L 5

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
```

FILEID**MCHECKUV1

```
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  UU      UU  VV      VV      11
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  UU      UU  VV      VV      11
MMMM  MMMM  CC          HH      HH  EE          CC          KK      KK  UU      UU  VV      VV    1111
MMMM  MMMM  CC          HH      HH  EE          CC          KK      KK  UU      UU  VV      VV    1111
MM  MM  MM  CC          HH      HH  EE          CC          KK    KK    UU      UU  VV      VV      11
MM  MM  MM  CC          HH      HH  EE          CC          KK    KK    UU      UU  VV      VV      11
MM      MM  CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK      UU      UU  VV      VV      11
MM      MM  CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK      UU      UU  VV      VV      11
MM      MM  CC          HH      HH  EE          CC          KK    KK    UU      UU  VV      VV      11
MM      MM  CC          HH      HH  EE          CC          KK    KK    UU      UU  VV      VV      11
MM      MM  CC          HH      HH  EE          CC          KK      KK  UU      UU    VV  VV        11          ....
MM      MM  CC          HH      HH  EE          CC          KK      KK  UU      UU    VV  VV        11          ....
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  UUUUUUUUUU      VV        111111        ....
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  UUUUUUUUUU      VV        111111        ....

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            .TITLE  MCHECKUV1 -- Micro-VAX I Machine Check
0000            .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY: SYSLOAUV1 - loadable CPU-dependent code
0000    ;
0000    ; ABSTRACT:
0000    ;       This module contains routines to handle Micro-VAX I specific
0000    ;       machine check errors.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       IPL = 31 (1D for memory errors)         Mode = KERNEL
0000    ;
0000    ; AUTHOR:  Kathleen D. Morse,   CREATION DATE: 13-Sep-1983
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-003 KDM0100         Kathleen D. Morse       1-May-1984
0000    ;               Add missing indirects to memory CSR references.
0000    ;
0000    ;       V03-002 KDM0096         Kathleen D. Morse       27-Mar-1984
0000    ;               Fix some bugs in the error logging routine.
0000    ;               Correct documentation of machine check codes.
0000    ;               Change branches to REFLECTCHK to cause a bugcheck,
0000    ;               since the VA and PSL may not always be accurate on
0000    ;               a memory error.
0000    ;
0000    ;--
```

```
                   0000         .SBTTL  DECLARATIONS
                   0000   ;
                   0000   ; INCLUDE FILES:
                   0000   ;
                   0000
                   0000   ;
                   0000   ; INCLUDED SYSTEM SYMBOL DEFINITIONS
                   0000   ;
                   0000         $EMBDEF <MC,SE>
                   0000         $IPLDEF
                   0000         $PCBDEF
                   0000         $PFNDEF
                   0000         $PHDDEF
                   0000         $PRDEF
                   0000         $PRUV1DEF
                   0000         $PSLDEF
                   0000         $PTEDEF
                   0000         $RPBDEF
                   0000         $VADEF
                   0000         $MCDEF
                   0000         $MCHKDEF
                   0000
                   0000   ;
                   0000   ; OWN STORAGE:
                   0000   ;
                   0000
         FFFFFFFC  0000   MC$L_RECOVMSK = -4                      ; These symbols define offsets from AP
         FFFFFFF8  0000   MC$L_PCPSLPTR = -8                      ; to locations on the stack; see
                   0000                                           ; functional description of MCHECKUV1.
                   0000
               00000000   .PSECT  MCHK$DATA,QUAD,WRT
                   0000   EXE$MCHK_ERRCNT::                       ; Used to locate error counters
                   0000                                           ; via SYS.MAP.
                   0000
                   0000   EXE$AL_MEMCSRS::                        ; Array of memory CSR addresses.
         00000000  0000         .LONG   0                         ;  Count of memory CSRs.
         00000044  0004         .BLKL   16                        ;  1 longword per possible CSR.
                   0044
                   0044   LAST_BADMCK:                            ; Time of last bad machine check code.
         00000000  0044         .LONG   0
         00000001  0048   BADMCK_MIN = 1                          ; Allowable time between bad machine
                   0048                                           ; check codes.
                   0048
                   0048   LAST_BADINT:                            ; Time of last illegal interrupt.
         00000000  0048         .LONG   0
         00000001  004C   BADINT_MIN = 1                          ; Allowable time between bad interrupts.
                   004C
                   004C   LAST_RDS:                               ; Time of last RDS error.
         00000000  004C         .LONG   0
         00000001  0050   RDS_MIN = 1                             ; Allowable time between RDS errors.
                   0050
                   0050   ;
                   0050   ; This data is used by ECC$REENABLE, which is periodically called to scan
                   0050   ; the memory controller CSRs for CRD errors.
                   0050   ;
                   0050   ECC$GW_REENAB:                          ; Time since CRD interrupts
             0000  0050         .WORD   0                         ; were last enabled.
```

```
          0052
00000384  0052  REENABTIME = 60*15                      ; Reenable CRD interrupts every
          0052                                          ; 15 minutes.
          0052
          0052  ECC$GW_CRDWATCH:                        ; Time since last scanned mem
    0000  0052          .WORD   0                       ; CSR for CRD errors.
          0054
0000003C  0054  CRDWATCHTIME = 60                       ; Scan for non-interrupt CRD
          0054                                          ; errors every 60 seconds.
          0054
          0054  ;
          0054  ; The following data is used by the CRD interrupt handler, EXE$LOGCRD, and
          0054  ; the memory CSR logging routine, EXE$LOGMEM.
          0054  ;
          0054  ECC$AB_MEMERR:                          ; Count CRD errors logged recently.
      00  0054          .BYTE   0                       ; (within REENABTIME)
          0055
00000006  0055  CRDLOGMAX = 6                           ; Maximum number of CRD errors to log.
          0055
          0055  ECC$AB_CRDCNT:                          ; Count recent CRD interrupts (within
      00  0055          .BYTE   0                       ; REENABTIME).
          0056
00000003  0056  CRDINTMAX = 3                           ; Maximum number of CRD interrupts
          0056                                          ; before interrupts are disabled.
          0056
          0056  MMG$L_CRDCNT:                           ; Count total CRD interrupts.
00000000  0056          .LONG   0                       ;
          005A
```

```
00000000        .PSECT  WIONONPAGED,QUAD,RD,WRT
    0000        .SBTTL  * MACHINE CHECK ENTRY POINT *
    0000
    0000 ;++
    0000 ; MCHECKUV1 -- Micro-VAX I Machine Check
    0000 ;
    0000 ; FUNCTIONAL DESCRIPTION:
    0000 ;
    0000 ;       All machine checks are vectored to this entry point.  By casing
    0000 ;       off of the machine check type code in the machine check logout
    0000 ;       stack, determine the recovery action (if any) appropriate for
    0000 ;       each error.
    0000 ;
    0000 ; Format of Micro-VAX I machine check logout stack:
    0000 ;
    0000 ;       On entry to this machine check handler, the stack is set up as follows:
    0000 ;
    0000 ;               00(SP): # bytes in machine check log on stack (always 0C hex)
    0000 ;               04(SP): machine check type code
    0000 ;               08(SP): 1st machine check parameter
    0000 ;               0C(SP): 2nd machine check parameter
    0000 ;               10(SP): exception PC
    0000 ;               14(SP): exception PSL
    0000 ;
    0000 ;       The meanings of the third and fourth longwords depend on the
    0000 ;       machine check type code.
    0000 ;
    0000 ;       As soon as the machine check handler is invoked, it sets up the stack
    0000 ;       as follows:
    0000 ;       +--------------------------------+          :(SP)
    0000 ;       |                .               |
    0000 ;       |                .               |
    0000 ;       |       saved R0 - R5, AP        |
    0000 ;       |                .               |
    0000 ;       |                .               |
    0000 ;       +--------------------------------+
    0000 ;       | pointer to PC/PSL of exception |          :MC$L_PCPSLPTR(AP)
    0000 ;       +--------------------------------+
    0000 ;       |       recovery mask            |          :MC$L_RECOVMSK(AP)
    0000 ;       +--------------------------------+
    0000 ;       | # bytes pushed on logout stack | :(AP)    :MC$L_BYTCNT(AP)
    0000 ;       +--------------------------------+
    0000 ;       |  machine check type code       |          :MC$L_TYPECODE(AP)
    0000 ;       +--------------------------------+
    0000 ;       |       1st parameter            |          :MC$L_P1(AP)
    0000 ;       +--------------------------------+
    0000 ;       |       2nd parameter            |          :MC$L_P2(AP)
    0000 ;       +--------------------------------+
    0000 ;       |       exception PC             |          :MC$L_PC(AP)
    0000 ;       +--------------------------------+
    0000 ;       |       exception PSL            |          :MC$L_PSL(AP)
    0000 ;       +--------------------------------+
    0000 ;
    0000 ;
    0000 ;       AP will point to the beginning of the machine check log on the stack.
    0000 ;       2 longwords are immediately pushed on top of the machine check log, and
    0000 ;       are referenced as negative offsets from AP.  These two longwords are
    0000 ;       input arguments to EXE$MCHK_BUGCHK, a routine that is called to check
```

```
                        0000   ;       for a user-declared machine check recovery block.  This routine is
                        0000   ;       called immediately before bugchecking, and expects the mask and the
                        0000   ;       pointer to the exception PC/PSL to be right on top of the machine check
                        0000   ;       log on the stack.
                        0000   ;--
                        0000           .ALIGN  LONG
                        0000   EXE$INT58::                                 ; These interrupts are other-processor-
                        0000   EXE$INT5C::                                 ; specific and should never be
                        0000                                               ; seen on Micro-VAX I.
                        0000   EXE$UBAERR_INT::
                        0000   EXE$RH780_INT::
              28   11   0000           BRB     BAD_TYPE
                        0002
                        0002
                        0002           .ALIGN  LONG
                        0004   EXE$MCHK::                                  ; Machine check handler.
              02   DD   0004           PUSHL   #MCHK$M_MCK                 ; Mask signals machine check.
           14 AE   DF   0006           PUSHAL  MC$L_PC+4(SP)               ; Push pointer to exception PC/PSL.
         103F 8F   BB   0009           PUSHR   #^M<R0,R1,R2,R3,R4,R5,AP>   ; Working registers.
   5C      5E 24   C1   000D           ADDL3   #<9*4>,SP,AP                ; AP points to mchk log frame.
                        0011
                        0011           CASE    MC$L_TYPECODE(AP), -        ; Case on machine check code.
                        0011                   <MEMCTLR_ERR, -             ; code  1 - memory controller bugchk
                        0011                   MEM_ERROR, -                ;       2 - unrecoverable read error
                        0011                   NX_MEM, -                   ;       3 - nonexistent memory
                        0011                   UNALIGNED_IO, -             ;       4 - unaligned ref to I/O space
                        0011                   PTE_READCHK, -              ;       5 - page table read error
                        0011                   PTE_WRITECHK, -             ;       6 - page table write error
                        0011                   CS_PARITY, -                ;       7 - control store parity error
                        0011                   MICRO_ERRORS, -             ;       8 - micromachine bugcheck
                        0011                   BAD_VECTOR, -               ;       9 - Q22 bus vector read error
                        0011                   STACK_ERR>, -               ;      10 - error writing param onto stk
                        0011                   TYPE=B
                        002A
                        002A   BAD_TYPE:                                   ; undefined exception
           52 03   D0   002A           MOVL    #<MCHK$M_MCK! -             ; Type code for check for
                        002D                   MCHK$M_LOG>,R2              ; recovery block.
            00E4   30   002D           BSBW    LOG_MCHECK                  ; Log a machine check.
                        0030   ;
                        0030   ; Check to see if bad machine checks are occurring too rapidly.
                        0030   ;
   50    0044'CF   DE   0030           MOVAL   W^LAST_BADMCK,R0            ; Address of time stamp.
           51 01   D0   0035           MOVL    #BADMCK_MIN,R1              ; Min allowable time between errors.
            0073   30   0038           BSBW    CHK_AND_RESUME              ; Try to resume.  Returns if
                        003B                                               ; unresumable opcode.
                        003B           BUG_CHECK  BADMCKCOD,FATAL          ; Bad machine check code.
```

```
003F            .SBTTL  MICROCODE DETECTED ERRORS
003F       ;++
003F       ; The following machine checks are caused by microcode-detected
003F       ; inconsistencies in the hardware.
003F       ;--
003F
003F       ;++
003F       ; Memory controller bugcheck:
003F       ;
003F       ; Machine-check code:  1
003F       ;
003F       ; Description:  An invalid state was reached in the memory controller
003F       ;               and it was unable to successfully complete the last
003F       ;               function.
003F       ;
003F       ; Parameters:
003F       ;
003F       ;       MC$L_P1(AP): contents of the memory controller register "virtual".
003F       ;                    This register usually contains the physical address
003F       ;                    of the last function.
003F       ;       MC$L_P2(AP): address that was presented to the memory controller
003F       ;                    at the start of the function.
003F       ;--
003F       MEMCTLR_ERR:
003F
003F       ;++
003F       ; Micromachine bug check:
003F       ;
003F       ; Machine-check code:  8
003F       ;
003F       ; Description:  An invalid state has been reached in the micromachine.
003F       ;               This is a catastrophic error.
003F       ;
003F       ; Parameters:
003F       ;
003F       ;       MC$L_P1(AP): 0
003F       ;       MC$L_P2(AP): 0
003F       ;--
003F       MICRO_ERRORS:                               ; Micromachine invalid state error.
```

```
003F            .SBTTL  MEMORY REFERENCE ERRORS
003F    ;++
003F    ; The following machine checks are caused by memory reference errors
003F    ; of one sort or another.
003F    ;--
003F
003F    ;++
003F    ; Illegal operation (Unaligned or non-longword reference to I/O space):
003F    ;
003F    ; Machine-check code:  4
003F    ;
003F    ; Description:  An attempt was made to access an unaligned word or a
003F    ;               longword in I/O space.
003F    ;
003F    ; Parameters:
003F    ;
003F    ;       MC$L_P1(AP): Physical address of the illegal I/O reference. (**)
003F    ;       MC$L_P2(AP): Address presented to the memory controller at
003F    ;                    the start of the function.
003F    ;
003F    ;       ** Note:
003F    ;               Physical and virtual addresses returned on memory
003F    ;               controller errors may not be the actual address of
003F    ;               the error if a page crossing occurs.  If the page
003F    ;               offset (i.e., bits <8:0>) are:
003F    ;
003F    ;                       00000001 and the data length was word, or
003F    ;                       00000001 and the data length was long, or
003F    ;                       00000010 and the data langth was long, or
003F    ;                       00000011 and the data length was long
003F    ;
003F    ;               then the page in which the error occurred could be
003F    ;               the one addresses or the one logically preceding the
003F    ;               one specified.
003F    ;--
003F    UNALIGNED_IO:                                   ; Unaligned reference to I/O space, or
003F                                                    ; non-longword ref to I/O space.
003F
003F    ;++
003F    ; Unrecoverable page table read error:
003F    ;
003F    ; Machine-check code:  5
003F    ;
003F    ; Description:  An unrecoverable error occurred while attempting to
003F    ;               read a page table entry.  This error may have been a
003F    ;               parity, ECC, or timeout error.
003F    ;
003F    ; Parameters:
003F    ;
003F    ;       MC$L_P1(AP): physical address of page table entry.
003F    ;       MC$L_P2(AP): virtual address associated with the
003F    ;                    page table entry (i.e., the address that
003F    ;                    caused the page table entry to be read).
003F    ;--
003F    PTE_READCHK:                                    ; Hard memory error on PTE read.
003F
003F    ;++
```

```
003F    ; Unrecoverable page table write error:
003F    ;
003F    ; Machine-check code:  6
003F    ;
003F    ; Description:  This error occurred while attempting to write the
003F    ;               modify bit in a page table entry.  This error reflects hardware
003F    ;               that is in an unrunnable state and should be treated as a write
003F    ;               timeout error.
003F    ;
003F    ; Parameters:
003F    ;
003F    ;       MC$L_P1(AP): physical address of page table entry.
003F    ;       MC$L_P2(AP): virtual address associated with the
003F    ;                    page table entry (i.e., the address that
003F    ;                    caused the page table entry to be read).
003F    ;
003F    ;--
003F    PTE_WRITECHK:                           ; Hard memory error on PTE write.
003F
003F    ;++
003F    ; Control store parity error:
003F    ;
003F    ; Machine-check code:  7
003F    ;
003F    ; Description:  A control store parity error has occurred.  This is a
003F    ;               catastrophic error.
003F    ;
003F    ; Parameters:
003F    ;
003F    ;       MC$L_P1(AP): 0
003F    ;       MC$L_P2(AP): 0
003F    ;--
003F    CS_PARITY:                              ; Control store parity error.
003F
003F    ;++
003F    ; Q22 bus vector read error:
003F    ;
003F    ; Machine-check code:  9
003F    ;
003F    ; Description:  An error was encountered wyhile attempting to read an
003F    ;               interrupt vector address from the Q22 bus.)
003F    ;
003F    ; Parameters:
003F    ;
003F    ;       MC$L_P1(AP): Virtual address referenced
003F    ;       MC$L_P2(AP): Bad CSR value
003F    ;--
003F    BAD_VECTOR:                             ; Q22 bus vector read error.
003F
003F    ;++
003F    ; Write parameter error:
003F    ;
003F    ; Machine-check code:  10
003F    ;
003F    ; Description:  An error was encountered during an exception while
003F    ;               attempting to write the user, supervisor, or executive
003F    ;               stack after having verified that the write would succeed
```

```
               003F  ;                         (i.e., chmx and emulation).
               003F  ;
               003F  ; Parameters:
               003F  ;
               003F  ;         MC$L_P1(AP): virtual address that was being written. (**)
               003F  ;         MC$L_P2(AP): 0
               003F  ;
               003F  ;         ** Note:
               003F  ;                 Physical and virtual addresses returned on memory
               003F  ;                 controller errors may not be the actual address of
               003F  ;                 the error if a page crossing occurs.  If the page
               003F  ;                 offset (i.e., bits <8:0>) are:
               003F  ;
               003F  ;                         00000001 and the data length was word, or
               003F  ;                         00000001 and the data length was long, or
               003F  ;                         00000010 and the data langth was long, or
               003F  ;                         00000011 and the data length was long
               003F  ;
               003F  ;                 then the page in which the error occurred could be
               003F  ;                 the one addresses or the one logically preceding the
               003F  ;                 one specified.
               003F  ;--
               003F  STACK_ERR:                                      ; Error writing parameter onto stack.
               003F
               003F
52    03   D0  003F          MOVL    #<MCHK$M_MCK! -                 ; Type code for recovery block
               0042                  MCHK$M_LOG>,R2                  ; check.
   00CF    30  0042          BSBW    LOG_MCHECK                      ; Log the machine check.
   0087    31  0045          BRW     REFLECTCHK                      ; Reflect exception/bugcheck,
               0048                                                  ; based on current process mode.
```

```
                        0048          .SBTTL  NON-EXISTENT MEMORY
                        0048    ;++
                        0048    ; Machine Checks due to non-existent memory or non-existent I/O space addresses
                        0048    ; may have their own specific recovery block.  A recovery block may prevent
                        0048    ; logging of an NXM machine check, and/or it may prevent bugchecking because
                        0048    ; of an NXM machine check.
                        0048    ;
                        0048    ; Non-existent memory:
                        0048    ;
                        0048    ; Machine-check code:  3
                        0048    ;
                        0048    ; Description:  A bus timeout occurred during the last memory
                        0048    ;               controller read function.
                        0048    ;
                        0048    ; Parameters:
                        0048    ;
                        0048    ;       MC$L_P1(AP): physical address of the non-existent memory (**)
                        0048    ;       MC$L_P2(AP): address presented to memory controller at the start
                        0048    ;                    of the function
                        0048    ;
                        0048    ;       ** Note:
                        0048    ;               Physical and virtual addresses returned on memory
                        0048    ;               controller errors may not be the actual address of
                        0048    ;               the error if a page crossing occurs.  If the page
                        0048    ;               offset (i.e., bits <8:0>) are:
                        0048    ;
                        0048    ;                       00000001 and the data length was word, or
                        0048    ;                       00000001 and the data length was long, or
                        0048    ;                       00000010 and the data langth was long, or
                        0048    ;                       00000011 and the data length was long
                        0048    ;
                        0048    ;               then the page in which the error occurred could be
                        0048    ;               the one addresses or the one logically preceding the
                        0048    ;               one specified.
                        0048    ;--
                        0048    NX_MEM:                                         ; Reference to non-existent memory.
        52    07    D0  0048            MOVL    #<MCHK$M_LOG! -                 ; Type code for checking for
                        004B                    MCHK$M_MCK! -                   ; recovery block.
                        004B                    MCHK$M_NEXM>,R2                 ;
            00C6    30  004B            BSBW    LOG_MCHECK                      ; Log the machine check.
                        004E
     FC AC    04    C8  004E            BISL    #MCHK$M_NEXM, -                 ; Indicate NXM in recovery mask on the
                        0052                    MC$L_RECOVMSK(AP)               ; stack.
            007A    31  0052            BRW     REFLECTCHK                      ; Reflect exception/bugcheck.
                        0055
```

```
                              0055          .SBTTL  UNCORRECTABLE MEMORY ERRORS
                              0055
                              0055  ;++
                              0055  ;
                              0055  ;       UNCORRECTABLE MEMORY READ ERRORS
                              0055  ;
                              0055  ; Since this memory error could not be corrected by the hardware, the
                              0055  ; physical memory page is unusable.  Mark the page bad, and reflect
                              0055  ; exception/bugcheck.
                              0055  ;--
                              0055
                              0055  ;++
                              0055  ; Unrecoverable read error:
                              0055  ;
                              0055  ; Machine-check code:  2
                              0055  ;
                              0055  ; Description:  An unrecoverable read error occurred on the last memory
                              0055  ;               controller function.  The error may have been a parity
                              0055  ;               or an ECC error depending on the type of memory present.
                              0055  ;
                              0055  ; Parameters:
                              0055  ;
                              0055  ;       MC$L_P1(AP): physical address of the page containing the error (**)
                              0055  ;       MC$L_P2(AP): address presented to the memory controller at the
                              0055  ;                    start of the function
                              0055  ;
                              0055  ;       ** Note:
                              0055  ;               Physical and virtual addresses returned on memory
                              0055  ;               controller errors may not be the actual address of
                              0055  ;               the error if a page crossing occurs.  If the page
                              0055  ;               offset (i.e., bits <8:0>) are:
                              0055  ;
                              0055  ;                       00000001 and the data length was word, or
                              0055  ;                       00000001 and the data length was long, or
                              0055  ;                       00000010 and the data langth was long, or
                              0055  ;                       00000011 and the data length was long
                              0055  ;
                              0055  ;               then the page in which the error occurred could be
                              0055  ;               the one addresses or the one logically preceding the
                              0055  ;               one specified.
                              0055  ;--
                              0055
                              0055          .ENABL  LSB
                              0055  MEM_ERROR:                                      ; Uncorrectable memory read error.
                              0055  ;
                              0055  ; Mark page bad.
                              0055  ;
 50   08 AC   F7 8F      78   0055          ASHL    #-9,MC$L_P1(AP),R0      ; Get PFN of bad memory location.
  00000000'GF     50     D1   005B          CMPL    R0,G^MMG$GL_MAXPFN      ; PFN data base for this page?
                  0C     1A   0062          BGTRU   10$                     ; No, cannot mark page bad.
 51     00000000'GF      DE   0064          MOVAL   G^PFN$AB_TYPE, R1       ; Get address of PFN TYPE array.
      00 B140     20     88   006B          BISB2   #PFN$M_BADPAG, -        ; Mark page bad.
                              0070                  @(R1)[R0]               ;
                              0070  10$:
                              0070  ;
                              0070  ; Log a machine check and a memory error.
                              0070  ;
           52     03     D0   0070          MOVL    #<MCHK$M_MCK! -         ; Type code for recovery block
```

```
                    0073   533              MCHK$M_LOG>,R2      ; check.
        009E   30   0073   534     BSBW     LOG_MCHECK          ; Log the machine check.
  53      08   9A   0076   535     MOVZBL   #EMB$K_HE,R3        ; Error type code for logging.
        00D8   30   0079   536     BSBW     EXE$LOGMEM          ; Log a hard memory error.
        0084   31   007C   537     BRW      BUGCHECK            ; Bugcheck since VA/PSL may not
                    007F   538                                  ; be accurate enough to tell what
                    007F   539                                  ; mode the acess was taken in.
                    007F   540     .DSABL   LSB
```

```
                          007F          .SBTTL  ASYNCHRONOUS WRITE ERROR INTERRUPT
                          007F   ;++
                          007F   ;
                          007F   ; An interrupt is generated to this vector whenever a Qbus write operation
                          007F   ; does not complete successfully.  There could be any number of reasons for
                          007F   ; this.  The Qbus adapter could be broken or it might have been a write to
                          007F   ; some non-existant address.
                          007F   ;
                          007F   ; There is no machine-check frame on the stack upon entry at this point.
                          007F   ;
                          007F   ; The SCB offset ^X60 is connected to this entrypoint.
                          007F   ;--
                          007F
                          007F          .ALIGN  LONG
                          0080
                          0080   EXE$LOGAWE::
                          0080   EXE$INT60::
      5E     0C      C2   0080          SUBL    #<4*3>,SP                 ; Allocate space for dummy mchk frame
             0C      DD   0083          PUSHL   #^X0C                     ; Length of machine-check frame
             07      DD   0085          PUSHL   #MCHK$M_LOG!MCHK$M_MCK!MCHK$M_NEXM ;Mask for PRTCTEST
          14 AE      DF   0087          PUSHAL  MC$L_PC+4(SP)             ; and PC pointer
        103F 8F      BB   008A          PUSHR   #^M<R0,R1,R2,R3,R4,R5,AP> ; Save registers
   5C   5E   24      C1   008E          ADDL3   #<9*4>,SP,AP              ; Point AP to dummy machine-check frame
        53   07      3C   0092          MOVZWL  #EMB$K_AW,R3              ; Error type
           00BC      30   0095          BSBW    EXE$LOGMEM                ; log the error
    51    10 AC      DE   0098          MOVAL   MC$L_PC(AP),R1            ; Get address of PC/PSL.
      00000000'GF    16   009C          JSB     G^EXE$MCHK_TEST           ; Is recovery block in effect?
          06 50      E8   00A2          BLBS    R0,10$                    ; Br on yes, do not log this error
      00000000'GF    D6   00A5          INCL    G^EXE$GL_MCHKERRS         ; Bump the global machine-check counter
           0055      31   00AB   10$:   BRW     BUGCHECK                  ; Bugcheck since VA/PSL may not
                          00AE                                            ; be accurate enough to tell what
                          00AE                                            ; mode the acess was taken in.
```

```
                        00AE          .SBTTL  * EXITS FROM MACHINE CHECK ROUTINES *
                        00AE          .SBTTL  CHK_AND_RESUME
                        00AE  ;++
                        00AE  ; CHK_AND_RESUME
                        00AE  ;
                        00AE  ; FUNCTIONAL DESCRIPTION:
                        00AE  ;
                        00AE  ;       Called to check time that this error last occurred.
                        00AE  ;       CHK_AND_RESUME has three possible exit paths:
                        00AE  ;               (1) If errors are occurring too rapidly, BUGCHECK.
                        00AE  ;       Else fall through to RESUME:
                        00AE  ;               (2) If opcode is unresumable, RSB.
                        00AE  ;               (3) Else resume: clear stack and REI to retry the instruction.
                        00AE  ; INPUTS:
                        00AE  ;
                        00AE  ;       R0: Address of longword which contains time error last occurred.
                        00AE  ;       R1: Minimum time that must have elapsed since the last error.
                        00AE  ;
                        00AE  ; IMPLICIT INPUTS:
                        00AE  ;
                        00AE  ;        EXE$GQ_SYSTIME
                        00AE  ;
                        00AE  ; OUTPUTS:
                        00AE  ;
                        00AE  ;       The longword pointed to by R0 is updated with the time of this error.
                        00AE  ;--
                        00AE  CHK_AND_RESUME:
       53    50    D0   00AE          MOVL    R0,R3                   ; Save R0
    00000000'EF    16   00B1          JSB     EXE$READ_TODR           ; Get current time in R0,
       52    50    D0   00B7          MOVL    R0,R2                   ; and move it to R2.
       50    53    D0   00BA          MOVL    R3,R0                   ; Restore R0.
 53    52    60    C3   00BD          SUBL3   (R0),R2,R3              ; How long since last error?
       51    53    D1   00C1          CMPL    R3,R1                   ; Compare against minimum threshhold.
             05    1A   00C4          BGTRU   10$                     ; Br if enough time has elapsed.
             8E    D5   00C6          TSTL    (SP)+                   ; Else clear return address from stack.
           0038    31   00C8          BRW     BUGCHECK                ; Errors recurring too fast; bugcheck.
                        00CB  10$:
       60    52    D0   00CB          MOVL    R2,(R0)                 ; Save time of latest error.
                   05   00CE          RSB                             ; Since no opcodes are resumable (all
                        00CF                                          ; possible retrying is done in micro-
                        00CF                                          ; code) just return to caller.
```

```
                          00CF         .SBTTL  REFLCTCHK
                          00CF   ;++
                          00CF   ;      REFLECT MACHINE CHECK TO USER
                          00CF   ;
                          00CF   ; This code is entered if the machine check was fatal.  It determines
                          00CF   ; if it was just fatal to the process which caused it (current process
                          00CF   ; is in USER or SUPER mode), or if it was fatal to the entire system
                          00CF   ; (current process is in EXEC or KERNEL mode).
                          00CF   ;
                          00CF   ; If current process is in USER or SUPER mode,
                          00CF   ;      set up an exception on user's stack and REI to it
                          00CF   ; If current process is in EXEC or KERNEL mode,
                          00CF   ;      issue a fatal bugcheck.
                          00CF   ;
                          00CF   ; CALLING SEQUENCE:
                          00CF   ;
                          00CF   ;      BRB/W -- NOTHING EXTRA CAN BE ON THE STACK!!
                          00CF   ;
                          00CF   ; STACK CONTENTS:
                          00CF   ;
                          00CF   ;      00(SP): saved R0,R1,R2,R3,R4,R5,AP
                          00CF   ;      1C(SP): 2 longword inputs for recovery block check
                          00CF   ;      24(SP): (also AP) machine check log -- 1st longword is a byte count.
                          00CF   ;--
                          00CF   REFLECTCHK:                                  ; Reflect exception according
                          00CF                                                ; to current access mode.
        03 14 AC    19  E0    00CF         BBS     #PSL$V_CURMOD+1, -         ; Branch if USER or SUPER.
                          00D4                 MC$L_PSL(AP),10$           ;
              002C  31    00D4         BRW     BUGCHECK                   ; EXEC or KERNEL; bugcheck.
                          00D7
                          00D7   10$:                                         ; SUPER or USER; exception.
          50    00  DB    00D7         MFPR    #PR$_KSP,R0                ; Get kernel stack pointer.
        70   10 AC  7D    00DA         MOVQ    MC$L_PC(AP),-(R0)          ; Push PC,PSL on kernel stack.
          00    50  DA    00DE         MTPR    R0,#PR$_KSP                ; Replace new kernel stack ptr.
        103F    8F  BA    00E1         POPR    #^M<R0,R1,R2,R3,R4,R5,AP>  ; Restore registers.
          5E    08  C0    00E5         ADDL    #<2*4>,SP                  ; Pop inputs for recovery block check.
          5E    8E  C0    00E8         ADDL    (SP)+,SP                   ; Pop mck log from stack.
                          00EB   ;
                          00EB   ; Set up an exception stack for current process.
                          00EB   ; The faulting PC,PSL pair are still on the interrupt stack.  Alter
                          00EB   ; them to look as if an exception has occurred.
                          00EB   ;
     6E   00000000'GF  9E    00EB         MOVAB   G^EXE$MCHECK,(SP)          ; Replace exception PC.
04 AE   04 AE   02  18  EF    00F2         EXTZV   #PSL$V_CURMOD, -           ; Zero exception PSL, except
                          00F9                 #PSL$S_CURMOD, -           ; for current access mode.
                          00F9                 4(SP),4(SP)
   04 AE   04 AE    16  9C    00F9         ROTL    #PSL$V_PRVMOD, -           ; Create a PSL of current mode
                          00FF                 4(SP),4(SP)                ; kernel, correct previous
                          00FF                                            ; mode, and IPL 0.
          26    0F  DA    00FF         MTPR    #^XF,#PRUV1$_MCESR         ; Clear "mcheck in progress" flag.
                    02    0102         REI                                ; Go to exception handler.
                          0103
```

```
                        0103            .SBTTL  BUGCHECK
                        0103    ;++
                        0103    ; If user has declared a recovery block, transfer control to it.
                        0103    ; Else issue a fatal bugcheck.
                        0103    ;
                        0103    ; CALLING SEQUENCE:
                        0103    ;
                        0103    ;       BRB/W -- NOTHING EXTRA CAN BE ON THE STACK!!!
                        0103    ;
                        0103    ; STACK CONTENTS ON ENTRY:
                        0103    ;
                        0103    ;       00(SP): saved R0,R1,R2,R3,R4,R5,AP
                        0103    ;       1C(SP): 2 longword inputs for recovery block check
                        0103    ;       24(SP): (also AP) machine check log
                        0103    ;--
                        0103    BUGCHECK:
        103F 8F     BA  0103            POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ; Restore registers.
                        0107    ;
                        0107    ; A fatal bugcheck is now inevitable unless a user has declared a machine
                        0107    ; check recovery block.
                        0107    ;
                        0107
          26   0F   DA  0107            MTPR    #^XF,#PRUV1$_MCESR      ; Clear "mcheck in progress" flag.
   00000000'GF      16  010A            JSB     G^EXE$MCHK_BUGCHK       ; If return, no recovery block.
                        0110            BUG_CHECK -                     ; Issue fatal bugcheck.
                        0110                    MACHINECHK,FATAL
```

```
                       0114          .SBTTL  * LOGGING ROUTINES FOR MACHINE CHECKS *
                       0114    ;++
                       0114    ; LOG_MCHECK -- format inputs to LOGGER
                       0114    ;
                       0114    ; INPUTS:
                       0114    ;
                       0114    ;       R2: a mask which specifies the type of error (hence, the type
                       0114    ;               of recovery block to check for)
                       0114    ;
                       0114    ; IMPLICIT INPUTS:
                       0114    ;
                       0114    ;       (AP): points to machine check log on stack
                       0114    ;
                       0114    ; OUTPUTS:
                       0114    ;
                       0114    ;       Error is formatted and logged in system error log.
                       0114    ;--
                       0114
                       0114    LOG_MCHECK:
                       0114    ;
                       0114    ; Test if a machine check recovery block that specifies no error
                       0114    ; logging is in effect.
                       0114    ;
     51    10 AC   DE  0114          MOVAL   MC$L_PC(AP),R1            ; R1 points to PC,PSL of abort.
      00000000'GF  16  0118          JSB     G^EXE$MCHK_TEST           ; Logging inhibited?
            01 50  E9  011E          BLBC    R0,10$                    ; Branch if no.
                   05  0121          RSB                               ; Else return.
                       0122    10$:                                    ; Set up inputs to LOGGER.
      00000000'GF  D6  0122          INCL    G^EXE$GL_MCHKERRS         ; Bump machine check error count
            53 02  D0  0128          MOVL    #EMB$K_MC,R3              ; Use Machine Check type code.
  54     08 6C     C1  012B          ADDL3   MC$L_BYTCNT(AP), -        ; Size of data to log: machine check
                       012F                  #<2*4>,R4                 ; stack + PC,PSL.
     55    04 AC   DE  012F          MOVAL   MC$L_TYPECODE(AP),R5      ; Address of data to log.
                       0133    ;++
                       0133    ; LOGGER - release error data to error logger
                       0133    ;
                       0133    ; INPUTS:
                       0133    ;
                       0133    ;       R3: error type
                       0133    ;       R4: number of bytes to log
                       0133    ;       R5: address of information to be logged
                       0133    ;
                       0133    ; OUTPUTS:
                       0133    ;
                       0133    ;       Error log is inserted into error log buffer.
                       0133    ;       If no error log buffer, return with error status in R0.
                       0133    ;       R0-R5 destroyed.
                       0133    ;--
                       0133
                       0133    LOGGER:
     51    54 10   C1  0133          ADDL3   #EMB$B_MC_SUMCOD,R4,R1    ; Add space for log header.
      00000000'GF  16  0137          JSB     G^ERL$ALLOCEMB            ; Get error logging buffer.
            13 50  E9  013D          BLBC    R0, 20$                   ; Br if failed to get buffer.
               52  DD  0140          PUSHL   R2                        ; Save buffer addr on stack.
      04 A2    53  B0  0142          MOVW    R3,EMB$W_MC_ENTRY(R2)     ; Set entry type.
 10 A2    65   54  28  0146          MOVC3   R4,(R5),EMB$B_MC_SUMCOD(R2) ; Transfer info to log.
               04  BA  014B          POPR    #^M<R2>                   ; Retreive buffer address.
```

```
00000000'GF  16  014D   755         JSB     G^ERL$RELEASEMB         ; Give buffer to logger.
                 0153   756 20$:
             05  0153   757         RSB
```

```
0154  ;++
0154  ; EXE$LOGMEM -- log memory Control and Status Registers
0154  ;
0154  ; FUNCTIONAL DESCRIPTION:
0154  ;
0154  ;       EXE$LOGMEM is called to log memory CSRs.  If called with R3 =
0154  ;       EMB$K_SE (log a soft memory error), look at the memory CSRs to see
0154  ;       if the CRD (soft error) bit was set; if not, don't log the CSRs.
0154  ;       If "too many" CRD errors have been logged recently, also don't log
0154  ;       the CSRs.
0154  ;
0154  ;       The format of the MSV-11P CSR is as follows:
0154  ;
0154  ;               15 14    11            5     2   0
0154  ;               +--------------------------------+
0154  ;               !P!X! ! ! error address ! ! !W! !N!
0154  ;               +--------------------------------+
0154  ;
0154  ;           P (bit 15) - parity error, set when parity error occurs.  This
0154  ;                        bit turns on a red LED on the module.  Set to 0
0154  ;                        by power up or BUS INIT and remains set unless
0154  ;                        rewritten or initialized.
0154  ;
0154  ;           X (bit 14) - extended CSR read enable, used on 22-bit address
0154  ;                        machines.  This bit is set by a program to request
0154  ;                        error address bits 18-21 be places in CSR bits 5-8.
0154  ;
0154  ;             (bits 13-12) - not used
0154  ;
0154  ;           error address (bits 11-5) - address bits 11-17.  To get address
0154  ;                        bits 18-21, bit 14 must be set by a program and the
0154  ;                        CSR re-read.
0154  ;
0154  ;             (bits 4-3) - not used
0154  ;
0154  ;           W (bit 2) - write wrong parity.  If this bit is set and a DATO
0154  ;                       or DATOB cycle to memory occurs, wrong parity data
0154  ;                       is written into the parity MOS RAMs.  This bit can
0154  ;                       be used to check the parity error logic as well as
0154  ;                       failed address information in the CSR.
0154  ;
0154  ;             (bit 1) - not used
0154  ;
0154  ;           N (bit 0) - parity error enable.  If set and parity error occurs,
0154  ;                       then BDAL 16L and BDAL 17L are asserted on the bus
0154  ;                       simultaneously with the data (error interrupt is
0154  ;                       generated).  This is a read/write bit reset to zero on
0154  ;                       power up or BUS INIT.
0154  ;
0154  ; INPUTS:
0154  ;
0154  ;       R3: errorlog type code
0154  ;
0154  ; IMPLICIT INPUTS:
0154  ;
0154  ;       EXE$AL_MEMCSRS - array of memory CSR addresses
0154  ;
```

```
                                    0154  ; OUTPUTS:
                                    0154  ;
                                    0154  ;       Create entry in errorlog buffer containing the three memory controller
                                    0154  ;       Control and Status Registers.
                                    0154  ;       CRD error logging may be disabled.
                                    0154  ;       R0: low bit signals success/failure
                                    0154  ;       R1-R5 destroyed.
                                    0154  ;--
                                    0154  EXE$LOGMEM::
                                    0154
                                    0154  ;
                                    0154  ; Now push all possible memory CSRs onto the stack.  The architectural
                                    0154  ; limit is 16 memory controllers.  Since the CSR address is not logged
                                    0154  ; with each CSR, leave zeroed cells for CSRs that had no errors.
                                    0154  ; This is to prevent some problem with trying to read a memory address
                                    0154  ; from a register that cannot provide one since it had no error.
                                    0154  ;
            5E  00000040 8F    C2   0154          SUBL    #<4*16>,SP                   ; Allocate max size err log buffer.
6E    0040 8F    00    6E  00    2C   015B          MOVC5   #0,(SP),#0,#<4*16>,(SP)      ; Zero out the entire err log buffer.
            51  00000000'GF    DE   0163          MOVAL   G^EXE$AL_MEMCSRS,R1          ; Get address of memory CSR array.
                    55   81    D0   016A          MOVL    (R1)+,R5                     ; Get count of memory CSRs.
                                    016D          DSBINT  #31,R0                       ; Block out all interrupts.
                    52   5E    D0   0173          MOVL    SP,R2                        ; Get address of buffer for CSRs.
                         54    D4   0176          CLRL    R4                           ; Initialize parity error bit counter.
                                    0178
         00 B1     8000 8F    B3   0178  10$:    BITW    #^X8000,@(R1)                ; Is parity error bit set?
                         17    13   017E          BEQL    20$                          ; Br if not set, no error.
               62     00 B1    B0   0180          MOVW    @(R1),(R2)                   ; Store memory CSR into buffer.
         00 B1     4000 8F    A8   0184          BISW    #^X4000,@(R1)                ; Set bit requesting other err adr bits.
                         54    D6   018A          INCL    R4                           ; Count number of error bits set.
            02 A2     00 B1    B0   018C          MOVW    @(R1),2(R2)                  ; Store memory CSR into buffer.
         00 B1     8000 8F    AA   0191          BICW    #^X8000,@(R1)                ; Clear parity error bit.
                    51   04    C0   0197  20$:    ADDL    #4,R1                        ; Get address of next memory CSR.
                    52   04    C0   019A          ADDL    #4,R2                        ; Point to next buffer cell.
                 D8 55    F5   019D          SOBGTR  R5,10$                       ; Loop through all CSRs.
                                    01A0
                                    01A0          ENBINT  R0                           ; Restore IPL.
                         54    DD   01A3          PUSHL   R4                           ; Push count of CSRs with errors.
                         22    13   01A5          BEQL    NOLOG                        ; Br if no parity errors found.
                00000000'GF    D6   01A7          INCL    G^EXE$GL_MEMERRS             ; Bump memory error counter
                                    01AD  ;
                                    01AD  ; A CRD error occurred.  Count it, and if we haven't logged a lot of CRD errors
                                    01AD  ; recently, log it.
                                    01AD  ;
                    06   53    91   01AD          CMPB    R3,#EMB$K_SE                 ; Looking for CRD errors?
                         0B    12   01B0          BNEQ    LOG_CSRS                     ; No.  Unconditionally log CSRs.
                    0054'CF    96   01B2          INCB    W^ECC$AB_MEMERR              ; Count # of CRD errors LOGGED recently.
               06   0054'CF    91   01B6          CMPB    W^ECC$AB_MEMERR, -           ; Already logged enough CRD errors
                                    01BB                  #CRDLOGMAX                   ; recently?
                         0C    1A   01BB          BGTRU   NOLOG                        ; Yes.  Skip the logging.
                                    01BD
                                    01BD  LOG_CSRS:
                         54    D6   01BD          INCL    R4                           ; Add in one for CSR count.
               54   54   02    78   01BF          ASHL    #2,R4,R4                     ; Size of errorlog buffer.
                    55   5E    D0   01C3          MOVL    SP,R5                        ; Point to error log buffer.
                       FF6A    30   01C6          BSBW    LOGGER                       ; Log memory CSRs.
                                    01C9  NOLOG:
```

```
        5E   00000044 8F   C0  01C9    873          ADDL    #<4*17>,SP                   ; Pop error log buffer off stack.
                           05  01D0    874          RSB
```

```
                           01D1          .SBTTL  ECC$REENABLE -- TIMER CALL FROM SYSTEM CLOCK ROUTINE
                           01D1     ;++
                           01D1     ; ECC$REENABLE -- TIMER CALL FROM SYSTEM CLOCK ROUTINE
                           01D1     ;
                           01D1     ; FUNCTIONAL DESCRIPTION:
                           01D1     ;
                           01D1     ;         This routine periodically scans memory controller CSRs for
                           01D1     ;         CRD errors.  CRD errors are normally reported by interrupt,
                           01D1     ;         but even when CRD interrupts are turned off this routine will
                           01D1     ;         still scan memory controller CSRs periodically, to report a
                           01D1     ;         representative sample of CRD errors.
                           01D1     ;
                           01D1     ;         Also, check if it is time to reenable CRD interrupts.
                           01D1     ;         CRD interrupts may have been disabled by the CRD interrupt handler,
                           01D1     ;         EXE$LOGCRD, if it determines that "too many" interrupts are being
                           01D1     ;         received.
                           01D1     ;
                           01D1     ; INPUTS:
                           01D1     ;
                           01D1     ;         NONE
                           01D1     ;
                           01D1     ; IMPLICIT OUTPUTS:
                           01D1     ;
                           01D1     ;         If a CRD error is found the memory controller CSRs will be logged.
                           01D1     ;         CRD (Corrected Read Data) interrupts may be enabled for all
                           01D1     ;         memory controllers.
                           01D1     ;--
                           01D1
                           01D1     ECC$REENABLE::
          0052'CF    B7    01D1             DECW    W^ECC$GW_CRDWATCH          ; Time to scan for CRD errors?
               0F    14    01D5             BGTR    REENAB_SCAN                ; Branch if no.
               3F    BB    01D7             PUSHR   #^M<R0,R1,R2,R3,R4,R5>     ; Save working registers.
     0052'CF   3C    B0    01D9             MOVW    #CRDWATCHTIME, -           ; Reset scan timer.
                           01DE                     W^ECC$GW_CRDWATCH          ;
          53   06    3C    01DE             MOVZWL  #EMB$K_SE,R3               ; Test for CRD error,
             FF70    30    01E1             BSBW    EXE$LOGMEM                 ; and log memory CSRs if found.
               3F    BA    01E4             POPR    #^M<R0,R1,R2,R3,R4,R5>     ; Restore registers.
                           01E6     ;
                           01E6     ; If any CRD errors were found, the memory controller CSRs were logged.
                           01E6     ; Now check to see if its time to enable CRD interrupts.  CRD interrupts are
                           01E6     ; enabled periodically, whether or not they were disabled by EXE$LOGCRD.
                           01E6     ;
                           01E6     REENAB_SCAN:
          0050'CF    B7    01E6             DECW    W^ECC$GW_REENAB            ; Has reenable time elapsed?
               31    14    01EA             BGTR    20$                        ; Branch if no.
0050'CF      0384 8F    B0 01EC             MOVW    #REENABTIME, -             ; Yes.  Reset reenable timer.
                           01F3                     W^ECC$GW_REENAB            ;
          0054'CF    94    01F3             CLRB    W^ECC$AB_MEMERR            ; Reset CRD log counter.
          0055'CF    94    01F7             CLRB    W^ECC$AB_CRDCNT            ; Reset CRD interrupt counter.
   00000000'GF  00'  E1    01FB             BBC     S^#EXE$V_CRDENABL, -       ; Br if SYSGEN parameter does
               1A          0202                     G^EXE$GL_FLAGS,20$         ; not specify CRD interrupts.
                           0203
          7E   50    7D    0203             MOVQ    R0,-(SP)                   ; Save working registers.
51   00000000'GF     DE    0206             MOVAL   G^EXE$AL_MEMCSRS,R1        ; Get address of memory CSR array.
          50   81    D0    020D             MOVL    (R1)+,R0                   ; Get count of memory CSRs.
     00 B1     01    A8    0210     10$:    BISW    #1,@(R1)                   ; Reenable CRD interrupts.
          51   04    C0    0214             ADDL    #4,R1                      ; Get VA of next memory controller CSR.
```

```
         F6 50   F5  0217   933          SOBGTR  R0,10$            ; Loop through all CSRs.
      50    8E   7D  021A   934          MOVQ    (SP)+,R0          ; Restore working registers.
                     021D   935
                 05  021D   936 20$:     RSB                       ; Return.
```

```
                                021E          .SBTTL  EXE$LOGCRD -- CORRECTED MEMORY DATA INTERRUPTS
                                021E  ;++
                                021E  ; This routine is called when a CRD -- Corrected Read Data -- interrupt
                                021E  ; is received from a memory controller.  Log all interrupts, and
                                021E  ; continue.  If too many CRD interrupts are logged, turn off CRD interrupts.
                                021E  ;--
                                021E          .ALIGN  LONG
                                0220  EXE$LOGCRD::
                                0220  EXE$INT54::
                       3F   BB  0220          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save working registers.
                  53   06   3C  0222          MOVZWL  #EMB$K_SE,R3            ; Soft memory error.
                     FF2C   30  0225          BSBW    EXE$LOGMEM              ; Log a memory error.
                  0056'CF   D6  0228          INCL    W^MMG$L_CRDCNT          ; Count total CRD interrupts.
                  0055'CF   96  022C          INCB    W^ECC$AB_CRDCNT         ; Count recent CRD interrupts.
            03    0055'CF   91  0230          CMPB    W^ECC$AB_CRDCNT, -      ; More than enough CRD interrupts
                                0235                  #CRDINTMAX              ; lately?
                       14   1B  0235          BLEQU   20$                     ; No, do not disable CRD interrupts.
                                0237
      51    00000000'GF     DE  0237          MOVAL   G^EXE$AL_MEMCSRS,R1     ; Get address of memory CSR array.
                  50   81   D0  023E          MOVL    (R1)+,R0                ; Get count of memory CSRs.
               00 B1   01   AA  0241  10$:    BICW    #1,@(R1)                ; Disable CRD interrupts.
                  51   04   C0  0245          ADDL    #4,R1                   ; Get VA of next memory controller CSR.
                    F6 50   F5  0248          SOBGTR  R0,10$                  ; Loop through all CSRs.
                                024B
                                024B  20$:
                       3F   BA  024B          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                            02  024D          REI                             ; Return from interrupt.
                                024E
                                024E          .END
```

```
BADINT_MIN                       = 00000001
BADMCK_MIN                       = 00000001
BAD_TYPE                           0000002A R       03
BAD_VECTOR                         0000003F R       03
BUG$_BADMCKCOD                     ********    X    03
BUG$_MACHINECHK                    ********    X    03
BUGCHECK                           00000103 R       03
CHK_AND_RESUME                     000000AE R       03
CRDINTMAX                        = 00000003
CRDLOGMAX                        = 00000006
CRDWATCHTIME                     = 0000003C
CS_PARITY                          0000003F R       03
ECC$AB_CRDCNT                      00000055 R       02
ECC$AB_MEMERR                      00000054 R       02
ECC$GW_CRDWATCH                    00000052 R       02
ECC$GW_REENAB                      00000050 R       02
ECC$REENABLE                       000001D1 RG      03
EMB$B_MC_SUMCOD                  = 00000010
EMB$K_AW                         = 00000007
EMB$K_HE                         = 00000008
EMB$K_MC                         = 00000002
EMB$K_SE                         = 00000006
EMB$W_MC_ENTRY                   = 00000004
ERL$ALLOCEMB                       ********    X    03
ERL$RELEASEMB                      ********    X    03
EXE$AL_MEMCSRS                     00000000 RG      02
EXE$GL_FLAGS                       ********    X    03
EXE$GL_MCHKERRS                    ********    X    03
EXE$GL_MEMERRS                     ********    X    03
EXE$INT54                          00000220 RG      03
EXE$INT58                          00000000 RG      03
EXE$INT5C                          00000000 RG      03
EXE$INT60                          00000080 RG      03
EXE$LOGAWE                         00000080 RG      03
EXE$LOGCRD                         00000220 RG      03
EXE$LOGMEM                         00000154 RG      03
EXE$MCHECK                         ********    X    03
EXE$MCHK                           00000004 RG      03
EXE$MCHK_BUGCHK                    ********    X    03
EXE$MCHK_ERRCNT                    00000000 RG      02
EXE$MCHK_TEST                      ********    X    03
EXE$READ_TODR                      ********    X    03
EXE$RH780_INT                      00000000 RG      03
EXE$UBAERR_INT                     00000000 RG      03
EXE$V_CRDENABL                     ********    X    03
LAST_BADINT                        00000048 R       02
LAST_BADMCK                        00000044 R       02
LAST_RDS                           0000004C R       02
LOGGER                             00000133 R       03
LOG_CSRS                           000001BD R       03
LOG_MCHECK                         00000114 R       03
MC$L_BYTCNT                        00000000
MC$L_P1                            00000008
MC$L_P2                            0000000C
MC$L_PC                            00000010
MC$L_PCPSLPTR                    = FFFFFFF8
MC$L_PSL                           00000014
MC$L_RECOVMSK                    = FFFFFFFC
MC$L_TYPECODE                      00000004
MCHK$M_LOG                       = 00000001
MCHK$M_MCK                       = 00000002
MCHK$M_NEXM                      = 00000004
MEMCTLR_ERR                        0000003F R       03
MEM_ERROR                          00000055 R       03
MICRO_ERRORS                       0000003F R       03
MMG$GL_MAXPFN                      ********    X    03
MMG$L_CRDCNT                       00000056 R       02
NOLOG                              000001C9 R       03
NX_MEM                             00000048 R       03
PFN$AB_TYPE                        ********    X    03
PFN$M_BADPAG                     = 00000020
PR$_IPL                          = 00000012
PR$_KSP                          = 00000000
PRUV1$_MCESR                     = 00000026
PSL$S_CURMOD                     = 00000002
PSL$V_CURMOD                     = 00000018
PSL$V_PRVMOD                     = 00000016
PTE_READCHK                        0000003F R       03
PTE_WRITECHK                       0000003F R       03
RDS_MIN                          = 00000001
REENABTIME                       = 00000384
REENAB_SCAN                        000001E6 R       03
REFLECTCHK                         000000CF R       03
STACK_ERR                          0000003F R       03
UNALIGNED_IO                       0000003F R       03
```

MOU
V04

```
                                   +----------------+
                                   ! Psect synopsis !
                                   +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000018  (   24.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
MCHK$DATA                       0000005A  (   90.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC QUAD
WIONONPAGED                     0000024E  (  590.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC QUAD

                                +------------------------+
                                ! Performance indicators !
                                +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    38    00:00:00.03    00:00:01.46
Command processing               141    00:00:00.44    00:00:02.84
Pass 1                           284    00:00:05.94    00:00:18.16
Symbol table sort                  0    00:00:00.81    00:00:03.85
Pass 2                           175    00:00:01.82    00:00:08.14
Symbol table output               11    00:00:00.06    00:00:00.19
Psect synopsis output              3    00:00:00.02    00:00:00.02
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals             654    00:00:09.12    00:00:34.66

The working set limit was 1650 pages.
50128 bytes (98 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 785 non-local and 14 local symbols.
966 source lines were read in Pass 1, producing 18 object records in Pass 2.
28 pages of virtual memory were used to define 27 macros.

                               +----------------------------+
                               ! Macro library statistics !
                               +----------------------------+

Macro library name                  Macros defined
------------------                  --------------
_$255$DUA28:[SYSLOA.OBJ]MC.MLB;1                 1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                  17
_$255$DUA28:[SYSLIB]STARLET.MLB;2                6
TOTALS (all libraries)                          24

884 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:MCHECKUV1/OBJ=OBJ$:MCHECKUV1 MSRC$:MCHECKUV1/UPDATE=(ENH$:MCHECKUV1)+EXECML$/LIB+LIB$:MC/LIB
```

0397 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LIOSUB750
LIS

MCHECKUV1
LIS

MCHECK730
LIS

MCHECK750
LIS

LIOSUB730
LIS

LIOSUB790
LIS

MCHECK780
LIS

MCF790
LIS

MCHECK790
LIS

LIOSUB780
LIS

LIOSUBUV1
LIS

0398 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MOUNTVER
LIS

OPDRVWS1
LIS

QUORUM
LIS

OPDRV790
LIS

OPDRIVER
LIS

REBLDLOCK
LIS