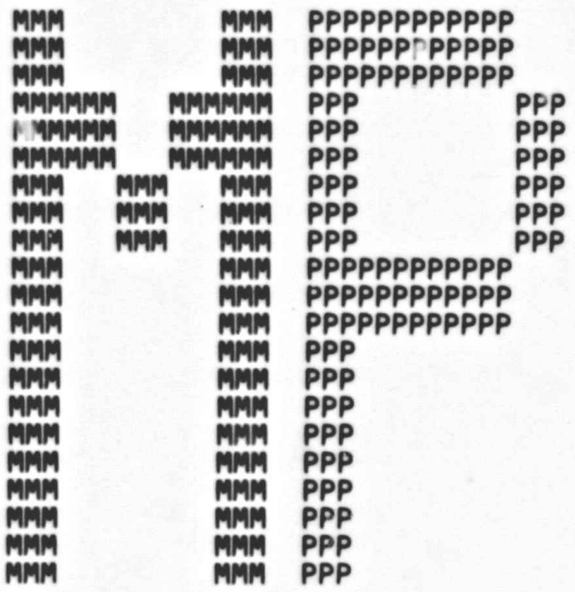
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM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

FILEID**MPINTEXC

```
MM      MM  PPPPPPPP     IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  XX      XX    CCCCCCCC
MM      MM  PPPPPPPP     IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  XX      XX    CCCCCCCC
MMMM  MMMM  PP      PP     II     NN      NN      TT      EE          XX      XX  CC
MMMM  MMMM  PP      PP     II     NN      NN      TT      EE          XX      XX  CC
MM  MM  MM  PP      PP     II     NNNN    NN      TT      EE            XX  XX    CC
MM  MM  MM  PP      PP     II     NNNN    NN      TT      EE            XX  XX    CC
MM      MM  PPPPPPPP       II     NN  NN  NN      TT      EEEEEEEE        XX      CC
MM      MM  PPPPPPPP       II     NN  NN  NN      TT      EEEEEEEE        XX      CC
MM      MM  PP             II     NN    NNNN      TT      EE            XX  XX    CC
MM      MM  PP             II     NN    NNNN      TT      EE            XX  XX    CC
MM      MM  PP             II     NN      NN      TT      EE          XX      XX  CC              ....
MM      MM  PP             II     NN      NN      TT      EE          XX      XX  CC              ....
MM      MM  PP          IIIIII    NN      NN      TT      EEEEEEEEEE  XX      XX    CCCCCCCC      ....
MM      MM  PP          IIIIII    NN      NN      TT      EEEEEEEEEE  XX      XX    CCCCCCCC      ....

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  MPINTEXC - SECONDARY PROCESSOR INTERRUPT AND EXCEPTIONS
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:  Unhandled exception and interrupt routines
0000    34 ;
0000    35 ; Environment: MODE=Kernel
0000    36 ;
0000    37 ; Author:  RICHARD I. HUSTVEDT, Creation date:  15-May-1979
0000    38 ;
0000    39 ; Modified by:
0000    40 ;
0000    41 ;       V03-005 KDM0030         Kathleen D. Morse       18-Nov-1982
0000    42 ;               Add logic that allows primary to continue execution
0000    43 ;               of secondary-specific code without turning into a
0000    44 ;               secondary processor.  Remove pagefault handler as no
0000    45 ;               longer needed.
0000    46 ;
0000    47 ;       V03-004 KDM0018         Kathleen D. Morse       08-Oct-1982
0000    48 ;               Move all logic to handle kernel mode system services
0000    49 ;               on secondary to MPCMOD.MAR.  This includes the AST
0000    50 ;               exit system service and the code that special-cased it.
0000    51 ;
0000    52 ;       V03-003 KDM0005         Kathleen D. Morse       31-Aug-1982
```

```
                                                L 5
0000     53 ;
0000     54 ;                  Add a secondary pagefault handler, to allow backing out
0000     55 ;                  of MPS$COMPAT if it takes a pagefault.
0000     56 ;
0000     57 ;          V03-002 RIH0001         Richard I. Hustvedt     1-Jun-1982
0000     58 ;                  Correct handling of AST queue by secondary processor to
0000     59 ;                  avoid losing some AST notifications by incorrectly computing
0000     60 ;                  PHD$B_ASTLVL.
0000     61 ;
0000     62 ;  01    -
0000     63 ;--
```

```
0000     65 ;
0000     66 ; INCLUDE FILES:
0000     67 ;
0000     68
0000     69 ;
0000     70 ; MACROS:
0000     71 ;
0000     72
0000     73
0000     74 ;
0000     75 ; Macro to define an interrupt service routine label
0000     76 ; for unexpected interrupts
0000     77 ;
0000     78          .MACRO  ISRDEF,VNUM
0000     79          .ALIGN  LONG                        ; Make all vectors long word alligned
0000     80 ERL$VEC'VNUM::                               ; Unexpected interrupt service rtn label
0000     81          .ENDM   ISRDEF
0000     82 ;
0000     83 ; Macro to define the interrupt service routine labels
0000     84 ; for an adapter
0000     85 ;
0000     86          .MACRO  ADPISR,SLOT
0000     87 VECTOR = SLOT * 4 + 256
0000     88          .REPT   4
0000     89          ISRDEF  \VECTOR
0000     90 VECTOR = VECTOR + <16 * 4>
0000     91          .ENDR
0000     92          BSBB    ADP_HANDLER                 ; Call interrupt service routine
0000     93          .BYTE   SLOT                        ; TR index for this int srv rtn
0000     94          .ENDM   ADPISR
0000     95
0000     96 ;
0000     97 ; EQUATED SYMBOLS:
0000     98 ;
0000     99
0000    100          $EMBDEF <UI>                        ; Error log message buffer offsets
0000    101          $IPLDEF                             ; Interrupt priority levels
0000    102          $PCBDEF                             ; Process control block offsets
0000    103          $PHDDEF                             ; Process header block offsets
0000    104          $PRDEF                              ; Define processor register numbers
0000    105          $PSLDEF                             ; Define PSL fields
0000    106          $RPBDEF                             ; Define reboot parameter block
0000    107          $SSDEF                              ; Define system status codes
0000    108
0000    109                                              ;***
0000    110 ;DEBUG=1                                     ;***If defined, enable unexpected
0000    111                                              ;*** interrupt identifies vector #
0000    112
0000    113 ;
0000    114 ; OWN STORAGE:
0000    115 ;
```

```
                        0000   117          .SBTTL  MPS$UNEXPINT - Unexpected interrupt routine
                        0000   118
                        0000   119 ;+
                        0000   120 ; ERL$VEC'VNUM - INTERRUPT SERVICE FOR SCB VECTOR VNUM.
                        0000   121 ; MPS$UNEXPINT - General unexpected interrupt service routine
                        0000   122 ;
                        0000   123 ; These interrupt service routines are executed for unused SCB vectors.
                        0000   124 ; If DEBUG is defined, each interrupt service calls ERL$UNEXP with
                        0000   125 ; the <vector offset>/2 into the SCB as a 1 byte argument.  If
                        0000   126 ; DEBUG is not defined, all interrupt service routines collapse to
                        0000   127 ; global labels equal to ERL$UNEXP.  There are enought interrupt
                        0000   128 ; service routines for the architectural page of the SCB, i.e.,
                        0000   129 ; 128 routines.
                        0000   130 ;
                        0000   131 ; INPUTS:
                        0000   132 ;
                        0000   133 ;       0(SP) - PC at interrupt
                        0000   134 ;       4(SP) - PSL at interrupt
                        0000   135 ;
                        0000   136 ; OUTPUTS:
                        0000   137 ;
                        0000   138 ;
                        0000   139 ;-
                        0000   140
                00000000 0000  141          .PSECT  $AEXNONPAGED,LONG
                        0000   142
                        0000   143          .ALIGN  LONG
                        0000   144
                        0000   145 ;
                        0000   146 ; All unused vectors in the SCB point to locations in the following table.
                        0000   147 ; There is one longword for each longword in the SCB.  The unused vectors
                        0000   148 ; may sometimes receive interrupts which must be handled in some way.  For
                        0000   149 ; all cpu-type interrupts, the routine MPS$UNEXPINT is executed.  For all
                        0000   150 ; adapter interrupts, the routine ADP_HANDLER is executed.  The former
                        0000   151 ; passes the process currently executing back to the primary or bugchecks,
                        0000   152 ; depending upon whether or not it is on the interrupt stack.  The latter
                        0000   153 ; routine creates and error log entry for the unexpected interrupt and REIs.
                        0000   154 ;
                        0000   155          .ALIGN  LONG
                        0000   156 CPU_UNEXP:
                00000000 0000  157          VNUM=000                        ; First vector = 0
                        0000   158          .REPT   64                      ; ISRs for cpu interrupts only
                        0000   159          ISRDEF  \VNUM                   ; Define ERL$VEC'VNUM label and ISR
                        0000   160          BSBW    MPS$UNEXPINT            ; Call unexpected interrupt service rtn
                        0000   161          .BYTE   <VNUM>/2                ; Identify vector offset into SCB
                        0000   162          VNUM=VNUM+4                     ; Next vector
                        0000   163          .ENDR
                        0100   164 ADP_UNEXP:
                00000000 0100  165          NEXUS = 0                       ; First adapter = 0
                        0100   166          .REPT   16                      ; ISR's for 16 adapters only
                        0100   167          ADPISR  \NEXUS                  ; Define ERL$VEC'VNUM labels and ISRs
                        0100   168          NEXUS = NEXUS + 1               ; Next adapter
                        0100   169          .ENDR
                        013F   170 ;
                        013F   171 ; Unexpected adapter interrupt handler:
                        013F   172 ;
                        013F   173 ; This routine is entered whenever an adapter on the secondary interrupts.
```

```
                                013F   174 ; It logs the unexpected interrupt by creating an error log entry and then
                                013F   175 ; attempts to clear the interrupt.
                                013F   176 ;
                                013F   177
                                013F   178                .ALIGN  LONG
                                0140   179 ADP_HANDLER:
  6E    00000102'8F    C2     0140   180                SUBL    #ADP_UNEXP+2,(SP)           ; Compute adapter offset
          6E    02     C6     0147   181                DIVL    #2,(SP)                     ; Compute adapter slot/TR number
                1F     BB     014A   182                PUSHR   #^M<R0,R1,R2,R3,R4>         ; Save registers
        53    14 AE    D0     014C   183                MOVL    5*4(SP),R3                  ; Retrieve slot number
54    00000000'FF43    D0     0150   184                MOVL    @MMG$GL_SBICONF[R3],R4      ; Get address of adapter registers
              04 A4    D4     0158   185                CLRL    4(R4)                       ; Disable adapter interrupts
          54    64     D0     015B   186                MOVL    (R4),R4                     ; Get the adapter's configuration reg
                       015E   187 10$:
          51    18     D0     015E   188                MOVL    #EMB$C_UI_LENGTH,R1         ; Set size of message to allocate
             FE9C'     30     0161   189                BSBW    MPS$ALLOCEMB                ; Allocate an error log buffer
            11 50      E9     0164   190                BLBC    R0,20$                      ; Branch if none available
04 A2     0061 8F      B0     0167   191                MOVW    #EMB$C_UI,EMB$W_UI_ENTRY(R2) ; Set message type
      10 A2    53     D0     016D   192                MOVL    R3,EMB$L_UI_TR(R2)          ; Set slot/TR number
      14 A2    54     D0     0171   193                MOVL    R4,EMB$L_UI_CSR(R2)         ; Set configuration register value
             FE88'     30     0175   194                BSBW    MPS$RELEASEMB               ; Release buffer
                1F     BA     0178   195 20$:           POPR    #^M<R0,R1,R2,R3,R4>         ; Restore registers
          5E    04     C0     017A   196                ADDL    #4,SP                       ; Remove slot number
                02     017D   197                REI                                         ;
                       017E   198
```

```
                        ;+
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ; This routine is the unexpected interrupt handler.  It will halt
                        ; if vector identification (DEBUG) is enabled.  On the top of the
                        ; stack will be the <vector offset>/2 into the SCB.
                        ;
                        ; At the time the routine is entered, the stack looks like the following:
                        ;
                        ;                 +---------------------------+
                        ;                 !   PC from a BSBW in SCB   !
                        ;                 +---------------------------+
                        ;                 !                           !
                        ;                 \         optional          \
                        ;                 \       parameters          \
                        ;                 !                           !
                        ;                 +---------------------------+
                        ;                 ! PC at time of exception   !
                        ;                 +---------------------------+
                        ;                 ! PSL at time of exception  !
                        ;                 +---------------------------+
                        ;
                        ; The secondary processor computes the address of the vector for
                        ; this particular interrupt in the primary's SCB.  It places this
                        ; PC and the current PSL on the stack and returns the process to
                        ; the primary processor.  A SVPCTX is done by the secondary, which
                        ; takes the PC-PSL pair and places them in the hardware PCB.  The
                        ; primary will immediately handle the interrupt when it starts
                        ; executing this process.
                        ;
                        ; This code is executed for any exception that happens on the secondary,
                        ; e.g., access violation, pagefault, change mode to kernel, etc.
                        ;
                        ; ENVIRONMENT:
                        ;
                        ;       Executes on the secondary processor.
                        ;
                        ;-

                                .ALIGN  LONG
                        MPS$UNEXPINT::                                  ;
      6E    00 BE   9A          MOVZBL  @(SP),(SP)                    ;***Overlay return with vector offset
         6E   02   C4          MULL    #2,(SP)                       ;***Convert arg to vector offset

                                .IF     DF,MPPFMSWT
                                BSBW    MPS$PFM_UNEXP                 ; Gather performance statistics
                                .ENDC

7E    6E    00000000'GF   C1          ADDL3   G^EXE$GL_SCB,(SP),-(SP) ; Compute proper PC for continue
                                                                      ;  in primary processor
      6E    00 BE   03    CB          BICL3   #3,@(SP),(SP)           ; Continuation PC is the SCB vector
                                                                      ; Location in the primary less int stk bit
               04 AE    DC          MOVPSL  4(SP)                   ; Save current PSL
         0B 04 AE   1A    E0          BBS     #PSL$V_IS,4(SP),10$    ; Br if on interrupt stack
   04 AE    001F0000 8F   CA          BICL    #PSL$M_IPL,4(SP)       ; Force IPL to zero

                        ;
                        ; Now drop the current process and ask the primary processor for another.
```

```
                     01A4    257 ;
FE59'  31            01A4    258          BRW     MPS$MPSCHED2            ; And get another
                     01A7    259
                     01A7    260 10$:     SECBUG_CHECK MPUNEXPINT,FATAL   ; Unexpected interrupt or exception
```

```
                                                E 6
                     01AC   262          .SBTTL AST DELIVERY INTERRUPT SERVICE
                     01AC   263 ;+
                     01AC   264 ; FUNCTIONAL DESCRIPTION:
                     01AC   265 ;
                     01AC   266 ; This routine is entered from the AST delivery interrupt.  If it
                     01AC   267 ; interrupts some kernel mode execution, then the AST delivery cannot
                     01AC   268 ; be handled now and it merely sets the hardware register ASTLVL so
                     01AC   269 ; that the interrupt is canceled.  If any other mode is interrupted, then
                     01AC   270 ; the process is folded up and returned to the primary processor.  When
                     01AC   271 ; the primary processor does a LDPCTX and an REI to start executing this
                     01AC   272 ; process, the REI finds that the hardware register ASTLVL differs from
                     01AC   273 ; that in the process header and the AST is delivered.
                     01AC   274 ;
                     01AC   275 ;
                     01AC   276 ; INPUTS:
                     01AC   277 ;
                     01AC   278 ;        (SP) - PC at time of interrupt
                     01AC   279 ;      4(SP) - PSL at time of interrupt
                     01AC   280 ;
                     01AC   281 ; ENVIRONMENT:
                     01AC   282 ;
                     01AC   283 ;        Executes on the secondary processor.
                     01AC   284 ;
                     01AC   285 ;-
                     01AC   286
                     01AC   287          .ALIGN  LONG                        ; Longword align interrupt routines
                     01AC   288 MPS$ASTDEL::
          07 AE  03  93 01AC   289          BITB    #<PSL$M_CURMOD -            ; Check if a kernel mode routine
                     01B0   290                  @<-PSL$V_CURMOD>>, -        ;  was interrupted
                     01B0   291                  4+<PSL$V_CURMOD@-3>(SP)     ;
                04   12 01B0   292          BNEQ    MPS$UNEXPINT1               ; Br if not kernel, go handle AST now
                13   04 DA 01B2   293          MTPR    #4,#PR$_ASTLVL              ; Don't handle the AST delivery now,
                     01B5   294                                                  ;  and disable the interrupt
                   02 01B5   295          REI                                 ; Return from interrupt
                     01B6   296
                     01B6   297 ;
                     01B6   298 ; The following is in the format of the SCB vectors.
                     01B6   299 ;
                     01B6   300 MPS$UNEXPINT1:
00020000 8F          DD 01B6   301          PUSHL   #<IPL$_ASTDEL @ PSL$V_IPL> ; Return process to primary to do AST
00000000'GF          9F 01BC   302          PUSHAB  G^SCH$ASTDEL                ; delivery and execution
                     01C2   303
                     01C2   304          .IF     DF,MPPFMSWT
                     01C2   305          BSBW    MPS$PFM_ASTDEL              ; Gather performance measurement data
                     01C2   306          .ENDC
                     01C2   307
                FE3B' 31 01C2   308          BRW     MPS$MPSCHED2                ; Go fold up the process & hand it back
```

```
                        .SBTTL  MPS$KERSTKNV - KERNEL STACK NOT VALID FAULT
                ;+
                ; MPS$KERSTKNV - KERNEL STACK NOT VALID FAULT
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ; This routine is automatically vectored to when a kernel stack not
                ; valid is detected during a change mode instruction, during an REI
                ; instruction, or during an attempt to push exception information on
                ; the kernel stack.  This exception runs on the interrupt stack.  The
                ; state of the stack on entry is:
                ;
                ;       00(SP) - Exception PC
                ;       04(SP) - Exception PSL
                ;
                ; A fatal kernel stack not valid bugcheck is declared.
                ;
                ; ENVIRONMENT:
                ;
                ;       Executes on the secondary processor.
                ;
                ;-

                        .ALIGN  LONG
01C8    334 MPS$KERSTKNV::                              ; Kernel stack not valid fault
01C8    335             SECBUG_CHECK MPKNLSTKNV,FATAL   ; Kernel stack not valid bugcheck
01CD    336
```

```
                     01CD   338          .SBTTL  SECONDARY PROCESSOR COMPATIBILITY MODE FAULT HANDLER
                     01CD   339 ;+
                     01CD   340 ; FUNCTIONAL DESCRIPTION:
                     01CD   341 ;
                     01CD   342 ; This routine is automatically vectored to when a compatibility mode
                     01CD   343 ; exception is detected for a process executing on the secondary.  The
                     01CD   344 ; state of the stack on entry is:
                     01CD   345 ;
                     01CD   346 ;       00(SP) - Compatibility exception code
                     01CD   347 ;       04(SP) - Exception PC
                     01CD   348 ;       08(SP) - Exception PSL
                     01CD   349 ;
                     01CD   350 ; Possible compatibility exception codes are:
                     01CD   351 ;
                     01CD   352 ;       0 - Reserved instruction execution
                     01CD   353 ;       1 - BPT instruction execution
                     01CD   354 ;       2 - IOT instruction execution
                     01CD   355 ;       3 - EMT instruction execution
                     01CD   356 ;       4 - Trap instruction execution
                     01CD   357 ;       5 - Illegal instruction execution
                     01CD   358 ;       6 - Odd address fault
                     01CD   359 ;       7 - Tbit trap
                     01CD   360 ;
                     01CD   361 ; The exception name followed by the number of exception arguments are
                     01CD   362 ; pushed on the stack.  Final processing is accomplished in common code.
                     01CD   363 ;
                     01CD   364 ; If there is no compatibility mode handler declared, then the process
                     01CD   365 ; is folded up and handed back to the primary, in such a way that the
                     01CD   366 ; primary will execute in EXCEPTION.
                     01CD   367 ;
                     01CD   368 ; Environment:
                     01CD   369 ;
                     01CD   370 ;       Executed by the secondary processor, in kernel mode, on kernel stack.
                     01CD   371 ;       If interrupted at any point, can continue execution on primary.
                     01CD   372 ;
                     01CD   373 ;-
                     01CD   374
                     01CD   375          .ALIGN  LONG
                     01D0   376 MPS$COMPAT::                                 ; Compatibility mode faults on secondary
00000000'9F   50  7D 01D0   377          MOVQ    R0,@#CTL$AL_CMCNTX          ; Save R0,R1 in compat context region
50  00000008'9F  DE 01D7   378          MOVAL   @#CTL$AL_CMCNTX+8,R0        ; Get addr of compatibility context area
        80  52   7D 01DE   379          MOVQ    R2,(R0)+                    ; Save R2 and R3
        80  54   7D 01E1   380          MOVQ    R4,(R0)+                    ; Save R4 and R5
        80  56   D0 01E4   381          MOVL    R6,(R0)+                    ; Save R6
        80  8E   D0 01E7   382          MOVL    (SP)+,(R0)+                 ; Save exception code & clean off stack
        60  8E   7D 01EA   383          MOVQ    (SP)+,(R0)                  ; Save PC/PSL and clean off stack
    03C00000 8F   DD 01ED   384          PUSHL   #<PSL$C_USER@PSL$V_PRVMOD>!<PSL$C_USER@PSL$V_CURMOD> ; Fabricate
                     01F3   385                                              ; a PSL for CME
00000000'9F   DD 01F3   386          PUSHL   @#CTL$GL_CMHANDLR               ; Compatibility mode handler address
        01   13 01F9   387          BEQL    20$                             ; Branch if none specified
             02 01FB   388          REI                                     ; Jump to compatibility handler
                     01FC   389
                     01FC   390 ;
                     01FC   391 ; No compatibility mode handler was declared.  Restore the stack and
                     01FC   392 ; saved register, and return process to primary to continue through
                     01FC   393 ; normal exception code.  R0 now points to the saved PC in the
                     01FC   394 ; compatibility context area.
```

```
                                01FC   395 ;
                                01FC   396
                    6E    60    7D 01FC   397 20$:    MOVQ    (R0),(SP)                ; Restore exception PC/PSL
                          70    DD 01FF   398          PUSHL   -(R0)                    ; Push exception code again
                 50    E4 A0    D0 0201   399          MOVL    -28(R0),R0               ; Restore R0 from context area
                7E   042C 8F    3C 0205   400          MOVZWL  #SS$_COMPAT,-(SP)        ; Set exception name
                          04    DD 020A   401          PUSHL   #4                       ; Set number of signal arguments
                                020C   402          IFPRIMARY <JMP G^EXE$EXCEPTION> ;IF PRIMARY, CONTINUE WITH EXCEPTION
                                0225   403                                          ;IF SECONDARY, RETURN PROCESS TO PRIMARY
          7E    10 AE   02    18 EF 0225   404          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,16(SP),-(SP) ;Create PC/PSL w/prev
                6E    6E    16 9C 022B   405          ROTL    #PSL$V_PRVMOD,(SP),(SP) ; mode correct & current mode of kernel
                 00000000'GF    9F 022F   406          PUSHAB  G^EXE$EXCEPTION          ; Adr where primary will execute
                          FDC8' 31 0235   407          BRW     MPS$MPSCHED2             ; Hand process back to primary
                                0238   408
```

```
                     0238   410          .SBTTL  SYMBOLS NEEDED FOR XDELTA
                     0238   411 ;
                     0238   412 ; The following symbols are defined so that XDELTA can be linked
                     0238   413 ; into MP.EXE for debugging the secondary processor.
                     0238   414 ;
00000018             0238   415 EXE$ROPRAND==ERL$VEC24
00000020             0238   416 EXE$ACVIOLAT==ERL$VEC32
00000024             0238   417 MMG$PAGEFAULT==ERL$VEC36
00000028             0238   418 EXE$TBIT==ERL$VEC40
0000002C             0238   419 EXE$BREAK==ERL$VEC44
                     0238   420          .END
```

```
                                                J 6

ADP_HANDLER                     00000140 R      02      ERL$VEC260                      00000104 RG     02
ADP_UNEXP                       00000100 R      02      ERL$VEC264                      00000108 RG     02
BUG$_MPKNLSTKNV                 ******** X      02      ERL$VEC268                      0000010C RG     02
BUG$_MPUNEXPINT                 ******** X      02      ERL$VEC272                      00000110 RG     02
CPU_UNEXP                       00000000 R      02      ERL$VEC276                      00000114 RG     02
CTL$AL_CMCNTX                   ******** X      02      ERL$VEC28                       0000001C RG     02
CTL$GL_CMHANDLR                 ******** X      02      ERL$VEC280                      00000118 RG     02
EMB$C_UI                    =   00000061                ERL$VEC284                      0000011C RG     02
EMB$C_UI_LENGTH             =   00000018                ERL$VEC288                      00000120 RG     02
EMB$L_UI_CSR                =   00000014                ERL$VEC292                      00000124 RG     02
EMB$L_UI_TR                 =   00000010                ERL$VEC296                      00000128 RG     02
EMB$W_UI_ENTRY              =   00000004                ERL$VEC300                      0000012C RG     02
ERL$VEC0                        00000000 RG     02      ERL$VEC304                      00000130 RG     02
ERL$VEC100                      00000064 RG     02      ERL$VEC308                      00000134 RG     02
ERL$VEC104                      00000068 RG     02      ERL$VEC312                      00000138 RG     02
ERL$VEC108                      0000006C RG     02      ERL$VEC316                      0000013C RG     02
ERL$VEC112                      00000070 RG     02      ERL$VEC32                       00000020 RG     02
ERL$VEC116                      00000074 RG     02      ERL$VEC320                      00000100 RG     02
ERL$VEC12                       0000000C RG     02      ERL$VEC324                      00000104 RG     02
ERL$VEC120                      00000078 RG     02      ERL$VEC328                      00000108 RG     02
ERL$VEC124                      0000007C RG     02      ERL$VEC332                      0000010C RG     02
ERL$VEC128                      00000080 RG     02      ERL$VEC336                      00000110 RG     02
ERL$VEC132                      00000084 RG     02      ERL$VEC340                      00000114 RG     02
ERL$VEC136                      00000088 RG     02      ERL$VEC344                      00000118 RG     02
ERL$VEC140                      0000008C RG     02      ERL$VEC348                      0000011C RG     02
ERL$VEC144                      00000090 RG     02      ERL$VEC352                      00000120 RG     02
ERL$VEC148                      00000094 RG     02      ERL$VEC356                      00000124 RG     02
ERL$VEC152                      00000098 RG     02      ERL$VEC36                       00000024 RG     02
ERL$VEC156                      0000009C RG     02      ERL$VEC360                      00000128 RG     02
ERL$VEC16                       00000010 RG     02      ERL$VEC364                      0000012C RG     02
ERL$VEC160                      000000A0 RG     02      ERL$VEC368                      00000130 RG     02
ERL$VEC164                      000000A4 RG     02      ERL$VEC372                      00000134 RG     02
ERL$VEC168                      000000A8 RG     02      ERL$VEC376                      00000138 RG     02
ERL$VEC172                      000000AC RG     02      ERL$VEC380                      0000013C RG     02
ERL$VEC176                      000000B0 RG     02      ERL$VEC384                      00000100 RG     02
ERL$VEC180                      000000B4 RG     02      ERL$VEC388                      00000104 RG     02
ERL$VEC184                      000000B8 RG     02      ERL$VEC392                      00000108 RG     02
ERL$VEC188                      000000BC RG     02      ERL$VEC396                      0000010C RG     02
ERL$VEC192                      000000C0 RG     02      ERL$VEC4                        00000004 RG     02
ERL$VEC196                      000000C4 RG     02      ERL$VEC40                       00000028 RG     02
ERL$VEC20                       00000014 RG     02      ERL$VEC400                      00000110 RG     02
ERL$VEC200                      000000C8 RG     02      ERL$VEC404                      00000114 RG     02
ERL$VEC204                      000000CC RG     02      ERL$VEC408                      00000118 RG     02
ERL$VEC208                      000000D0 RG     02      ERL$VEC412                      0000011C RG     02
ERL$VEC212                      000000D4 RG     02      ERL$VEC416                      00000120 RG     02
ERL$VEC216                      000000D8 RG     02      ERL$VEC420                      00000124 RG     02
ERL$VEC220                      000000DC RG     02      ERL$VEC424                      00000128 RG     02
ERL$VEC224                      000000E0 RG     02      ERL$VEC428                      0000012C RG     02
ERL$VEC228                      000000E4 RG     02      ERL$VEC432                      00000130 RG     02
ERL$VEC232                      000000E8 RG     02      ERL$VEC436                      00000134 RG     02
ERL$VEC236                      000000EC RG     02      ERL$VEC44                       0000002C RG     02
ERL$VEC24                       00000018 RG     02      ERL$VEC440                      00000138 RG     02
ERL$VEC240                      000000F0 RG     02      ERL$VEC444                      0000013C RG     02
ERL$VEC244                      000000F4 RG     02      ERL$VEC448                      00000100 RG     02
ERL$VEC248                      000000F8 RG     02      ERL$VEC452                      00000104 RG     02
ERL$VEC252                      000000FC RG     02      ERL$VEC456                      00000108 RG     02
ERL$VEC256                      00000100 RG     02      ERL$VEC460                      0000010C RG     02
```

```
ERL$VEC464                      00000110 RG    02
ERL$VEC468                      00000114 RG    02
ERL$VEC472                      00000118 RG    02
ERL$VEC476                      0000011C RG    02
ERL$VEC48                       00000030 RG    02
ERL$VEC480                      00000120 RG    02
ERL$VEC484                      00000124 RG    02
ERL$VEC488                      00000128 RG    02
ERL$VEC492                      0000012C RG    02
ERL$VEC496                      00000130 RG    02
ERL$VEC500                      00000134 RG    02
ERL$VEC504                      00000138 RG    02
ERL$VEC508                      0000013C RG    02
ERL$VEC52                       00000034 RG    02
ERL$VEC56                       00000038 RG    02
ERL$VEC60                       0000003C RG    02
ERL$VEC64                       00000040 RG    02
ERL$VEC68                       00000044 RG    02
ERL$VEC72                       00000048 RG    02
ERL$VEC76                       0000004C RG    02
ERL$VEC8                        00000008 RG    02
ERL$VEC80                       00000050 RG    02
ERL$VEC84                       00000054 RG    02
ERL$VEC88                       00000058 RG    02
ERL$VEC92                       0000005C RG    02
ERL$VEC96                       00000060 RG    02
EXE$ACVIOLAT                  = 00000020 RG    02
EXE$BREAK                     = 0000002C RG    02
EXE$EXCEPTION                   ******** X     02
EXE$GL_RPB                      ******** X     02
EXE$GL_SCB                      ******** X     02
EXE$ROPRAND                   = 00000018 RG    02
EXE$TBIT                      = 00000028 RG    02
IPL$_ASTDEL                   = 00000002
MMG$GL_SBICONF                  ******** X     02
MMG$PAGEFAULT                 = 00000024 RG    02
MPS$ALLOCEMB                    ******** X     02
MPS$ASTDEL                      000001AC RG    02
MPS$COMPAT                      000001D0 RG    02
MPS$KERSTKNV                    000001C8 RG    02
MPS$MPSCHED2                    ******** X     02
MPS$RELEASEMB                   ******** X     02
MPS$SECBUGCHK                   ******** X     02
MPS$UNEXPINT                    00000180 RG    02
MPS$UNEXPINT1                   000001B6 R     02
NEXUS                         = 00000010
PR$_ASTLVL                    = 00000013
PR$_SCBB                      = 00000011
PSL$C_USER                    = 00000003
PSL$M_CURMOD                  = 03000000
PSL$M_IPL                     = 001F0000
PSL$S_CURMOD                  = 00000002
PSL$V_CURMOD                  = 00000018
PSL$V_IPL                     = 00000010
PSL$V_IS                      = 0000001A
PSL$V_PRVMOD                  = 00000016
RPB$L_SCBB                    = 000000B0

SCH$ASTDEL                      ********   X   02
SS$_COMPAT                    = 0000042C
VECTOR                        = 0000023C
VNUM                          = 00000100
```

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
$AEXNONPAGED	00000238 (568.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.09	00:00:01.16
Command processing	140	00:00:00.89	00:00:05.14
Pass 1	357	00:00:11.84	00:00:32.69
Symbol table sort	0	00:00:01.64	00:00:03.15
Pass 2	96	00:00:02.44	00:00:06.45
Symbol table output	22	00:00:00.19	00:00:00.72
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	649	00:00:17.12	00:00:49.34

The working set limit was 1500 pages.
65054 bytes (128 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1132 non-local and 5 local symbols.
425 source lines were read in Pass 1, producing 21 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MP.OBJ]MP.MLB;1	4
_255DUA28:[SYS.OBJ]LIB.MLB;1	8
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	18

1140 GETS were required to define 18 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:MPINTEXC/OBJ=OBJ$:MPINTEXC MSRC$:MPPREFIX/UPDATE=(ENH$:MPPREFIX)+MSRC$:MPINTEXC/UPDATE=(ENH$:MPINTEXC)+EXECML$/LIB+LI

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MPERRLOG
LIS

MPSCBVEC
LIS

MPINT
LIS

MPPFM
LIS

MPDAT
LIS

MPPWRFAIL
LIS

MPMCHECK
LIS

MPINTEXC
LIS

MPLOG
LIS

MPERRMSG
LIS

MPSCHED
LIS

MPSHWPFM
LIS

MPLOAD
LIS

MPINIT
LIS