YH BOOTSTRAP ROM OPTION. IT COMPARES THE CONTENTS OF EACH OF THE 256(10) LOCATIONS STORED IN THE ROM WITH THE CONTENTS OF A 256(10) WORD CORE MEMORY BUFFER TO VERIFY THAT EACH LOCATION IN THE ROM CAN BE UNIQUELY ADDRESSED AND CONTAINS THE CORRECT DATA. ALL ERRORS ARE REPORTED VIA THE CONSOLE DEVICE.

2.0 REQUIREMENTS

HARDWARE: A PDP11 COMPUTER WITH A BM873-YH OPTION

STORAGE:: BMF REQUIRES:
1. DECIMAL WORDS: 368
2. OCTAL WORDS: 0560
3. OCTAL BYTES: 1340

3.0 PASS DEFINITION

THE INITIAL PASS CONSISTS OF EXECUTING THE BASIC TEST SEQUENCE ONE TIME BEFORE REPORTING END OF PASS.
SUBSEQUENT PASSES OF THE BMFB MODULE CONSISTS OF 100(8) ITERATIONS OF THE BASIC TEST SEQUENCE DESCRIBED IN PARA. 7 BELOW.

4.0 EXECUTION TIME

PASS TIME VARIES DEPENDENT UPON CPU TYPE AND THE CONFIGURATION BEING EXERCISED. WHEN RUNNING ALONE ON A PDP11/40 THE FIRST PASS SHOULD TAKE LESS THAN 10 SECONDS AND SUBSEQUENT PASSES LESS THAN ONE MINUTE.

5.0 CONFIGURATION OPTIONS

DEFAULT PARAMETERS:

DVA: 173000

REQUIRED PARAMETERS:

NONE

6.0 DEVICE OPTION SETUP

NONE REQUIRED

7.0 MODULE OPERATION

TEST SEQUENCE:

1. R1 IS SET UP TO POINT TO THE FIRST WORD IN THE ROM
2. R2 IS SET UP TO POINT TO THE CORRESPONDING WORD IN THE CORE MEMORY BUFFER.

   THE ADDRESS IN R1 IS CHECKED FOR EQUALITY TO EITHER 173024 OR 173224 AND IF FOUND EQUAL GOES TO STEP (5) - IF NOT IT PROCEEDS WITH STEP (3). THESE TWO ADDRESSES ARE NOT CHECKED BECAUSE THEIR CONTENTS AS READ ON THE BUS WILL VARY DEPENDENT UPON WHICH PARTICULAR "LOAD" BUTTON HAD BEEN INITIALLY DEPRESSED TO LOAD THE PROGRAM.
3. R1 AND R2 ARE USED TO COMPARE A ROM WORD WITH ITS CORE IMAGE COUNTERPART. IF THE WORDS DON'T COMPARE A SUB-ROUTINE IS CALLED TO SET UP THE ERROR INFORMATION AND REPORT IT VIA A "DATER" CALL TO THE MONITOR.
4. STEP (3) IS REPEATED.
5. R1 AND R2 ARE UPDATED TO POINT TO THE NEXT WORD AND A TEST MADE ON R2 TO SEE IF 256(10) WORDS HAVE BEEN CHECKED. IF YES, GO TO STEP (6) IF NOT REPEAT (3) THRU (5).
6. A PASS COUNTER IS DECREMENTED AND TESTED TO SEE IF 100(8) ITERATIONS OF STEPS (1) THRU (5) HAVE OCCURRED - IF YES GO TO STEP (7) IF NOT REPEAT (1) THRU (5).
7. REPORT END OF PASS AND REPEAT (1) THRU (6).

8.0 OPERATOR OPTIONS

(NONE)

9.0 NON-STANDARD PRINTOUTS

(NONE)

```
                        BKMOD   <BMFB >,173000,,,,,100,155
                        MODULE  40020,BMFB ,173000,,,,,100,155
                        .TITLE  BMFB DEC/X11 SYSTEM EXERCISER MODULE
                ;       DDXCOM  VERSION  6       23-MAY-78
                                .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 046502  041106      040 MODNAM: .ASCII  /BMFB / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN                ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 173000                  ADDR:   173000+0                    ;1ST DEVICE ADDR.
000010' 000000                  VECTOR: +0                          ;1ST DEVICE VECTOR.
000012'     000                 BR1:    .BYTE   PRTY+0              ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY+0              ;2ND BR LEVEL.
000014' 000001                  DVID1:  +1                          ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                        ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                        ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                        ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                        ;SWITCH REGISTER 4
                                ;*****************************************************************************
000026' 040020                  STAT:   40020                       ;STATUS WORD.
000030' 000224'                 INIT:   START                       ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP                       ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                           ;PASS COUNTER.
000036' 000100                  ICONT:  100                         ;# OF ITERATIONS PER PASS=100
000040' 000000                  ICOUNT: 0                           ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                           ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                           ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                           ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                           ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                           ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                           ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                             ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                           ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                           ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                        ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                        ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                        ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                        ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                        ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                        ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                        ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                        ;ADDR OF CURRENT CSR.
000102'                         SBADR:                              ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                        ;CONTENTS OF CSR.
000104'                         BASADR:                             ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                        ;STATUS REG CONTENTS.
000106'                         ERRTYP:                             ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                        ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                        ;ACTUAL DATA.
000112' 000230'                 RSTRT:  RESTRT                      ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                        ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                        ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                        ;# OF INTERRUPTS PER ITERATION
000122' 000155                  IDNUM:  155                         ;MODULE IDENTIFICATION NUMBER=155
        000040                          .REPT   SPSIZ               ;MODULE STACK STARTS HERE.
                                        .NLIST
```



```
                                        .WORD   0
                                        .LIST
                                        .ENDR
000224'                         MODSP:  ;*****************************************************************************
```

```
                                ;
000224' 016705  177556          START:  MOV     ADDR,R5         ;GET FIRST ROM ADDRESS INTO R5
                                ;
000230'                         RESTRT:
000230' 010501                  AGAIN:  MOV     R5,R1           ;R1 POINTS TO ROM WORD
000232' 012702  000336'                 MOV     #RMTAB,R2       ;R2 POINTS TO ROM IMAGE IN CORE
000236' 022701  173024          1$:     CMP     #173024,R1      ;ROM ADDRESS = 173024 ??
000242' 001413                          BEQ     3$              ;BR IF YES
000244' 022701  173224                  CMP     #173224,R1      ;ROM ADDRESS = 173224 ??
000250' 001410                          BEQ     3$              ;BR IF YES
000252' 021112                  2$:     CMP     (R1),(R2)       ;CHECK ONE LOCATION
000254' 001402                          BEQ     4$              ;BR IF [ROM] = [CORE]
000256' 004767  000026                  JSR     PC,RMERR        ;GO SETUP AND REPORT ERROR
000262' 021112                          CMP     (R1),(R2)       ;CHECK IT AGAIN
000264' 001402                          BEQ     3$              ;BR IF [ROM] = [CORE]
000266' 004767  000016                  JSR     PC,RMERR        ;GO SETUP AND REPORT ERROR
000272' 022122                  3$:     CMP     (R1)+,(R2)+     ;ADD +2 TO BOTH POINTERS
000274' 022702  001336'                 CMP     #TABEND,R2      ;DONE LAST WORD ??
000300' 001356                          BNE     1$              ;BR IF NOT
000302'                         4$:
000302' 104413  000000'                 ENDITS,BEGIN            ;SIGNAL END OF ITERATION.
                                                                ;MONITOR SHALL TEST END OF PASS
000306' 000750                  BR      AGAIN
                                ;
                                ;
                                ;THIS ROUTINE SETS UP AND REPORTS ALL DATA COMPARE ERRORS
                                ;
000310' 010267  177566          RMERR:  MOV     R2,SBADR        ;SAVE THE ADDR. OF GOOD DATA
000314' 010167  177564                  MOV     R1,WASADR       ;SAVE ADDR. OF THE BAD DATA
000320' 011167  177564                  MOV     (R1),WAS        ;GET WAS DATA
000324' 011267  177556                  MOV     (R2),SHD        ;GET THE SHD DATA
                                ;*****************************************************************
000330' 104404  000000'                 DATERS,BEGIN            ;DATA ERROR!!!
                                ;*****************************************************************
000334' 000207                  RTS     PC                      ;CONTINUE CHECKING
                                ;
                                ;256(10) WORD TABLE THAT STORES A CORE IMAGE OF THE CONTENTS OF THE ROM
                                ;
                                ;
                                ;
000336'                         RMTAB:
000336' 010037                          ;173000         010037  BUTON1:
000340' 000040                          ;173002         000040
000342' 013700                          ;173004         013700          MOV     R0,R0TOR7+0     ;SAVE R0 IN LOCATION 40
000344' 177570                          ;173006         177570
000346' 032700                          ;173010         032700          MOV     SWR,R0          ;GET SWITCH REGISTER
000350' 000001                          ;173012         000001
000352' 001007                          ;173014         001007          BIT     #BIT0,R0        ;IS LOW-ORDER BIT SET?
000354' 000513                          ;173016         000513
                                                                        BNE     BUTONX          ;YES-- LOOK AT CONTENTS
                                                                BUTON3: BR      RFGSAV          ;NO-- SAVE R1-R7 IN 42-5
000356' 005000                          ;173020         005000          CLR     R0              ;SAV LOAD FROM FLOPPY, U
000360' 000404                          ;173022         000404          BR      BUTONX          ;GO TO COMMON CODE FOR 3
                                        ;173024                         FILLTO  24
000362' 173000                          173000          ;173024 173000  .WORD   ROMORG,PR7
```



```
000364' 000340                          000340  ;173026 000340
                                                ;173030         BUTON2:
000366' 012700                          012700  ;173030 012700          MOV     #BIT7,R0        ;BIT 7 MEANS LOAD FROM R
000370' 000200                          000200  ;173032 000200
                                                ;173034         BUTONX:
000372' 010005                          010005  ;173034 010005          MOV     R0,R5           ;SAVE PARAMETER FOR BOOT
000374' 106300                          106300  ;173036 106300          ASLB    R0              ;LEFT-ALIGN SPEED FIELD
000376' 122700                          122700  ;173040 122700          CMPB    #3*BIT4,R0      ;IS SPEED 0, 1, OR 2?
000400' 000060                          000060  ;173042 000060
000402' 101001                          101001  ;173044 101001          BHI     10$             ;YES-- UNIT IS UNIT TO U
000404' 005000                          005000  ;173046 005000          CLR     R0              ;NO-- USE UNIT #0
                                                ;173050         10$:
000406' 000300                          000300  ;173050 000300          SWAB    R0              ;GET UNIT # IN LOW BYTE
000410' 042700                          042700  ;173052 042700          BIC     #^C7,R0         ;TRIM TO 3 BITS 2, 1, 0
000412' 177770                          177770  ;173054 177770
000414' 105705                          105705  ;173056 105705          TSTB    R5              ;WHERE SHOULD WE BOOT FR
000416' 100553                          100553  ;173060 100553          BMI     RPBOOT          ;BIT 7 = 1 -- BOOT FROM
000420' 012737                          012737  ;173062 012737          MOV     #TCROOT,4       ;SET TIMEOUT TRAP TO TRY
000422' 173304                          173304  ;173064 173304
000424' 000004                          000004  ;173066 000004
000426' 005037                          005037  ;173070 005037          CLR     6               ; . .
000430' 000006                          000006  ;173072 000006
                                                ;173074         RXBOOT:
000432' 012706                          012706  ;173074 012706          MOV     #RETRY,SP       ;SET RETRY COUNT
000434' 000014                          000014  ;173076 000014
000436' 012701                          012701  ;173100 012701          MOV     #RXEPA+RXCS,R1  ;ADDRESS CONTROL STATUS
000440' 177170                          177170  ;173102 177170
000442' 010003                          010003  ;173104 010003          MOV     R0,R3           ;COPY UNIT #
                                                ;173106         RXRTRY:
000444' 005705                          005705  ;173106 005705          TST     R5              ;INDEFINITE RETRY?
000446' 100402                          100402  ;173110 100402          BMI     RXRSET          ;YES-- TRY FAITHFULLY
000450' 005306                          005306  ;173112 005306          DEC     SP              ;NO-- DECREMENT RETRY CO
000452' 002445                          002445  ;173114 002445          BLT     RXEHLT          ;GIVE UP IF RUN OUT
                                                ;173116         RXRSET:
000454' 000005                          000005  ;173116 000005          RESET                   ;CLEAR THE WORLD
                                                ;173120         20$:
000456' 032711                          032711  ;173120 032711          BIT     #RXDONE,(R1)    ;WAIT UNTIL READY FOR FU
000460' 000040                          000040  ;173122 000040
000462' 001775                          001775  ;173124 001775          BEQ     20$             ;NOT YET-- WAIT
                                                ;173126         RXPERF:
000464' 010300                          010300  ;173126 010300          MOV     R3,R0           ;GET UNIT #
000466' 001402                          001402  ;173130 001402          BEQ     5$              ;ZERO-- USE ZERO
000470' 012700                          012700  ;173132 012700          MOV     #RXUNIT,R0      ;NON-ZERO-- ASSUME UNIT
000472' 000020                          000020  ;173134 000020
                                                ;173136         5$:
000474' 052700                          052700  ;173136 052700          BIS     #PXREAD+RXGO,R0 ;SET READ FUNCTION
000476' 000007                          000007  ;173140 000007
000500' 010102                          010102  ;173142 010102          MOV     R1,P2           ;COPY ADDRESS OF RXCS
000502' 010022                          010022  ;173144 010022          MOV     R0,(R2)+        ;START READ FUNCTION, R2
                                                ;173146         10$:
000504' 105711                          105711  ;173146 105711          TSTB    (R1)            ;READY?
000506' 100376                          100376  ;173150 100376          BPL     10$             ;NO-- WAIT
000510' 012712                          012712  ;173152 012712          MOV     #1,(R2)         ;SET SECTOR #
000512' 000001                          000001  ;173154 000001
                                                ;173156         20$:
000514' 105711                          105711  ;173156 105711          TSTB    (R1)            ;READY FOR TRACK?
```

```
000516' 100376                  100376  ;173160 100376          BPL     20$             ;NO-- WAIT
000520' 012712                  012712  ;173162 012712          MOV     #1,(R2)         ;SET TRACK #
000522' 000001                  000001  ;173164 000001
                                        ;173166          30$:
000524' 032711                  032711  ;173166 032711          BIT     #RXERR!RXDONE,(R1) ;DONE OR ERROR?
000526' 100040                  100040  ;173170 100040          BEQ     30$             ;NO-- WAIT
000530' 001775                  001775  ;173172 001775          BMI     RXRTRY          ;YES-- ERROR IN FUNCTION
000532' 100744                  100744  ;173174 100744  RXEMSL:
                                        ;173176
000534' 012711                  012711  ;173176 012711          MOV     #RXEMPT+RXGO,(R1) ;START EMPTY
000536' 000003                  000003  ;173170 000003
000540' 005004                  005004  ;173202 005004          CLR     R4              ;ALWAYS START TRANSFER A
                                        ;173204          10$:
000542' 132711                  132711  ;173204 132711          BITB    #RXTREQ!RXDONE,(R1) ;READY FOR WORD, OR
000544' 000240                  000240  ;173206 000240
000546' 001775                  001775  ;173210 001775          BEQ     10$             ;NOT READY-- WAIT SOME M
000550' 100153                  100153  ;173212 100153          BPL     CLRPC           ;DONE--- GO TO LOCATION 0
000552' 111224                  111224  ;173214 111224          MOVB    (R2),(R4)+      ;NOT DONE-- GET A BYTE F
000554' 000772                  000772  ;173216 000772          BR      10$             ;WAIT FOR NEXT BYTE
                                        ;173220          FILLTO  224
000556' 000000                  000000  ;173220 000000          .BYTE   0
                                        ;173221 000              .BYTE   0
000560' 000000                  000000  ;173222 000000          .BYTE   0
                                        ;173223 000              .BYTE   0
000562' 173000                  173000  ;173224 173000          .WORD   ROMORG,PR7
000564' 000340                  000340  ;173226 000340
                                        ;173230          RXEHLT:
000566' 012711                  012711  ;173230 012711          MOV     #RXRERR+RXGO,(R1) ;DO A READ ERROR REGIS
000570' 000017                  000017  ;173232 000017
                                        ;173234          10$:
000572' 032711                  032711  ;173234 032711          BIT     #RXDONE,(R1)    ;WAIT UNTIL ERROR ASSEMB
000574' 000040                  000040  ;173236 000040          BEQ     10$             ;...
000576' 001775                  001775  ;173240 001775          MOV     (R2),R0         ;GET ERROR REGISTER
000600' 011200                  011200  ;173242 011200          BR      HALTED          ;HALT AND DISPLAY ERRORS
000602' 000541                  000541  ;173244 000541
                                        ;173246          REGSAV:
000604' 010037                  010037  ;173246 010037          MOV     R0,ROTOR7+16    ;SAVE R0 AS PC IN 56
000606' 000056                  000056  ;173250 000056
000610' 012700                  012700  ;173252 012700          MOV     #ROTOR7+16,R0   ;R0 NOW POINTS TO 56
000612' 000056                  000056  ;173254 000056
000614' 010640                  010640  ;173256 010640          MOV     SP,-(R0)        ;SAVE SP IN 54
000616' 010540                  010540  ;173260 010540          MOV     R5,-(R0)        ;SAVE R5 IN 52
000620' 010440                  010440  ;173262 010440          MOV     R4,-(R0)        ;SAVE R4 IN 50
000622' 010340                  010340  ;173264 010340          MOV     R3,-(R0)        ;SAVE R3 IN 46
000624' 010240                  010240  ;173266 010240          MOV     R2,-(R0)        ;SAVE R2 IN 44
000626' 010140                  010140  ;173270 010140          MOV     R1,-(R0)        ;SAVE R1 IN 42
000630' 014000                  014000  ;173272 014000          MOV     -(R0),R0        ;RESTORE R0 FROM 40
000632' 000177                  000177  ;173274 000177          JMP     @ROTOR7+16      ;GO TO SAVED PC
000634' 004556                  004556  ;173276 004556
                                        ;173300          TCBOT0:
000636' 005000                  005000  ;173300 005000          CLR     R0              ;HERE TO START WITH A FL
000640' 005005                  005005  ;173302 005005          CLR     R5              ;...
                                        ;173304          TCBOOT:
000642' 012706                  012706  ;173304 012706          MOV     #RETRY,SP       ;INIT RETRY COUNTER
000644' 000014                  000014  ;173306 000014
000646' 012701                  012701  ;173310 012701          MOV     #TCEPA+TCCM,R1  ;POINT TO COMMAND REGIST
```



```
000650' 177342                  177342  ;173312 177342
                                        ;173314          TCRTRY:
000652' 005705                  005705  ;173314 005705          TST     R5              ;INDEFINITE RETRY?
000654' 100402                  100402  ;173316 100402          BMI     10$             ;YES-- TRY HARDER
000656' 005306                  005306  ;173320 005306          DEC     SP              ;NO-- DECREMENT COUNT
000660' 002427                  002427  ;173322 002427          BLT     TCEHLT          ;TOO MANY-- GIVE UP
                                        ;173324          10$:
000662' 000005                  000005  ;173324 000005          RESET                   ;CLEAR TC11
000664' 010003                  010003  ;173326 010003          MOV     R0,R3           ;GET UNIT NUMBER
000666' 000303                  000303  ;173330 000303          SWAB    R3              ;TO BITS 10-8
000670' 010304                  010304  ;173332 010304          MOV     R3,R4           ;COPY FOR READ FUNCTION
000672' 052703                  052703  ;173334 052703          BIS     #TCREV+TCRNUM+TCGO,R3 ;START TAPE TOWARD
000674' 004003                  004003  ;173336 004003
000676' 010311                  010311  ;173340 010311          MOV     R3,(R1)         ;...
                                        ;173342          20$:
000700' 005711                  005711  ;173342 005711          TST     (R1)            ;ERROR?
000702' 100376                  100376  ;173344 100376          BPL     20$             ;NO-- WAIT FOR END-ZONE
000704' 005761                  005761  ;173346 005761          TST     TCST-TCCM(R1)   ;END-ZONE UP YET?
000706' 177776                  177776  ;173350 177776
000710' 100360                  100360  ;173352 100360          BPL     TCRTRY          ;NO-- MUST BE OTHER ERRO
000712' 012761                  012761  ;173354 012761          MOV     #-256.,TCWC-TCCM(R1) ;SET WORD COUNT
000714' 177400                  177400  ;173356 177400
000716' 000002                  000002  ;173360 000002
000720' 052704                  052704  ;173362 052704          BIS     #TCREAD+TCGO,R4 ;START READ, FORWARD
000722' 000005                  000005  ;173364 000005
000724' 010411                  010411  ;173366 010411          MOV     R4,(R1)         ;...
                                        ;173370          30$:
000726' 105711                  105711  ;173370 105711          TSTB    (R1)            ;TRANSFER DONE?
000730' 100376                  100376  ;173372 100376          BPL     30$             ;NO-- WAIT SOME MORE
000732' 005711                  005711  ;173374 005711          TST     (R1)            ;YES-- ERROR?
000734' 100746                  100746  ;173376 100746          BMI     TCRTRY          ;YES-- RETRY
000736' 000460                  000460  ;173400 000460          BR      CLRPC           ;NO-- DONE-- GOTO LOCATI
                                        ;173402          TCEHLT:
000740' 016100                  016100  ;173402 016100          MOV     TCST-TCCM(R1),R0 ;GET STATUS REGISTER
000742' 177776                  177776  ;173404 177776
000744' 000460                  000460  ;173406 000460          BR      HALTED          ;AND STOP
                                        ;173410          RPBOOT:
000746' 012706                  012706  ;173410 012706          MOV     #RETRY,SP       ;RETRY RETRY TIMES
000750' 000014                  000014  ;173412 000014
000752' 012701                  012701  ;173414 012701          MOV     #RPEPA+RPCS1,R1 ;ADDRESS RPCS1 IN R1
000754' 176700                  176700  ;173416 176700
000756' 012702                  012702  ;173420 012702          MOV     #RPECCI,R2      ;SET ECC INHIBIT, 20 SEC
000760' 004000                  004000  ;173422 004000
                                        ;173424          RPRTRY:
000762' 005705                  005705  ;173424 005705          TST     R5              ;INFINITE RETRY?
000764' 100402                  100402  ;173426 100402          BMI     10$             ;YES-- TRY AGAIN
000766' 005306                  005306  ;173430 005306          DEC     SP              ;RETRY COUNT EXHAUSTED?
000770' 002444                  002444  ;173432 002444          BLT     RPEHLT          ;YES-- GIVE UP
                                        ;173434          10$:
000772' 000005                  000005  ;173434 000005          RESET                   ;ZAP!!
001774' 110061                  110061  ;173436 110061          MOVB    R0,RPCS2(R1)    ;SELECT PROPER UNIT #
001776' 000010                  000010  ;173440 000010
001000' 032711                  032711  ;173442 032711          BIT     #RPDVA,(R1)     ;IS DRIVE AVAILABLE TO U
001002' 004000                  004000  ;173444 004000
001004' 001766                  001766  ;173446 001766          BEQ     RPRTRY          ;NO-- TRY AGAIN
001006' 012711                  012711  ;173450 012711          MOV     #RPPRST+RPGO,(R1) ;DO 'READ-IN PRESET' F
```

```
001010' 000021    000021 ;173452 000021          CLR   RPDC(R1)          ;SET CYLINDER 0
001012' 005061    005061 ;173454 005061
001014' 000034    000034 ;173456 000034          CLR   RPDA(R1)          ; TRACK 0, SECTOR 0
001016' 005061    005061 ;173460 005061
001020' 000006    000006 ;173462 000006
001022' 050261    050261 ;173464 050261          BIS   R2,RPOF(R1)       ;SET INHIBIT ECC, 22-SEC
001024' 000032    000032 ;173466 000032
001026' 012761    012761 ;173470 012761          MOV   #-256.,RPWC(R1) ;SET UP WORD COUNT TO PR
001030' 177400    177400 ;173472 177400
001032' 000002    000002 ;173474 000002
001034' 012711    012711 ;173476 012711          MOV   #RPREAD+RPGO,(R1) ;START READ FUNCTION
001036' 000071    000071 ;173470 000071
                         ;173502         20$:
001040' 105711    105711 ;173502 105711          TSTB  (R1)              ;READY?
001042' 100376    100376 ;173504 100376          BPL   20$               ;NO-- WAIT UNTIL IT IS
001044' 032761    032761 ;173506 032761          BIT   #RPFER,RPER1(R1) ;FORMAT ERROR?
001046' 000020    000020 ;173510 000020
001050' 000014    000014 ;173512 000014
001052' 001403    001403 ;173514 001403          BEQ   30$               ;NO-- TRY AGAIN
001054' 052702    052702 ;173516 052702          BIS   #RPFM22,R2        ;YES-- TRY FOR 22 SECTOR
001056' 010000    010000 ;173520 010000
001060' 000740    000740 ;173522 000740          BR    RPRTRY            ;TRY AGAIN
                         ;173524         30$:
001062' 032711    032711 ;173524 032711          BIT   #RPTRE!RPMCPE,(R1) ;TRANSFER OR MBC PARI
001064' 060000    060000 ;173526 060000
001066' 001335    001335 ;173530 001335          BNE   RPRTRY            ;YES-- ERROR-- TRY AGAIN
001070' 032761    032761 ;173532 032761          BIT   #RPATA!RPERR,RPDS(R1) ;ATTN OR OTHER ERR
001072' 140000    140000 ;173534 140000
001074' 000012    000012 ;173536 000012
001076' 001331    001331 ;173540 001331          BNE   RPRTRY            ;YES-- ERROR-- TRY AGAIN
                         ;173542         CLRPC:
001100' 005007    005007 ;173542 005007          CLR   PC                ;JMP 0
                         ;173544         RPEHLT:
001102' 016100    016100 ;173544 016100          MOV   RPDS(R1),R0       ;DISPLAY DRIVE STATUS
001104' 000012    000012 ;173546 000012
                         ;173550         HALTED:
001106' 000000    000000 ;173550 000000          HALT                    ;DIE
001110' 000776    000776 ;173552 000776          BR    HALTED            ;STAY DEAD
                         ;173554         BUTON4:
001112' 010037    010037 ;173554 010037          MOV   R0,ROTOR7+0       ;SAVE R0 IN 40
001114' 000040    000040 ;173556 000040
001116' 012700    012700 ;173560 012700          MOV   #10$,R0           ;SET RETURN ADDRESS IN R
001120' 173566    173566 ;173562 173566
001122' 000630    000630 ;173564 000630          BR    REGSAV            ;SAVE R1-R7
                         ;173566         10$:
001124' 005005    005005 ;173566 005005          CLR   R5                ;ADDRESS LOCATION ZERO
001126' 012500    012500 ;173570 012500          MOV   (R5)+,R0          ;SAVE 0 IN R0
001130' 012501    012501 ;173572 012501          MOV   (R5)+,R1          ;SAVE 2 IN R1
001132' 011502    011502 ;173574 011502          MOV   (R5),R2           ;SAVE 4 IN R2
001134' 012725    012725 ;173576 012725          MOV   #21$,(R5)+        ;SET NXM TRAP ADDRESS IN
001136' 173612    173612 ;173570 173612
001140' 011503    011503 ;173602 011503          MOV   (R5),R3           ;SAVE 6 IN R3
001142' 005015    005015 ;173604 005015          CLR   (R5)              ;SET PS FOR TRAP
                         ;173606         20$:
001144' 012704    012704 ;173606 012704          MOV   #DLYCNT-DTESIZ,R4 ;POINT TO DTE # -1'S D
001146' 174340    174340 ;173610 174340
```



```
                         ;173612         21$:
001150' 012706    012706 ;173612 012706          MOV   #4,SP             ;SET SP TO 4, STACK IS L
001152' 000004    000004 ;173614 000004
                         ;173616         22$:
001154' 062704    062704 ;173616 062704          ADD   #DTESIZ,R4        ;BUMP TO NEXT DTE'S EXTF
001156' 000040    000040 ;173620 000040
001160' 105704    105704 ;173622 105704          TSTB  R4                ;IS THIS THE END OF THE
001162' 100770    100770 ;173624 100770          BMI   20$               ;YES-- START ALL OVER, U
001164' 032764    032764 ;173626 032764          BIT   #TO11DB,STAT-DLYCNT(R4) ;DOORBELL RINGIN
001166' 004000    004000 ;173630 004000
001170' 000034    000034 ;173632 000034
001172' 001770    001770 ;173634 001770          BEQ   22$               ;NO-- TRY NEXT DTE
001174' 026417    026417 ;173636 026417          CMP   TO10BC-DLYCNT(R4),(PC) ;DOES THIS ONE HA
001176' 000014    000014 ;173640 000014
001200' 001365    001365 ;173642 001365          BNE   22$               ;NO-- TRY ANOTHER DTE
001202' 010315    010315 ;173644 010315          MOV   R3,(R5)           ;RESTORE LOCATION 6
001204' 010245    010245 ;173646 010245          MOV   R2,-(R5)          ; 4
001206' 010145    010145 ;173650 010145          MOV   R1,-(R5)          ; 2
001210' 010045    010045 ;173652 010045          MOV   R0,-(R5)          ; 0
001212' 012700    012700 ;173654 012700          MOV   #DTESAV,R0        ;POINT TO SAVE AREA
001214' 000130    000130 ;173656 000130
                         ;173660         29$:
001216' 012420    012420 ;173660 012420          MOV   (R4)+,(R0)+       ;SAVE A REGISTER
001220' 022700    022700 ;173662 022700          CMP   #TO11DT-DLYCNT+DTESAV,R0 ;FINISHED?
001222' 000156    000156 ;173664 000156
001224' 103374    103374 ;173666 103374          BHIS  29$               ;NO-- SAVE SOME MORE
                         ;173670                 ADDX  DIAG2-TO11DT-2,R4 ;R4 POINTS TO DIAG2 RE
001226' 005724    005724 ;173670 005724          TST   (R4)+
001230' 010401    010401 ;173672 010401          MOV   R4,R1             ; SO DOES R1
001232' 012700    012700 ;173674 012700          MOV   #DRESET,R0        ;SETUP R0 FOR "DIAGNOSTI
001234' 000100    000100 ;173676 000100
001236' 010021    010021 ;173700 010021          MOV   R0,(R1)+          ;R1 POINTS TO STATUS REG
001240' 005061    005061 ;173702 005061          CLR   DLYCNT-STAT(R1) ;SET DTE20 FOR MAXIMUM D
001242' 177744    177744 ;173704 177744
001244' 005061    005061 ;173706 005061          CLR   TO10AD-STAT(R1) ;START DUMPING -11 MEMOR
001246' 177764    177764 ;173710 177764
                         ;173712         30$:
001250' 032711    032711 ;173712 032711          BIT   #TO11DB,(R1)      ;IS DOORBELL RINGING (TR
001252' 004000    004000 ;173714 004000
001254' 001775    001775 ;173716 001775          BEQ   30$               ;NO-- WAIT FOR DOORBELL
001256' 010014    010014 ;173720 010014          MOV   R0,(R4)           ;YES-- CLEAR DOORBELL AN
001260' 005061    005061 ;173722 005061          CLR   TO11AD-STAT(R1) ;START INPUT TO LOCATION
001262' 177766    177766 ;173724 177766
001264' 012761    012761 ;173726 012761          MOV   #IFLOP!<<-256.>&7777>,TO11BC-STAT(R1) ;2
001266' 107400    107400 ;173730 107400
001270' 177762    177762 ;173732 177762
                         ;173734         40$:
001272' 032711    032711 ;173734 032711          BIT   #TO11DN,(R1)      ;TRANSFER COMPLETE?
001274' 000200    000200 ;173736 000200
001276' 001775    001775 ;173740 001775          BEQ   40$               ;NO-- WAIT UNTIL DONE
001300' 005007    005007 ;173742 005007          CLR   PC                ;GO TO LOADED CODE, STAR
                         ;173744                 FILLTO 1000
001302' 000000    000000 ;173744    000          .BYTE 0
                         ;173745    000          .BYTE 0
001304' 000000    000000 ;173746    000          .BYTE 0
                         ;173747    000          .BYTE 0
```

```
   539  001306' 000000                      000000  ;173750  000          .BYTE  0
   540                                              ;173751  000          .BYTE  0
   541  001310' 000000                      000000  ;173752  000          .BYTE  0
   542                                              ;173753  000          .BYTE  0
   543  001312' 000000                      000000  ;173754  000          .BYTE  0
   544                                              ;173755  000          .BYTE  0
   545  001314' 000000                      000000  ;173756  000          .BYTE  0
   546                                              ;173757  000          .BYTE  0
   547  001316' 000000                      000000  ;173760  000          .BYTE  0
   548                                              ;173761  000          .BYTE  0
   549  001320' 000000                      000000  ;173762  000          .BYTE  0
   550                                              ;173763  000          .BYTE  0
   551  001322' 000000                      000000  ;173764  000          .BYTE  0
   552                                              ;173765  000          .BYTE  0
   553  001324' 000000                      000000  ;173766  000          .BYTE  0
   554                                              ;173767  000          .BYTE  0
   555  001326' 000000                      000000  ;173770  000          .BYTE  0
   556                                              ;173771  000          .BYTE  0
   557  001330' 000000                      000000  ;173772  000          .BYTE  0
   558                                              ;173773  000          .BYTE  0
   559  001332' 000000                      000000  ;173774  000          .BYTE  0
   560                                              ;173775  000          .BYTE  0
   561  001334' 000000                      000000  ;173776  000          .BYTE  0
   562                                              ;173777  000          .BYTE  0
   563  001336' 177777                  TABEND: 177777  ;174000
   564
   565          000001                              .END
```


CROSS REFERENCE TABLE -- USER SYMBOLS

```
ACSP     000102R          185#
ADDR     000006R          151#    204
ADDR22=  001000           203#
AGAIN    000230R          207#    225
ASB      000106R          189#    233*
ASTAT    000104R          187#
AWAS     000110R          190#    232*
BEGIN    000000R          148#    223     235
BIT0   = 000001           203#
BIT1   = 000002           203#
BIT10  = 002000           203#
BIT11  = 004000           203#
BIT12  = 010000           203#
BIT13  = 020000           203#
BIT14  = 040000           203#
BIT15  = 100000           203#
BIT2   = 000004           203#
BIT3   = 000010           203#
BIT4   = 000020           203#
BIT5   = 000040           203#
BIT6   = 000100           203#
BIT7   = 000200           203#
BIT8   = 000400           203#
BIT9   = 001000           203#
BMERR    000310R          215     218     230#
BMTAB    000336R          208     245#
BREAK$=  104407           203#
BR1      000012R          153#
BR2      000013R          154#
BTOD$  = 104421           203#
CDATA$=  104412           203#
CONFIG   000056R          173#
CSRA     000100R          183#
DATCK$=  104411           203#
DATER$=  104404           203#    235
DVID1    000014R          155#
ENDIT$=  104413           203#    223
END$   = 104410           203#
ERRTYP   000106R          188#
EXIT$  = 104400           203#
GETPA$=  104415           203#
GWBUF$=  104414           203#
HRDCNT   000044R          168#
HRDER$=  104405           203#
HRDPAS   000050R          170#
ICUNT    000036R          165#
ICUNT    000040R          166#
IONUM    000122R          195#
INIT     000030R          162#
INTP     000120R          194#
MAP22$=  104416           203#
MODNAM   000000R          149#
MODSP    000224R          163     201#
MSGN$  = 104403           203#
MSGS$  = 104402           203#
MSG$   = 104401           203#
```

```
NULL   = 000000            203#
OPEN   = 000000            150     156     157     158     159     176     177     178#    179     180     181     182     183
                           185     187     189     190     192     193     194     203#
OTOA$  = 104420            203#
PASCNT = 000034R           164#
PTWOS  = 000004            203#
POPSP  = 005726            203#
POPSP2= 022626             203#
PRTY   = 000000            153     154     203#
PRTY0  = 000000            203#
PRTY1  = 000040            203#
PRTY2  = 000100            203#
PRTY3  = 000140            203#
PRTY4  = 000200            203#
PRTY5  = 000240            203#
PRTY6  = 000300            203#
PRTY7  = 000340            203#
PS     = 177776            203#
PSW    = 177776            203#
PUSH   = 005746            203#
PUSH2  = 024645            203#
RAND$  = 104417            203#
RANNUM  000054R            172#
RESTRT  000230R            191     206#
RES1    000056R            174#
RES2    000060R            175#
RSTRT   000112R            191#
SBADR   000102R            184#    230*
SOFCNT  000042R            167#
SOFERR$= 104406            203#
SOFPAS  000046R            169#
SPOINT  000032R            163#
SPSIZ  = 000040            16#     196
SR1     000016R            156#
SR2     000020R            157#
SR3     000022R            158#
SR4     000024R            159#
START   000224R            162     204#
STAT    000026R            161#
SVR0    000062R            176#
SVR1    000064R            177#
SVR2    000066R            178#
SVR3    000070R            179#
SVR4    000072R            180#
SVR5    000074R            181#
SVR6    000076R            182#
SYSCNT  000052R            171#
TABEND  001336R            220     563#
TRPDFD= 000022             203#
VECTOR  000010R            152#
WASADR  000104R            186#    231*
WDFR    000116R            193#
WDTO    000114R            192#
XFLAG   000005R            150#
```



```
. ABS.  000000      000
        001340      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XRMFB0,XRMFB0/SOL/CRF:SYM=DDXCOM,XRMFB0
 RUN-TIME:  1  1  .2 SECONDS
 RUN-TIME RATIO: 21/3=5.5
 CORE USED:  7K  (13 PAGES)
```