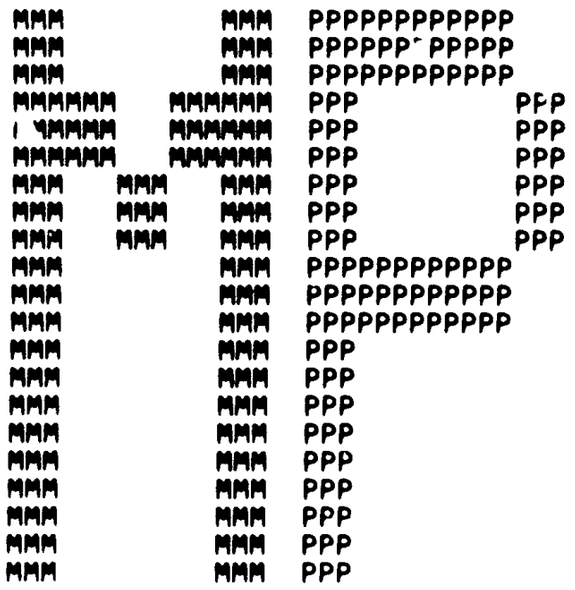
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPP PPPPP
MMM MMM PPPPPPPPPPPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP

```
MM      MM  PPPPPPPP      IIIIII    NN      NN  TTTTTTTTTT
MM      MM  PPPPPPPP      IIIIII    NN      NN  TTTTTTTTTT
MMMM  MMMM  PP      PP      II      NN      NN      TT
MMMM  MMMM  PP      PP      II      NN      NN      TT
MM  MM  MM  PP      PP      II      NNNN    NN      TT
MM  MM  MM  PP      PP      II      NNNN    NN      TT
MM      MM  PPPPPPPP        II      NN  NN  NN      TT
MM      MM  PPPPPPPP        II      NN  NN  NN      TT
MM      MM  PP              II      NN    NNNN      TT
MM      MM  PP              II      NN    NNNN      TT
MM      MM  PP              II      NN      NN      TT        ....
MM      MM  PP              II      NN      NN      TT        ....
MM      MM  PP              II      NN      NN      TT        ....
MM      MM  PP            IIIIII    NN      NN      TT        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1 ;
0000     2 ; Version:      'V04-000'
0000     3 ;
0000     4
0000     5         .MCALL  MFPR
0000     1         .TITLE  MPINT - MULTI-PROCESSOR INTERRUPT HANDLER
0000     2         .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ;
0000    31 ; Facility:  Executive , Hardware fault handling
0000    32 ;
0000    33 ; Abstract:  This module contains the VAX multiport memory interrupt handler.
0000    34 ;
0000    35 ; Environment: MODE=Kernel,  Interrupt
0000    36 ;
0000    37 ; Author:  RICHARD I. HUSTVEDT, Creation date:  15-May-1979
0000    38 ;
0000    39 ; Modified by:
0000    40 ;
0000    41 ;       V03-007 KDM0026         Kathleen D. Morse       14-Oct-1982
0000    42 ;               Conditionalize time-out logic based on debugging switch
0000    43 ;               so that taking a breakpoint on the secondary does not
0000    44 ;               make the primary turn it off.
0000    45 ;
0000    46 ;       V03-006 KDM0018         Kathleen D. Morse       13-Oct-1982
0000    47 ;               Add logic to primary code for secondary wait check
0000    48 ;               request.
0000    49 ;
0000    50 ;       V03-005 KDM0020         Kathleen D. Morse       04-Oct-1982
0000    51 ;               Add time-out logic to primary code that requests the
0000    52 ;               secondary to do an invalidate of a system space address.
```

```
0000     53 ;
0000     54 ;       V03-004 KDM0012         Kathleen D. Morse       20-Sep-1982
0000     55 ;               Add second error log buffer and clear MA780 interrupt
0000     56 ;               before checking for reason of interrupt.
0000     57 ;
0000     58 ; 01    -
0000     59 ;--
```

```
                                                I 4
                     0000     61          .SBTTL  DEFINITIONS
                     0000     62 ;
                     0000     63 ; INCLUDE FILES:
                     0000     64 ;
                     0000     65
                     0000     66 ;
                     0000     67 ; MACROS:
                     0000     68 ;
                     0000     69
                     0000     70 ;
                     0000     71 ; EQUATED SYMBOLS:
                     0000     72 ;
                     0000     73          $BUGDEF                         ; Bugcheck indicator bit definitions
                     0000     74          $EMBDEF                         ; Error log message buffer definitions
                     0000     75          $EMBDEF SS                      ; Sys srv err log message buffer defs
                     0000     76          $IPLDEF                         ; Interrupt priority levels
                     0000     77          $LCKDEF                         ; Interlock bit definitons
                     0000     78          $MPMDEF                         ; Multiport memory registers
                     0000     79          $MPSDEF                         ; Secondary processor state definitions
                     0000     80          $PCBDEF                         ; Define PCB offsets
                     0000     81          $PHDDEF                         ; Define process header offsets
                     0000     82          $PRDEF                          ; Define processor registers
                     0000     83          $PTEDEF                         ; Define page table entry format
                     0000     84          $RPBDEF                         ; Define reboot parameter block
                     0000     85          $VADEF                          ; Virtual address definition
                     0000     86
00000004             0000     87 MAX_PORTS = 4                            ; Maximum number of ports
                     0000     88
                     0000     89 ;
                     0000     90 ; OWN STORAGE:
                     0000     91 ;
                 00000000     92          .PSECT  AEXENONPAGED,LONG       ;
```

```
                                                     J 4
                                0000     94          .SBTTL  MPS$MAINIT - INITIALIZE MULTI-PORT MEMORY ADAPTER
                                0000     95 ;++
                                0000     96 ;
                                0000     97 ; FUNCTIONAL DESCRIPTION:
                                0000     98 ;
                                0000     99 ; This routine is called at system initialization and after a power
                                0000    100 ; recovery restart to initialize the port adapter by clearing any
                                0000    101 ; errors and enabling all interrupts.
                                0000    102 ;
                                0000    103 ;
                                0000    104 ; OUPUTS:
                                0000    105 ;
                                0000    106 ;       Any errors in port are cleared and all interrupts are enabled.
                                0000    107 ;
                                0000    108 ;--
                                0000    109 MPS$MAINIT::                                ;
                          3F    BB    0000    110         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers
                54  0000'CF     D0    0002    111         MOVL    W^MPS$AL_MPMBASE,R4     ; Get base of MPM registers
                00400000 8F     D0    0007    112         MOVL    #MPM$M_CSR_PU,-         ; Clear any power-up status
                          64          000D    113                 MPM$L_CSR(R4)           ;
                                D0    000E    114         MOVL    #MPM$M_CR_ERRS!-        ; Clear any port errors and
                                      000F    115                 MPM$M_CR_MIE,-          ; Enable master interrupt
        04 A4   FF000001 8F           000F    116                 MPM$L_CR(R4)            ;
                                D0    0016    117         MOVL    #MPM$M_SR_SS!-          ; Clear any status errors and
                                      0017    118                 MPM$M_SR_IDL!-          ;  disable error interrupts
                                      0017    119                 MPM$M_SR_IT!-           ;
                                      0017    120                 MPM$M_SR_AGP!-          ;
                                      0017    121                 MPM$M_SR_MXF!-          ;
                                      0017    122                 MPM$M_SR_ACA,-          ;
        08 A4   D000E000 8F           0017    123                 MPM$L_SR(R4)            ;
                  50   0C A4     D0    001E    124         MOVL    MPM$L_INV(R4),R0        ; Get invalidation register
           50   800FFFFF 8F     CA    0022    125         BICL    #^C<MPM$M_INV_STADR>,R0 ; Clear all but starting address
                                      0029    126         ASSUME  MPM$V_INV_ID EQ 0       ; Cached nexus id's start at 0
            0C A4    50    01    C9    0029    127         BISL3   #1@0,R0,MPM$L_INV(R4)   ; Set cpu (nexus 0) as cached
                                D0    002E    128         MOVL    #MPM$M_ERR_ELR!-        ; Clear any errors
                                      002F    129                 MPM$M_ERR_IMP,-         ;
        10 A4   9000000C 8F           002F    130                 MPM$L_ERR(R4)           ;
                00000400 8F     D0    0036    131         MOVL    #MPM$M_CSR1_MIA,-       ; Clear any error
                      18 A4           003C    132                 MPM$L_CSR1(R4)          ;
                      1C A4     D4    003E    133         CLRL    MPM$L_MR(R4)            ; Clear any diagnostic settings
                   50   64     D0    0041    134         MOVL    MPM$L_CSR(R4),R0        ; Get CSR register
                        00     EF    0044    135         EXTZV   #MPM$V_CSR_PORT,-       ; Get port number
                50   50   02          0046    136                 #MPM$S_CSR_PORT,R0,R0   ;
                     50   04     C4    0049    137         MULL    #MAX_PORTS,R0           ; Compute interrupt enable bit #
                     50   10     C0    004C    138         ADDL    #MPM$V_IIE_CTL,R0       ; ...
            24 A4    0F   50     78    004F    139         ASHL    R0,#^XF,MPM$L_IIE(R4)   ; Enable interport interrupts
                                      0054    140                                         ;  from all ports
                          3F    BA    0054    141         POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
                                05    0056    142         RSB                             ; Return
                                      0057    143
```

```
                     0057   145          .SBTTL  MPS$INTPRIM - INTERRUPT PRIMARY PROCESSOR
                     0057   146 ;++
                     0057   147 ; FUNTIONAL DESCRIPTION:
                     0057   148 ;
                     0057   149 ; MPS$INTPRIM is called to cause an interrupt to the primary processor.
                     0057   150 ;
                     0057   151 ; CALLING SEQUENCE:
                     0057   152 ;
                     0057   153 ;       BSB/JSB MPS$INTPRIM
                     0057   154 ;
                     0057   155 ; INPUT PARAMETERS:
                     0057   156 ;
                     0057   157 ;       NONE
                     0057   158 ;
                     0057   159 ; OUTPUT PARAMETERS:
                     0057   160 ;
                     0057   161 ;       NONE
                     0057   162 ;--
                     0057   163
                     0057   164 MPS$INTPRIM::                                ;
0000'DF   0000'CF    D0   0057   165          MOVL    W^MPS$GL_PRIMSKT,@W^MPS$GL_MPMIIR ; Trigger primary interrupt
                     05   005E   166          RSB                                        ; And return
                          005F   167
```

```
                     005F   169          .SBTTL  MPS$INTSCND - INTERRUPT SECONDARY PROCESSOR
                     005F   170 ;++
                     005F   171 ; FUNCTIONAL DESCRIPTION:
                     005F   172 ;
                     005F   173 ; MPS$INTSCND is called to interrupt the secondary processor.
                     005F   174 ;
                     005F   175 ; CALLING SEQUENCE:
                     005F   176 ;
                     005F   177 ;       BSB/JSB MPS$INTSCND
                     005F   178 ;
                     005F   179 ; INPUT PARAMETERS:
                     005F   180 ;
                     005F   181 ;       NONE
                     005F   182 ;
                     005F   183 ; OUTPUT PARAMETERS:
                     005F   184 ;
                     005F   185 ;       NONE
                     005F   186 ;--
                     005F   187
                     005F   188 MPS$INTSCND::                                ;
0000'DF   0000'CF    D0     005F   189          MOVL    W^MPS$GL_SCNDMSKT,@W^MPS$GL_MPMIIR ; Trigger secondary interrupt
                     05     0066   190          RSB                                         ; And return
                            0067   191
```

```
                                0067   193          .SBTTL  MPS$PINTSR - PRIMARY PROCESSOR INTERRUPT SERVICE ROUTINE
                                0067   194 ;++
                                0067   195 ; FUNCTIONAL DESCRIPTION:
                                0067   196 ;
                                0067   197 ; MPS$PINTSR is entered via the interrupt vector for the MA780 in
                                0067   198 ; the primary processor in response to a call to MPS$INTPRIM.
                                0067   199 ;
                                0067   200 ;--
                                0067   201
                                0067   202          .ALIGN  LONG
                                0068   203 MPS$PINTSR::                                ; Primary interrupt service routine
                          50    DD    0068   204          PUSHL   R0                      ; Save R0
                    0000'CF     D0    006A   205          MOVL    W^MPS$AL_MPMBASE,R0     ; Get base of MPM registers
                20  A0  0000'CF     D0    006F   206          MOVL    W^MPS$GL_PRIMSKC,MPM$L_IIR(R0)  ; Clear pending interrupt
                          50  8ED0    0075   207          POPL    R0                      ; Restore R0
                04 0000'CF    00    E7    0078   208          BBCCI   #MPS$V_SECBUGCHK,W^MPS$GL_SECREQFLG,10$ ; Br if no bugchk to do
                                007E   209          .LIST   MEB
                                007E   210          BUG_CHECK MPBADMCK,FATAL        ; Jump to bugcheck code
                             FEFF  007E                      .WORD   ^XFEFF
                             0004' 0080                      .IIF IDN <FATAL>,<FATAL> , .WORD        BUG$_MPBADMCK!4
                         00000080  0082   211 MPS$GW_BUGCHKCOD == .-2                  ; Location for secondary to place the
                                0082   212                                             ;  type of bugcheck it is requesting
                                0082   213          .NLIST  MEB
                46 0000'CF    01    E7    0082   214 10$:     BBCCI   #MPS$V_SECERRLOG,W^MPS$GL_SECREQFLG,50$ ; Br if no errlog to do
                              3F    BB    0088   215          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers
                2F 0000'CF    00    E7    008A   216          BBCCI   #MPS$V_ERLBUF1,W^MPS$GL_ERLBUFIND,30$ ; Br if no entry in buf 1
                   53   0000'CF    9E    0090   217          MOVAB   W^MPS$AL_ERLBUF1,R3     ; Get address of error log entry
                     51    FC  A3    3C    0095   218 20$:     MOVZWL  EMB$W_SIZE(R3),R1       ; Find size of error log entry
                   55    51    10    C3    0099   219          SUBL3   #EMB$C_HD_LENGTH,R1,R5  ; Remember size of entry to move in
                    00000000'GF    16    009D   220          JSB     G^ERL$ALLOCEMB          ; Allocate an error log buffer
                          19  50    E9    00A3   221          BLBC    R0,30$                  ; Br if none available
                       62   63    D0    00A6   222          MOVL    EMB$L_HD_SID(R3),EMB$L_HD_SID(R2) ; Set system ID in error msg
                 04 A2    04  A3    B0    00A9   223          MOVW    EMB$W_HD_ENTRY(R3),EMB$W_HD_ENTRY(R2) ; Set msg type in errlog
                              52    DD    00AE   224          PUSHL   R2                      ; Remember address of buffer
             10 A2    10 A3    55    28    00B0   225          MOVC3   R5,EMB$C_HD_LENGTH(R3),EMB$C_HD_LENGTH(R2) ; Move msg into buf
                              52  8ED0    00B6   226          POPL    R2                      ; Restore buffer address
                    00000000'GF    16    00B9   227          JSB     G^ERL$RELEASEMB         ; Release the error log buffer
                07 0000'CF    01    E7    00BF   228 30$:     BBCCI   #MPS$V_ERLBUF2,W^MPS$GL_ERLBUFIND,40$ ; Br if no entry in buf 2
                   53   0000'CF    9E    00C5   229          MOVAB   W^MPS$AL_ERLBUF2,R3     ; Get address of error log entry
                              C9    11    00CA   230          BRB     20$                     ; Join common code
                              3F    BA    00CC   231 40$:     POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
                                00CE   232 ;
                                00CE   233 ; Nothing to be done at device IPL.  This is either a spurious
                                00CE   234 ; interrupt, or an event flag wait check request from the secondary,
                                00CE   235 ; or a legitimate reschedule request from the secondary.  Cause the
                                00CE   236 ; reschedule software interrupt and check for requested work at that
                                00CE   237 ; IPL.
                                00CE   238 ;
                                00CE   239 50$:     SOFTINT #5                      ; Request IPL 5 interrupt
                             02 00D1   240          REI                             ; And return
                                00D2   241
```

```
                                00D2   243          .SBTTL  MPS$SINTSR - SECONDARY INTERRUPT SERVICE ROUTINE
                                00D2   244 ;++
                                00D2   245 ; FUNCTIONAL DESCRIPTION:
                                00D2   246 ;
                                00D2   247 ; MPS$SINTSR is entered in response to an interrupt on the secondary
                                00D2   248 ; processor.  The interrupt was sent for one of the following reasons:
                                00D2   249 ;
                                00D2   250 ;       1) An AST was sent to the process currently
                                00D2   251 ;          running on the secondary
                                00D2   252 ;          (Primary processor is executing QAST.)
                                00D2   253 ;
                                00D2   254 ;       2) A system space address was invalidated by
                                00D2   255 ;          the primary processor
                                00D2   256 ;          (Primary processor is executing FREWSL or PAGEFAULT.)
                                00D2   257 ;
                                00D2   258 ;       3) The primary wants to bugcheck.
                                00D2   259 ;
                                00D2   260 ; The secondary processor, not knowing which reason the interrupt
                                00D2   261 ; was sent, does the appropriate work to handle all the reasons.
                                00D2   262 ; (Since the code is small, there is no need to figure out the real
                                00D2   263 ; reason for the interrupt.)  The following list corresponds to the
                                00D2   264 ; work done to handle the above conditions causing an interrupt:
                                00D2   265 ;
                                00D2   266 ;       1) The ASTLVL for the process currently running
                                00D2   267 ;          on the secondary is updated
                                00D2   268 ;
                                00D2   269 ;       2) An invalidate is done for the system space
                                00D2   270 ;          address indicated by MPS$GL_INVALID
                                00D2   271 ;
                                00D2   272 ;       3) First, fold up the current process.
                                00D2   273 ;          Second, load the loop address into the RPB.
                                00D2   274 ;          Third, acknowlege the bugcheck request.
                                00D2   275 ;          Fourth, halt to turn off mapping.  Execution continues
                                00D2   276 ;          if restart is enabled, by the console program executing
                                00D2   277 ;          RESTAR.CMD.
                                00D2   278 ;--
                                00D2   279
                                00D2   280          .ALIGN  LONG
                                00D4   281 MPS$SINTSR::                             ; Secondary interrupt service routine
                          50    DD    00D4   282          PUSHL   R0                      ; Save R0
     50    0000'CF    D0    00D6   283          MOVL    W^MPS$AL_MPMBASE,R0     ; Get base of MPM registers
20   A0    0000'CF    D0    00DB   284          MOVL    W^MPS$GL_SCNDMSKC,MPM$L_IIR(R0); Clear pending interrupt
                          50 8ED0   00E1   285          POPL    R0                      ; Restore R0
00 0000'CF    00    E6    00E4   286          BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,5$ ; Flush cache queue
26 0000'CF    00    E0    00EA   287 5$:       BBS     #BUG$V_BUGCHK,W^MPS$GL_BUGCHECK,10$ ; Br if bugcheck requested
4B 0000'CF    00    E0    00F0   288          BBS     #MPS$V_STOPREQ,W^MPS$GL_STOPFLAG,50$ ; Br if STOP/CPU requested
                                00F6   289 ;
                                00F6   290 ; Update the ASTLVL for the process currently running on the secondary.
                                00F6   291 ;
                          50    DD    00F6   292          PUSHL   R0                      ; Save R0
     50    0000'CF    D0    00F8   293          MOVL    W^MPS$GL_CURPCB,R0      ; Get current PCB address
        50    6C    A0    00FD   294          MOVL    PCB$L_PHD(R0),R0        ; Get PHD address
     50    00CF    C0    0101   295          MOVB    PHD$B_ASTLVL(R0),R0     ; And fetch ASTLVL
           13    50    DA    0106   296          MTPR    R0,#PR$_ASTLVL          ; Update current value
                                0109   297 ;
                                0109   298 ; Invalidate the system space address that is contained in MPS$GL_INVALID.
                                0109   299 ;
```

```
                                0109   300          INVALID W^MPS$GL_INVALID        ; Invalidate requested page
                0000'CF   D4    010E   301          CLRL    W^MPS$GL_INVALID        ; And acknowledge it
                     50 8ED0    0112   302          POPL    R0                      ; Restore R0
                        02      0115   303          REI                             ; And continue
                                0116   304
                                0116   305 ;
                                0116   306 ; Primary processor has requested a bugcheck.  The secondary must fold
                                0116   307 ; up the process it is running and loop quietly in a safe place out of
                                0116   308 ; the way of the primary.
                                0116   309 ;
          00 0000'CF    00      E6    0116   310 10$:     BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,20$ ; Flush cache queue
             04   0000'CF       D1    011C   311 20$:     CMPL    W^MPS$GL_STATE,#MPS$K_EXECSTATE ; Was LDPCTX done?
                       01       12    0121   312          BNEQ    30$                     ; Br if not done, don't do SVPCTX
                                07    0123   313          SVPCTX                          ; Save state of current process
          50    00000000'GF     D0    0124   314 30$:     MOVL    G^EXE$GL_RPB,R0         ; Get address of RPB
OOFC CO   60    00000100 8F     C1    012B   315          ADDL3   #RPB$B_WAIT,RPB$L_BASE(R0),RPB$L_BUGCHK(R0) ; Load loop adr
             0000'CF    06      D0    0135   316          MOVL    #MPS$K_STOPSTATE,W^MPS$GL_STATE ; Indicate processor not active
          00 0000'CF    01      E6    013A   317          BBSSI   #BUG$V_ACK1,W^MPS$GL_BUGCHECK,40$ ; Acknowlege bugcheck request
                        00      0140   318 40$:     HALT                            ; This halt causes the secondary to
                                0141   319                                          ; start executing RESTAR.CMD on the
                                0141   320                                          ; console device if restart is enabled.
                                0141   321 ;
                                0141   322 ; A STOP/CPU was issued.  The secondary must return its current process,
                                0141   323 ; if any, load a wait loop into the RPB, and halt.
                                0141   324 ;
          00 0000'CF    00      E6    0141   325 50$:     BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,60$ ; Flush cache queue
             04   0000'CF       D1    0147   326 60$:     CMPL    W^MPS$GL_STATE,#MPS$K_EXECSTATE ; Is there a current process?
                       06       12    014C   327          BNEQ    70$                     ; Br if no current state to save
                                07    014E   328          SVPCTX                          ; Save state of current process
             0000'CF    02      D0    014F   329          MOVL    #MPS$K_DROPSTATE,W^MPS$GL_STATE ; Primary must take process bark
          50    00000000'GF     D0    0154   330 70$:     MOVL    G^EXE$GL_RPB,R0         ; Get address of RPB
OOFC CO   60    00000100 8F     C1    015B   331          ADDL3   #RPB$B_WAIT,RPB$L_BASE(R0),RPB$L_BUGCHK(R0) ; Load loop adr
                0000'CF         D4    0165   332          CLRL    W^MPS$GL_INVALID        ; Indicate no invalidate to wait on
          00 0000'CF    01      E6    0169   333          BBSSI   #MPS$V_STOPACK1,W^MPS$GL_STOPFLAG,80$ ; Acknowlege STOP request
                        00      016F   334 80$:     HALT                            ; Stop the secondary
```

```
                     0170   336          .SBTTL  MPS$INVALID - Relay invalidate request to secondary
                     0170   337 ;++
                     0170   338 ; FUNCTIONAL DESCRIPTION:
                     0170   339 ;
                     0170   340 ; MPS$INVALID relays a translation buffer invalidate request to
                     0170   341 ; the secondary processor and waits for acknowledgement before
                     0170   342 ; proceeding.  Since P0 pages are only referenced by the processor
                     0170   343 ; currently executing a process, only system pages need to be
                     0170   344 ; invalidated by both the primary and secondary processors at
                     0170   345 ; the same time.
                     0170   346 ;
                     0170   347 ; This code is hooked into the pagefault exception handling code.
                     0170   348 ;
                     0170   349 ;--
                     0170   350
                     0170   351 MPS$INVALID::                                ;
          03 A3  84 8F  8A   0170   352          BICB    #<PTE$M_VALID!PTE$M_MODIFY>@-24,3(R3); Clear valid and modify
                     0175   353                                              ; (Replaced instruction)
                     0175   354          INVALID R2                          ; Invalidate for primary processor
             38 52  1F  E1   0178   355          BBC     #VA$V_SYSTEM,R2,60$     ; Only invalidate for system space
                     017C   356          ASSUME  MPS$K_STOPSTATE GT MPS$K_INITSTATE
       00 0000'CF  00  E6   017C   357          BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,10$ ; Flush cache queue
          05 0000'CF  D1   0182   358 10$:     CMPL    W^MPS$GL_STATE,#MPS$K_INITSTATE ; Secondary active?
                       2B  18   0187   359          BGEQ    60$                     ; Br if no, secondary not responding
          0000'CF  52  D0   0189   360          MOVL    R2,W^MPS$GL_INVALID     ; Set address to invalidate
                     018E   361
                     018E   362          .IF     DF,MPPFMSWT
                     018E   363          INCL    W^PFM$L_CNT_INVAL       ; Add one to perf meas invalidate ctr
                     018E   364          .ENDC
                     018E   365
                   FECE  30   018E   366          BSBW    MPS$INTSCND             ; Interrupt secondary processor
                      5A  DD   0191   367          PUSHL   R10                     ; Save R10
       5A  00E4E1C0 8F  D0   0193   368          MOVL    #15000000,R10           ; Initialize time-out counter
                   02 5A  F4   019A   369 20$:     SOBGEQ  R10,30$                 ; Repeat loop, waiting for secondary ack
                     019D   370
                     019D   371          .IF     NDF,MPDBGSWT
                      1B  11   019D   372          BRB     70$                     ; Go log failure and turn off secondary
                     019F   373          .IFF    ;MPDBGSWT DEFINED
                     019F   374          BRB     30$                     ; Don't turn off secondary, just loop
                     019F   375                                          ;  if debugging as breakpoints would
                     019F   376                                          ;  cause the secondary to get turned off
                     019F   377          .ENDC
                     019F   378
       00 0000'CF  00  E6   019F   379 30$:     BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,40$ ; Flush cache queue
          0000'CF  D5   01A5   380 40$:     TSTL    W^MPS$GL_PFAILTIM       ; Has secondary powerfailed?
                   06  12   01A9   381          BNEQ    50$                     ; Br if yes, don't wait for him
                     01AB   382
                     01AB   383          .IF     DF,MPPFMSWT
                     01AB   384          INCL    W^PFM$L_CNT_IWAIT       ; Inc perf meas invalidate loop counter
                     01AB   385          .ENDC
                     01AB   386
          0000'CF  D5   01AB   387          TSTL    W^MPS$GL_INVALID        ; Acknowledged yet?
                   E9  12   01AF   388          BNEQ    20$                     ; No, continue waiting
                      5A 8ED0  01B1   389 50$:     POPL    R10                     ; Restore R10
    00000000'GF   17   01B4   390 60$:     JMP     G^MMG$FRE_TRYSKIP       ; Continue with page fault
                     01BA   391
                     018A   392 ;
```

```
                                01BA   393 ; The secondary did not acknowledge the invalidate request.  Therefore,
                                01BA   394 ; the primary assumes it has died.  A message is placed in the error log
                                01BA   395 ; and an indicator is incremented showing that this failure occurred.
                                01BA   396 ; Then the multi-processing code is unhooked from the running system,
                                01BA   397 ; making the primary ignore any further activity from the secondary.
                                01BA   398 ; The pool space containing the multi-processing code is left untouched
                                01BA   399 ; just in case the secondary is eventually resurrected and tries to
                                01BA   400 ; continue executing.  If this happens, some unexpected interrupt will
                                01BA   401 ; probably be logged by the primary but nothing will have been lost,
                                01BA   402 ; except whatever process the secondary may have been running.
                                01BA   403 ;
                                01BA   404 ; This design allows a gradual degradation of the system to a single
                                01BA   405 ; processor 11/780, instead of forcing a bugcheck.
                                01BA   406 ;
                  0000'CF   D6  01BA   407 70$:    INCL    W^MPS$GL_INV_NACK       ; Indicate secondary did not acknowledge
                        3F  BB  01BE   408         PUSHR   #^M<R0,R1,R2,R3,R4,R5> ; Save registers for MOVC
               51   00'8F  9A  01C0   409         MOVZBL  #MPS$C_INV_NACK,R1      ; Size of ASCII message text
                    51  15  C0  01C4   410         ADDL    #<EMB$K_SS_LENGTH+3>,R1 ; Add in overhead for message
                    51  03  CA  01C7   411         BICL    #3,R1                   ; Buffer size modulo 4
              00000000'GF  16  01CA   412         JSB     G^ERL$ALLOCEMB          ; Allocate error log buffer
                   1E  50  E9  01D0   413         BLBC    R0,80$                  ; If failure, just unhook MP code
                04 A2  27  B0  01D3   414         MOVW    #EMB$C_SS,EMB$W_SS_ENTRY(R2) ; Set type of error log message
           10 A2   0000'8F  B0  01D7   415         MOVW    #MPS$C_INV_NACK,EMB$W_SS_MSGSZ(R2) ; Set size of ASCII text msg
                        52  DD  01DD   416         PUSHL   R2                      ; Save buffer address
    12 A2  0000'CF  0000'8F  28  01DF   417         MOVC    #MPS$C_INV_NACK,W^MPS$T_INV_NACK,EMB$B_SS_MSGTXT(R2) ; Msg txt
                        52 8ED0  01E8   418         POPL    R2                      ; Restore buffer address
              00000000'GF  16  01EB   419         JSB     G^ERL$RELEASEMB         ; Release error log buffer
                               01F1   420
                               01F1   421 ;
                               01F1   422 ; Now unhook the multi-processing code and restore the system to
                               01F1   423 ; a single processor 11/780, vanilla VMS system.
                               01F1   424 ;
 00 00000000'EF   00  E6  01F1   425 80$:    BBSSI   #MPS$V_STOPREQ,MPS$GL_STOPFLAG,90$ ; Indic primary forced a stop
    5A  00000000'GF  D0  01F9   426 90$:    MOVL    G^EXE$GL_MP,R10         ; Get address of MP code
                FDFD'  30  0200   427         BSBW    W^MPS$UNHOOK            ; Unhook MP code from VMS code
              043F 8F  BA  0203   428         POPR    #^M<R0,R1,R2,R3,R4,R5,R10> ; Restore registers
                 FFAA  31  0207   429         BRW     60$                     ; Continue with normal VMS code
```

```
                     020A   431          .SBTTL  MPS$BUGCHECK - Relay bugcheck request to secondary and wait
                     020A   432 ;++
                     020A   433 ; FUNCTIONAL DESCRIPTION:
                     020A   434 ;
                     020A   435 ; MPS$BUGCHECK makes sure that the secondary is out of the way before
                     020A   436 ; the primary procedes with the bugcheck logic.  It sets a flag to
                     020A   437 ; indicate a bugcheck is requested.  Then interrupts the secondary to
                     020A   438 ; make it notice the flag.  The primary then waits for the secondary
                     020A   439 ; to acknowlege the bugcheck request.
                     020A   440 ;
                     020A   441 ; ENVIRONMENT:
                     020A   442 ;
                     020A   443 ;       Executed by the primary processor.
                     020A   444 ;       IPL = 31
                     020A   445 ;
                     020A   446 ;--
                     020A   447
                     020A   448 MPS$BUGCHECK::                                ;
00 0000'CF    00     E6  020A   449          BBSSI   #BUG$V_BUGCHK,W^MPS$GL_BUGCHECK,10$ ; Indicate bugcheck request
                     0210   450          ASSUME  MPS$K_STOPSTATE GT MPS$K_INITSTATE
   05   0000'CF     D1  0210   451 10$:     CMPL    W^MPS$GL_STATE,#MPS$K_INITSTATE ; Is secondary active?
             1B      18  0215   452          BGEQ    50$                     ; Br on not active, don't request bugchk
00 0000'CF    00     E6  0217   453          BBSSI   #BUG$V_BUGCHK,W^MPS$GL_BUGCHECK,20$ ; Indicate bugcheck request
          FE3F      30  021D   454 20$:     BSBW    W^MPS$INTSCND           ; Interrupt secondary to notice request
50   00E4E1C0 8F     D0  0220   455          MOVL    #15000000,R0            ; Wait a significant amount of time
02 0000'CF    01     E7  0227   456 30$:     BBCCI   #BUG$V_ACK1,W^MPS$GL_BUGCHECK,40$ ; Wait for secondary acknowlege
             03      11  022D   457          BRB     50$                     ; Secondary done, continue with bugchk
        F5   50      F4  022F   458 40$:     SOBGEQ  R0,30$                  ; Repeat as secondary not acknowleged
   00000000'9F       17  0232   459 50$:     JMP     @#EXE$INIBOOTADP        ; Continue with normal bugcheck code
```

```
                                0238   461          .SBTTL  MPS$SECBUGCHK - Relay secondary's bugcheck request to primary
                                0238   462 ;++
                                0238   463 ; FUNCTIONAL DESCRIPTION:
                                0238   464 ;
                                0238   465 ; MPS$SECBUGCHK is executed when the secondary processor wants to initiate
                                0238   466 ; a bugcheck.  It sets a flag indicating a bugcheck is requested and
                                0238   467 ; interrupts the primary to make it notice the flag.  The secondary then
                                0238   468 ; waits for the primary to interrupt it with the actual bugcheck request
                                0238   469 ; by executing a self-branch.
                                0238   470 ;
                                0238   471 ; INPUTS:
                                0238   472 ;
                                0238   473 ;       The return address pushed on the stack by calling this routine
                                0238   474 ;       is the address of the bugcheck code being requested.
                                0238   475 ;
                                0238   476 ; OUTPUTS:
                                0238   477 ;
                                0238   478 ;       None
                                0238   479 ;
                                0238   480 ; ENVIRONMENT:
                                0238   481 ;
                                0238   482 ;       Executed by the secondary processor.
                                0238   483 ;
                                0238   484 ;--
                                0238   485
                                0238   486 MPS$SECBUGCHK::                        ;
          FE42 CF    00 BE    B0  0238   487          MOVW    @(SP),W^MPS$GW_BUGCHKCOD ; Set type of bugcheck requested
        50   00000000'GF    D0  023E   488          MOVL    G^EXE$GL_RPB,R0          ; Get address of RPB
00FC C0 60   00000100 8F    C1  0245   489          ADDL3   #RPB$B_WAIT,RPB$L_BASE(R0),RPB$L_BUGCHK(R0) ; Load loop adr
                            024F   490          SETIPL  #IPL$_SYNCH              ; Lower IPL, enabling inter-proc intrpt
         00 0000'CF    00  E6  0252   491          BBSSI   #MPS$V_SECBUGCHK,W^MPS$GL_SECREQFLG,10$ ; Set request flag
                  FDFC  30  0258   492 10$:       BSBW    W^MPS$INTPRIM            ; Interrupt primary processor
         00 0000'CF    00  E6  025B   493          BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,20$ ; Flush cache queue
            05 0000'CF    D1  0261   494 20$:       CMPL    W^MPS$GL_STATE,#MPS$K_INITSTATE ; Secondary active?
                     02   18  0266   495          BGEQ    40$                      ; Br if not active
                     FE   11  0268   496 30$:       BRB     30$                      ; Wait for interrupt from primary to
                          026A   497                                              ;  handle the bugcheck
                     00   026A   498 40$:       HALT                             ; This halt causes the secondary to
                          026B   499                                              ; start executing RESTAR.CMD on the
                          026B   500                                              ; console device if restart is enabled.
                          026B   501
                          026B   502          .END
```

Symbol table

```
BUG$V_ACK1                      =  00000001
BUG$V_BUGCHK                    =  00000000
BUG$_MPBADMCK                      ********    X   02
EMB$B_SS_MSGTXT                 =  00000012
EMB$C_HD_LENGTH                 =  00000010
EMB$C_SS                        =  00000027
EMB$K_SS_LENGTH                 =  00000012
EMB$L_HD_SID                    =  00000000
EMB$W_HD_ENTRY                  =  00000004
EMB$W_SIZE                      =  FFFFFFFC
EMB$W_SS_ENTRY                  =  00000004
EMB$W_SS_MSGSZ                  =  00000010
ERL$ALLOCEMB                       ********    X   02
ERL$RELEASEMB                      ********    X   02
EXE$GL_MP                          ********    X   02
EXE$GL_RPB                         ********    X   02
EXE$INIBOOTADP                     ********    X   02
IPL$_SYNCH                      =  00000008
LCK$V_INTERLOCK                 =  00000000
MAX_PORTS                       =  00000004
MMG$FRE_TRYSKIP                    ********    X   02
MPM$L_CR                        =  00000004
MPM$L_CSR                       =  00000000
MPM$L_CSR1                      =  00000018
MPM$L_ERR                       =  00000010
MPM$L_IIE                       =  00000024
MPM$L_IIR                       =  00000020
MPM$L_INV                       =  0000000C
MPM$L_MR                        =  0000001C
MPM$L_SR                        =  00000008
MPM$M_CR_ERRS                   =  FF000000
MPM$M_CR_MIE                    =  00000001
MPM$M_CSR1_MIA                  =  00000400
MPM$M_CSR_PU                    =  00400000
MPM$M_ERR_ELR                   =  10000000
MPM$M_ERR_IMP                   =  80000000
MPM$M_INV_STADR                 =  7FF00000
MPM$M_SR_ACA                    =  80000000
MPM$M_SR_AGP                    =  10000000
MPM$M_SR_IDL                    =  00004000
MPM$M_SR_IT                     =  00008000
MPM$M_SR_MXF                    =  40000000
MPM$M_SR_SS                     =  00002000
MPM$S_CSR_PORT                  =  00000002
MPM$V_CSR_PORT                  =  00000000
MPM$V_IIE_CTL                   =  00000010
MPM$V_INV_ID                    =  00000000
MPS$AL_ERLBUF1                     ********    X   02
MPS$AL_ERLBUF2                     ********    X   02
MPS$AL_MPMBASE                     ********    X   02
MPS$BUGCHECK                       0000020A RG     02
MPS$C_INV_NACK                     ********    X   02
MPS$GL_BUGCHECK                    ********    X   02
MPS$GL_CURPCB                      ********    X   02
MPS$GL_ERLBUFIND                   ********    X   02
MPS$GL_INTERLOCK                   ********    X   02
MPS$GL_INVALID                     ********    X   02
MPS$GL_INV_NACK                    ********    X   02
MPS$GL_MPMIIR                      ********    X   02
MPS$GL_PFAILTIM                    ********    X   02
MPS$GL_PRIMSKC                     ********    X   02
MPS$GL_PRIMSKT                     ********    X   02
MPS$GL_SCNDMSKC                    ********    X   02
MPS$GL_SCNDMSKT                    ********    X   02
MPS$GL_SECREQFLG                   ********    X   02
MPS$GL_STATE                       ********    X   02
MPS$GL_STOPFLAG                    ********    X   02
MPS$GW_BUGCHKCOD                =  00000080 RG     02
MPS$INTPRIM                        00000057 RG     02
MPS$INTSCND                        0000005F RG     02
MPS$INVALID                        00000170 RG     02
MPS$K_DROPSTATE                 =  00000002
MPS$K_EXECSTATE                 =  00000004
MPS$K_INITSTATE                 =  00000005
MPS$K_STOPSTATE                 =  00000006
MPS$MAINIT                         00000000 RG     02
MPS$PINTSR                         00000068 RG     02
MPS$SECBUGCHK                      00000238 RG     02
MPS$SINTSR                         000000D4 RG     02
MPS$T_INV_NACK                     ********    X   02
MPS$UNHOOK                         ********    X   02
MPS$V_ERLBUF1                   =  00000000
MPS$V_ERLBUF2                   =  00000001
MPS$V_SECBUGCHK                 =  00000000
MPS$V_SECERRLOG                 =  00000001
MPS$V_STOPACK1                  =  00000001
MPS$V_STOPREQ                   =  00000000
PCB$L_PHD                       =  0000006C
PHD$B_ASTLVL                    =  000000CF
PR$_ASTLVL                      =  00000013
PR$_IPL                         =  00000012
PR$_SIRR                        =  00000014
PR$_TBIS                        =  0000003A
PTE$M_MODIFY                    =  04000000
PTE$M_VALID                     =  80000000
RPB$B_WAIT                      =  00000100
RPB$L_BASE                      =  00000000
RPB$L_BUGCHK                    =  000000FC
VA$V_SYSTEM                     =  0000001F
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
AEXENONPAGED	0000026B (619.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.10	00:00:01.15
Command processing	131	00:00:00.85	00:00:04.28
Pass 1	288	00:00:08.89	00:00:28.05
Symbol table sort	0	00:00:01.20	00:00:01.96
Pass 2	109	00:00:02.14	00:00:06.64
Symbol table output	12	00:00:00.09	00:00:00.38
Psect synopsis output	2	00:00:00.02	00:00:00.23
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	576	00:00:13.29	00:00:42.70

The working set limit was 1500 pages.
47795 bytes (94 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 791 non-local and 32 local symbols.
507 source lines were read in Pass 1, producing 17 object records in Pass 2.
30 pages of virtual memory were used to define 29 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MP.OBJ]MP.MLB;1	4
_255DUA28:[SYS.OBJ]LIB.MLB;1	17
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	26

950 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:MPINT/OBJ=OBJ$:MPINT MSRC$:MPPREFIX/UPDATE=(ENH$:MPPREFIX)+MSRC$:MPINT/UPDATE=(ENH$:MPINT)+EXECML$/LIB+LIB$:MP.MLB/LI
```

0248 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MPERRLOG
LIS

MPSCBVEC
LIS

MPINT
LIS

MPPFM
LIS

MPDAT
LIS

MPPWRFAIL
LIS

MPMCHECK
LIS

MPINTEXC
LIS

MPLOG
LIS

MPERRMSG
LIS

MPSCHED
LIS

MPSHWPFM
LIS

MPLOAD
LIS

MPINIT
LIS