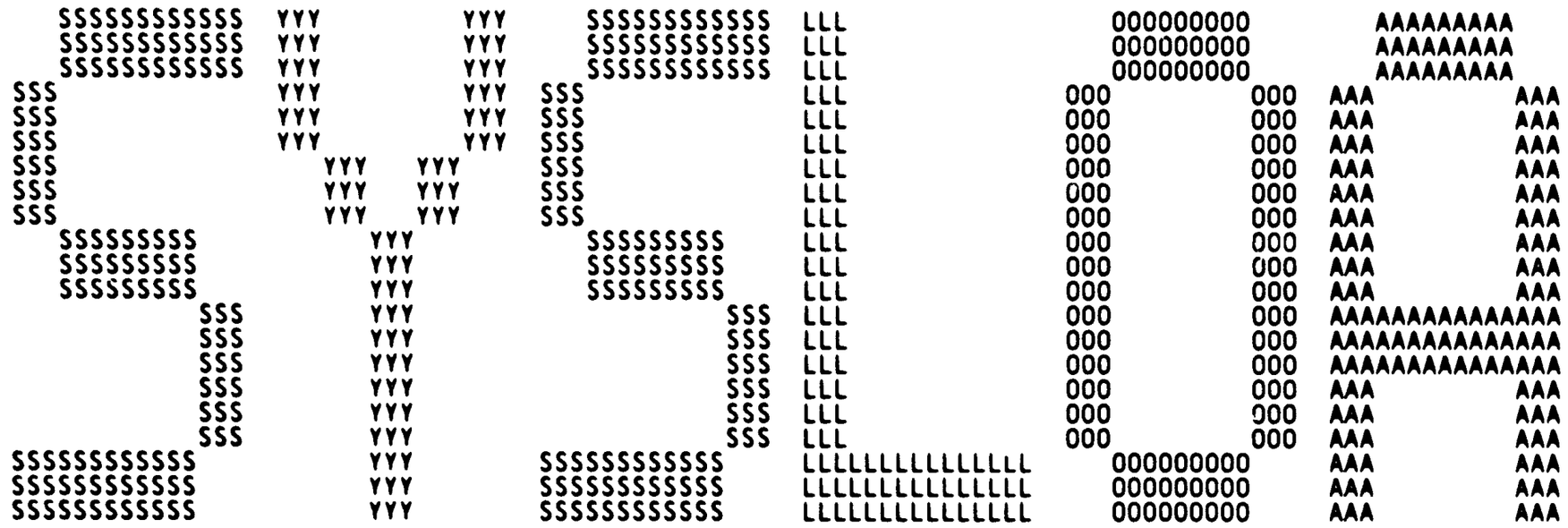
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA

FILEID**MCHECKUV1

Table of contents

```
0000     1          .TITLE  MCHECKUV1 -- Micro-VAX I Machine Check
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY: SYSLOAUV1 - loadable CPU-dependent code
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;       This module contains routines to handle Micro-VAX I specific
0000    34 ;       machine check errors.
0000    35 ;
0000    36 ; ENVIRONMENT:
0000    37 ;       IPL = 31 (1D for memory errors)         Mode = KERNEL
0000    38 ;
0000    39 ; AUTHOR:  Kathleen D. Morse,   CREATION DATE: 13-Sep-1983
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       V03-003 KDM0100         Kathleen D. Morse       1-May-1984
0000    44 ;               Add missing indirects to memory CSR references.
0000    45 ;
0000    46 ;       V03-002 KDM0096         Kathleen D. Morse       27-Mar-1984
0000    47 ;               Fix some bugs in the error logging routine.
0000    48 ;               Correct documentation of machine check codes.
0000    49 ;               Change branches to REFLECTCHK to cause a bugcheck,
0000    50 ;               since the VA and PSL may not always be accurate on
0000    51 ;               a memory error.
0000    52 ;
0000    53 ;--
```

```
                     0000     55          .SBTTL  DECLARATIONS
                     0000     56 ;
                     0000     57 ; INCLUDE FILES:
                     0000     58 ;
                     0000     59
                     0000     60 ;
                     0000     61 ; INCLUDED SYSTEM SYMBOL DEFINITIONS
                     0000     62 ;
                     0000     63          $EMBDEF <MC,SE>
                     0000     64          $IPLDEF
                     0000     65          $PCBDEF
                     0000     66          $PFNDEF
                     0000     67          $PHDDEF
                     0000     68          $PRDEF
                     0000     69          $PRUV1DEF
                     0000     70          $PSLDEF
                     0000     71          $PTEDEF
                     0000     72          $RPBDEF
                     0000     73          $VADEF
                     0000     74          $MCDEF
                     0000     75          $MCHKDEF
                     0000     76
                     0000     77 ;
                     0000     78 ; OWN STORAGE:
                     0000     79 ;
                     0000     80
FFFFFFFC             0000     81 MC$L_RECOVMSK = -4                      ; These symbols define offsets from AP
FFFFFFF8             0000     82 MC$L_PCPSLPTR = -8                      ; to locations on the stack; see
                     0000     83                                         ; functional description of MCHECKUV1.
                     0000     84
                     0000     85          .PSECT  MCHK$DATA,QUAD,WRT
                     0000     86 EXE$MCHK_ERRCNT::                       ; Used to locate error counters
                     0000     87                                         ; via SYS.MAP.
                     0000     88
                     0000     89 EXE$AL_MEMCSRS::                        ; Array of memory CSR addresses.
00000000             0000     90          .LONG   0                      ;  Count of memory CSRs.
00000044             0004     91          .BLKL   16                     ;  1 longword per possible CSR.
                     0044     92
                     0044     93 LAST_BADMCK:                            ; Time of last bad machine check code.
00000000             0044     94          .LONG   0
00000001             0048     95 BADMCK_MIN = 1                          ; Allowable time between bad machine
                     0048     96                                         ; check codes.
                     0048     97
                     0048     98 LAST_BADINT:                            ; Time of last illegal interrupt.
00000000             0048     99          .LONG   0
00000001             004C    100 BADINT_MIN = 1                          ; Allowable time between bad interrupts.
                     004C    101
                     004C    102 LAST_RDS:                              ; Time of last RDS error.
00000000             004C    103          .LONG   0
00000001             0050    104 RDS_MIN = 1                            ; Allowable time between RDS errors.
                     0050    105
                     0050    106 ;
                     0050    107 ; This data is used by ECC$REENABLE, which is periodically called to scan
                     0050    108 ; the memory controller CSRs for CRD errors.
                     0050    109 ;
                     0050    110 ECC$GW_REENAB:                         ; Time since CRD interrupts
0000                 0050    111          .WORD   0                      ; were last enabled.
```

```
                     0052   112
00000384             0052   113 REENABTIME = 60*15                  ; Reenable CRD interrupts every
                     0052   114                                     ; 15 minutes.
                     0052   115
                     0052   116 ECC$GW_CRDWATCH:                    ; Time since last scanned mem
0000                 0052   117         .WORD   0                   ; CSR for CRD errors.
                     0054   118
0000003C             0054   119 CRDWATCHTIME = 60                   ; Scan for non-interrupt CRD
                     0054   120                                     ; errors every 60 seconds.
                     0054   121
                     0054   122 ;
                     0054   123 ; The following data is used by the CRD interrupt handler, EXE$LOGCRD, and
                     0054   124 ; the memory CSR logging routine, EXE$LOGMEM.
                     0054   125 ;
                     0054   126 ECC$AB_MEMERR:                      ; Count CRD errors logged recently.
00                   0054   127         .BYTE   0                   ; (within REENABTIME)
                     0055   128
00000006             0055   129 CRDLOGMAX = 6                       ; Maximum number of CRD errors to log.
                     0055   130
                     0055   131 ECC$AB_CRDCNT:                     ; Count recent CRD interrupts (within
00                   0055   132         .BYTE   0                   ; REENABTIME).
                     0056   133
00000003             0056   134 CRDINTMAX = 3                       ; Maximum number of CRD interrupts
                     0056   135                                     ; before interrupts are disabled.
                     0056   136
                     0056   137 MMG$L_CRDCNT:                      ; Count total CRD interrupts.
00000000             0056   138         .LONG   0                   ;
                     005A   139
```

```
00000000   141          .PSECT  WIONONPAGED,QUAD,RD,WRT
     0000  142          .SBTTL  * MACHINE CHECK ENTRY POINT *
     0000  143
     0000  144 ;++
     0000  145 ; MCHECKUV1 -- Micro-VAX I Machine Check
     0000  146 ;
     0000  147 ; FUNCTIONAL DESCRIPTION:
     0000  148 ;
     0000  149 ;       All machine checks are vectored to this entry point.  By casing
     0000  150 ;       off of the machine check type code in the machine check logout
     0000  151 ;       stack, determine the recovery action (if any) appropriate for
     0000  152 ;       each error.
     0000  153 ;
     0000  154 ; Format of Micro-VAX I machine check logout stack:
     0000  155 ;
     0000  156 ;       On entry to this machine check handler, the stack is set up as follows:
     0000  157 ;
     0000  158 ;               00(SP): # bytes in machine check log on stack (always 0C hex)
     0000  159 ;               04(SP): machine check type code
     0000  160 ;               08(SP): 1st machine check parameter
     0000  161 ;               0C(SP): 2nd machine check parameter
     0000  162 ;               10(SP): exception PC
     0000  163 ;               14(SP): exception PSL
     0000  164 ;
     0000  165 ;       The meanings of the third and fourth longwords depend on the
     0000  166 ;       machine check type code.
     0000  167 ;
     0000  168 ;       As soon as the machine check handler is invoked, it sets up the stack
     0000  169 ;       as follows:
     0000  170 ;       +-------------------------------+       :(SP)
     0000  171 ;       |                               |
     0000  172 ;       .                               .
     0000  173 ;       .       saved R0 - R5, AP       .
     0000  174 ;       .                               .
     0000  175 ;       |                               |
     0000  176 ;       +-------------------------------+
     0000  177 ;       | pointer to PC/PSL of exception|       :MC$L_PCPSLPTR(AP)
     0000  178 ;       +-------------------------------+
     0000  179 ;       |       recovery mask           |       :MC$L_RECOVMSK(AP)
     0000  180 ;       +-------------------------------+
     0000  181 ;       | # bytes pushed on logout stack| :(AP) :MC$L_BYTCNT(AP)
     0000  182 ;       +-------------------------------+
     0000  183 ;       |  machine check type code      |       :MC$L_TYPECODE(AP)
     0000  184 ;       +-------------------------------+
     0000  185 ;       |       1st parameter           |       :MC$L_P1(AP)
     0000  186 ;       +-------------------------------+
     0000  187 ;       |       2nd parameter           |       :MC$L_P2(AP)
     0000  188 ;       +-------------------------------+
     0000  189 ;       |       exception PC            |       :MC$L_PC(AP)
     0000  190 ;       +-------------------------------+
     0000  191 ;       |       exception PSL           |       :MC$L_PSL(AP)
     0000  192 ;       +-------------------------------+
     0000  193 ;
     0000  194 ;       AP will point to the beginning of the machine check log on the stack.
     0000  195 ;       2 longwords are immediately pushed on top of the machine check log, and
     0000  196 ;       are referenced as negative offsets from AP.  These two longwords are
     0000  197 ;       input arguments to EXE$MCHK_BUGCHK, a routine that is called to check
```

```
                                0000   198 ;       for a user-declared machine check recovery block.  This routine is
                                0000   199 ;       called immediately before bugchecking, and expects the mask and the
                                0000   200 ;       pointer to the exception PC/PSL to be right on top of the machine check
                                0000   201 ;       log on the stack.
                                0000   202 ;--
                                0000   203         .ALIGN  LONG
                                0000   204 EXE$INT58::                                ; These interrupts are other-processor-
                                0000   205 EXE$INT5C::                                ; specific and should never be
                                0000   206                                           ; seen on Micro-VAX I.
                                0000   207 EXE$UBAERR_INT::
                                0000   208 EXE$RH780_INT::
                          28    11  0000   209         BRB     BAD_TYPE
                                0002   210
                                0002   211
                                0002   212         .ALIGN  LONG
                                0004   213 EXE$MCHK::                                ; Machine check handler.
                          02    DD  0004   214         PUSHL   #MCHK$M_MCK             ; Mask signals machine check.
                        14 AE   DF  0006   215         PUSHAL  MC$L_PC+4(SP)           ; Push pointer to exception PC/PSL.
                    103F 8F     BB  0009   216         PUSHR   #^M<R0,R1,R2,R3,R4,R5,AP> ; Working registers.
                5C  5E  24      C1  000D   217         ADDL3   #<9*4>,SP,AP            ; AP points to mchk log frame.
                                0011   218
                                0011   219         CASE    MC$L_TYPECODE(AP), -    ; Case on machine check code
                                0011   220                 <MEMCTLR_ERR, -         ; code  1 - memory controller bugchk
                                0011   221                 MEM_ERROR, -            ;       2 - unrecoverable read error
                                0011   222                 NX_MEM, -               ;       3 - nonexistent memory
                                0011   223                 UNALIGNED_IO, -         ;       4 - unaligned ref to I/O space
                                0011   224                 PTE_READCHK, -          ;       5 - page table read error
                                0011   225                 PTE_WRITECHK, -         ;       6 - page table write error
                                0011   226                 CS_PARITY, -            ;       7 - control store parity error
                                0011   227                 MICRO_ERRORS, -         ;       8 - micromachine bugcheck
                                0011   228                 BAD_VECTOR, -           ;       9 - Q22 bus vector read error
                                0011   229                 STACK_ERR>, -           ;      10 - error writing param onto stk
                                0011   230                 TYPE=B
                                002A   231
                                002A   232 BAD_TYPE:                                ; undefined exception
                      52    03  D0   002A   233         MOVL    #<MCHK$M_MCK! -         ; Type code for check for
                                002D   234                 MCHK$M_LOG>,R2          ; recovery block.
                         00E4   30   002D   235         BSBW    LOG_MCHECK              ; Log a machine check.
                                0030   236 ;
                                0030   237 ; Check to see if bad machine checks are occurring too rapidly.
                                0030   238 ;
                50  0044'CF     DE   0030   239         MOVAL   W^LAST_BADMCK,R0        ; Address of time stamp.
                    51    01    D0   0035   240         MOVL    #BADMCK_MIN,R1          ; Min allowable time between errors.
                         0073   30   0038   241         BSBW    CHK_AND_RESUME          ; Try to resume.  Returns if
                                003B   242                                           ; unresumable opcode.
                                003B   243         BUG_CHECK  BADMCKCOD,FATAL      ; Bad machine check code.
```

```
                        .SBTTL  MICROCODE DETECTED ERRORS
                ;++
                ; The following machine checks are caused by microcode-detected
                ; inconsistencies in the hardware.
                ;--

                ;++
                ; Memory controller bugcheck:
                ;
                ; Machine-check code:  1
                ;
                ; Description:  An invalid state was reached in the memory controller
                ;               and it was unable to successfully complete the last
                ;               function.
                ;
                ; Parameters:
                ;
                ;       MC$L_P1(AP): contents of the memory controller register "virtual".
                ;                    This register usually contains the physical address
                ;                    of the last function.
                ;       MC$L_P2(AP): address that was presented to the memory controller
                ;                    at the start of the function.
                ;--
                MEMCTLR_ERR:

                ;++
                ; Micromachine bug check:
                ;
                ; Machine-check code:  8
                ;
                ; Description:  An invalid state has been reached in the micromachine.
                ;               This is a catastrophic error.
                ;
                ; Parameters:
                ;
                ;       MC$L_P1(AP): 0
                ;       MC$L_P2(AP): 0
                ;--
                MICRO_ERRORS:                           ; Micromachine invalid state error.
```

```
                        .SBTTL  MEMORY REFERENCE ERRORS
                ;++
                ; The following machine checks are caused by memory reference errors
                ; of one sort or another.
                ;--

                ;++
                ; Illegal operation (Unaligned or non-longword reference to I/O space):
                ;
                ; Machine-check code:  4
                ;
                ; Description:  An attempt was made to access an unaligned word or a
                ;               longword in I/O space.
                ;
                ; Parameters:
                ;
                ;       MC$L_P1(AP): Physical address of the illegal I/O reference. (**)
                ;       MC$L_P2(AP): Address presented to the memory controller at
                ;                    the start of the function.
                ;
                ;       ** Note:
                ;               Physical and virtual addresses returned on memory
                ;               controller errors may not be the actual address of
                ;               the error if a page crossing occurs.  If the page
                ;               offset (i.e., bits <8:0>) are:
                ;
                ;                       00000001 and the data length was word, or
                ;                       00000001 and the data length was long, or
                ;                       00000010 and the data langth was long, or
                ;                       00000011 and the data length was long
                ;
                ;               then the page in which the error occurred could be
                ;               the one addresses or the one logically preceding the
                ;               one specified.
                ;--
UNALIGNED_IO:                                   ; Unaligned reference to I/O space, or
                                                ; non-longword ref to I/O space.

                ;++
                ; Unrecoverable page table read error:
                ;
                ; Machine-check code:  5
                ;
                ; Description:  An unrecoverable error occurred while attempting to
                ;               read a page table entry.  This error may have been a
                ;               parity, ECC, or timeout error.
                ;
                ; Parameters:
                ;
                ;       MC$L_P1(AP): physical address of page table entry.
                ;       MC$L_P2(AP): virtual address associated with the
                ;                    page table entry (i.e., the address that
                ;                    caused the page table entry to be read).
                ;--
PTE_READCHK:                                    ; Hard memory error on PTE read.

                ;++
```

```
003F    342 ; Unrecoverable page table write error:
003F    343 ;
003F    344 ; Machine-check code:  6
003F    345 ;
003F    346 ; Description:  This error occurred while attempting to write the
003F    347 ;               modify bit in a page table entry.  This error reflects hardware
003F    348 ;               that is in an unrunnable state and should be treated as a write
003F    349 ;               timeout error.
003F    350 ;
003F    351 ; Parameters:
003F    352 ;
003F    353 ;       MC$L_P1(AP): physical address of page table entry.
003F    354 ;       MC$L_P2(AP): virtual address associated with the
003F    355 ;                    page table entry (i.e., the address that
003F    356 ;                    caused the page table entry to be read).
003F    357 ;
003F    358 ;--
003F    359 PTE_WRITECHK:                            ; Hard memory error on PTE write.
003F    360
003F    361 ;++
003F    362 ; Control store parity error:
003F    363 ;
003F    364 ; Machine-check code:  7
003F    365 ;
003F    366 ; Description:  A control store parity error has occurred.  This is a
003F    367 ;               catastrophic error.
003F    368 ;
003F    369 ; Parameters:
003F    370 ;
003F    371 ;       MC$L_P1(AP): 0
003F    372 ;       MC$L_P2(AP): 0
003F    373 ;--
003F    374 CS_PARITY:                              ; Control store parity error.
003F    375
003F    376 ;++
003F    377 ; Q22 bus vector read error:
003F    378 ;
003F    379 ; Machine-check code:  9
003F    380 ;
003F    381 ; Description:  An error was encountered wyhile attempting to read an
003F    382 ;               interrupt vector address from the Q22 bus.)
003F    383 ;
003F    384 ; Parameters:
003F    385 ;
003F    386 ;       MC$L_P1(AP): Virtual address referenced
003F    387 ;       MC$L_P2(AP): Bad CSR value
003F    388 ;--
003F    389 BAD_VECTOR:                             ; Q22 bus vector read error.
003F    390
003F    391 ;++
003F    392 ; Write parameter error:
003F    393 ;
003F    394 ; Machine-check code:  10
003F    395 ;
003F    396 ; Description:  An error was encountered during an exception while
003F    397 ;               attempting to write the user, supervisor, or executive
003F    398 ;               stack after having verified that the write would succeed
```

```
                          003F   399 ;                (i.e., chmx and emulation).
                          003F   400 ;
                          003F   401 ; Parameters:
                          003F   402 ;
                          003F   403 ;        MC$L_P1(AP): virtual address that was being written. (**)
                          003F   404 ;        MC$L_P2(AP): 0
                          003F   405 ;
                          003F   406 ;        ** Note:
                          003F   407 ;                Physical and virtual addresses returned on memory
                          003F   408 ;                controller errors may not be the actual address of
                          003F   409 ;                the error if a page crossing occurs.  If the page
                          003F   410 ;                offset (i.e., bits <8:0>) are:
                          003F   411 ;
                          003F   412 ;                        00000001 and the data length was word, or
                          003F   413 ;                        00000001 and the data length was long, or
                          003F   414 ;                        00000010 and the data langth was long, or
                          003F   415 ;                        00000011 and the data length was long
                          003F   416 ;
                          003F   417 ;                then the page in which the error occurred could be
                          003F   418 ;                the one addresses or the one logically preceding the
                          003F   419 ;                one specified.
                          003F   420 ;--
                          003F   421 STACK_ERR:                                   ; Error writing parameter onto stack.
                          003F   422
                          003F   423
                 52  03   D0   003F   424         MOVL    #<MCHK$M_MCK! -         ; Type code for recovery block
                          0042   425                 MCHK$M_LOG>,R2              ; check.
                     00CF 30   0042   426         BSBW    LOG_MCHECK              ; Log the machine check.
                     0087 31   0045   427         BRW     REFLECTCHK              ; Reflect exception/bugcheck,
                               0048   428                                         ; based on current process mode.
```

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

```
                                        .SBTTL  UNCORRECTABLE MEMORY ERRORS

                                ;++
                                ;       UNCORRECTABLE MEMORY READ ERRORS
                                ;
                                ; Since this memory error could not be corrected by the hardware, the
                                ; physical memory page is unusable.  Mark the page bad, and reflect
                                ; exception/bugcheck.
                                ;--

                                ;++
                                ; Unrecoverable read error:
                                ;
                                ; Machine-check code:  2
                                ;
                                ; Description:  An unrecoverable read error occurred on the last memory
                                ;               controller function.  The error may have been a parity
                                ;               or an ECC error depending on the type of memory present.
                                ;
                                ; Parameters:
                                ;
                                ;       MC$L_P1(AP): physical address of the page containing the error (**)
                                ;       MC$L_P2(AP): address presented to the memory controller at the
                                ;                    start of the function
                                ;
                                ;       ** Note:
                                ;               Physical and virtual addresses returned on memory
                                ;               controller errors may not be the actual address of
                                ;               the error if a page crossing occurs.  If the page
                                ;               offset (i.e., bits <8:0>) are:
                                ;
                                ;                       00000001 and the data length was word, or
                                ;                       00000001 and the data length was long, or
                                ;                       00000010 and the data langth was long, or
                                ;                       00000011 and the data length was long
                                ;
                                ;               then the page in which the error occurred could be
                                ;               the one addresses or the one logically preceding the
                                ;               one specified.
                                ;--

                                        .ENABL  LSB
                                MEM_ERROR:                              ; Uncorrectable memory read error.
                                ;
                                ; Mark page bad.
                                ;
50   08 AC   F7 8F    78         ASHL    #-9,MC$L_P1(AP),R0              ; Get PFN of bad memory location.
  00000000'GF   50    D1         CMPL    R0,G^MMG$GL_MAXPFN              ; PFN data base for this page?
                0C    1A         BGTRU   10$                             ; No, cannot mark page bad.
  51    00000000'GF   DE         MOVAL   G^PFN$AB_TYPE, R1               ; Get address of PFN TYPE array.
      00 B140   20    88         BISB2   #PFN$M_BADPAG, -                ; Mark page bad.
                                        @(R1)[R0]                       ;
                                10$:
                                ;
                                ; Log a machine check and a memory error.
                                ;
          52    03    D0        MOVL    #<MCHK$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    543          .SBTTL  ASYNCHRONOUS WRITE ERROR INTERRUPT
                         007F    544 ;++
                         007F    545 ;
                         007F    546 ; An interrupt is generated to this vector whenever a Qbus write operation
                         007F    547 ; does not complete successfully.  There could be any number of reasons for
                         007F    548 ; this.  The Qbus adapter could be broken or it might have been a write to
                         007F    549 ; some non-existant address.
                         007F    550 ;
                         007F    551 ; There is no machine-check frame on the stack upon entry at this point.
                         007F    552 ;
                         007F    553 ; The SCB offset ^X60 is connected to this entrypoint.
                         007F    554 ;--
                         007F    555
                         007F    556          .ALIGN  LONG
                         0080    557
                         0080    558 EXE$LOGAWE::
                         0080    559 EXE$INT60::
        5E    0C   C2    0080    560          SUBL    #<4*3>,SP                       ; Allocate space for dummy mchk frame
              0C   DD    0083    561          PUSHL   #^X0C                           ; Length of machine-check frame
              07   DD    0085    562          PUSHL   #MCHK$M_LOG!MCHK$M_MCK!MCHK$M_NEXM ;Mask for PRTCTEST
           14 AE   DF    0087    563          PUSHAL  MC$L_PC+4(SP)                   ; and PC pointer
         103F 8F   BB    008A    564          PUSHR   #^M<R0,R1,R2,R3,R4,R5,AP> ; Save registers
5C       5E   24   C1    008E    565          ADDL3   #<9*4>,SP,AP                    ; Point AP to dummy machine-check frame
         53   07   3C    0092    566          MOVZWL  #EMB$K_AW,R3                    ; Error type
            006C   30    0095    567          BSBW    EXE$LOGMEM                      ; log the error
      51    10 AC   DE    0098    568          MOVAL   MC$L_PC(AP),R1                  ; Get address of PC/PSL.
 00000000'GF   16    009C    569          JSB     G^EXE$MCHK_TEST                 ; Is recovery block in effect?
           06 50   E8    00A2    570          BLBS    R0,10$                          ; Br on yes, do not log this error
 00000000'GF   D6    00A5    571          INCL    G^EXE$GL_MCHKERRS               ; Bump the global machine-check counter
            0055   31    00AB    572 10$:     BRW     BUGCHECK                        ; Bugcheck since VA/PSL may not
                         00AE    573                                                  ; be accurate enough to tell what
                         00AE    574                                                  ; mode the acess was taken in.
```

```
                        00AE    576          .SBTTL  * EXITS FROM MACHINE CHECK ROUTINES *
                        00AE    577          .SBTTL  CHK_AND_RESUME
                        00AE    578 ;++
                        00AE    579 ; CHK_AND_RESUME
                        00AE    580 ;
                        00AE    581 ; FUNCTIONAL DESCRIPTION:
                        00AE    582 ;
                        00AE    583 ;       Called to check time that this error last occurred.
                        00AE    584 ;       CHK_AND_RESUME has three possible exit paths:
                        00AE    585 ;               (1) If errors are occurring too rapidly, BUGCHECK.
                        00AE    586 ;       Else fall through to RESUME:
                        00AE    587 ;               (2) If opcode is unresumable, RSB.
                        00AE    588 ;               (3) Else resume: clear stack and REI to retry the instruction.
                        00AE    589 ; INPUTS:
                        00AE    590 ;
                        00AE    591 ;       R0: Address of longword which contains time error last occurred.
                        00AE    592 ;       R1: Minimum time that must have elapsed since the last error.
                        00AE    593 ;
                        00AE    594 ; IMPLICIT INPUTS:
                        00AE    595 ;
                        00AE    596 ;        EXE$GQ_SYSTIME
                        00AE    597 ;
                        00AE    598 ; OUTPUTS:
                        00AE    599 ;
                        00AE    600 ;       The longword pointed to by R0 is updated with the time of this error.
                        00AE    601 ;--
                        00AE    602 CHK_AND_RESUME:
          53    50      D0      00AE    603          MOVL    R0,R3                   ; Save R0
00000000'EF     16      00B1    604          JSB     EXE$READ_TODR           ; Get current time in R0,
          52    50      D0      00B7    605          MOVL    R0,R2                   ; and move it to R2.
          50    53      D0      00BA    606          MOVL    R3,R0                   ; Restore R0.
      53  52    60      C3      00BD    607          SUBL3   (R0),R2,R3              ; How long since last error?
          51    53      D1      00C1    608          CMPL    R3,R1                   ; Compare against minimum threshhold.
                05      1A      00C4    609          BGTRU   10$                     ; Br if enough time has elapsed.
                8E      D5      00C6    610          TSTL    (SP)+                   ; Else clear return address from stack.
              0038      31      00C8    611          BRW     BUGCHECK                ; Errors recurring too fast; bugcheck.
                        00CB    612 10$:
          60    52      D0      00CB    613          MOVL    R2,(R0)                 ; Save time of latest error.
                05      00CE    614          RSB                             ; Since no opcodes are resumable (all
                        00CF    615                                          ; possible retrying is done in micro-
                        00CF    616                                          ; code) just return to caller.
```

```
                                00CF    618          .SBTTL  REFLCTCHK
                                00CF    619 ;++
                                00CF    620 ;       REFLECT MACHINE CHECK TO USER
                                00CF    621 ;
                                00CF    622 ; This code is entered if the machine check was fatal.  It determines
                                00CF    623 ; if it was just fatal to the process which caused it (current process
                                00CF    624 ; is in USER or SUPER mode), or if it was fatal to the entire system
                                00CF    625 ; (current process is in EXEC or KERNEL mode).
                                00CF    626 ;
                                00CF    627 ; If current process is in USER or SUPER mode,
                                00CF    628 ;       set up an exception on user's stack and REI to it
                                00CF    629 ; If current process is in EXEC or KERNEL mode,
                                00CF    630 ;       issue a fatal bugcheck.
                                00CF    631 ;
                                00CF    632 ; CALLING SEQUENCE:
                                00CF    633 ;
                                00CF    634 ;       BRB/W -- NOTHING EXTRA CAN BE ON THE STACK!!
                                00CF    635 ;
                                00CF    636 ; STACK CONTENTS:
                                00CF    637 ;
                                00CF    638 ;       00(SP): saved R0,R1,R2,R3,R4,R5,AP
                                00CF    639 ;       1C(SP): 2 longword inputs for recovery block check
                                00CF    640 ;       24(SP): (also AP) machine check log -- 1st longword is a byte count.
                                00CF    641 ;--
                                00CF    642 REFLECTCHK:                             ; Reflect exception according
                                00CF    643                                         ; to current access mode.
03 14 AC    19    E0            00CF    644          BBS     #PSL$V_CURMOD+1, -     ; Branch if USER or SUPER.
                                00D4    645                  MC$L_PSL(AP),10$       ;
          002C    31            00D4    646          BRW     BUGCHECK               ; EXEC or KERNEL; bugcheck.
                                00D7    647
                                00D7    648 10$:                                    ; SUPER or USER; exception.
          50    00    DB        00D7    649          MFPR    #PR$_KSP,R0            ; Get kernel stack pointer.
       70   10 AC    7D        00DA    650          MOVQ    MC$L_PC(AP),-(R0)      ; Push PC,PSL on kernel stack.
          00    50    DA        00DE    651          MTPR    R0,#PR$_KSP            ; Replace new kernel stack ptr.
          103F  8F    BA        00E1    652          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ; Restore registers.
          5E    08    C0        00E5    653          ADDL    #<2*4>,SP              ; Pop inputs for recovery block check.
          5E    8E    C0        00E8    654          ADDL    (SP)+,SP               ; Pop mck log from stack.
                                00EB    655 ;
                                00EB    656 ; Set up an exception stack for current process.
                                00EB    657 ; The faulting PC,PSL pair are still on the interrupt stack.  Alter
                                00EB    658 ; them to look as if an exception has occurred.
                                00EB    659 ;
    6E    00000000'GF   9E      00EB    660          MOVAB   G^EXE$MCHECK,(SP)      ; Replace exception PC.
04 AE   04 AE   02    18  EF   00F2    661          EXTZV   #PSL$V_CURMOD, -       ; Zero exception PSL, except
                                00F9    662                  #PSL$S_CURMOD, -       ; for current access mode.
                                00F9    663                  4(SP),4(SP)
 04 AE   04 AE   16    9C      00F9    664          ROTL    #PSL$V_PRVMOD, -       ; Create a PSL of current mode
                                00FF    665                  4(SP),4(SP)            ; kernel, correct previous
                                00FF    666                                         ; mode, and IPL 0.
          26    0F    DA        00FF    667          MTPR    #^XF,#PRUV1$_MCESR     ; Clear "mcheck in progress" flag.
                      02        0102    668          REI                            ; Go to exception handler.
                                0103    669
```

```
                     0103   671          .SBTTL  BUGCHECK
                     0103   672 ;++
                     0103   673 ; If user has declared a recovery block, transfer control to it.
                     0103   674 ; Else issue a fatal bugcheck.
                     0103   675 ;
                     0103   676 ; CALLING SEQUENCE:
                     0103   677 ;
                     0103   678 ;       BRB/W -- NOTHING EXTRA CAN BE ON THE STACK!!!
                     0103   679 ;
                     0103   680 ; STACK CONTENTS ON ENTRY:
                     0103   681 ;
                     0103   682 ;       00(SP): saved R0,R1,R2,R3,R4,R5,AP
                     0103   683 ;       1C(SP): 2 longword inputs for recovery block check
                     0103   684 ;       24(SP): (also AP) machine check log
                     0103   685 ;--
                     0103   686 BUGCHECK:
          103F 8F    BA 0103   687          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ; Restore registers.
                     0107   688 ;
                     0107   689 ; A fatal bugcheck is now inevitable unless a user has declared a machine
                     0107   690 ; check recovery block.
                     0107   691 ;
                     0107   692
            26   0F  DA 0107   693          MTPR    #^XF,#PRUV1$_MCESR      ; Clear "mcheck in progress" flag.
00000000'GF   16     010A   694          JSB     G^EXE$MCHK_BUGCHK       ; If return, no recovery block.
                     0110   695          BUG_CHECK -                     ; Issue fatal bugcheck.
                     0110   696                  MACHINECHK,FATAL
```

```
                     0114   698          .SBTTL  * LOGGING ROUTINES FOR MACHINE CHECKS *
                     0114   699 ;++
                     0114   700 ; LOG_MCHECK -- format inputs to LOGGER
                     0114   701 ;
                     0114   702 ; INPUTS:
                     0114   703 ;
                     0114   704 ;       R2: a mask which specifies the type of error (hence, the type
                     0114   705 ;           of recovery block to check for)
                     0114   706 ;
                     0114   707 ; IMPLICIT INPUTS:
                     0114   708 ;
                     0114   709 ;       (AP): points to machine check log on stack
                     0114   710 ;
                     0114   711 ; OUTPUTS:
                     0114   712 ;
                     0114   713 ;       Error is formatted and logged in system error log.
                     0114   714 ;--
                     0114   715
                     0114   716 LOG_MCHECK:
                     0114   717 ;
                     0114   718 ; Test if a machine check recovery block that specifies no error
                     0114   719 ; logging is in effect.
                     0114   720 ;
      51   10 AC   DE  0114   721          MOVAL   MC$L_PC(AP),R1          ; R1 points to PC,PSL of abort.
00000000'GF   16  0118   722          JSB     G^EXE$MCHK_TEST         ; Logging inhibited?
        01 50   E9  011E   723          BLBC    R0,10$                  ; Branch if no.
                05  0121   724          RSB                             ; Else return.
                    0122   725 10$:                                     ; Set up inputs to LOGGER.
 00000000'GF   D6  0122   726          INCL    G^EXE$GL_MCHKERRS       ; Bump machine check error count
        53   02  0128   727          MOVL    #EMB$K_MC,R3            ; Use Machine Check type code.
   54   08   6C  012B   728          ADDL3   MC$L_BYTCNT(AP), -      ; Size of data to log: machine check
                    012F   729                  #<2*4>,R4               ; stack + PC,PSL.
    55   04 AC   DE  012F   730          MOVAL   MC$L_TYPECODE(AP),R5    ; Address of data to log.
                    0133   731 ;++
                    0133   732 ; LOGGER - release error data to error logger
                    0133   733 ;
                    0133   734 ; INPUTS:
                    0133   735 ;
                    0133   736 ;       R3: error type
                    0133   737 ;       R4: number of bytes to log
                    0133   738 ;       R5: address of information to be logged
                    0133   739 ;
                    0133   740 ; OUTPUTS:
                    0133   741 ;
                    0133   742 ;       Error log is inserted into error log buffer.
                    0133   743 ;       If no error log buffer, return with error status in R0.
                    0133   744 ;       R0-R5 destroyed.
                    0133   745 ;--
                    0133   746
                    0133   747 LOGGER:
   51   54   10   C1  0133   748          ADDL3   #EMB$B_MC_SUMCOD,R4,R1  ; Add space for log header.
    00000000'GF   16  0137   749          JSB     G^ERL$ALLOCEMB          ; Get error logging buffer.
          13 50   E9  013D   750          BLBC    R0, 20$                 ; Br if failed to get buffer.
             52   DD  0140   751          PUSHL   R2                      ; Save buffer addr on stack.
    04 A2    53   B0  0142   752          MOVW    R3,EMB$W_MC_ENTRY(R2)   ; Set entry type.
10 A2   65   54   28  0146   753          MOVC3   R4,(R5),EMB$B_MC_SUMCOD(R2) ; Transfer info to log.
             04   BA  014B   754          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   759 ;++
0154   760 ; EXE$LOGMEM -- log memory Control and Status Registers
0154   761 ;
0154   762 ; FUNCTIONAL DESCRIPTION:
0154   763 ;
0154   764 ;       EXE$LOGMEM is called to log memory CSRs.  If called with R3 =
0154   765 ;       EMB$K_SE (log a soft memory error), look at the memory CSRs to see
0154   766 ;       if the CRD (soft error) bit was set; if not, don't log the CSRs.
0154   767 ;       If "too many" CRD errors have been logged recently, also don't log
0154   768 ;       the CSRs.
0154   769 ;
0154   770 ;       The format of the MSV-11P CSR is as follows:
0154   771 ;
0154   772 ;               15 14    11             5     2   0
0154   773 ;               +-----------------------------------+
0154   774 ;               |P|X| | | error address | | |W| |N|
0154   775 ;               +-----------------------------------+
0154   776 ;
0154   777 ;           P (bit 15) - parity error, set when parity error occurs.  This
0154   778 ;                        bit turns on a red LED on the module.  Set to 0
0154   779 ;                        by power up or BUS INIT and remains set unless
0154   780 ;                        rewritten or initialized.
0154   781 ;
0154   782 ;           X (bit 14) - extended CSR read enable, used on 22-bit address
0154   783 ;                        machines.  This bit is set by a program to request
0154   784 ;                        error address bits 18-21 be places in CSR bits 5-8.
0154   785 ;
0154   786 ;             (bits 13-12) - not used
0154   787 ;
0154   788 ;           error address (bits 11-5) - address bits 11-17.  To get address
0154   789 ;                        bits 18-21, bit 14 must be set by a program and the
0154   790 ;                        CSR re-read.
0154   791 ;
0154   792 ;             (bits 4-3) - not used
0154   793 ;
0154   794 ;           W (bit 2) - write wrong parity.  If this bit is set and a DATO
0154   795 ;                       or DATOB cycle to memory occurs, wrong parity data
0154   796 ;                       is written into the parity MOS RAMs.  This bit can
0154   797 ;                       be used to check the parity error logic as well as
0154   798 ;                       failed address information in the CSR.
0154   799 ;
0154   800 ;             (bit 1) - not used
0154   801 ;
0154   802 ;           N (bit 0) - parity error enable.  If set and parity error occurs,
0154   803 ;                       then BDAL 16L and BDAL 17L are asserted on the bus
0154   804 ;                       simultaneously with the data (error interrupt is
0154   805 ;                       generated).  This is a read/write bit reset to zero on
0154   806 ;                       power up or BUS INIT.
0154   807 ;
0154   808 ; INPUTS:
0154   809 ;
0154   810 ;       R3: errorlog type code
0154   811 ;
0154   812 ; IMPLICIT INPUTS:
0154   813 ;
0154   814 ;       EXE$AL_MEMCSRS - array of memory CSR addresses
0154   815 ;
```

```
                                0154   816 ; OUTPUTS:
                                0154   817 ;
                                0154   818 ;       Create entry in errorlog buffer containing the three memory controller
                                0154   819 ;       Control and Status Registers.
                                0154   820 ;       CRD error logging may be disabled.
                                0154   821 ;       R0: low bit signals success/failure
                                0154   822 ;       R1-R5 destroyed.
                                0154   823 ;--
                                0154   824 EXE$LOGMEM::
                                0154   825
                                0154   826 ;
                                0154   827 ; Now push all possible memory CSRs onto the stack.  The architectural
                                0154   828 ; limit is 16 memory controllers.  Since the CSR address is not logged
                                0154   829 ; with each CSR, leave zeroed cells for CSRs that had no errors.
                                0154   830 ; This is to prevent some problem with trying to read a memory address
                                0154   831 ; from a register that cannot provide one since it had no error.
                                0154   832 ;
          5E   00000040 8F   C2 0154   833         SUBL    #<4*16>,SP                      ; Allocate max size err log buffer.
6E  0040 8F   00   6E   00   2C 015B   834         MOVC5   #0,(SP),#0,#<4*16>,(SP)         ; Zero out the entire err log buffer.
          51   00000000'GF   DE 0163   835         MOVAL   G^EXE$AL_MEMCSRS,R1             ; Get address of memory CSR array.
                    55   81   D0 016A   836         MOVL    (R1)+,R5                        ; Get count of memory CSRs.
                                016D   837         DSBINT  #31,R0                          ; Block out all interrupts.
                    52   5E   D0 0173   838         MOVL    SP,R2                           ; Get address of buffer for CSRs.
                         54   D4 0176   839         CLRL    R4                              ; Initialize parity error bit counter.
                                0178   840
       00 B1    8000 8F   B3 0178   841 10$:    BITW    #^X8000,@(R1)                   ; Is parity error bit set?
                      17   13 017E   842         BEQL    20$                             ; Br if not set, no error.
             62    00 B1   B0 0180   843         MOVW    @(R1),(R2)                      ; Store memory CSR into buffer.
       00 B1    4000 8F   A8 0184   844         BISW    #^X4000,@(R1)                   ; Set bit requesting other err adr bits.
                      54   D6 018A   845         INCL    R4                              ; Count number of error bits set.
          02 A2    00 B1   B0 018C   846         MOVW    @(R1),2(R2)                     ; Store memory CSR into buffer.
       00 B1    8000 8F   AA 0191   847         BICW    #^X8000,@(R1)                   ; Clear parity error bit.
                51   04   C0 0197   848 20$:    ADDL    #4,R1                           ; Get address of next memory CSR.
                52   04   C0 019A   849         ADDL    #4,R2                           ; Point to next buffer cell.
                  D8 55   F5 019D   850         SOBGTR  R5,10$                          ; Loop through all CSRs.
                             01A0   851
                             01A0   852         ENBINT  R0                              ; Restore IPL.
                     54   DD 01A3   853         PUSHL   R4                              ; Push count of CSRs with errors.
                     22   13 01A5   854         BEQL    NOLOG                           ; Br if no parity errors found.
            00000000'GF   D6 01A7   855         INCL    G^EXE$GL_MEMERRS                ; Bump memory error counter
                             01AD   856 ;
                             01AD   857 ; A CRD error occurred.  Count it, and if we haven't logged a lot of CRD errors
                             01AD   858 ; recently, log it.
                             01AD   859 ;
                06   53   91 01AD   860         CMPB    R3,#EMB$K_SE                    ; Looking for CRD errors?
                     0B   12 01B0   861         BNEQ    LOG_CSRS                        ; No.  Unconditionally log CSRs.
                0054'CF   96 01B2   862         INCB    W^ECC$AB_MEMERR                 ; Count # of CRD errors LOGGED recently.
          06    0054'CF   91 01B6   863         CMPB    W^ECC$AB_MEMERR, -              ; Already logged enough CRD errors
                             01BB   864                 #CRDLOGMAX                      ; recently?
                     0C   1A 01BB   865         BGTRU   NOLOG                           ; Yes.  Skip the logging.
                             01BD   866
                             01BD   867 LOG_CSRS:
                     54   D6 01BD   868         INCL    R4                              ; Add in one for CSR count.
          54   54   02   78 01BF   869         ASHL    #2,R4,R4                        ; Size of errorlog buffer.
                55   5E   D0 01C3   870         MOVL    SP,R5                           ; Point to error log buffer.
                   FF6A   30 01C6   871         BSBW    LOGGER                          ; Log memory CSRs.
                             01C9   872 NOLOG:
```

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

```
                        01D1    876          .SBTTL  ECC$REENABLE -- TIMER CALL FROM SYSTEM CLOCK ROUTINE
                        01D1    877 ;++
                        01D1    878 ; ECC$REENABLE -- TIMER CALL FROM SYSTEM CLOCK ROUTINE
                        01D1    879 ;
                        01D1    880 ; FUNCTIONAL DESCRIPTION:
                        01D1    881 ;
                        01D1    882 ;       This routine periodically scans memory controller CSRs for
                        01D1    883 ;       CRD errors.  CRD errors are normally reported by interrupt,
                        01D1    884 ;       but even when CRD interrupts are turned off this routine will
                        01D1    885 ;       still scan memory controller CSRs periodically, to report a
                        01D1    886 ;       representative sample of CRD errors.
                        01D1    887 ;
                        01D1    888 ;       Also, check if it is time to reenable CRD interrupts.
                        01D1    889 ;       CRD interrupts may have been disabled by the CRD interrupt handler,
                        01D1    890 ;       EXE$LOGCRD, if it determines that "too many" interrupts are being
                        01D1    891 ;       received.
                        01D1    892 ;
                        01D1    893 ; INPUTS:
                        01D1    894 ;
                        01D1    895 ;       NONE
                        01D1    896 ;
                        01D1    897 ; IMPLICIT OUTPUTS:
                        01D1    898 ;
                        01D1    899 ;       If a CRD error is found the memory controller CSRs will be logged.
                        01D1    900 ;       CRD (Corrected Read Data) interrupts may be enabled for all
                        01D1    901 ;       memory controllers.
                        01D1    902 ;--
                        01D1    903
                        01D1    904 ECC$REENABLE::
          0052'CF   B7  01D1    905          DECW    W^ECC$GW_CRDWATCH       ; Time to scan for CRD errors?
                0F  14  01D5    906          BGTR    REENAB_SCAN             ; Branch if no.
                3F  BB  01D7    907          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save working registers.
     0052'CF    3C  B0  01D9    908          MOVW    #CRDWATCHTIME, -        ; Reset scan timer.
                        01DE    909                  W^ECC$GW_CRDWATCH       ;
          53    06  3C  01DE    910          MOVZWL  #EMB$K_SE,R3            ; Test for CRD error,
              FF70  30  01E1    911          BSBW    EXE$LOGMEM              ; and log memory CSRs if found.
                3F  BA  01E4    912          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                        01E6    913 ;
                        01E6    914 ; If any CRD errors were found, the memory controller CSRs were logged.
                        01E6    915 ; Now check to see if its time to enable CRD interrupts.  CRD interrupts are
                        01E6    916 ; enabled periodically, whether or not they were disabled by EXE$LOGCRD.
                        01E6    917 ;
                        01E6    918 REENAB_SCAN:
          0050'CF   B7  01E6    919          DECW    W^ECC$GW_REENAB         ; Has reenable time elapsed?
                31  14  01EA    920          BGTR    20$                     ; Branch if no.
     0050'CF    0384 8F  B0  01EC    921          MOVW    #REENABTIME, -          ; Yes.  Reset reenable timer.
                        01F3    922                  W^ECC$GW_REENAB         ;
          0054'CF   94  01F3    923          CLRB    W^ECC$AB_MEMERR         ; Reset CRD log counter.
          0055'CF   94  01F7    924          CLRB    W^ECC$AB_CRDCNT         ; Reset CRD interrupt counter.
 00000000'GF    00'  E1  01FB    925          BBC     S^#EXE$V_CRDENABL, -    ; Br if SYSGEN parameter does
                1A      0202    926                  G^EXE$GL_FLAGS,20$      ; not specify CRD interrupts.
                        0203    927
          7E    50  7D  0203    928          MOVQ    R0,-(SP)                ; Save working registers.
   51   00000000'GF  DE  0206    929          MOVAL   G^EXE$AL_MEMCSRS,R1     ; Get address of memory CSR array.
          50    81  D0  020D    930          MOVL    (R1)+,R0                ; Get count of memory CSRs.
        00 B1   01  A8  0210    931 10$:     BISW    #1,@(R1)                ; Reenable CRD interrupts.
          51    04  C0  0214    932          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   938          .SBTTL  EXE$LOGCRD -- CORRECTED MEMORY DATA INTERRUPTS
                        021E   939 ;++
                        021E   940 ; This routine is called when a CRD -- Corrected Read Data -- interrupt
                        021E   941 ; is received from a memory controller.  Log all interrupts, and
                        021E   942 ; continue.  If too many CRD interrupts are logged, turn off CRD interrupts.
                        021E   943 ;--
                        021E   944          .ALIGN  LONG
                        0220   945 EXE$LOGCRD::
                        0220   946 EXE$INT54::
                  3F    BB    0220   947          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save working registers.
            53    06    3C    0222   948          MOVZWL  #EMB$K_SE,R3            ; Soft memory error.
              FF2C    30    0225   949          BSBW    EXE$LOGMEM              ; Log a memory error.
          0056'CF     D6    0228   950          INCL    W^MMG$L_CRDCNT          ; Count total CRD interrupts.
          0055'CF     96    022C   951          INCB    W^ECC$AB_CRDCNT         ; Count recent CRD interrupts.
    03    0055'CF     91    0230   952          CMPB    W^ECC$AB_CRDCNT, -      ; More than enough CRD interrupts
                        0235   953                  #CRDINTMAX              ; lately?
                  14    1B    0235   954          BLEQU   20$                     ; No, do not disable CRD interrupts.
                        0237   955
51  00000000'GF     DE    0237   956          MOVAL   G^EXE$AL_MEMCSRS,R1     ; Get address of memory CSR array.
            50    81    D0    023E   957          MOVL    (R1)+,R0                ; Get count of memory CSRs.
         00 B1    01    AA    0241   958 10$:     BICW    #1,@(R1)                ; Disable CRD interrupts.
            51    04    C0    0245   959          ADDL    #4,R1                   ; Get VA of next memory controller CSR.
              F6  50    F5    0248   960          SOBGTR  R0,10$                  ; Loop through all CSRs.
                        024B   961
                        024B   962 20$:
                  3F    BA    024B   963          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                  02    02    024D   964          REI                             ; Return from interrupt.
                        024E   965
                        024E   966          .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
```

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
_255DUA28:[SYSLOA.OBJ]MC.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	17
_255DUA28:[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
```

LIOSUB750
LIS

MCHECKUV1
LIS

MCHECK730
LIS

MCHECK750
LIS

LIOSUB730
LIS

LIOSUB790
LIS

MCHECK780
LIS

MCF790
LIS

MCHECK790
LIS

LIOSUB780
LIS

LIOSUBUV1
LIS

MOUNTVER
LIS

OPDRVWS1
LIS

QUORUM
LIS

OPDRV790
LIS

OPDRIVER
CIS

REBLDLOCK
LIS