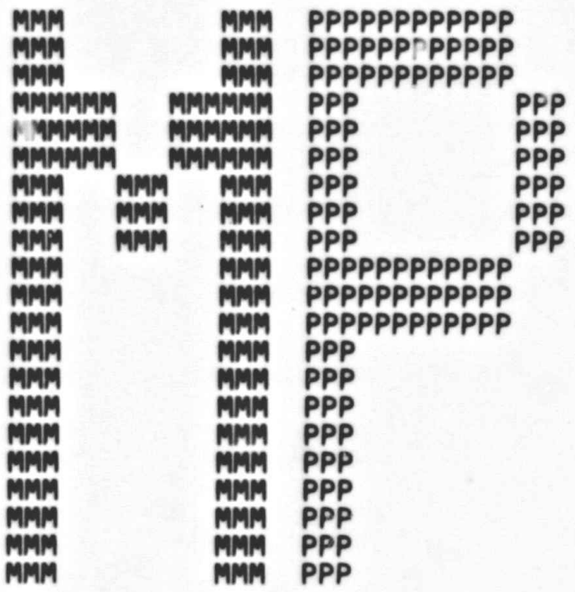
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP

FILEID**MPINIT

```
MM      MM  PPPPPPPP      IIIIII    NN      NN    IIIIII    TTTTTTTTTT
MM      MM  PPPPPPPP      IIIIII    NN      NN    IIIIII    TTTTTTTTTT
MMMM  MMMM  PP      PP      II      NN      NN      II        TT
MMMM  MMMM  PP      PP      II      NN      NN      II        TT
MM  MM  MM  PP      PP      II      NNNN    NN      II        TT
MM  MM  MM  PP      PP      II      NNNN    NN      II        TT
MM      MM  PPPPPPPP        II      NN  NN  NN      II        TT
MM      MM  PPPPPPPP        II      NN  NN  NN      II        TT
MM      MM  PP              II      NN    NNNN      II        TT
MM      MM  PP              II      NN    NNNN      II        TT
MM      MM  PP              II      NN      NN      II        TT        ....
MM      MM  PP              II      NN      NN      II        TT        ....
MM      MM  PP          IIIIII    NN      NN    IIIIII      TT        ....
MM      MM  PP          IIIIII    NN      NN    IIIIII      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  MPINIT - SECONDARY PROCESSOR INITIALIZATION
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:  MPS$INIT contains the routine MPS$START that initializes
0000    34 ;            a secondary processor.
0000    35 ;
0000    36 ; Environment:  MODE=Kernel , IPL=31
0000    37 ;
0000    38 ; Author:  RICHARD I. HUSTVEDT, Creation date:  15-May-1979
0000    39 ;
0000    40 ; Modified by:
0000    41 ;
0000    42 ;       V03-005 KDM0066         Kathleen D. Morse       3-Aug-1983
0000    43 ;               Change PR$_NICR to cpu-specific symbol PR780$_NICR.
0000    44 ;
0000    45 ;       V03-004 KDM0019         Kathleen D. Morse       05-Oct-1982
0000    46 ;               Allow secondary to bugcheck without waiting for primary
0000    47 ;               to set it to INIT state, for all restart codes except
0000    48 ;               power recovery and halt.
0000    49 ;
0000    50 ;       V03-003 KDM0011         Kathleen D. Morse       31-Aug-1982
0000    51 ;               Set AP in MPS$GL_SAVEDAP.
0000    52 ;
```

```
0000     53 ; 01    -
0000     54 ;--
0000     55
```

```
                     0000     57 ;
                     0000     58 ; INCLUDE FILES:
                     0000     59 ;
                     0000     60
                     0000     61 ;
                     0000     62 ; MACROS:
                     0000     63 ;
                     0000     64
                     0000     65 ;
                     0000     66 ; EQUATED SYMBOLS:
                     0000     67 ;
                     0000     68          $IPLDEF                         ; Define interrupt priority levels
                     0000     69          $LCKDEF                         ; Lock bit definitions
                     0000     70          $MPMDEF                         ; Define MA780 registers
                     0000     71          $MPSDEF                         ; Secondary processor states
                     0000     72          $PRDEF                          ; Define processor register numbers
                     0000     73          $PR780DEF                       ; Define 11/780-specific IPR numbers
                     0000     74          $RPBDEF                         ; Define restart parameter block offsets
                     0000     75
00000000             0000     76 TCSR             =       0               ; Offset to terminal transmitter CSR
00000002             0000     77 TDBR             =       2               ; Offset to terminal transmitter DBR
                     0000     78
00000013             0000     79 CONTROL_S        =       19              ; Decimal equivalent for cntl-S
00000011             0000     80 CONTROL_Q        =       17              ; Decimal equivalent for cntl-Q
0000000D             0000     81 CR               =       13              ; Decimal equivalent of carriage-return
0000000A             0000     82 LF               =       10              ; Decimal equivalent of line-feed
                     0000     83
0000006D             0000     84 FPLA_VLOC        =       ^0155           ; Offset to FPLA version number location
0000006A             0000     85 PCS_VLOC         =       ^0152           ; Offset to PCS version number location
0000006B             0000     86 WCSP_VLOC        =       ^0153           ; Offset to WCS primary version location
0000006C             0000     87 WCSS_VLOC        =       ^0154           ; Offset to WCS secondary version loc
                     0000     88
00000003             0000     89 RESTRT_POWERUP   =       3               ; Power recovery restart code
00000004             0000     90 RESTRT_IVLISTK   =       4               ; Interrupt stack not valid
00000005             0000     91 RESTRT_DBLERR    =       5               ; Double error restart code
00000006             0000     92 RESTRT_HALT      =       6               ; Halt restart code
00000007             0000     93 RESTRT_ILLVEC    =       7               ; Illegal vector code
00000008             0000     94 RESTRT_NOUSRWCS  =       8               ; No user WCS restart code
00000009             0000     95 RESTRT_ERRHALT   =       9               ; Error halt restart code
0000000A             0000     96 RESTRT_CHM       =       10              ; CHMx with IS=1 restart code
                     0000     97
                     0000     98          .LIST   MEB                     ; Show macro expansions
```

```
                                0000   100          .SBTTL  EXE$MPSTART - Initialize secondary processor
                                0000   101 ;++
                                0000   102 ; Functional Description:
                                0000   103 ;
                                0000   104 ;       EXE$MPSTART is given control by the boot or restart command file
                                0000   105 ;       for a secondary processor startup.
                                0000   106 ;       Initial entry to EXE$MPSTART is made with memory
                                0000   107 ;       management disabled IPL=31 with the stack pointer set to the high
                                0000   108 ;       end of the page containing the restart control block.
                                0000   109 ;
                                0000   110 ; Calling Sequence:
                                0000   111 ;
                                0000   112 ;       JMP     @RPB$L_MPSTART-^X200(SP)
                                0000   113 ;
                                0000   114 ; Input Parameters:
                                0000   115 ;
                                0000   116 ;       SP - Address of RPB+^x200
                                0000   117 ;
                                0000   118 ;--
                                0000   119
                           00000000   120          .PSECT  $AAEXENONPAGED,PAGE     ; Must be in page aligned psect
                                0000   121 EXE$MPSTART::                           ; Initial entry point
                                0000   122 5$:      MFPR    #PR$_TXCS,R6            ; Get console transmitter status
                     56    22   DB   0000                MFPR    #PR$_TXCS,R6
                   F9 56    07   E1   0003   123          BBC     #7,R6,5$                ; Wait until ready
               23  00000F03 8F   DA   0007   124          MTPR    #^XF03,#PR$_TXDB        ; Send code to clear warmstart inhibit
                                000E   125 6$:      MFPR    #PR$_TXCS,R6            ; Get console transmitter status
                     56    22   DB   000E                MFPR    #PR$_TXCS,R6
                   F9 56    07   E1   0011   126          BBC     #7,R6,6$                ; Wait until console accepts request
                55    01F7'CF   9E   0015   127          MOVAB   W^VERSVECT,R5           ; Get address of version vector
                56    01C4'CF   9E   001A   128          MOVAB   W^MPS$GB_CPUDATA,R6     ; Get address of secondary's cpu data
                                001F   129          MFPR    #PR$_SID,(R6)+          ; Get system ID for secondary
                      86    3E   DB   001F                MFPR    #PR$_SID,(R6)+
                      51    85   9A   0022   130 10$:     MOVZBL  (R5)+,R1                ; Get offset to version code
                            08   13   0025   131          BEQL    30$                     ; 0 ends the list of version codes
                               01A2 30   0027   132 20$:     BSBW    GETCONLOC               ; Ask console for value
                      86    50   90   002A   133          MOVB    R0,(R6)+                ; Store it away
                            F3   11   002D   134          BRB     10$                     ; Repeat for next version code
                55    FE00  CE   9E   002F   135 30$:     MOVAB   -512(SP),R5             ; Compute base of RPB
                54    00AC  C5   D0   0034   136          MOVL    RPB$L_SBR(R5),R4        ; Get base of SPT
                      0C    54   DA   0039   137          MTPR    R4,#PR$_SBR             ; Set SPT base register
                0D    00B8  C5   DA   003C   138          MTPR    RPB$L_SLR(R5),#PR$_SLR  ;  and length register
                                0041   139 ;        MTPR    RPB$L_SISR(R5),#PR$_SISR; Restore Software interrupt state
                11    01C0'CF   DA   0041   140          MTPR    W^MPS$GL_SCBB,#PR$_SCBB ; Restore pointer to System Control Block
                                0046   141 ;        MTPR    RPB$L_PCBB(R5),#PR$_PCBB; Restore pointer to current PCB
                  53    50  A5   D0   0046   142          MOVL    RPB$L_SVASPT(R5),R3     ; Get virtual address of SPT
                51    01B8'CF   D0   004A   143          MOVL    W^MPS$GL_STRTVA,R1      ; VA in this physical page
           51    80000000 8F   CA   004F   144          BICL    #^X80000000,R1          ; Clear system bit
            51     51    F7  8F   78   0056   145          ASHL    #-9,R1,R1               ; and convert to VPN
                50    A2  AF   9E   005B   146          MOVAB   EXE$MPSTART,R0          ; Physical address of EXE$MPSTART
            50  50     F7  8F   78   005F   147          ASHL    #-9,R0,R0               ; Convert to physical page number
                  51    50   C2   0064   148          SUBL    R0,R1                   ; Compute delta VPN-PFN
                  53   6341  DE   0067   149          MOVAL   (R3)[R1],R3             ; Now compute base address for POPT
           09    00010000 8F   DA   006B   150          MTPR    #^X10000,#PR$_P0LR      ; Set dummy P0 length
                      08    53   DA   0072   151          MTPR    R3,#PR$_P0BR            ; Set base for P0 page table
                56    01BC'CF   D0   0075   152          MOVL    W^MPS$GL_ISP,R6         ; Get Saved interrupt stack pointer
                                007A   153          INVALID                         ; Clear translation buffer
```

```
                39   00    DA   007A                          MTPR    #0,S^#PR$_TBIA
                38   01    DA   007D   154                    MTPR    #1,#PR$_MAPEN           ; Enable memory management
                01B8'DF    17   0080   155                    JMP     @W^MPS$GL_STRTVA        ; Set PC to system space
                                0084   156
                                0084   157 MPS$STRTVA::                                     ;
                5E   56    D0   0084   158                    MOVL    R6,SP                   ; Now restore correct stack pointer
                                0087   159
                                0087   160                    .IF     DF,MPDBGSWT
                                0087   161                    BSBW    INI$BRK                 ;***** Initial secondary breakpoint
                                0087   162                    .ENDC
                                0087   163
                03   5C    D1   0087   164                    CMPL    AP,#RESTRT_POWERUP      ; Is this a power recovery?
                     0B    13   008A   165                    BEQL    9$                      ; Br if yes, wait for synch w/primary
                06   5C    D1   008C   166                    CMPL    AP,#RESTRT_HALT         ; Is this a halt restart?
                     06    13   008F   167                    BEQL    9$                      ; Br if yes, wait for synch w/primary
          0D 0000'CF    00    E6   0091   168                    BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,11$ ; Flush cache queue
          00 0000'CF    00    E6   0097   169 9$:                 BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,10$ ; Flush cache queue
             05 0000'CF    D1   009D   170 10$:                  CMPL    W^MPS$GL_STATE,#MPS$K_INITSTATE ; Is secondary ready for init?
                     F3    12   00A2   171                    BNEQ    9$                      ; Loop until primary sets 2ndary ready
                                00A4   172
                0000'CF    5C   D0   00A4   173 11$:           MOVL    AP,W^MPS$GL_SAVEDAP     ; Save value of AP for future use
                                00A9   174
             55  00000004'GF    9E   00A9   175                MOVAB   G^EXE$GB_CPUDATA+4,R5   ; Point past SID field for primary
                56 01C8'CF    9E   00B0   176                    MOVAB   W^MPS$GB_CPUDATA+4,R6   ; Point past SID field for secondary
                86   85    91   00B5   177                    CMPB    (R5)+,(R6)+             ; Check FPLA version number
                     0C    13   00B8   178                    BEQL    12$                     ; Br if secondary does matches primary
                     5B    D4   00BA   179                    CLRL    R11                     ; Indicate console terminal
             51  00000266'EF    9E   00BC   180                MOVAB   FPLA_MISMATCH,R1        ; Get address of error message
                       0191    30   00C3   181                    BSBW    MPS$OUTZSTRING          ; Output message to secondary console
                86   85    91   00C6   182 12$:                 CMPB    (R5)+,(R6)+             ; Check PCS version number
                     0C    13   00C9   183                    BEQL    13$                     ; Br if secondary does matches primary
                     5B    D4   00CB   184                    CLRL    R11                     ; Indicate console terminal
             51  00000299'EF    9E   00CD   185                MOVAB   PCS_MISMATCH,R1         ; Get address of error message
                       0180    30   00D4   186                    BSBW    MPS$OUTZSTRING          ; Output message to secondary console
                   86   85    B1   00D7   187 13$:             CMPW    (R5)+,(R6)+             ; Check WCS version number
                     0C    13   00DA   188                    BEQL    14$                     ; Br if secondary does matches primary
                     5B    D4   00DC   189                    CLRL    R11                     ; Indicate console terminal
             51  000002CA'EF    9E   00DE   190                MOVAB   WCS_MISMATCH,R1         ; Get address of error message
                       016F    30   00E5   191                    BSBW    MPS$OUTZSTRING          ; Output message to secondary console
                01   01C7'CF    91   00E8   192 14$:           CMPB    W^MPS$GB_CPUDATA+3,#1   ; Check that secondary is an 11/780
                     0C    13   00ED   193                    BEQL    15$                     ; Br if secondary is an 11/780
                     5B    D4   00EF   194                    CLRL    R11                     ; Indicate console terminal
             51  000002FB'EF    9E   00F1   195                MOVAB   CPU_NOT_780,R1          ; Get address of error message
                       015C    30   00F8   196                    BSBW    MPS$OUTZSTRING          ; Output message to secondary console
                                00FB   197 15$:
             19  FFFFD8F0 8F    DA   00FB   198                MTPR    #-<10*1000>,S^#PR780$_NICR ; Load next interval register
             18  800000D1 8F    DA   0102   199                MTPR    #^X800000D1,S^#PR$_ICCS ; Clear error and start clock
                       FEF4'    30   0109   200                    BSBW    MPS$MAINIT              ; Initialze multi-port memory
                50   0000'CF    D0   010C   201                    MOVL    W^MPS$AL_MPMBASE,R0     ; Get base for MA780 registers
                     51   60    D0   0111   202                    MOVL    MPM$L_CSR(R0),R1        ; Fetch configuration register
                          00    EF   0114   203                    EXTZV   #MPM$V_CSR_PORT,-       ; Get port number
                51   51   02         0116   204                            #MPM$S_CSR_PORT,R1,R1   ;
             52  00001111 8F    51   78   0119   205                ASHL    R1,#^XT111,R2           ; Generate proper trigger mask
                0000'CF    52    10   78   0121   206                ASHL    #MPM$V_IIR_CTL,R2,W^MPS$GL_SCNDMSKT ; Align and store it
                     51    04   C4   0127   207                    MULL    #4,R1                   ; Compute interrupt enable bit #
                0000'CF    0F   51   78   012A   208                ASHL    R1,#^XF,W^MPS$GL_SCNDMSKC ; Generate clear mask
                50   0000'CF    D0   0130   209                    MOVL    W^MPS$AL_MPMBASE,R0     ; Get base adr of MA780 registers
```

```
                                                     H 3
                20 A0   0000'CF   D0  0135   210          MOVL    W^MPS$GL_SCNDMSKC,MPM$L_IIR(R0) ; Clear any pending interrupt
                                  013B   211          SETIPL  #IPL$_SYNCH              ; Lower IPL for a short time
                        12   08   DA  013B                MTPR    #IPL$_SYNCH,S^#PR$_IPL
                                  013E   212          SETIPL  #IPL$_POWER              ; so that waiting powerfails can occur
                        12   1F   DA  013E                MTPR    #IPL$_POWER,S^#PR$_IPL
             00 0000'CF   00   E6  0141   213          BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,18$ ; Flush cache
             01 0000'CF   00   E1  0147   214 18$:     BBC     #MPS$V_STOPREQ,W^MPS$GL_STOPFLAG,19$ ; Halt if STOP/CPU request
                              00  014D   215          HALT
                                  014E   216 19$:
                03   0000'CF   D1  014E   217          CMPL    W^MPS$GL_SAVEDAP,#RESTRT_POWERUP ; Is this a power recovery?
                           4C   13  0153   218          BEQL    110$                    ; Br if yes, power recovery
                           4F   19  0155   219          BLSS    120$                    ; Br if normal cold startup
                                0157   220          CASE    W^MPS$GL_SAVEDAP,<-     ; Else switch on restart code
                                0157   221                  20$,-                   ; 4 => Interrupt stack not valid
                                0157   222                  30$,-                   ; 5 => CPU double error halt
                                0157   223                  120$,-                  ; 6 => Halt instruction
                                0157   224                  50$,-                   ; 7 => Illegal I/E vector
                                0157   225                  60$,-                   ; 8 => No user WCS
                                0157   226                  70$,-                   ; 9 => Error pending on Halt
                                0157   227                  80$,-                   ;10 => CHM on ISTK halt
                                0157   228                  90$,-                   ;11 => CHM vector <1:0> .NE. 0
                                0157   229                  100$,-                  ;12 => SCB physical read error
                                0157   230                  >,LIMIT=#RESTRT_IVLISTK ;
        08'     04   0000'CF   AF  0157                CASEW   W^MPS$GL_SAVEDAP,#RESTRT_IVLISTK,S^#<<30004$-30003$>/2>-1
                                015D        30003$:
                          0017' 015D                  .SIGNED_WORD    20$-30003$
                          001C' 015F                  .SIGNED_WORD    30$-30003$
                          0049' 0161                  .SIGNED_WORD    120$-30003$
                          0026' 0163                  .SIGNED_WORD    50$-30003$
                          002B' 0165                  .SIGNED_WORD    60$-30003$
                          0030' 0167                  .SIGNED_WORD    70$-30003$
                          0035' 0169                  .SIGNED_WORD    80$-30003$
                          003A' 016B                  .SIGNED_WORD    90$-30003$
                          003F' 016D                  .SIGNED_WORD    100$-30003$
                                016F        30004$:
                                016F   231          SECBUG_CHECK    MPUNKRSTRT,FATAL ; Unknown restart code
                    FE8E'  30   016F                        BSBW    W^MPS$SECBUGCHK
                          0004' 0172                        .IIF    IDN <FATAL>,<FATAL> , .WORD BUG$_MPUNKRSTRT!4
                                0174   232 20$:     SECBUG_CHECK    MPIVLISTK,FATAL ; Invalid interrupt stack
                    FE89'  30   0174                        BSBW    W^MPS$SECBUGCHK
                          0004' 0177                        .IIF    IDN <FATAL>,<FATAL> , .WORD BUG$_MPIVLISTK!4
                                0179   233 30$:     SECBUG_CHECK    MPDBLERR,FATAL  ; Double error halt
                    FE84'  30   0179                        BSBW    W^MPS$SECBUGCHK
                          0004' 017C                        .IIF    IDN <FATAL>,<FATAL> , .WORD BUG$_MPDBLERR!4
                                017E   234 40$:     SECBUG_CHECK    MPHALT,FATAL    ; Halt instruction
                    FE7F'  30   017E                        BSBW    W^MPS$SECBUGCHK
                          0004' 0181                        .IIF    IDN <FATAL>,<FATAL> , .WORD BUG$_MPHALT!4
                                0183   235 50$:     SECBUG_CHECK    MPILLVEC,FATAL  ; Illegal Vector code
                    FE7A'  30   0183                        BSBW    W^MPS$SECBUGCHK
                          0004' 0186                        .IIF    IDN <FATAL>,<FATAL> , .WORD BUG$_MPILLVEC!4
                                0188   236 60$:     SECBUG_CHECK    MPNOUSRWCS,FATAL ; No user WCS for vector
                    FE75'  30   0188                        BSBW    W^MPS$SECBUGCHK
                          0004' 018B                        .IIF    IDN <FATAL>,<FATAL> , .WORD BUG$_MPNOUSRWCS!4
                                018D   237 70$:     SECBUG_CHECK    MPERRHALT,FATAL ; Error pending on halt
                    FE70'  30   018D                        BSBW    W^MPS$SECBUGCHK
                          0004' 0190                        .IIF    IDN <FATAL>,<FATAL> , .WORD BUG$_MPERRHALT!4
                                0192   238 80$:     SECBUG_CHECK    MPCHMONIS,FATAL ; CHM on interrupt stack
```

```
                FE6B'  30  0192                      BSBW    W^MPS$SECBUGCHK
                    0004' 0195                       .IIF IDN <FATAL>,<FATAL> , .WORD BUG$_MPCHMONIS!4
                          0197  239 90$:    SECBUG_CHECK    MPCHMVEC,FATAL  ; CHM vector <1:0> .NE. 0
                FE66'  30  0197                      BSBW    W^MPS$SECBUGCHK
                    0004' 019A                       .IIF IDN <FATAL>,<FATAL> , .WORD BUG$_MPCHMVEC!4
                          019C  240 100$:   SECBUG_CHECK    MPSCBRDERR,FATAL ; SCB physical read error.
                FE61'  30  019C                      BSBW    W^MPS$SECBUGCHK
                    0004' 019F                       .IIF IDN <FATAL>,<FATAL> , .WORD BUG$_MPSCBRDERR!4
                FE5C'  30  01A1  241 110$:   BSBW    MPS$WARMSTART           ; Log power recovery in the error log
                  03   11  01A4  242           BRB     130$                    ; Continue with common code
                FE57'  30  01A6  243 120$:   BSBW    MPS$COLDSTART           ; Log normal boot of secondary
            0000'CF    D4  01A9  244 130$:   CLRL    W^MPS$GL_PFAILTIM       ; Indicate no power fail in progress
                          01AD  245           SETIPL  #IPL$_SYNCH             ; Drop IPL
                12  08    DA  01AD                      MTPR    #IPL$_SYNCH,S^#PR$_IPL
    0000'CF     01    D0  01B0  246           MOVL    #MPS$K_IDLESTATE,W^MPS$GL_STATE ; Indicate ready for work
                FE48'  31  01B5  247           BRW     MPS$MPSCHED1            ; Go ask for something to do
                          01B8  248 ;
                          01B8  249 ; These data fields are accessed by the secondary initialization
                          01B8  250 ; routine before it has memory management enabled.  They MUST reside
                          01B8  251 ; in the same physical page as the code, since non-paged pool is not
                          01B8  252 ; guaranteed to be physically contiguous.
                          01B8  253 ;
                          01B8  254 MPS$GL_STRTVA::                         ; Virtual address of starting instr
     00000084'            01B8  255           .LONG   MPS$STRTVA             ;
                          01BC  256 MPS$GL_ISP::                            ; Start of interrupt stack
     00000000'            01BC  257           .LONG   MPS$AL_INTSTK          ;
                          01C0  258 MPS$GL_SCBB::                           ; Physical address of SCB base
     00000000             01C0  259           .LONG   0                      ;
                          01C4  260 MPS$GB_CPUDATA::                        ; Secondary processor cpu data
     00000000             01C4  261           .LONG   0                      ;  SID
                          01C8  262 CPUVERS:
     00000000             01C8  263           .LONG   0                      ;  FPLA, PCS, WCS versions
     00000004             01CC  264 CPUVERSLEN=.-CPUVERS
```

```
                                01CC   266          .SBTTL  GETCONLOC - Routine to read console information location
                                01CC   267 ;++
                                01CC   268 ;
                                01CC   269 ; Functional Description:
                                01CC   270 ;
                                01CC   271 ;       GETCONLOC is used to access the locations in console memory containing
                                01CC   272 ;       values such as WCS and FPLA version numbers.
                                01CC   273 ;
                                01CC   274 ; Input Parameters:
                                01CC   275 ;
                                01CC   276 ;       R1 - Location code
                                01CC   277 ;
                                01CC   278 ; Output Parameters:
                                01CC   279 ;
                                01CC   280 ;       R0 - Value contained in console cell
                                01CC   281 ;--
                                01CC   282 GETCONLOC:                                           ;
      51   0300 C1   9E        01CC   283          MOVAB   ^X300(R1),R1                        ; Set code to read console memory
                                01D1   284 10$:    MFPR    #PR$_TXCS,R0                        ; Get transmit status register
           50   22   DB        01D1               MFPR    #PR$_TXCS,R0
        F9 50   07   E1        01D4   285          BBC     #7,R0,10$                           ; Wait for done
           23   51   DA        01D8   286          MTPR    R1,#PR$_TXDB                        ; Request data from console
                                01DB   287 20$:    MFPR    #PR$_TXCS,R0                        ; Read transmit status register
           50   22   DB        01DB               MFPR    #PR$_TXCS,R0
        F9 50   07   E1        01DE   288          BBC     #7,R0,20$                           ; Wait for done
                                01E2   289 30$:    MFPR    #PR$_RXCS,R0                        ; Get receiver status
           50   20   DB        01E2               MFPR    #PR$_RXCS,R0
        F9 50   07   E1        01E5   290          BBC     #7,R0,30$                           ; And wait for done
                                01E9   291          MFPR    #PR$_RXDB,R0                        ; Now read data value
           50   21   DB        01E9               MFPR    #PR$_RXDB,R0
   03   50   04   08   ED      01EC   292          CMPZV   #8,#4,R0,#3                         ; Is this a valid response?
                                01F1   293 ;        BNEQ    10$                             ; No, try again
                           01   01F1   294          NOP                                     ;****TEMP
                           01   01F2   295          NOP                                     ;****TEMP, until 780 console works
   50   50   9A        05   01F3   296          MOVZBL  R0,R0                               ; Zero extend data
                                01F6   297          RSB                                     ;
                                01F7   298 VERSVECT:                                        ; Vector of version offsets
                           6D   01F7   299          .BYTE   FPLA_VLOC                       ; FPLA Version offset
                           6A   01F8   300          .BYTE   PCS_VLOC                        ; PCS Version offset
                           6C   01F9   301          .BYTE   WCSS_VLOC                       ; WCS Secondary version offset
                           6B   01FA   302          .BYTE   WCSP_VLOC                       ; WCS Primary version offset
                 00000004   01FB   303 VERSVECLEN=.-VERSVECT
                                01FB   304          ASSUME VERSVECLEN EQ CPUVERSLEN
                           00   01FB   305          .BYTE   0                               ; End of list
                                01FC   306 ONE_PAGE:
                                01FC   307          ASSUME <ONE_PAGE - EXE$MPSTART> LE 512
```

```
                                01FC    309          .SBTTL MPS$OUTCHAR - Output character
                                01FC    310 ;+
                                01FC    311 ;
                                01FC    312 ; Functional Description:
                                01FC    313 ;
                                01FC    314 ;       This routine is called via a JSB to output a character to a
                                01FC    315 ;       specified device.
                                01FC    316 ;
                                01FC    317 ; Inputs:
                                01FC    318 ;
                                01FC    319 ;       R0 = Character to output
                                01FC    320 ;       R11 = Output CSR address (0 implies console terminal)
                                01FC    321 ;
                                01FC    322 ; OUTPUTS:
                                01FC    323 ;
                                01FC    324 ;       Character is output to the specified device.  If the character
                                01FC    325 ;       is a carriage return and the output device is the console terminal,
                                01FC    326 ;       then a sufficient number of fill characters are also output.
                                01FC    327 ;-
                                01FC    328
                                01FC    329 MPS$OUTCHAR::                           ; Output a character
                      51    DD  01FC    330          PUSHL   R1                     ; Get a scratch register
                      5B    D5  01FE    331          TSTL    R11                    ; Is this the console terminal?
                      15    12  0200    332          BNEQ    20$                    ; Br if not console terminal
                                0202    333          MFPR    #PR$_RXCS,R1           ; Read receive control register
                            51  20  DB  0202                   MFPR    #PR$_RXCS,R1
                        31  51  07  E0  0205    334          BBS     #7,R1,80$              ; Branch if received a character
                                0209    335 10$:
                                0209    336          MFPR    #PR$_TXCS,R1           ; Read transmit control register
                            51  22  DB  0209                   MFPR    #PR$_TXCS,R1
                         F9 51  07  E1  020C    337          BBC     #7,R1,10$              ; Loop until ready
                            F7  13  0210    338          BEQL    10$                    ; Br if not ready
                            23  50  DA  0212    339          MTPR    R0,#PR$_TXDB           ; Output character
                                0B  11  0215    340          BRB     30$                    ;
                      6B    0080  8F  B3  0217    341 20$:     BITW    #^X080,TCSR(R11)       ; Is device ready?
                                F9  13  021C    342          BEQL    20$                    ; Br if not ready
                            02  AB  50  90  021E    343          MOVB    R0,TDBR(R11)           ; Output character
                                50  0D  91  0222    344 30$:     CMPB    #CR,R0                 ; Is this a carriage return?
                                    0F  12  0225    345          BNEQ    60$                    ; Br if not
                                    5B  D5  0227    346          TSTL    R11                    ; Is this the console terminal?
                                    0B  12  0229    347          BNEQ    60$                    ; Br if not
                                    50  D4  022B    348 40$:     CLRL    R0                     ; Set fill character
                                    02  DD  022D    349          PUSHL   #2                     ; Set fill count
                                    CB  10  022F    350 50$:     BSBB    MPS$OUTCHAR            ; Output a fill character
                                 FB 6E  F5  0231    351          SOBGTR  (SP),50$               ; Any more fills to output?
                                    8E  D5  0234    352          TSTL    (SP)+                  ; Clean stack
                                    51  8ED0  0236    353 60$:     POPL    R1                     ; Restore scratch register
                                        05  0239    354          RSB
                                            023A    355 ;
                                            023A    356 ; Received an input character while output was taking place.  Check to see
                                            023A    357 ; if it was an XOFF (Control-S) character.
                                            023A    358 ;
                                            023A    359 80$:
                                            023A    360          MFPR    #PR$_RXDB,R1           ; Get the character
                                51  21  DB  023A                   MFPR    #PR$_RXDB,R1
                    13  51  07  00  ED  023D    361          CMPZV   #0,#7,R1,#CONTROL_S    ; Is it a control-S?
                                C5  12  0242    362          BNEQ    10$                    ; Br on not, to output
```

```
                                0244   363 ;
                                0244   364 ; Received an XOFF (Control-S) character.  Wait until receiving an XON
                                0244   365 ; before continuing.  Throw away any input characters that arrive before
                                0244   366 ; the XON (Control-Q).
                                0244   367 ;
                                0244   368 90$:
                                0244   369          MFPR    #PR$_RXCS,R1                ; Have we received a character?
                          51    20   DB  0244               MFPR    #PR$_RXCS,R1
                      F9  51    07   E1  0247   370          BBC     #7,R1,90$           ; Br on no, loop until we have
                                0248   371          MFPR    #PR$_RXDB,R1                ; Get the input character
                          51    21   DB  0248               MFPR    #PR$_RXDB,R1
                  11  51    07   00   ED  024E   372          CMPZV   #0,#7,R1,#CONTROL_Q ; Is it a control-Q?
                              EF   12      0253   373          BNEQ    90$                 ; Br on no, go wait for another char
                              B2   11      0255   374          BRB     10$                 ; Got it.  Now continue output
```

```
                     0257   376          .SBTTL   MPS$OUTZSTRING - OUTPUT ZERO TERMINATED STRING
                     0257   377 ;+
                     0257   378 ;
                     0257   379 ; Functional Description:
                     0257   380 ;
                     0257   381 ;       This routine is called via a JSB to output a string that
                     0257   382 ;       is terminated by a zero byte.
                     0257   383 ;
                     0257   384 ; Inputs:
                     0257   385 ;
                     0257   386 ;       R1 = Address of zero terminated string
                     0257   387 ;       R11 = Output device CSR address
                     0257   388 ;
                     0257   389 ; OUTPUTS:
                     0257   390 ;
                     0257   391 ;       Characters from the specified string are output until a
                     0257   392 ;       zero byte is encountered.
                     0257   393 ;-
                     0257   394
                     0257   395 MPS$OUTZSTRING::                        ; Output zero terminated string
52   FF 8F   9A      0257   396          MOVZBL  #255,R2                 ; Set maximum allowable string length
   50   81   9A      025B   397 10$:     MOVZBL  (R1)+,R0                ; Get next character from input string
        05   13      025E   398          BEQL    20$                     ; Br if end of string
        9A   10      0260   399          BSBB    MPS$OUTCHAR             ; Output character
     F6 52   F5      0262   400          SOBGTR  R2,10$                  ; Any more characters to output?
             05      0265   401 20$:     RSB                             ;
```

```
                                                0266    403          .SBTTL  Secondary processor's error messages
                                                0266    404 ;
                                                0266    405 ; This is the ascii text for all error messages output to by
                                                0266    406 ; the secondary processor, on its console terminal.
                                                0266    407 ;
                                                0266    408 FPLA_MISMATCH:
                                          0A 0D 0266    409          .ASCII  <CR><LF>
20 2C 41 4C 50 46 2D 46 2D 50 4D 25     0268    410          .ASCIZ  \%MP-F-FPLA, FPLA mismatch with primary processor\
63 74 61 6D 73 69 6D 20 41 4C 50 46     0274
61 6D 69 72 70 20 68 74 69 77 20 68     0280
72 6F 73 73 65 63 6F 72 70 20 79 72     028C
                                    00  0298
                                        0299    411 PCS_MISMATCH:
                                          0A 0D 0299    412          .ASCII  <CR><LF>
50 20 2C 53 43 50 2D 46 2D 50 4D 25     029B    413          .ASCIZ  \%MP-F-PCS, PCS mismatch with primary processor\
20 68 63 74 61 6D 73 69 6D 20 53 43     02A7
79 72 61 6D 69 72 70 20 68 74 69 77     02B3
   00 72 6F 73 73 65 63 6F 72 70 20     02BF
                                        02CA    414 WCS_MISMATCH:
                                          0A 0D 02CA    415          .ASCII  <CR><LF>
57 20 2C 53 43 57 2D 46 2D 50 4D 25     02CC    416          .ASCIZ  \%MP-F-WCS, WCS mismatch with primary processor\
20 68 63 74 61 6D 73 69 6D 20 53 43     02D8
79 72 61 6D 69 72 70 20 68 74 69 77     02E4
   00 72 6F 73 73 65 63 6F 72 70 20     02F0
                                        02FB    417 CPU_NOT_780:
                                          0A 0D 02FB    418          .ASCII  <CR><LF>
73 20 2C 53 43 57 2D 46 2D 50 4D 25     02FD    419          .ASCIZ  \%MP-F-WCS, secondary processor is not an 11/780\
6F 72 70 20 79 72 61 64 6E 6F 63 65     0309
6F 6E 20 73 69 20 72 6F 73 73 65 63     0315
00 30 38 37 2F 31 31 20 6E 61 20 74     0321
```

```
                                B 4
                     032D   421          .SBTTL  INI$BRK Initial Breakpoint
                     032D   422 ;++
                     032D   423 ; FUNCTIONAL DESCRIPTION:
                     032D   424 ;
                     032D   425 ;          INI$BRK is a routine to give control to XDELTA for debugging purposes.
                     032D   426 ;--
                     032D   427
                     032D   428 INI$BRK::                                    ; Initial breakpoint
               03    032D   429          BPT                                ;
               05    032E   430          RSB                                ;
                     032F   431
                     032F   432
                     032F   433
                     032F   434
                     032F   435 ;++
                     032F   436 ; FUNCTIONAL DESCRIPTION:
                     032F   437 ;
                     032F   438 ; MPS$XDELTAINT is a routine to allow XDELTA to get control
                     032F   439 ; via the software interrupt ^XF.
                     032F   440 ;
                     032F   441 ;--
                     032F   442
                     032F   443          .ALIGN  LONG
                     0330   444 MPS$XDELTAINT::                              ; XDELTA software interrupt routine
          FB   10    0330   445          BSBB    INI$BRK                    ; Call breakpoint routine
               02    0332   446          REI                                ; Return from software interrupt
                     0333   447          .END
```

```
BUG$_MPCHMONIS          ********  X   02
BUG$_MPCHMVEC           ********  X   02
BUG$_MPDBLERR           ********  X   02
BUG$_MPERRHALT          ********  X   02
BUG$_MPHALT             ********  X   02
BUG$_MPILLVEC           ********  X   02
BUG$_MPIVLISTK          ********  X   02
BUG$_MPNOUSRWCS         ********  X   02
BUG$_MPSCBRDERR         ********  X   02
BUG$_MPUNKRSTRT         ********  X   02
CONTROL_Q             = 00000011
CONTROL_S             = 00000013
CPUVERS                 000001C8  R   02
CPUVERSLEN            = 00000004
CPU_NOT_780             000002FB  R   02
CR                    = 0000000D
EXE$GB_CPUDATA          ********  X   02
EXE$MPSTART             00000000 RG   02
FPLA_MISMATCH           00000266  R   02
FPLA_VLOC             = 0000006D
GETCONLOC               000001CC  R   02
INI$BRK                 0000032D RG   02
IPL$_POWER            = 0000001F
IPL$_SYNCH            = 00000008
LCK$V_INTERLOCK       = 00000000
LF                    = 0000000A
MPM$L_CSR             = 00000000
MPM$L_IIR             = 00000020
MPM$S_CSR_PORT        = 00000002
MPM$V_CSR_PORT        = 00000000
MPM$V_IIR_CTL         = 00000010
MPS$AL_INTSTK           ********  X   02
MPS$AL_MPMBASE          ********  X   02
MPS$COLDSTART           ********  X   02
MPS$GB_CPUDATA          000001C4  RG  02
MPS$GL_INTERLOCK        ********  X   02
MPS$GL_ISP              000001BC  RG  02
MPS$GL_PFAILTIM         ********  X   02
MPS$GL_SAVEDAP          ********  X   02
MPS$GL_SCBB             000001C0  RG  02
MPS$GL_SCNDMSKC         ********  X   02
MPS$GL_SCNDMSKT         ********  X   02
MPS$GL_STATE            ********  X   02
MPS$GL_STOPFLAG         ********  X   02
MPS$GL_STRTVA           000001B8  RG  02
MPS$K_IDLESTATE       = 00000001
MPS$K_INITSTATE       = 00000005
MPS$MAINIT              ********  X   02
MPS$MPSCHED1            ********  X   02
MPS$OUTCHAR             000001FC  RG  02
MPS$OUTZSTRING          00000257  RG  02
MPS$SECBUGCHK           ********  X   02
MPS$STRTVA              00000084  RG  02
MPS$V_STOPREQ         = 00000000
MPS$WARMSTART           ********  X   02
MPS$XDELTAINT           00000330  RG  02
ONE_PAGE                000001FC  R   02
PCS_MISMATCH            00000299  R   02
PCS_VLOC              = 0000006A
PR$_ICCS              = 00000018
PR$_IPL               = 00000012
PR$_MAPEN             = 00000038
PR$_P0BR              = 00000008
PR$_P0LR              = 00000009
PR$_RXCS              = 00000020
PR$_RXDB              = 00000021
PR$_SBR               = 0000000C
PR$_SCBB              = 00000011
PR$_SID               = 0000003E
PR$_SLR               = 0000000D
PR$_TBIA              = 00000039
PR$_TXCS              = 00000022
PR$_TXDB              = 00000023
PR780$_NICR           = 00000019
RESTRT_CHM            = 0000000A
RESTRT_DBLERR         = 00000005
RESTRT_ERRHALT        = 00000009
RESTRT_HALT           = 00000006
RESTRT_ILLVEC         = 00000007
RESTRT_IVLISTK        = 00000004
RESTRT_NOUSRWCS       = 00000008
RESTRT_POWERUP        = 00000003
RPB$L_SBR             = 000000AC
RPB$L_SLR             = 000000B8
RPB$L_SVASPT          = 00000050
TCSR                  = 00000000
TDBR                  = 00000002
VERSVECLEN            = 00000004
VERSVECT                000001F7  R   02
WCSP_VLOC             = 0000006B
WCSS_VLOC             = 0000006C
WCS_MISMATCH            000002CA  R   02
```

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
$AAEXENONPAGED	00000333 (819.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.09	00:00:00.36
Command processing	129	00:00:00.89	00:00:05.45
Pass 1	236	00:00:06.15	00:00:17.53
Symbol table sort	0	00:00:00.52	00:00:00.90
Pass 2	106	00:00:01.65	00:00:05.37
Symbol table output	12	00:00:00.07	00:00:00.14
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	524	00:00:09.40	00:00:29.79

The working set limit was 1500 pages.
36134 bytes (71 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 380 non-local and 41 local symbols.
452 source lines were read in Pass 1, producing 17 object records in Pass 2.
20 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MP.OBJ]MP.MLB;1	16
_255DUA28:[SYS.OBJ]LIB.MLB;1	6
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	28

736 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:MPINIT/OBJ=OBJ$:MPINIT MSRC$:MPPREFIX/UPDATE=(ENH$:MPPREFIX)+MSRC$:MPINIT/UPDATE=(ENH$:MPINIT)+EXECML$/LIB+LIB$:MP.ML
```

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