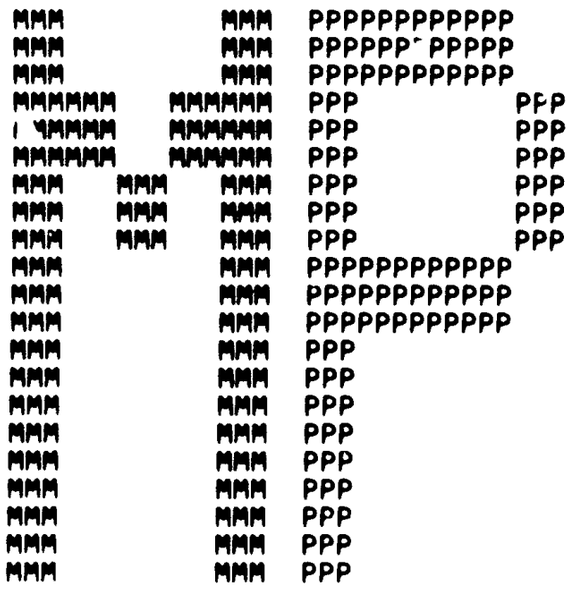
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPP PPPPP
MMM MMM PPPPPPPPPPPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM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

FILEID**MPLOAD

```
MM      MM  PPPPPPPP    LL            OOOOOO      AAAAAA    DDDDDDDD
MM      MM  PPPPPPPP    LL            OOOOOO      AAAAAA    DDDDDDDD
MMMM  MMMM  PP      PP  LL          OO      OO  AA      AA  DD      DD
MMMM  MMMM  PP      PP  LL          OO      OO  AA      AA  DD      DD
MM  MM  MM  PP      PP  LL          OO      OO  AA      AA  DD      DD
MM  MM  MM  PP      PP  LL          OO      OO  AA      AA  DD      DD
MM      MM  PPPPPPPP    LL          OO      OO  AA      AA  DD      DD
MM      MM  PPPPPPPP    LL          OO      OO  AA      AA  DD      DD
MM      MM  PP          LL          OO      OO  AAAAAAAAAA  DD      DD
MM      MM  PP          LL          OO      OO  AAAAAAAAAA  DD      DD
MM      MM  PP          LL          OO      OO  AA      AA  DD      DD      ....
MM      MM  PP          LL          OO      OO  AA      AA  DD      DD      ....
MM      MM  PP          LLLLLLLLLL    OOOOOO    AA      AA  DDDDDDDD        ....
MM      MM  PP          LLLLLLLLLL    OOOOOO    AA      AA  DDDDDDDD        ....

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  MPLOAD - LOAD AND CONNECT CODE FOR MULTIPROCESSING
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 ; Facility:  Executive , Hardware fault handling
0000    31 ;
0000    32 ; Abstract:
0000    33 ;
0000    34 ;       MPLOAD is the main control module for the installation of
0000    35 ;       the code required for multiprocessing.  It changes mode to kernel,
0000    36 ;       acquires sufficient pool space and moves the code into it.
0000    37 ;       With all interrupts disabled, the necessary hooks are inserted
0000    38 ;       into the exec to connect several replacement routines.
0000    39 ;
0000    40 ; Environment: MODE=Kernel
0000    41 ;
0000    42 ; Author:  RICHARD I. HUSTVEDT, Creation date:  15-MAY-1979
0000    43 ;
0000    44 ; Modified by:
0000    45 ;
0000    46 ;       V03-009 KDM0024         Kathleen D. Morse       10-Oct-1982
0000    47 ;               Alter CHMx vectors in secondary SCB if system service
0000    48 ;               inhibit was requested at boot time.
0000    49 ;
0000    50 ;       V03-008 KDM0020         Kathleen D. Morse       04-Oct-1982
0000    51 ;               Create a subroutine that unhooks the multi-processing
0000    52 ;               code from the system, that can be invoked from the
```

```
                     ;               primary's invalidate time-out logic.
                     ;
                     ;       V03-007 KDM0003         Kathleen D. Morse       06-Aug-1982
                     ;               Make STOP/CPU provide correct address to EXE$DEANONPAGED
                     ;               when the pool block was < 32 bytes off a page boundary.
                     ;
                     ; 01    -
                     ;--

                     ; INCLUDE FILES:
                     ;

                     ; MACROS:
                     ;

                             .MACRO  HOOK,LOC,TARGET                 ;
                             .LONG   LOC                             ;
                             JMP     @#TARGET                        ;
                             .BLKB   6                               ; Normal VMS contents of hook location
                             .ENDM   HOOK                            ;

                             .MACRO  HOOKJSB,LOC,TARGET
                             .LONG   LOC
                             JSB     @#TARGET
                             .BLKB   6                               ; Normal VMS contents of hook location
                             .ENDM   HOOKJSB

                     ; EQUATED SYMBOLS:
                     ;

                             $CLIMSGDEF                              ; Define CLI error messages
                             $DSCDEF                                 ; Define string descriptor fields
                             $DYNDEF                                 ; Define dynamic structure types
                             $IPLDEF                                 ; Define interrupt priority levels
                             $IRPDEF                                 ; Define structure header fields
                             $LCKDEF                                 ; Lock bit definitions
                             $MPMDEF                                 ; Define MA780 registers
                             $MPSDEF                                 ; Define secondary processor states
                             $NDTDEF                                 ; Define nexus type codes
                             $PCBDEF                                 ; Process control block definitions
                             $PHDDEF                                 ; Process header definitions
                             $PRDEF                                  ; Define processor register numbers
                             $PTEDEF                                 ; Define page table entry format
                             $RPBDEF                                 ; Define restart param block offsets
                             $SSDEF                                  ; Define system error messages
                             $STATEDEF                               ; State definitions
                             $VADEF                                  ; Define virtual address fields

                     ; OWN STORAGE:
                     ;
00000000             .PSECT  $$$$$$BEGIN,PAGE                        ; Base PSECT
0000                 MPS$BEGIN::                                     ;
```

```
                     0000   110 MPS$GL_POOLDSC::                        ; Structure descriptor for MP code
            00000000 0000   111         .LONG   0                       ; Adr of non-paged pool alloc for MP
            000001F0' 0004  112         .LONG   <<<<MPS$END-MPS$BEGIN>+511>@-4>@4> ; Size of pool used
                01F0' 0008  113         .WORD   <<<<MPS$END-MPS$BEGIN>+511>@-4>@4> ; Structure size
                  62  000A  114         .BYTE   DYN$C_LOADCODE          ; Structure type of loadable code
                  03  000B  115         .BYTE   DYN$C_LC_MP             ; Structure sub-type of multi-processing
                     000C   116 MPS$HOOKTBL::                           ; Addresses to be locked in WS
000000AC'00000014' 000C   117         .LONG   HOOKBASE,HOOKEND        ;
                     0014   118 ;
                     0014   119 ;       The convention for the following logical names is:
                     0014   120 ;
                     0014   121 ;               1) All hooks in the exec are prefixed by MPH$
                     0014   122 ;               2) If the entire routine is replaced, then the
                     0014   123 ;                  rest of the hook name is the same as the normal
                     0014   124 ;                  routine name
                     0014   125 ;               3) If there are only a few lines of new MP code to
                     0014   126 ;                  be inserted, the hook name ends with HK and the
                     0014   127 ;                  continuation point (if any) hook name ends with CONT
                     0014   128 ;               4) All MP routines are prefixed with MPS$
                     0014   129 ;               5) If the MP routine entirely replaces an exec routine,
                     0014   130 ;                  then the rest of the MP routine name is the same as
                     0014   131 ;                  the normal routine name
                     0014   132 ;               6) If there are only a few lines of new MP code, then
                     0014   133 ;                  the rest of the MP routine name is a new name
                     0014   134 ;
                     0014   135 HOOKBASE:                               ; Base of hook table
                     0014   136         HOOK    MPH$SCHED,MPS$SCHED     ;
                     0024   137         HOOK    MPH$RESCHED,MPS$RESCHED ; Reschedule interrupt
                     0034   138         HOOK    MPH$QAST,MPS$QAST       ; Queue AST
                     0044   139         HOOK    MPH$INVALIDHK,MPS$INVALID ; Invalidate TB
                     0054   140         HOOKJSB MPH$BUGCHKHK,MPS$BUGCHECK ; Check for 2ndary during bugchk
                     0064   141         HOOK    MPH$ASTDELHK,MPS$ASTSCHEDCHK ; Resched process to 2ndary chk
                     0074   142         HOOK    MPH$NEWLVLHK,MPS$ASTNEWLVL ; Set new PR ASTLVL
            00000000 0084   143         .LONG   0                       ; End of JMP/JSB type hooks
                     0088   144 SCB_IPL14:                               ; Hook for SCB IPL=14 vector
            00000000 0088   145         .LONG   0                       ;  Adr of longword
            00000000 008C   146         .LONG   0                       ;  Old contents of longword
                     0090   147 SCB_IPL16:                               ; Hook for SCB IPL=16 vector
            00000000 0090   148         .LONG   0                       ;  Adr of longword
            00000000 0094   149         .LONG   0                       ;  Old contents of longword
                     0098   150 SCB_VEC94:                              ; Hook for XDELTA (SOFTINT 5)
            00000000 0098   151         .LONG   0                       ;  Adr of longword
            00000000 009C   152         .LONG   0                       ;  Old contents of longword
                     00A0   153 SCB_VECBC:                              ; Hook for SOFTINT F
            00000000 00A0   154         .LONG   0                       ;  Adr of longword
            00000000 00A4   155         .LONG   0                       ;  Old contents of longword
            00000000 00A8   156         .LONG   0                       ; Empty hook at ends table
                     00AC   157 HOOKEND:
                 00000000   158         .PSECT  _END,PAGE               ; End PSECT
                     0000   159 MPS$END::                               ;
                     0000   160
                 00000000   161         .PSECT  __MPLOAD,LONG           ;
                     0000   162
                     0000   163 LOCKRANGE:                              ; Addresses to be locked in WS
00000525'000004E1' 0000   164         .LONG   LOCKSTART,LOCKEND       ;
                     0008   165
                     0008   166 STOPRANGE:                              ; Addresses to be locked in WS
```

```
000005DC'00000555'  0008   167          .LONG   STOPSTART,STOPEND       ;
                    0010   168
                    0010   169 ;
                    0010   170 ; Output buffer for SHOW CPU displays.
                    0010   171 ;
            00000000  0010   172 OUTPUT_LENGTH:  .LONG   0
                    0014   173
                    0014   174 OUTPUT_BUFFER:
            0000005A  0014   175          .BLKB   70
                    005A   176
                    005A   177 OUTPUT_BUF_DSC:
            00000046  005A   178          .LONG   70
            00000014' 005E   179          .ADDRESS        OUTPUT_BUFFER
                    0062   180
                    0062   181 DCL_LINE_DSC:
            00000000  0062   182          .LONG   0
            00000000  0066   183          .LONG   0
                    006A   184
                    006A   185 STATE_VALUE:
            00000000  006A   186          .LONG   0
                    006E   187
                    006E   188
                    006E   189 ;
                    006E   190 ; Array of ascid descriptor addresses for secondary
                    006E   191 ; processor states.  Ordered from minimum to maximum
                    006E   192 ; value of the state.  Indexed via numeric state value - 1.
                    006E   193 ;
                    006E   194 STATES:
        000000CB'   006E   195          .ADDRESS IDLE_STATE_DSC
        000000BF'   0072   196          .ADDRESS DROP_STATE_DSC
        000000A4'   0076   197          .ADDRESS BUSY_STATE_DSC
        000000B0'   007A   198          .ADDRESS EXEC_STATE_DSC
        00000086'   007E   199          .ADDRESS INIT_STATE_DSC
        00000098'   0082   200          .ADDRESS STOP_STATE_DSC
                    0086   201
                    0086   202 INIT_STATE_DSC:
41 49 54 49 4E 49 0000008E'010E0000'  0086   203          .ASCID  /INITIALIZE/
                  45 5A 49 4C  0094
                    0098   204 STOP_STATE_DSC:
      50 4F 54 53 000000A0'010E0000'  0098   205          .ASCID  /STOP/
                    00A4   206 BUSY_STATE_DSC:
      59 53 55 42 000000AC'010E0000'  00A4   207          .ASCID  /BUSY/
                    00B0   208 EXEC_STATE_DSC:
54 55 43 45 58 45 000000B8'010E0000'  00B0   209          .ASCID  /EXECUTE/
                        45  00BE
                    00BF   210 DROP_STATE_DSC:
      50 4F 52 44 000000C7'010E0000'  00BF   211          .ASCID  /DROP/
                    00CB   212 IDLE_STATE_DSC:
      45 4C 44 49 000000D3'010E0000'  00CB   213          .ASCID  /IDLE/
                    00D7   214 GET_VERB_DSC:
   42 52 45 56 24 000000DF'010E0000'  00D7   215          .ASCID  /$VERB/
                    00E4   216 GET_LINE_DSC:
   45 4E 49 4C 24 000000EC'010E0000'  00E4   217          .ASCID  /$LINE/
                    00F1   218
                    00F1   219 STATE_CTL_DSC:
74 41 5F 21 2F 21 000000F9'010E0000'  00F1   220          .ASCID  \!/!_Attached processor is in the !AS state.\
65 63 6F 72 70 20 64 65 68 63 61 74  00FF
```

```
74 20 6E 69 20 73 69 20 72 6F 73 73  010B
65 74 61 74 73 20 53 41 21 20 65 68  0117
                              2E     0123
                                     0124    221
                                     0124    222 ILSTATE_CTL_DSC:
74 41 5F 21 2F 21 0000012C'010E0000' 0124    223         .ASCID  \!/!_Attached processor is in an illegal state of !XL\
65 63 6F 72 70 20 64 65 68 63 61 74  0132
61 20 6E 69 20 73 69 20 72 6F 73 73  013E
74 73 20 6C 61 67 65 6C 6C 69 20 6E  014A
       4C 58 21 20 66 6F 20 65 74 61  0156
                                     0160    224
                                     0160    225
                                     0160    226         .LIST   MEB                     ; Show macro expansions
```

```
                     0160   228          .SBTTL  MPDCL - Multi-processing DCL command line handling
                     0160   229 ;++
                     0160   230 ; Functional Description:
                     0160   231 ;
                     0160   232 ; This routine contains the entry point for MP.EXE, MPSDCL.  This code
                     0160   233 ; determines which command verb was requested (START, STOP, or SHOW)
                     0160   234 ; and branches to the appropriate code.  It reports an error if the
                     0160   235 ; invocation was not via one of these three verbs.
                     0160   236 ;
                     0160   237 ; Calling Sequence:
                     0160   238 ;
                     0160   239 ;       CALLx   MPS$DCL
                     0160   240 ;
                     0160   241 ; Input Parameters:
                     0160   242 ;
                     0160   243 ;       None
                     0160   244 ;
                     0160   245 ; Environment:
                     0160   246 ;
                     0160   247 ;       Executed by the primary processor.
                     0160   248 ;
                     0160   249 ; Return Status:
                     0160   250 ;
                     0160   251 ;       SS$_NORMAL - normal completion
                     0160   252 ;       SS$_DEVOFFLINE - device is not in configuration (returned if not
                     0160   253 ;               on an 11/780, or secondary is not started)
                     0160   254 ;       SS$_SHMNOTCNCT - shared memory not connected (returned if no MA780
                     0160   255 ;               is found for START/CPU)
                     0160   256 ;       CLI$_ABVERB - unrecognized command (returned if MP.EXE is not invoked
                     0160   257 ;               via START, STOP, or SHOW DCL commands)
                     0160   258 ;
                     0160   259 ;--
                     0160   260 ERRORX2:
          50   84 8F  9A  0160   261          MOVZBL  #SS$_DEVOFFLINE,R0      ; Report device not in configuration
                     0164   262 ERRORX5:
                        04  0164   263          RET                             ; Exit with error status
                     0165   264 MPS$DCL::
                      0000  0165   265          .WORD   0                       ; Entry mask
                     0167   266          .ENABL  LSB
00000000'GF    01    91  0167   267          CMPB    #1,G^EXE$GB_CPUTYPE     ; Is this an 11/780?
                  F0   12  016E   268          BNEQ    ERRORX2                 ; Br if not, error exit
     FEF0 CF    02    90  0170   269          MOVB    #DSC$K_CLASS_D,DSC$B_CLASS+DCL_LINE_DSC ; Initialize descriptor
          FEE9 CF   9F  0175   270          PUSHAB  DCL_LINE_DSC            ; Buffer dsc adr for command line
          FF67 CF   9F  0179   271          PUSHAB  GET_LINE_DSC            ; Dsc adr of callback item requested
00000000'GF    02    FB  017D   272          CALLS   #2,G^CLI$GET_VALUE      ; Request command line
            DD 50    E9  0184   273          BLBC    R0,ERRORX5              ; Br if error
                     0187   274 ;
                     0187   275 ; Determine which DCL command was issued:  SHOW/CPU, START/CPU, or SHOW/CPU.
                     0187   276 ; If none of these was issued, then exit with an error status.
                     0187   277 ;
          52   FEDB CF   D0  0187   278          MOVL    DCL_LINE_DSC+DSC$A_POINTER,R2 ; Get address of DCL command line
            53 8F    82   91  018C   279          CMPB    (R2)+,#^A\S\            ; First character must be an S
                     18   12  0190   280          BNEQ    ERRORX3                 ; Br if unrecognized verb
            48 8F    62   91  0192   281          CMPB    (R2),#^A\H\             ; Check for SHOW verb
                     36   13  0196   282          BEQL    SHOW_CPU                ; Br to execute SHOW/CPU command
            54 8F    82   91  0198   283          CMPB    (R2)+,#^A\T\            ; Check for STOP or START verbs
                     0C   12  019C   284          BNEQ    ERRORX3                 ; Br if unrecognized verb
```

```
                41 8F    62    91   019E   285          CMPB    (R2),#^A\A\             ; Check for START verb
                         0E    13   01A2   286          BEQL    START_CPU               ; Br to execute START/CPU command
                4F 8F    62    91   01A4   287          CMPB    (R2),#^A\O\             ; Check for STOP verb
                         0B    13   01A8   288          BEQL    STOP_CPU                ; Br to execute STOP/CPU command
                               01AA   289 ERRORX3:
     50    00038090 8F    D0    01AA   290          MOVL    #CLI$_IVVERB,R0         ; Report unrecognized command
                               01B1   291 ERRORX4:
                         04    01B1   292          RET                             ; Exit with error status
                               01B2   293 START_CPU:                                ; START/CPU DCL command execution
                      0078    31    01B2   294          BRW     MPS$LOAD                ; Continue loading MP code
                               01B5   295
                               01B5   296 STOP_CPU:                                 ; STOP/CPU DCL command execution
                      037D    31    01B5   297          BRW     MPS$UNLOAD              ; STOP/CPU DCL command execution
                               01B8   298
                               01B8   299 ;
                               01B8   300 ; GETDATA - This routine goes into kernel mode and copies
                               01B8   301 ; needed data into a local buffer.
                               01B8   302 ;
                               01B8   303 GETDATA:
                      0000     01B8   304          .WORD   0                       ; Entry mask
              50      01       9A    01BA   305          MOVZBL  #SS$_NORMAL,R0          ; Assume success
     51    00000000'GF    D0    01BD   306          MOVL    G^EXE$GL_MP,R1          ; Get adr of loaded MP code
                         9A    13    01C4   307          BEQL    ERRORX2                 ; Br if MP not loaded
  FE9D CF    0000'C1    D0    01C6   308          MOVL    MPS$GL_STATE(R1),STATE_VALUE ; Copy state variable
                         04    01CD   309          RET                             ; Exit from kernel mode routine
                               01CE   310 ;
                               01CE   311 ; The command was SHOW CPU.  Change into kernel mode to
                               01CE   312 ; access the state variable for the secondary processor.
                               01CE   313 ;
                               01CE   314 SHOW_CPU:
                               01CE   315          $CMKRNL_S GETDATA               ; Acquire secondary state
                      00       DD    01CE                    PUSHL   #0
                   E5 AF       DF    01D0                    PUSHAL  GETDATA
  00000000'GF      02          FB    01D3                    CALLS   #2,G^SYS$CMKRNL
                   D4 50       E9    01DA   316          BLBC    R0,ERRORX4              ; Exit with error status
                               01DD   317 ;
                               01DD   318 ; Now format and output data.
                               01DD   319 ;
                               01DD   320 10$:
        52    FF43 CF          9E    01DD   321          MOVAB   ILSTATE_CTL_DSC,R2      ; Assume illegal state
        50    FE84 CF          D0    01E2   322          MOVL    STATE_VALUE,R0          ; Get state value
              01       50      D1    01E7   323          CMPL    R0,#MPS$K_IDLESTATE     ; Is this at least minimum value?
                       12      19    01EA   324          BLSS    20$                     ; Br if bad value
              06       50      D1    01EC   325          CMPL    R0,#MPS$K_STOPSTATE     ; Is this at least maximum value?
                       0D      14    01EF   326          BGTR    20$                     ; Br if bad value
                       50      D7    01F1   327          DECL    R0                      ; Convert value to an index
     50    FE76 CF40          D0    01F3   328          MOVL    STATES[R0],R0           ; Get address of ascii state text
        52    FEF4 CF          9E    01F9   329          MOVAB   STATE_CTL_DSC,R2        ; Output ascii state name
                               01FE   330 20$:     $FAO_S  CTRSTR=(R2),-
                               01FE   331                  OUTLEN=OUTPUT_LENGTH,-
                               01FE   332                  OUTBUF=OUTPUT_BUF_DSC,-
                               01FE   333                  P1=R0                   ; Format output line
                      50       DD    01FE                    PUSHL   R0
                   FE56 CF     7F    0200                    PUSHAQ  OUTPUT_BUF_DSC
                   FE08 CF     3F    0204                    PUSHAW  OUTPUT_LENGTH
                      62       7F    0208                    PUSHAQ  (R2)
  00000000'GF      04          FB    020A                    CALLS   #$$T2,G^SYS$FAO
```

```
                 9D 50    E9  0211   334          BLBC    R0,ERRORX4                  ; Exit with error status
      FE3F CF    FDF8 CF  D0  0214   335          MOVL    OUTPUT_LENGTH,OUTPUT_BUF_DSC ; Initialize output buffer dsc
                 FE3B CF  7F  021B   336          PUSHAQ  OUTPUT_BUF_DSC  ; Dsc adr for output buffer
       00000000'GF    01  FB  021F   337          CALLS   #1,G^LIB$PUT_OUTPUT         ; Output secondary processor state
                 88 50    E9  0226   338          BLBC    R0,ERRORX4                  ; Exit with error status
                          0229   339
                          0229   340 EXIT:
                 50    01 9A  0229   341          MOVZBL  #SS$_NORMAL,R0              ; Exit with success code
                          04  022C   342          RET                                 ; Exit
                              022D   343          .DSABL  LSB
```

```
                                022D   345          .SBTTL
                                022D   346 ;++
                                022D   347 ; Functional Description:
                                022D   348 ;
                                022D   349 ;       MPS$LOAD is linked together with all of the code required for multi-
                                022D   350 ;       processing. The necessary amount of non-paged pool is allocated
                                022D   351 ;       and rounded up to page boundary.  Code is then moved into this
                                022D   352 ;       block of pool.  All of this code must be PIC although a limited
                                022D   353 ;       amount of relocation will be done on data cells and the SCB for
                                022D   354 ;       the secondary processor.
                                022D   355 ;
                                022D   356 ; Calling Sequence:
                                022D   357 ;
                                022D   358 ;       BRW     MPS$LOAD
                                022D   359 ;
                                022D   360 ; Input Parameters:
                                022D   361 ;
                                022D   362 ;       None
                                022D   363 ;
                                022D   364 ; Environment:
                                022D   365 ;
                                022D   366 ;       Executed by the primary processor.
                                022D   367 ;
                                022D   368 ;
                                022D   369 ;--
                                022D   370
                                022D   371 MPS$LOAD::                           ; Load multi-processing code for START
                                022D   372          $LKWSET_S       INADR=LOCKRANGE ; Lock critical code into WS
                          00    DD   022D                   PUSHL   #0
                          00    DD   022F                   PUSHL   #0
                    FDCB  CF    7F   0231                   PUSHAQ  LOCKRANGE
00000000'GF             03    FB   0235                   CALLS   #3,G^SYS$LKWSET
                      21 50     E9   023C   373          BLBC    R0,ERRORX1              ; Exit if unable to lock pages
                                023F   374          $LKWSET_S       INADR=MPS$HOOKTBL ; Lock critical data into WS
                          00    DD   023F                   PUSHL   #0
                          00    DD   0241                   PUSHL   #0
              0000000C'EF       7F   0243                   PUSHAQ  MPS$HOOKTBL
00000000'GF             03    FB   0249                   CALLS   #3,G^SYS$LKWSET
                      0D 50     E9   0250   375          BLBC    R0,ERRORX1              ; Br if error
                                0253   376          $CMKRNL_S W^MPS$LOADK           ; Execute kernel routine
                          00    DD   0253                   PUSHL   #0
                  0279'CF       DF   0255                   PUSHAL  W^MPS$LOADK
00000000'GF             02    FB   0259                   CALLS   #2,G^SYS$CMKRNL
                                0260   377 ERRORX1:
                             04 0260   378          RET                             ; Return
                                0261   379
                                0261   380
                                0261   381 LOCAL_MEM_ERR:
50      00000000'8F       D0   0261   382          MOVL    #ERR$_LCLMEMUSED,R0     ; Local memory cannot be used for MP
                             04 0268   383          RET
                                0269   384
                                0269   385 MA780_NOT_USED:
50      00000000'8F       D0   0269   386          MOVL    #ERR$_MA780_REQ,R0      ; MA780 memory required for MP
                             04 0270   387          RET
                                0271   388
                                0271   389 MA780_CNCT_ERR:
50      00000000'8F       D0   0271   390          MOVL    #ERR$_SHMDBLUSE,R0      ; MA780 memory used for MP and MA780
```

```
                     04   0278   391          RET                             ;  system interconnect at same time
                          0279   392
                          0279   393
                          0279   394
                     0FFC 0279   395 MPS$LOADK::     .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          027B   396          .ENABL  LSB
          00000000'GF    D5   027B   397          TSTL    G^EXE$GL_MP             ; Is secondary already started?
                    A6   12   0281   398          BNEQ    EXIT                    ; Br on yes, don't do anything
          00000000'GF    D5   0283   399          TSTL    G^EXE$GL_SHBLIST        ; Is MA780 in use as sys intercnct?
                    E6   12   0289   400          BNEQ    MA780_CNCT_ERR          ; Br if MA780 already in use
       50 00000000'GF    D0   028B   401          MOVL    G^EXE$GL_RPB,R0         ; Get address of RPB
    CA 30 A0    0C    E0   0292   402          BBS     #RPB$V_USEMPM,RPB$L_BOOTR5(R0),LOCAL_MEM_ERR ; Br if booted with
    CD 30 A0    0B    E1   0297   403          BBC     #RPB$V_MPM,RPB$L_BOOTR5(R0),MA780_NOT_USED   ;  wrong memory
       51    0008'CF    3C   029C   404          MOVZWL  W^MPS$GL_POOLDSC+IRP$W_SIZE,R1 ; Size of loadable code
        00000000'GF    16   02A1   405          JSB     G^EXE$ALONONPAGED       ; Allocate necessary block
                          02A7   406
                          02A7   407          .IF     DF,MPDBGSWT
                          02A7   408          SETIPL  #0                      ;***** Drop IPL for debugging
                          02A7   409          .ENDC
                          02A7   410
              B6 50    E9   02A7   411          BLBC    R0,ERRORX1              ; Exit if none available
           50    52    D0   02AA   412          MOVL    R2,R0                   ; Remember starting address of pool
       0000'CF   52    D0   02AD   413          MOVL    R2,W^MPS$GL_POOLDSC     ; Assume nothing to return to pool
       52   01FF C2    9E   02B2   414          MOVAB   511(R2),R2              ; Round up to page boundary
    52   000001FF 8F    CA   02B7   415          BICL    #^X1FF,R2               ; Set to page boundary
           5A    52    D0   02BE   416          MOVL    R2,R10                  ; Save address of block
    00000000'GF  52    D0   02C1   417          MOVL    R2,G^EXE$GL_MP          ; Set exec pointer to MP code
       51   52    50    C3   02C8   418          SUBL3   R0,R2,R1                ; Compute size of unused piece of pool
                 4F    13   02CC   419          BEQL    10$                     ; Br if piece of pool was page-aligned
          08 A0    51    B0   02CE   420          MOVW    R1,IRP$W_SIZE(R0)       ; Set size of unused piece
     0A A0    0362 8F    B0   02D2   421          MOVW    #<DYN$C_LC_MP@8+DYN$C_LOADCODE>,IRP$B_TYPE(R0) ; Set type
           20    51    D1   02D8   422          CMPL    R1,#32                  ; Is piece too small to bother returning
                 4C    15   02DB   423          BLEQ    10$                     ; Br if too small
       0000'CF   52    D0   02DD   424          MOVL    R2,W^MPS$GL_POOLDSC     ; Set page-aligned pool adr
       0004'CF   51    C2   02E2   425          SUBL2   R1,W^MPS$GL_POOLDSC+4   ; Set page-aligned pool size
       0008'CF   51    C2   02E7   426          SUBL2   R1,W^MPS$GL_POOLDSC+IRP$W_SIZE ; Set page-aligned size
                          02EC   427          ASSUME  IRP$W_SIZE LE 10
       62   0000'CF   D0   02EC   428          MOVL    W^MPS$GL_POOLDSC,(R2)   ; Initialize enough of pool block
    04 A2   0004'CF   D0   02F1   429          MOVL    W^MPS$GL_POOLDSC+4,4(R2) ;  to allow its deallocate if an
    08 A2   0008'CF   D0   02F7   430          MOVL    W^MPS$GL_POOLDSC+8,8(R2) ;  error exit is required
        00000000'GF    16   02FD   431          JSB     G^EXE$DEANONPAGED       ; Return unused piece of pool
              17 50    E8   0303   432          BLBS    R0,10$                  ; Br if successful
                          0306   433 DEA_ERR_EXIT:
                    50    DD   0306   434          PUSHL   R0                      ; Save error exit status
       50   0000'CF   D0   0308   435          MOVL    W^MPS$GL_POOLDSC,R0     ; Get address of pool allocated
        00000000'GF    16   030D   436          JSB     G^EXE$DEANONPAGED       ; Release block of pool
                    50 8ED0   0313   437          POPL    R0                      ; Restore exit status
        00000000'GF    D4   0316   438          CLRL    G^EXE$GL_MP             ; Indicate no MP code loaded
                     04   031C   439          RET                             ; Exit with error status
                          031D   440
                          031D   441 10$:
       50   0000'CF   9E   031D   442          MOVAB   W^MPS$BEGIN,R0          ; Address of start of code segment
       59   5A    50   C3   0322   443          SUBL3   R0,R10,R9               ; Relocation offset is difference
                          0326   444 ;
                          0326   445 ; The addresses in the SCB for the secondary processor are now relocated
                          0326   446 ; by adding the relocation value.  Any SCB pointer already a system space
                          0326   447 ; address is correct and need not be relocated.
```

```
                                0326   448 ;
            51   0200'CF    9E  0326   449          MOVAB   W^SCB$AL_BASE+512,R1    ; Get base of SCB
              50     80 8F    9A  032B   450          MOVZBL  #128,R0                 ; Do all 128 vectors
                                032F   451          .DSABL  LSB
                                032F   452 SCB_LOOP:                                ;
                       71    D5  032F   453          TSTL    -(R1)                   ; Is vector in system space?
                       03    19  0331   454          BLSS    10$                     ; Yes, skip it
                   61    59    C0  0333   455          ADDL    R9,(R1)                 ; Other wise relocate pointer to point
                     F6  50    F5  0336   456 10$:     SOBGTR  R0,SCB_LOOP             ; to code segment. Do all 128 vectors
                                0339   457 ;
                                0339   458 ; Locate all the multiport memory controllers and initialize the SCB
                                0339   459 ; vectors for the inter-processor interrupts.  The first MA780 is used for
                                0339   460 ; the multi-processing scheduling interrupt while the other MA780s are
                                0339   461 ; vectored to an unexpected interrupt logging routine.  The error interrupt
                                0339   462 ; vectors for the MA780s are initialized for the secondary also, while
                                0339   463 ; those for the primary were initialized in the normal VMS boot procedures.
                                0339   464 ;
                                0339   465 LOC_MPM:                                  ; Initialize vectors for MA780 memories
                       50    D4  0339   466          CLRL    R0                      ; Initialize index
            52   0000'CF    9E  033B   467          MOVAB   W^MPS$INT58,R2          ; Get adr of error interrupt logger
                 52    59    C0  0340   468          ADDL    R9,R2                   ; Relocate for eventual location
                 52    01    C8  0343   469          BISL    #1,R2                   ; Set interrupt stack bit in vector
         53   00000000'GF    D0  0346   470          MOVL    G^MMG$GL_SBICONF,R3     ; Get address of SBI config array
            54   0000'CF    9E  034D   471          MOVAB   W^MPS$AL_MPMBASE,R4     ; Get address of first MA780 base
         55   00000000'GF    D0  0352   472          MOVL    G^EXE$GL_SCB,R5         ; Base address for primary SCB
         56   00000000'GF    D0  0359   473          MOVL    G^EXE$GL_RPB,R6         ; Get base of RPB
            57   0000'CF    DE  0360   474          MOVAL   W^SCB$AL_BASE,R7        ; Base address for secondary SCB
                                0365   475
   10 00000000'GF    00'    E1  0365   476          BBC     S^#EXE$V_SSINHIBIT,G^EXE$GL_FLAGS,5$ ; If sys srv are being
   44 A7   00000000'GF    DE  036D   477          MOVAL   G^EXE$CMODEXECX,^X44(R7) ;  inhibited, then revector the
   40 A7   00000000'GF    DE  0375   478          MOVAL   G^MPS$CMODKRNLX,^X40(R7) ;  entry points for CHMK and CHME
                                037D   479 5$:
                                037D   480
51    0090 C640    03    8B  037D   481 10$:     BICB3   #3,RPB$B_CONFREG(R6)[R0],R1 ; Get a type byte
            51    40 8F    91  0384   482          CMPB    #NDT$_MPM0,R1           ; Is it a multiport memory?
                       7D    12  0388   483          BNEQ    15$                     ; Br if not an MA780
                                038A   484
      58    0100 C740    DE  038A   485          MOVAL   256(R7)[R0],R8          ; Compute address of first vector
          40 A8    52    D0  0390   486          MOVL    R2,64(R8)               ; Set IPL=X15 vector (error interrrupt)
        00C0 C8    52    D0  0394   487          MOVL    R2,192(R8)              ; Set IPL=X17 vector (error interrupt)
                                0399   488
               0000'CF    D5  0399   489          TSTL    W^MPS$AL_MPMBASE        ; Is this the first MA780?
                       51    12  039D   490          BNEQ    12$                     ; Br if not
                                039F   491
            51   0000'CF    9E  039F   492          MOVAB   W^MPS$SINTSR,R1         ; Get adr of secondary interrupt rtn
                 51    59    C0  03A4   493          ADDL    R9,R1                   ; Relocate for eventual location
                 51    01    C8  03A7   494          BISL    #1,R1                   ; Set interrupt stack bit in vector
                 68    51    D0  03AA   495          MOVL    R1,(R8)                 ; Set IPL=X14 vector (inter-proc intrpt)
            0080 C8    51    D0  03AD   496          MOVL    R1,128(R8)              ; Set IPL=X16 vector (inter-proc intrpt)
                                03B2   497
            51   0000'CF    9E  03B2   498          MOVAB   W^MPS$PINTSR,R1         ; Get adr of primary interrupt routine
                 51    59    C0  03B7   499          ADDL    R9,R1                   ; Relocate for eventual location.
                 51    01    C8  03BA   500          BISL    #1,R1                   ; Set interrupt stack bit in vector
      58    0100 C540    DE  03BD   501          MOVAL   256(R5)[R0],R8          ; Compute address of first vector
   00000088'EF    58    D0  03C3   502          MOVL    R8,SCB_IPL14            ; Remember adr for unload logic
   0000008C'EF    68    D0  03CA   503          MOVL    (R8),SCB_IPL14+4        ; Remember contents for unload
                 68    51    D0  03D1   504          MOVL    R1,(R8)                 ; Set IPL=X14 vector (inter-proc intrpt)
```

```
00000090'EF   0080 C8   9E  03D4   505          MOVAB   128(R8),SCB_IPL16      ; Remember adr for unload logic
00000094'EF   0080 C8   D0  03DD   506          MOVL    128(R8),SCB_IPL16+4    ; Remember contents for unload
         0080 C8   51   DC  03E6   507          MOVL    R1,128(R8)             ; Set IPL=X16 vector (inter-proc intrpt)
                   16   11  03EB   508          BRB     14$                    ; Continue with common code
                                03ED   509
                FF8D   31  03ED   510 11$:     BRW     10$                    ; Branch assist
                                03F0   511
         51   0000'CF   9E  03F0   512 12$:     MOVAB   W^MPS$UNEXPINT,R1      ; Get adr of unexpected interrupt rtn
              51   59   C0  03F5   513          ADDL    R9,R1                  ; Relocate for eventual location
              51   01   C8  03F8   514          BISL    #1,R1                  ; Set interrupt stack bit in vector
              68   51   D0  03FB   515          MOVL    R1,(R8)                ; Set IPL=X14 vector (inter-proc intrpt)
         0080 C8   51   D0  03FE   516          MOVL    R1,128(R8)             ; Set IPL=X16 vector (inter-proc intrpt)
                                0403   517
            84   6340   D0  0403   518 14$:     MOVL    (R3)[R0],(R4)+         ; Remember adr for this MA780s registers
                                0407   519
           E2 50   10   F2  0407   520 15$:     AOBLSS  #16,R0,11$             ; Try all 16
                                040B   521
         50   0000'CF   D0  040B   522          MOVL    W^MPS$AL_MPMBASE,R0    ; Get base of MA780 registers
                   08   12  0410   523          BNEQ    FOUND_MPM              ; Found at least one MA780
         50   037C 8F   3C  0412   524          MOVZWL  #SS$_SHMNOTCNCT,R0     ; Indicate failure to find an MA780
                FEEC   31  0417   525          BRW     DEA_ERR_EXIT           ; Error exit with pool deall
                                041A   526 FOUND_MPM:                          ;
         51   0000'CF   9E  041A   527          MOVAB   W^MPS$RESCHEDIPL5,R1   ; Get address of primary fork rtn
              51   01   C8  041F   528          BISL    #1,R1                  ; Set interrupt stack bit in vector
00000098'EF   0094 C5   9E  0422   529          MOVAB   ^X94(R5),SCB_VEC94     ; Remember adr for unload logic
0000009C'EF   0094 C5   D0  042B   530          MOVL    ^X94(R5),SCB_VEC94+4   ; Remember contents for unload
000000A0'EF   00BC C5   9E  0434   531          MOVAB   ^XBC(R5),SCB_VECBC     ; Remember adr for unload logic
000000A4'EF   00BC C5   D0  043D   532          MOVL    ^XBC(R5),SCB_VECBC+4   ; Remember contents for unload
    00BC C5   0094 C5   D0  0446   533          MOVL    ^X94(R5),^XBC(R5)      ; Make XDELTA respond to softint ^XF
    0094 C5   51   59   C1  044D   534          ADDL3   R9,R1,^X94(R5)         ; Relocate for eventual location & store
         0000'CF   20   C1  0453   535          ADDL3   #MPM$L_IIR,W^MPS$AL_MPMBASE,- ; Compute address of
              0000'CF        0458   536                  W^MPS$GL_MPMIIR        ; interrupt request register
                                045B   537                                     ; Interrupt routines
                   52   60   D0  045B   538          MOVL    MPM$L_CSR(R0),R2       ; Read configuration register
                        00   EF  045E   539          EXTZV   #MPM$V_CSR_PORT,-      ; Get port number
              52   52   02       0460   540                  #MPM$S_CSR_PORT,R2,R2  ;
                                0463   541
              53   52   10   C1  0463   542          ADDL3   #MPM$V_IIR_CTL,R2,R3   ; Set to control field
0000'CF    00001111 8F   53   78  0467   543          ASHL    R3,#^XF111,W^MPS$GL_PRIMSKT ; Generate interrupt trigger mask
                   52   04   C4  0471   544          MULL    #4,R2                  ; Compute bit position for trigger
         0000'CF   0F   52   78  0474   545          ASHL    R2,#^XF,W^MPS$GL_PRIMSKC; Align and store mask for clear
                                047A   546 ;
                                047A   547 ; Compute the physical addresses for the secondary SCB and the Secondary
                                047A   548 ; initialization routine starting address.
                                047A   549 ;
                                047A   550 SET_PHYS:                           ;
          55   00000000'GF   D0  047A   551          MOVL    G^MMG$GL_SPTBASE,R5    ; Get base of SPT
               51   0000'CF   9E  0481   552          MOVAB   W^SCB$AL_BASE,R1       ; VA of Secondary processor SCB
                    51   59   C0  0486   553          ADDL    R9,R1                  ; Relocate to eventual location
         51    51   15   09   EF  0489   554          EXTZV   #VA$V_VPN,#VA$S_VPN,R1,R1 ; Get virtual page number
                   50   6541   D0  048E   555          MOVL    (R5)[R1],R0            ; Fetch PTE for it
         50    50   15   00   EF  0492   556          EXTZV   #PTE$V_PFN,#PTE$S_PFN,R0,R0 ; Isolate page number
         0000'CF    50   09   78  0497   557          ASHL    #9,R0,@^MPS$GL_SCBB    ; Save physical SCB address
               51   0000'CF   9E  049D   558          MOVAB   W^EXE$MPSTART,R1       ; VA of initialization routine
                    51   59   C0  04A2   559          ADDL    R9,R1                  ; Relocate to eventual location
         51    51   15   09   EF  04A5   560          EXTZV   #VA$V_VPN,#VA$S_VPN,R1,R1 ; Get virtual page number
                   50   6541   D0  04AA   561          MOVL    (R5)[R1],R0            ; Fetch PTE for it
```

```
                50    50    15    00    EF  04AE   562          EXTZV   #PTE$V_PFN,#PTE$S_PFN,R0,R0 ; Isolate page number
                  10  A6    50    09    78  04B3   563          ASHL    #9,R0,RPB$L_HALTPC(R6)  ; Save starting physical address
                                              04B8   564 ;
                                              04B8   565 ; Relocate some stray locations
                                              04B8   566 ;
                                              04B8   567 MISC_RELOC:                              ;
                 00000000'EF    59    C0    04B8   568          ADDL    R9,XDELIBRK             ; Address for initial breakpoint
                 00000000'EF    59    C0    04BF   569          ADDL    R9,MPS$GL_STRTVA        ; Address for jump into S0 space
                 00000000'EF    59    C0    04C6   570          ADDL    R9,MPS$GL_ISP           ; Interrupt stack for secondary
                                              04CD   571 ;
                                              04CD   572 ; Remember the time and date that this code was loaded.  This field
                                              04CD   573 ; is used by the MONITOR utility to determine if the multi-processing
                                              04CD   574 ; code has been reloaded during its last sampling interval (and therefore,
                                              04CD   575 ; the cpu time accumulation cells have been re-initialized).
                                              04CD   576 ;
          0000'CF    00000000'GF    7D    04CD   577          MOVQ    G^EXE$GQ_SYSTIME,W^MPS$GQ_MPSTRTIM ; Get time in 64 bits
                                              04D6   578 ;
                                              04D6   579 ; Now move the code into the pool segment after all relocation is done.
                                              04D6   580 ;
                                              04D6   581 MOVE_CODE:                               ;
                    5B    0000'8F    3C    04D6   582          MOVZWL  #<MPS$END-MPS$BEGIN>,R11 ; Size of MP code in bytes
                6A  0000'CF    5B    28    04DB   583          MOVC3   R11,W^MPS$BEGIN,(R10)   ; Move code to allocated pool space
                                              04E1   584
                                              04E1   585          .IF     DF,MPDBGSWT
                                              04E1   586          NOP                             ;***** Instruction for debug breakpoint
                                              04E1   587          .ENDC
                                              04E1   588
                                              04E1   589 ;
                                              04E1   590 ; Begin locked down code that will execute at IPL=31 to actually
                                              04E1   591 ; install needed hooks into the running system.
                                              04E1   592 ;
                                              04E1   593 LOCKSTART:                               ;
                                              04E1   594          SETIPL  #31                     ;
                      12    1F    DA    04E1                     MTPR    #31,S^#PR$_IPL
                 00000000'GF    16    04E4   595          JSB     G^INI$WRITABLE          ; Set writable for installing hooks
          55     00000014'EF    9E    04EA   596          MOVAB   HOOKBASE,R5             ; Get base of hook table
                51    55    59    C1    04F1   597          ADDL3   R9,R5,R1                ; Get address of loaded mp code
                      50    85    D0    04F5   598 10$:     MOVL    (R5)+,R0                ; Fetch address to install hook
                            17    13    04F8   599          BEQL    20$                     ; Done if address is zero
                  0A  A1    60    B0    04FA   600          MOVW    (R0),10(R1)             ; Save normal VMS contents for STOP/CPU
                      80    85    B0    04FE   601          MOVW    (R5)+,(R0)+             ; Move JMP opcode + operand specifier
                  0C  A1    60    D0    0501   602          MOVL    (R0),12(R1)             ; Save normal VMS contents for STOP/CPU
                80    85    59    C1    0505   603          ADDL3   R9,(R5)+,(R0)+          ; Set address
                      55    06    C0    0509   604          ADDL2   #6,R5                   ; Point to next hook
                      51    10    C0    050C   605          ADDL2   #16,R1                  ; Point to next hook
                            E4    11    050F   606          BRB     10$                     ; Continue installing hooks
                         0003    31    0511   607 20$:     BRW     HOOKS_DONE              ; Branch assist for hooks completed
                                              0514   608 ;
                                              0514   609 ; This code is loaded into the RPB.  It is used as a safe place for
                                              0514   610 ; the secondary to wait while the primary does a bugcheck.  The first
                                              0514   611 ; longword is modified by the primary after it reboots.
                                              0514   612 ;
                                              0514   613          .ALIGN  LONG
                           00000510           0514   614 RPB_BUGCHK=.-4                      ; Address of longword that is modified
                                              0514   615                                     ; by primary and secondary
                                              0514   616 MPS$RPB_WAIT::                      ;
                      F9  BF    17    0514   617          JMP     @RPB_BUGCHK             ; RPB loop for secondary to execute
```

```
                                0517   618
                00000003        0517   619 RPB_LOOPSIZ = .-MPS$RPB_WAIT            ; Number of bytes for code for RPB loop
                                0517   620
                                0517   621 ;
                                0517   622 ; Now load loop code into the RPB for bugcheck.  This must be loaded
                                0517   623 ; in two steps.  First, the loop code is loaded and then the loop address.
                                0517   624 ;
                                0517   625 HOOKS_DONE:                             ;
  0100 C6   FFF8 CF    03   28  0517   626         MOVC    #RPB_LOOPSIZ,W^MPS$RPB_WAIT,RPB$B_WAIT(R6) ; Load JMP loop
      00FC C6   10 A6   D0   51F  0517   627         MOVL    RPB$[_HALTPC(R6),RPB$L_BUGCHK(R6) ; Now start the secondary
                                0525   628                                         ; executing at EXE$MPSTART (phys. adr)
                                0525   629 ;
                                0525   630 LOCKEND:                               ; End of locked code
                                0525   631 ;
                                0525   632 ; Installation of the multi-processing code is now complete
                                0525   633 ;
                                0525   634 INSTALL_DONE:
            00000000'GF   16    0525   635         JSB     G^INI$RDONLY            ; Reset protection on system pages
                                052B   636         SETIPL  #0                      ; Drop IPL
                12   00   DA   052B                MTPR    #0,S^#PR$_IPL
                   FACF'  30   052E   637         BSBW    MPS$MAINIT              ; Initialize MA780 for interrupts
                50   01   D0   0531   638         MOVL    #1,R0                   ; Set status to success
                             04   0534   639         RET                             ; and return
```

```
                     0535    641          .SBTTL  MPS$UNLOAD - STOP/CPU DCL command
                     0535    642
                     0535    643 ;++
                     0535    644 ; Functional Description:
                     0535    645 ;
                     0535    646 ;       MPS$UNLOAD is linked together with all of the code required for
                     0535    647 ;       multi-processing.  This routine causes the secondary processor
                     0535    648 ;       to halt and then unloads the multi-processing code (i.e., restores
                     0535    649 ;       all the executive hook locations to their original values and
                     0535    650 ;       restores the SCB to its single processor VMS values.
                     0535    651 ;
                     0535    652 ; Calling Sequence:
                     0535    653 ;
                     0535    654 ;       BRW     MPS$UNLOAD
                     0535    655 ;
                     0535    656 ; Input Parameters:
                     0535    657 ;
                     0535    658 ;       EXE$GL_MP - Points to multi-processing code loaded in pool
                     0535    659 ;       EXE$GL_RPB - Points to RPB
                     0535    660 ;
                     0535    661 ; Environment:
                     0535    662 ;
                     0535    663 ;       Executed by the primary processor.
                     0535    664 ;
                     0535    665 ;
                     0535    666 ;--
                     0535    667
                     0535    668 MPS$UNLOAD::                            ; Unload multi-processing code for STOP
                     0535    669          $LKWSET_S       INADR=STOPRANGE ; Lock critical code into WS
          00    DD   0535                       PUSHL   #0
          00    DD   0537                       PUSHL   #0
        FACB    CF   0539                       PUSHAQ  STOPRANGE
00000000'GF     03   053D                       CALLS   #3,G^SYS$LKWSET
          0D    50   E9   0544    670          BLBC    R0,EXIT2                ; Exit if unable to lock pages
                     0547    671          $CMKRNL_S W^MPS$UNLOADK         ; Execute kernel routine
          00    DD   0547                       PUSHL   #0
      0555'CF   DF   0549                       PUSHAL  W^MPS$UNLOADK
00000000'GF     02   054D                       CALLS   #2,G^SYS$CMKRNL
                     0554    672 EXIT2:
                04   0554    673          RET                             ; Return
                     0555    674
                     0555    675
                     0555    676 STOPSTART:
                     0555    677
              0FFC   0555    678 MPS$UNLOADK::   .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0557    679          SETIPL  #IPL$_SYNCH             ; Synchronize on primary
        12    08   DA   0557                       MTPR    #IPL$_SYNCH,S^#PR$_IPL
        50    01   9A   055A    680          MOVZBL  #SS$_NORMAL,R0          ; Assume success
5A  00000000'GF   D0   055D    681          MOVL    G^EXE$GL_MP,R10         ; Is multi-proc code loaded?
                03   12   0564    682          BNEQ    5$                      ; Br on yes, continue unloading
              006F   31   0566    683 4$:      BRW     100$                    ; Branch assist
 F7 0000'CA     00   E6   0569    684 5$:      BBSSI   #MPS$V_STOPREQ,MPS$GL_STOPFLAG(R10),4$ ; Set STOP request flg
 00 0000'CA     00   E6   056F    685 10$:     BBSSI   #LCK$V_INTERLOCK,MPS$GL_INTERLOCK(R10),15$ ; Flush cache
    05  0000'CA   D1   0575    686 15$:     CMPL    MPS$GL_STATE(R10),#MPS$K_INITSTATE ; Is secondary active?
                3F   18   057A    687          BGEQ    50$                     ; Br if not active, no process to return
          0000'CA   16   057C    688          JSB     MPS$INTSCND(R10)        ; Interrupt secondary with STOP request
 FA 0000'CA     01   E7   0580    689 20$:     BBCC!   #MPS$V_STOPACK1,MPS$GL_STOPFLAG(R10),20$ ; Wait for secondary
```

```
                00 0000'CA    00    E6  0586    690                 BBSSI   #LCK$V_INTERLOCK,MPS$GL_INTERLOCK(R10),30$ ; Flush cache
                   01    0000'CA    D1  058C    691 30$:            CMPL    MPS$GL_STATE(R10),#MPS$K_IDLESTATE ; Is secondary active?
                               28    13  0591    692                 BEQL    50$                             ; Br if not active, no process to return
                                         0593    693 ;
                                         0593    694 ; Return process executing on the secondary to the scheduling queues.
                                         0593    695 ;
                      51    0000'CA    D0  0593    696                 MOVL    MPS$GL_CURPCB(R10),R1           ; Get PCB address for process
                         52    0B A1    9A  0598    697                 MOVZBL  PCB$B_PRI(R1),R2                ; Get current priority of process
             00 00000000'GF    52    E2  059C    698                 BBSS    R2,G^SCH$GL_COMQS,40$           ; Indicate something in sched queue
                         2C A1    0C    B0  05A4    699 40$:            MOVW    #SCH$C_COM,PCB$W_STATE(R1) ; Indicate process is computable
                53    00000000'GF    42    7E  05A8    700                 MOVAQ   G^SCH$AQ_COMT[R2],R3            ; Get tail of queue
                            93    61    0E  05B0    701                 INSQUE  (R1),@(R3)+                     ; Place process in scheduling queue
                      0000'CA    06    D0  05B3    702                 MOVL    #MPS$K_STOPSTATE,MPS$GL_STATE(R10) ; Set secondary stopped
                                         05B8    703                 SOFTINT #IPL$_SCHED                     ; Request primary reschedule
                                  14    03  DA  05B8                    MTPR    #IPL$_SCHED,S^#PR$_SIRR
                                         05BB    704 ;
                                         05BB    705 ; Now unload the multi-processing code.
                                         05BB    706 ;
                      0000'CA    06    D0  05BB    707 50$:            MOVL    #MPS$K_STOPSTATE,MPS$GL_STATE(R10) ; Change INIT state to STOP
                            0000'CA    16  05C0    708                 JSB     MPS$UNHOOK(R10)                 ; Call routine to unhook MP code
                                         05C4    709                 SETIPL  #IPL$_ASTDEL                    ; Lower IPL for deallocate of pool
                                  12    02  DA  05C4                    MTPR    #IPL$_ASTDEL,S^#PR$_IPL
                      50    0000'CA    D0  05C7    710                 MOVL    MPS$GL_POOLDSC(R10),R0          ; Set address of block to return
                       00000000'GF    16  05CC    711                 JSB     G^EXE$DEANONPAGED               ; Return block of nonpaged pool
                            03 50    E9  05D2    712                 BLBC    R0,100$                         ; Branch if error
                                  50    01  9A  05D5    713                 MOVZBL  #SS$_NORMAL,R0                  ; Return success code
                                         05D8    714 100$:           SETIPL  #0                              ; Restore IPL
                                  12    00  DA  05D8                    MTPR    #0,S^#PR$_IPL
                                         04    05DB    715                 RET                             ;
                                         05DC    716
                                         05DC    717 STOPEND:
```

```
                                05DC   719          .SBTTL  MPS$UNHOOK - Unhook multi-processing code from system
                                05DC   720
                                05DC   721 ;++
                                05DC   722 ; Functional Description:
                                05DC   723 ;
                                05DC   724 ;       MPS$UNHOOK is a part of the loadable multi-processing code.
                                05DC   725 ;       It unhooks the multi-processing code from the VMS system and
                                05DC   726 ;       resets the system to be a single processor 11/780, executing
                                05DC   727 ;       a vanilla VMS system.
                                05DC   728 ;
                                05DC   729 ;       This code is invoked for two reasons:  1) a STOP/CPU DCL command
                                05DC   730 ;       and  2) an invalidate request time-out.  It must be loaded into
                                05DC   731 ;       pool due to the latter usage.
                                05DC   732 ;
                                05DC   733 ; Calling Sequence:
                                05DC   734 ;
                                05DC   735 ;       JSB     MPS$UNHOOK
                                05DC   736 ;
                                05DC   737 ; Input Parameters:
                                05DC   738 ;
                                05DC   739 ;       R10 - address of multi-processing code in non-paged pool
                                05DC   740 ;
                                05DC   741 ; Environment:
                                05DC   742 ;
                                05DC   743 ;       Executed by the primary processor.
                                05DC   744 ;
                                05DC   745 ; Side-effects:
                                05DC   746 ;
                                05DC   747 ;       R0,R1 destroyed.  EXE$GL_MP is cleared.
                                05DC   748 ;
                                05DC   749 ;--
                                05DC   750
                           00000000   751          .PSECT  $AEXNONPAGED,LONG
                                0000   752 MPS$UNHOOK::
                                0000   753          DSBINT                                  ; Prevent all system events
                          7E    12   DB  0000                  MFPR    S^#PR$_IPL,-(SP)
                          12    1F   DA  0003                  MTPR    #31,S^#PR$_IPL
                50   00000000'GF   9E  0006   754          MOVAB   G^SCH$GL_NULLPCB,R0 ; Get address of null process PCB
                  50    6C   A0  D0  000D   755          MOVL    PCB$L_PHD(R0),R0                ; Get address of null process PHD
             38   A0    0000'CA   C2  0011   756          SUBL    MPS$GL_NULLCPU(R10),PHD$L_CPUTIM(R0) ; Del secondary null time
                00000000'GF   16  0017   757          JSB     G^INI$WRITABLE                  ; Make executive writable
               51    0014'CA   9E  001D   758          MOVAB   HOOKBASE(R10),R1                ; Get address of exec hook table
                     50   61   D0  0022   759 60$:     MOVL    (R1),R0                         ; Get address of a hook
                          0D   13  0025   760          BEQL    70$                             ; 0 ends the JMP/JSB type hooks
                    80    0A   A1  B0  0027   761          MOVW    10(R1),(R0)+                    ; Replace the JMP/JSB opcode
                    80    0C   A1  D0  002B   762          MOVL    12(R1),(R0)+                    ; Replace the longword destination
                     51   10   C0  002F   763          ADDL2   #16,R1                          ; Point to next hook in table
                          EE   11  0032   764          BRB     60$                             ; Repeat for each hook
                     51   04   C0  0034   765 70$:     ADDL    #4,R1                           ; Skip the end indicator
                     50   81   D0  0037   766 80$:     MOVL    (R1)+,R0                        ; Get address of SCB vector changed
                          05   13  003A   767          BEQL    90$                             ; 0 ends the SCB vector changes
                     60   81   D0  003C   768          MOVL    (R1)+,(R0)                      ; Replace SCB vector
                          F6   11  003F   769          BRB     80$                             ; Repeat for each SCB change
                00000000'GF   D4  0041   770 90$:     CLRL    G^EXE$GL_MP                     ; Reset pointer to multi-proc code
                00000000'GF   16  0047   771          JSB     G^INI$RDONLY                    ; Restore protection on executive pages
                                0047   772          ENBINT                                  ; Lower IPL for deallocate of pool
                          12   8E   DA  0047                  MTPR    (SP)+,S^#PR$_IPL
```

```
05  0050    773          RSB                                ; Return to caller
    0051    774
    0051    775
    0051    776          .END    MPS$DCL
```

```
$$T2                    = 00000004
BUSY_STATE_DSC            000000A4 R      04
CLI$GET_VALUE             ******** X    04
CLI$_IVVERB             = 00038090
DCL_LINE_DSC              00000062 R      04
DEA_ERR_EXIT              00000306 R      04
DROP_STATE_DSC            000000BF R      04
DSC$A_POINTER           = 00000004
DSC$B_CLASS             = 00000003
DSC$K_CLASS_D           = 00000002
DYN$C_LC_MP             = 00000003
DYN$C_LOADCODE          = 00000062
ERR$_CLMEMUSED            ******** X    04
ERR$_MA780_REQ            ******** X    04
ERR$_SHMDBUSE             ******** X    04
ERRORX1                   00000260 R      04
ERRORX2                   00000160 R      04
ERRORX3                   000001AA R      04
ERRORX4                   000001B1 R      04
ERRORX5                   00000164 R      04
EXE$ALONONPAGED           ******** X    04
EXE$CMODEXECX             ******** X    04
EXE$DEANONPAGED           ******** X    04
EXE$GB_CPUTYPE            ******** X    04
EXE$GL_FLAGS              ******** X    04
EXE$GL_MP                 ******** X    04
EXE$GL_RPB                ******** X    04
EXE$GL_SCB                ******** X    04
EXE$GL_SHBLIST            ******** X    04
EXE$GQ_SYSTIME            ******** X    04
EXE$MPSTART               ******** X    04
EXE$V_SSINHIBIT           ******** X    04
EXEC_STATE_DSC            000000B0 R      04
EXIT                      00000229 R      04
EXIT2                     00000554 R      04
FOUND_MPM                 0000041A R      04
GETDATA                   000001B8 R      04
GET_LINE_DSC              000000E4 R      04
GET_VERB_DSC              000000D7 R      04
HOOKBASE                  00000014 R      02
HOOKEND                   000000AC R      02
HOOKS_DONE                00000517 R      04
IDLE_STATE_DSC            000000CB R      04
ILSTATE_CTL_DSC           00000124 R      04
INI$RDONLY                ******** X    04
INI$WRITABLE              ******** X    04
INIT_STATE_DSC            00000086 R      04
INSTALL_DONE              00000525 R      04
IPL$_ASTDEL             = 00000002
IPL$_SCHED              = 00000003
IPL$_SYNCH              = 00000008
IRP$B_TYPE              = 0000000A
IRP$W_SIZE              = 00000008
LCK$V_INTERLOCK         = 00000000
LIB$PUT_OUTPUT            ******** X    04
LOCAL_MEM_ERR             00000261 R      04
LOCKEND                   00000525 R      04
LOCKRANGE                 00000000 R      04
LOCKSTART                 000004E1 R      04
LOC_MPM                   00000339 R      04
MA780_CNCT_ERR            00000271 R      04
MA780_NOT_USED            00000269 R      04
MISC_RELOC                00000488 R      04
MMG$GL_SBICONF            ********   X    04
MMG$GL_SPTBASE            ********   X    04
MOVE_CODE                 000004D6 R      04
MPH$ASTDELHK              ********   X    02
MPH$BUGCHKHK              ********   X    02
MPH$INVALIDHK             ********   X    02
MPH$NEWLVLHK              ********   X    02
MPH$QAST                  ********   X    02
MPH$RESCHED               ********   X    02
MPH$SCHED                 ********   X    02
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_MPMBASE            ********   X    04
MPS$ASTNEWLVL             ********   X    02
MPS$ASTSCHEDCHK           ********   X    02
MPS$BEGIN                 00000000 RG     02
MPS$BUGCHECK              ********   X    02
MPS$CMODKRNLX             ********   X    04
MPS$DCL                   00000165 RG     04
MPS$END                   00000000 RG     03
MPS$GL_CURPCB             ********   X    04
MPS$GL_INTERLOCK          ********   X    04
MPS$GL_ISP                ********   X    04
MPS$GL_MPMIIR             ********   X    04
MPS$GL_NULLCPU            ********   X    05
MPS$GL_POOLDSC            00000000 RG     02
MPS$GL_PRIMSKC            ********   X    04
MPS$GL_PRIMSKT            ********   X    04
MPS$GL_SCBB               ********   X    04
MPS$GL_STATE              ********   X    04
MPS$GL_STOPFLAG           ********   X    04
MPS$GL_STRTVA             ********   X    04
MPS$GQ_MPSTRTIM           ********   X    04
MPS$HOOKTBL               0000000C RG     02
MPS$INT58                 ********   X    04
MPS$INTSCND               ********   X    04
MPS$INVALID               ********   X    02
MPS$K_IDLESTATE         = 00000001
MPS$K_INITSTATE         = 00000005
MPS$K_STOPSTATE         = 00000006
MPS$LOAD                  0000022D RG     04
MPS$LOADK                 00000279 RG     04
MPS$MAINIT                ********   X    04
MPS$PINTSR                ********   X    04
MPS$QAST                  ********   X    02
MPS$RESCHED               ********   X    02
MPS$RESCHEDIPL5           ********   X    04
MPS$RPB_WAIT              00000514 RG     04
```

```
MPS$SCHED                       ********    X   02        VA$V_VPN                        = 00000009
MPS$SINTSR                      ********    X   04        XDELIBRK                          ********    X   04
MPS$UNEXPINT                    ********    X   04
MPS$UNHOOK                      00000000 RG     05
MPS$UNLOAD                      00000535 RG     04
MPS$UNLOADK                     00000555 RG     04
MPS$V_STOPACK1                = 00000001
MPS$V_STOPREQ                 = 00000000
NDT$_MPM0                     = 00000040
OUTPUT_BUFFER                   00000014 R      04
OUTPUT_BUF_DSC                  0000005A R      04
OUTPUT_LENGTH                   00000010 R      04
PCB$B_PRI                     = 0000000B
PCB$L_PHD                     = 0000006C
PCB$W_STATE                   = 0000002C
PHD$L_CPUTIM                  = 00000038
PR$_IPL                       = 00000012
PR$_SIRR                      = 00000014
PTE$S_PFN                     = 00000015
PTE$V_PFN                     = 00000000
RPB$B_CONFREG                 = 00000090
RPB$B_WAIT                    = 00000100
RPB$L_BOOTR5                  = 00000030
RPB$L_BUGCHK                  = 000000FC
RPB$L_HALTPC                  = 00000010
RPB$V_MPM                     = 0000000B
RPB$V_USEMPM                  = 0000000C
RPB_BUGCHK                    = 00000510 R      04
RPB_LOOPSIZ                   = 00000003
SCB$AL_BASE                     ********    X   04
SCB_IPL14                       00000088 R      02
SCB_IPL16                       00000090 R      02
SCB_LOOP                        0000032F R      04
SCB_VEC94                       00000098 R      02
SCB_VECBC                       000000A0 R      02
SCH$AQ_COMT                     ********    X   04
SCH$C_COM                     = 0000000C
SCH$GL_COMQS                    ********    X   04
SCH$GL_NULLPCB                  ********    X   05
SET_PHYS                        0000047A R      04
SHOW_CPU                        000001CE R      04
SS$_DEVOFFLINE                = 00000084
SS$_NORMAL                    = 00000001
SS$_SHMNOTCNCT                = 0000037C
START_CPU                       000001B2 R      04
STATES                          0000006E R      04
STATE_CTL_DSC                   000000F1 R      04
STATE_VALUE                     0000006A R      04
STOPEND                         000005DC R      04
STOPRANGE                       00000008 R      04
STOPSTART                       00000555 R      04
STOP_CPU                        000001B5 R      04
STOP_STATE_DSC                  00000098 R      04
SYS$CMKRNL                      ********   GX   04
SYS$FAO                         ********    X   04
SYS$LKWSET                      ********   GX   04
VA$S_VPN                      = 00000015
```

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
$$$$$$BEGIN	000000AC (172.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
_END	00000000 (0.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
_MPLOAD	000005DC (1500.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$AEXNONPAGED	00000051 (81.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.08	00:00:00.80
Command processing	136	00:00:00.87	00:00:05.33
Pass 1	464	00:00:17.71	00:00:43.37
Symbol table sort	0	00:00:02.80	00:00:05.53
Pass 2	165	00:00:03.78	00:00:08.57
Symbol table output	22	00:00:00.22	00:00:00.84
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	827	00:00:25.49	00:01:04.47

The working set limit was 1800 pages.
101303 bytes (198 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1786 non-local and 25 local symbols.
781 source lines were read in Pass 1, producing 26 object records in Pass 2.
37 pages of virtual memory were used to define 35 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MP.OBJ]MP.MLB;1	4
_255DUA28:[SYS.OBJ]LIB.MLB;1	15
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	31

1914 GETS were required to define 31 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:MPLOAD/OBJ=OBJ$:MPLOAD MSRC$:MPPREFIX/UPDATE=(ENH$:MPPREFIX)+MSRC$:MPLOAD/UPDATE=(ENH$:MPLOAD)+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