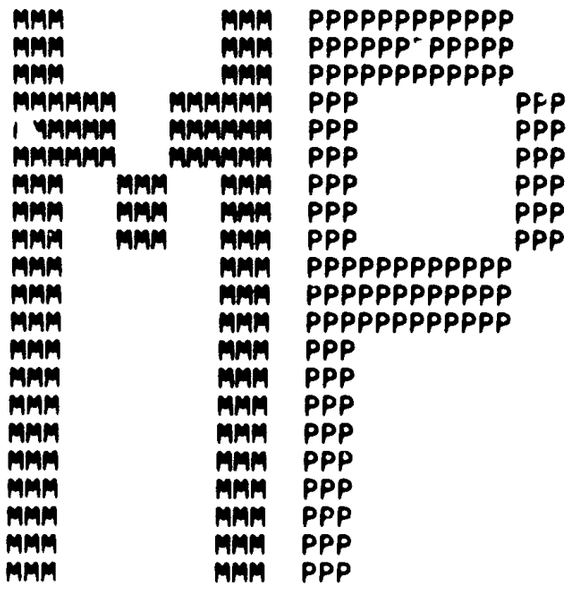
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPP PPPPP
MMM MMM PPPPPPPPPPPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMMMMM MMMMMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM MMM PPP PPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPPPPPPPPPPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM MMM PPP
MMM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

```
MM      MM  PPPPPPPP    TTTTTTTTTT    IIIIII    MM      MM  EEEEEEEEEE  RRRRRRRR
MM      MM  PPPPPPPP    TTTTTTTTTT    IIIIII    MM      MM  EEEEEEEEEE  RRRRRRRR
MMMM  MMMM  PP      PP      TT        II      MMMM  MMMM  EE          RR      RR
MMMM  MMMM  PP      PP      TT        II      MMMM  MMMM  EE          RR      RR
MM  MM  MM  PP      PP      TT        II      MM  MM  MM  EE          RR      RR
MM  MM  MM  PP      PP      TT        II      MM  MM  MM  EE          RR      RR
MM      MM  PPPPPPPP        TT        II      MM      MM  EEEEEEEE    RRRRRRRR
MM      MM  PPPPPPPP        TT        II      MM      MM  EEEEEEEE    RRRRRRRR
MM      MM  PP              TT        II      MM      MM  EE          RR  RR
MM      MM  PP              TT        II      MM      MM  EE          RR  RR
MM      MM  PP              TT        II      MM      MM  EE          RR    RR        ....
MM      MM  PP              TT        II      MM      MM  EE          RR    RR        ....
MM      MM  PP              TT      IIIIII    MM      MM  EEEEEEEEEE  RR      RR      ....
MM      MM  PP              TT      IIIIII    MM      MM  EEEEEEEEEE  RR      RR      ....

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  MPTIMER - MP TIMER INTERRUPT HANDLER
0000     2         .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ;
0000    31 ; Facility:  Executive , Hardware fault handling
0000    32 ;
0000    33 ; Abstract:  Timer interrupt handler for secondary processor
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 ;       V03-008 KDM0094         Kathleen D. Morse       22-Mar-1984
0000    42 ;               Do not reschedule a real-time process that goes through
0000    43 ;               quantum end on the secondary processor.
0000    44 ;
0000    45 ;       V03-007 ROW0151         Ralph O. Weber          29-DEC-1982
0000    46 ;               Fix SOFTINT in MPS$HWCLKINT to use IPL$_TIMERFORK and
0000    47 ;               MPS$SWTIMINT to begin by raising IPL to IPL$_TIMER.  This
0000    48 ;               change is required to match similar changes on the primary.
0000    49 ;               The net effect of these changes is to raise IPL$_SYNCH to 8.
0000    50 ;
0000    51 ;       V03-006 KDM0025         Kathleen D. Morse       10-Oct-1982
0000    52 ;               Change secondary specific routines to be MPS$xxx instead
```

```
0000     53 ;                  of EXE$xxx (MPS$SWTIMINT, MPS$HWCLKINT).
0000     54 ;
0000     55 ;        V03-005 KDM0010         Kathleen D. Morse       31-Aug-1982
0000     56 ;                Lower IPL to SYNCH before rescheduling for quantum end.
0000     57 ;
0000     58 ;        V03-004 KDM0009         Kathleen D. Morse       31-Aug-1982
0000     59 ;                Do an interlocked operation before testing MPS$GL_STATE.
0000     60 ;
0000     61 ;--
0000     62
0000     63 ;
0000     64 ; MACRO LIBRARY CALLS
0000     65 ;
0000     66
0000     67          $CADEF                          ; Define conditional assembly parameters
0000     68          $IPLDEF                         ; Define interrupt priority levels
0000     69          $LCKDEF                         ; Interlock bit definitions
0000     70          $MPSDEF                         ; Secondary processor state definitions
0000     71          $PCBDEF                         ; Define PCB offsets
0000     72          $PHDDEF                         ; Define PHD offsets
0000     73          $PRDEF                          ; Define processor registers
0000     74          $PRIDEF                         ; Define priority increments
0000     75          $PSLDEF                         ; Define processor status fields
```

```
                                0000     77          .SBTTL  HARDWARE CLOCK INTERRUPTS
                                0000     78 ;+
                                0000     79 ; EXE$HWCLKINT - HARDWARE CLOCK INTERRUPT
                                0000     80 ;
                                0000     81 ; This routine is automatically vectored to when the clock count register
                                0000     82 ; overflows.  The accounting integral of the current process in incremented,
                                0000     83 ; and the quantum of the current process is incremented.  If the process quantum
                                0000     84 ; transists to zero then a software interrupt is requested on the timer level.
                                0000     85 ; The clock interrupt is then dismissed.
                                0000     86 ;
                                0000     87 ; INPUTS:
                                0000     88 ;
                                0000     89 ;       (SP) - PC at time of interrupt
                                0000     90 ;     4(SP) - PSL at time of interrupt
                                0000     91 ;
                                0000     92 ; ENVIRONMENT:
                                0000     93 ;
                                0000     94 ;       Executes on secondary processor.
                                0000     95 ; The routine executes at IPL ^X18.
                                0000     96 ;
                                0000     97 ;-
                                0000     98
                           00000000     99          .PSECT  A$EXENONPAGED,LONG
                                0000    100 MPS$HWCLKINT::                          ; Hardware clock interrupt routine
                         7E     50  7D  0000    101          MOVQ    R0,-(SP)           ; Save register R0 and R1
                 18 800000C1  8F  DA  0003    102          MTPR    #^X800000C1,#PR$_ICCS ; Clr err & intrpt and re-enable
                                     000A    103
                           00000002  000A    104          .IF NE  CA$_MEASURE
                          50  0F AE  98  000A    105          CVTBL   15(SP),R0          ; Get upper byte of saved PSL
                                 69  19  000E    106          BLSS    70$                ; Br if CM bit set
                          50  F8 8F  8A  0010    107 10$:     BICB    #^XF8,R0           ; Convert extended byte to index
                           0000'CF40  D6  0014    108          INCL    W^MPS$AL_CPUTIME[R0] ; Increment statistics vector
                                     0019    109          .ENDC
                                     0019    110
                                     0019    111          .IF     DF,MPPFMSWT
                                     0019    112          BSBW    MPS$PFM_RUNTIME    ; Gather performance measurement data
                                     0019    113          .ENDC
                                     0019    114
                    00 0000'CF    00  E6  0019    115          BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,15$ ; Flush cache queue
                       01   0000'CF  D1  001F    116 15$:     CMPL    W^MPS$GL_STATE,#MPS$K_IDLESTATE ; Is a process actually running?
                                 3F  13  0024    117          BEQL    60$                ; Br on no, go accumulate null time
                          04   0000'CF  D1  0026    118          CMPL    W^MPS$GL_STATE,#MPS$K_EXECSTATE ; Is a process actually running?
                                 11  12  002B    119          BNEQ    20$                ; Br on no, don't accumulate time in PHD
                          50   0000'CF  D0  002D    120          MOVL    W^MPS$GL_CURPCB,R0 ; Get address of current process PCB
                            51   6C A0  D0  0032    121          MOVL    PCB$L_PHD(R0),R1   ; Get address of process header
                                 38 A1  D6  0036    122          INCL    PHD$L_CPUTIM(R1)   ; Increment accounting integral
                                 3C A1  B6  0039    123          INCW    PHD$W_QUANT(R1)    ; Increment time quantum
                                 04  18  003C    124          BGEQ    30$                ; If EQL quantum ran out
                               50   8E  7D  003E    125 20$:     MOVQ    (SP)+,R0           ; Restore R0 and R1
                                    02  0041    126          REI                        ;
                                     0042    127
                            10   0B A0  91  0042    128 30$:     CMPB    PCB$B_PRI(R0),#16  ; Is this a real-time process?
                                 05  19  0046    129          BLSS    40$                ; Br if yes, it is real-time
                                     0048    130          SOFTINT #IPL$_TIMERFORK    ; Request software timer fork int
                                 F1  11  004B    131          BRB     20$                ; Join common code
                       00 24 A0  03  E5  004D    132 40$:     BBCC    #PCB$V_INQUAN,PCB$L_STS(R0),50$ ; Clear initial quan flag
3C A1    00000000'GF  B0  0052    133 50$:     MOVW    G^SCH$GW_QUAN,PHD$W_QUANT(R1) ; Set new quantum
```

```
                0118 C0   00000000'GF   D0   005A   134          MOVL    G^EXE$GL_ABSTIM,PCB$L_WAITIME(R0) ; Record event time
                          D9   11   0063   135          BRB     20$                       ; Join common code
                                    0065   136
                     0000'CF   D6   0065   137 60$:     INCL    W^MPS$GL_NULLCPU          ; Accumulate null cpu time for secondary
                50   00000000'GF   9E   0069   138          MOVAB   G^SCH$GL_NULLPCB,R0       ; Get address of PCB for null process
                     50   6C A0   D0   0070   139          MOVL    PCB$L_PHD(R0),R0          ; Get address of process header
                          38 A0   D6   0074   140          INCL    PHD$L_CPUTIM(R0)          ; Increment accounting integral
                          C5   11   0077   141          BRB     20$                       ; Join common code
                                    0079   142
                          00000002  0079   143          .IF NE  CA$_MEASURE
                          50   05   9A   0079   144 70$:     MOVZBL  #5,R0                     ; Insert CM index
                               92   11   007C   145          BRB     10$                       ; Joint common code
                                         007E   146          .ENDC
```

```
                     148          .SBTTL  SOFTWARE TIMER INTERRUPTS
                     149 ;+
                     150 ; EXE$SWTIMINT - SOFTWARE TIMER INTERRUPTS
                     151 ;
                     152 ; This routine is automatically vectored to when a software interrupt
                     153 ; is requested on the timer level.  Timer interrupts are requested when
                     154 ; the current process has exceeded its cpu time quantum.
                     155 ;
                     156 ; A kernel mode PSL and a PC pointing to MPS$QEND are placed on the kernel
                     157 ; stack so that the primary will continue executing quantum end for this
                     158 ; process.
                     159 ;
                     160 ; INPUTS:
                     161 ;
                     162 ;       (SP) - PC at time of interrupt
                     163 ;      4(SP) - PSL at time of interrupt
                     164 ;
                     165 ; ENVIRONMENT:
                     166 ;
                     167 ;       Executes on secondary processor.
                     168 ;
                     169 ;       This routine is entered at IPL$_TIMERFORK.  It immediately raises
                     170 ;       to IPL$_TIMER (which equals IPL$_SYNCH) and completes execution at
                     171 ;       IPL$_SYNCH.
                     172 ;
                     173 ;-
                     174
                     175          .ENABL  LSB
                     176          .ALIGN  LONG
                     177          ASSUME  IPL$_SYNCH EQ IPL$_TIMER
                     178
                     179 MPS$SWTIMINT::                          ; Software timer interrupt routine
                     180          SETIPL  #IPL$_TIMER             ; Raise to normal IPL.
  07 AE    03    93  0083 181          BITB    #<<PSL$M_CURMOD> -      ; Check if this interrupt
                     0087 182                  @-<PSL$V_CURMOD>>, -    ;  occurred during
                     0087 183                  4+<PSL$V_CURMOD@-3>(SP) ;  kernel mode execution
        3C    13   0087 184          BEQL    20$                     ; Br if kernel mode, forget interrupt
        50    DD   0089 185          PUSHL   R0                      ; Save register
50 0000'CF    D0   008B 186          MOVL    W^MPS$GL_CURPCB,R0      ; A second check for quantum must be
  50   6C A0    D0   0090 187          MOVL    PCB$L_PHD(R0),R0        ;  done here in case the process found
                     0094 188          SETIPL  #IPL$_POWER             ; Prevent powerfail window here
    0000'CF    D5   0097 189          TSTL    W^MPS$GL_PFAILTIM       ; Check if powerfail occured
        05    12   009B 190          BNEQ    5$                      ; Br if it did, always give process back
     3C A0    B5   009D 191          TSTW    PHD$W_QUANT(R0)         ;  by the hardware clock routine was
        24    19   00A0 192          BLSS    30$                     ;  rescheduled before this interrupt.
    50   6E    D0   00A2 193 5$:      MOVL    (SP),R0                 ; Restore register
        6E    D4   00A5 194          CLRL    (SP)                    ; Make PSL look like kernel mode
                     00A7 195
                     00A7 196          .IF     DF,MPPFMSWT
                     00A7 197          BSBW    MPS$PFM_QEND            ; Gather performance measurement data
                     00A7 198          .ENDC
                     00A7 199
                     00A7 200          SETIPL  #IPL$_SYNCH             ; Lower to SYNCH before scheduling
      FF53'   30     00AA 201          BSBW    MPS$MPSCHED2            ; Go return process to primary
                     00AD 202                                          ;  and make primary start execution at
                     00AD 203                                          ;  MPS$QEND, which must follow this line
                     00AD 204
```

```
                        00AD    205
                        00AD    206
                        00AD    207 ;+
                        00AD    208 ; MPS$QEND - QUANTUM END PERFORMED BY PRIMARY ON BEHALF OF SECONDARY
                        00AD    209 ;
                        00AD    210 ; This routine is the place that the primary starts executing when the
                        00AD    211 ; secondary passes a process to it that has just finished its quantum.
                        00AD    212 ;
                        00AD    213 ; ENVIRONMENT:
                        00AD    214 ;
                        00AD    215 ;       Executed by primary processor.
                        00AD    216 ;
                        00AD    217 ;-
                        00AD    218
                        00AD    219 MPS$QEND:
                        00AD    220         SETIPL  #IPL$_SYNCH                 ; Synch scheduler with other reporting
                  3F    BB    00B0    221         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers R0 through R5
54  00000000'GF       D0    00B2    222         MOVL    G^SCH$GL_CURPCB,R4      ; Get current process PCB address
    55    6C  A4      D0    00B9    223         MOVL    PCB$L_PHD(R4),R5        ; Get address of process header
    00000000'GF       16    00BD    224         JSB     G^SCH$QEND              ; Perform normal quantum end
                  3F    BA    00C3    225 10$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers R0 thru R5
                  02    00C5    226 20$:    REI                             ; Return from interrupt
                        00C6    227
        50        8E    D0    00C6    228 30$:    MOVL    (SP)+,R0                ; Restore register
                  02    00C9    229         REI
                        00CA    230
                        00CA    231
                        00CA    232         .END
```

```
CA$_MEASURE                     = 00000002
EXE$GL_ABSTIM                     ********    X   02
IPL$_POWER                      = 0000001F
IPL$_SYNCH                      = 00000008
IPL$_TIMER                      = 00000008
IPL$_TIMERFORK                  = 00000007
LCK$V_INTERLOCK                 = 00000000
MPS$AC_CPUTIME                    ********    X   02
MPS$GL_CURPCB                     ********    X   02
MPS$GL_INTERLOCK                  ********    X   02
MPS$GL_NULLCPU                    ********    X   02
MPS$GL_PFAILTIM                   ********    X   02
MPS$GL_STATE                      ********    X   02
MPS$HWCLKINT                      00000000 RG     02
MPS$K_EXECSTATE                 = 00000004
MPS$K_IDLESTATE                 = 00000001
MPS$MPSCHED2                      ********    X   02
MPS$QEND                          000000AD R      02
MPS$SWTIMINT                      00000080 RG     02
PCB$B_PRI                       = 0000000B
PCB$L_PHD                       = 0000006C
PCB$L_STS                       = 00000024
PCB$L_WAITIME                   = 00000118
PCB$V_INQUAN                    = 00000003
PHD$L_CPUTIM                    = 00000038
PHD$W_QUANT                     = 0000003C
PR$_ICCS                        = 00000018
PR$_IPL                         = 00000012
PR$_SIRR                        = 00000014
PSL$M_CURMOD                    = 03000000
PSL$V_CURMOD                    = 00000018
SCH$GL_CURPCB                     ********    X   02
SCH$GL_NULLPCB                    ********    X   02
SCH$GW_QUAN                       ********    X   02
SCH$QEND                          ********    X   02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
A$EXENONPAGED	000000CA (202.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.09	00:00:00.38
Command processing	134	00:00:00.82	00:00:05.07
Pass 1	211	00:00:05.19	00:00:14.50
Symbol table sort	0	00:00:00.63	00:00:00.77
Pass 2	61	00:00:01.13	00:00:04.11

Symbol table output	7	00:00:00.06	00:00:00.14
Psect synopsis output	1	00:00:00.02	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	451	00:00:07.94	00:00:25.04

The working set limit was 1350 pages.
25383 bytes (50 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 429 non-local and 12 local symbols.
237 source lines were read in Pass 1, producing 13 object records in Pass 2.
20 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MP.OBJ]MP.MLB;1	3
_255DUA28:[SYS.OBJ]LIB.MLB;1	7
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	16

555 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:MPTIMER/OBJ=OBJ$:MPTIMER MSRC$:MPPREFIX/UPDATE=(ENH$:MPPREFIX)+MSRC$:MPTIMER/UPDATE=(ENH$:MPTIMER)+EXECML$/LIB+LIB$:M
```

MSCP

MSCP
MAP

ADDUNIT
LIS

MSCP
LIS

MPWAIT
LIS

MPTIMER
LIS

XDELTA
LIS

MSCPDEF
MAR