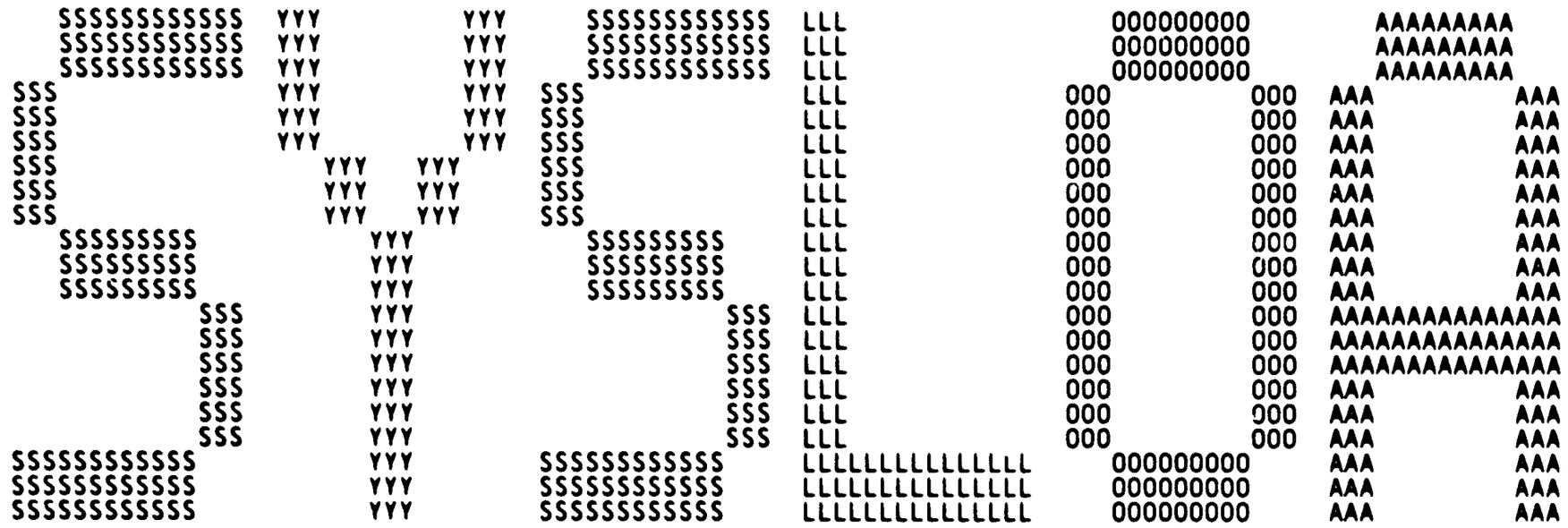
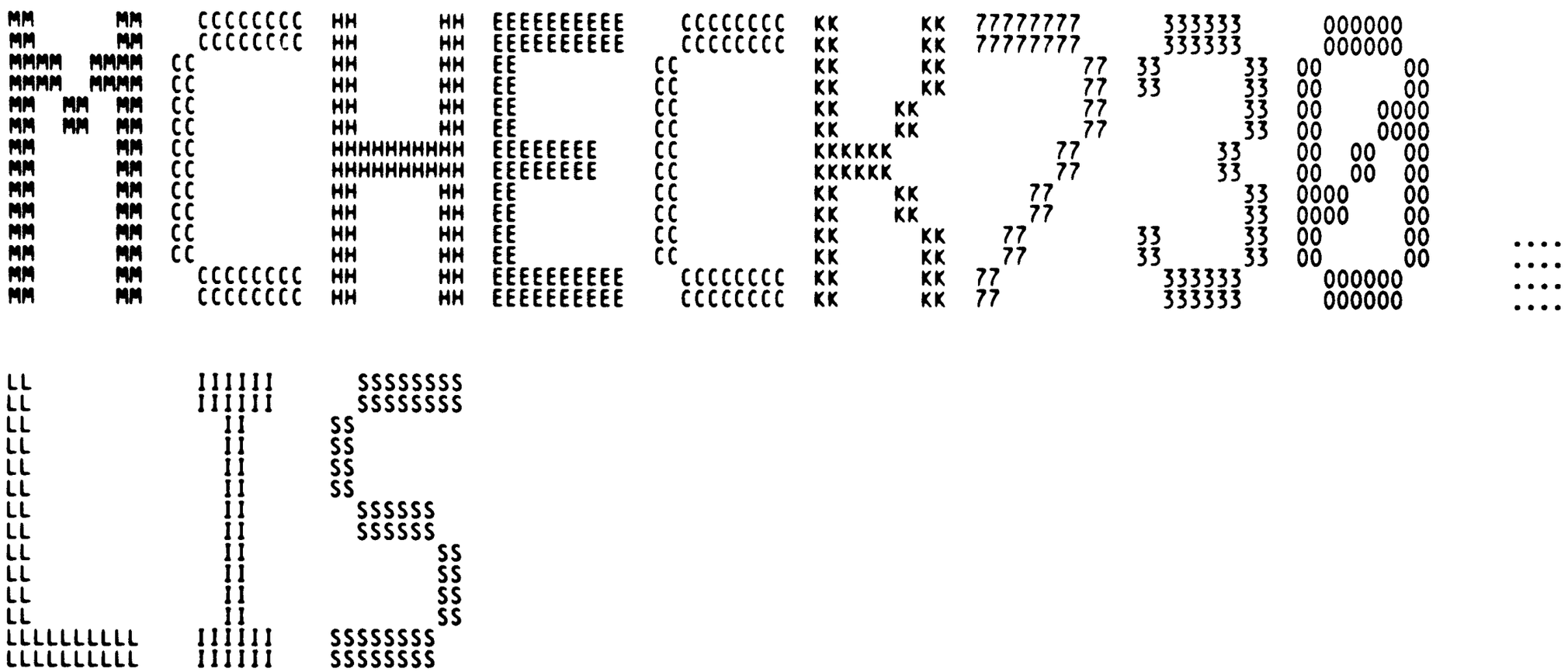

SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 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


MM MM CCCCCCCC HH HH EEEEEEEEEE CCCCCCCC KK KK 77777777 333333 000000
MM MM CCCCCCCC HH HH EEEEEEEEEE CCCCCCCC KK KK 77777777 333333 000000
MMMM MMMM CC HH HH EE CC KK KK 77 33 33 00 00
MMMM MMMM CC HH HH EE CC KK KK 77 33 33 00 00
MM MM MM CC HH HH EE CC KK KK 77 33 00 0000
MM MM MM CC HH HH EE CC KK KK 77 33 00 0000
MM MM CC HHHHHHHHHH EEEEEEEE CC KKKKK 77 33 00 00 00
MM MM CC HHHHHHHHHH EEEEEEEE CC KKKKK 77 33 00 00 00
MM MM CC HH HH EE CC KK KK 77 33 0000 00
MM MM CC HH HH EE CC KK KK 77 33 0000 00
MM MM CC HH HH EE CC KK KK 77 33 33 00 00
MM MM CC HH HH EE CC KK KK 77 33 33 00 00
MM MM CCCCCCCC HH HH EEEEEEEEEE CCCCCCCC KK KK 77 333333 000000
MM MM CCCCCCCC HH HH EEEEEEEEEE CCCCCCCC KK KK 77 333333 000000
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

```
0000     1          .TITLE  MCHECK730 -- NEBULA 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: SYSLOA730 - loadable CPU-dependent code
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;       This module contains routines to handle VAX 11/730 specific
0000    34  ;       machine check errors.
0000    35  ;
0000    36  ; ENVIRONMENT:
0000    37  ;       IPL = 31        Mode = KERNEL
0000    38  ;
0000    39  ; AUTHOR:  TRUDY MATTHEWS,      CREATION DATE: 25-Feb-1981
0000    40  ;
0000    41  ; MODIFIED BY:
0000    42  ;
0000    43  ;       V03-004 KDM0053         Kathleen D. Morse       11-Jul-1983
0000    44  ;               Change references to cpu-specific IPRs to use the
0000    45  ;               new cpu-specific $PR730DEF definitions.
0000    46  ;
0000    47  ;       V03-003 KTA3018         Kerbey T. Altmann       01-Nov-1982
0000    48  ;               Change name of psect for data.
0000    49  ;
0000    50  ;       V03-002 MSH0001         Maryann Hinden          09-Oct-1982
0000    51  ;               Change EXE$DW780_INT to EXE$UBAERR_INT.
0000    52  ;
0000    53  ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    54  ;               Added $VADEF.
0000    55  ;
0000    56  ;
0000    57  ;--
```

```
                     59          .SBTTL  DECLARATIONS
                     60 ;
                     61 ; INCLUDE FILES:
                     62 ;
                     63
                     64 ;
                     65 ; INCLUDED SYSTEM SYMBOL DEFINITIONS
                     66 ;
                     67          $EMBDEF <MC,SE>
                     68          $IPLDEF
                     69          ;$MPMDEF
                     70          $PCBDEF
                     71          $PFNDEF
                     72          $PHDDEF
                     73          $PRDEF
                     74          $PR730DEF
                     75          $PSLDEF
                     76          $PTEDEF
                     77          $RPBDEF
                     78          $VADEF
                     79 ; put these definitions into SYSDEF?
                     80          $MCDEF
                     81          $MCHKDEF
                     82          $MEMDEF
                     83          $CSR0DEF
                     84          $CSR1DEF
                     85          $CSR2DEF
                     86
                     87 ;
                     88 ; OWN STORAGE:
                     89 ;
                     90
FFFFFFFC             91 MC$L_RECOVMSK = -4                      ; These symbols define offsets from AP
FFFFFFF8             92 MC$L_PCPSLPTR = -8                      ; to locations on the stack; see
                     93                                         ; functional description of MCHECK730.
                     94
                     95          .PSECT  MCHK$DATA,QUAD,WRT
                     96 ;****** WHERE IS THIS TO BE DOCUMENTED???? ********
                     97 EXE$MCHK_ERRCNT::                       ; Used to locate error counters
                     98                                         ; via SYS.MAP.
                     99
                    100 EXE$GL_TB1OLD::                        ; Time of last TB parity error.
00000000             101          .LONG   0
                    102 EXE$GL_TB2OLD::                        ; Time of next-to-last TB error.
00000000            103          .LONG   0
0000000A            104 TB_THRESHOLD = 10                      ; Allowable time between TB errors
                    105                                        ; (in 10 millisecond units).
                    106
                    107
                    108 LAST_BADMCK:                           ; Time of last bad machine check code.
00000000            109          .LONG   0
00000001            110 BADMCK_MIN = 1                         ; Allowable time between bad machine
                    111                                        ; check codes.
                    112
                    113 LAST_BADINT:                           ; Time of last illegal interrupt.
00000000            114          .LONG   0
00000001            115 BADINT_MIN = 1                         ; Allowable time between bad interrupts.
```

```
                                                H 5
                     0010   116
                     0010   117 LAST_FPAPARITY:                          ; Time of last FPA parity error.
00000000             0010   118         .LONG   0
00000001             0014   119 FPAPARITY_MIN = 1                        ; Allowable time between FPA errors.
                     0014   120
                     0014   121 LAST_RDS:                                ; Time of last RDS error.
00000000             0014   122         .LONG   0
00000001             0018   123 RDS_MIN = 1                              ; Allowable time between RDS errors.
                     0018   124
                     0018   125 ;
                     0018   126 ; This data is used by ECC$REENABLE, which is periodically called to scan
                     0018   127 ; the memory controller CSRs for CRD errors.
                     0018   128 ;
                     0018   129 ECC$GW_REENAB:                           ; Time since CRD interrupts
0000                 0018   130         .WORD   0                       ; were last enabled.
                     001A   131
00000384             001A   132 REENABTIME = 60*15                       ; Reenable CRD interrupts every
                     001A   133                                         ; 15 minutes.
                     001A   134
                     001A   135 ECC$GW_CRDWATCH:                        ; Time since last scanned mem
0000                 001A   136         .WORD   0                       ; CSR for CRD errors.
                     001C   137
0000003C             001C   138 CRDWATCHTIME = 60                       ; Scan for non-interrupt CRD
                     001C   139                                         ; errors every 60 seconds.
                     001C   140
                     001C   141 ;
                     001C   142 ; The following data is used by the CRD interrupt handler, EXE$LOGCRD, and
                     001C   143 ; the memory CSR logging routine, EXE$LOGMEM.
                     001C   144 ;
                     001C   145 ECC$AB_MEMERR:                          ; Count CRD errors logged recently.
00                   001C   146         .BYTE   0                       ; (within REENABTIME)
                     001D   147
00000006             001D   148 CRDLOGMAX = 6                            ; Maximum number of CRD errors to log.
                     001D   149
                     001D   150 ECC$AB_CRDCNT:                          ; Count recent CRD interrupts (within
00                   001D   151         .BYTE   0                       ; REENABTIME).
                     001E   152
00000003             001E   153 CRDINTMAX = 3                            ; Maximum number of CRD interrupts
                     001E   154                                         ; before interrupts are disabled.
                     001E   155
                     001E   156 MMG$L_CRDCNT:                           ; Count total CRD interrupts.
00000000             001E   157         .LONG   0                       ;
                     0022   158
                     0022   159 ;
                     0022   160 ; This is a table that maps one bit for each opcode in the VAX
                     0022   161 ; instruction set.  If the corresponding bit is set, that opcode
                     0022   162 ; may be safely restarted if interrupted by a machine check.
                     0022   163 ;
                     0022   164 RESUME_TABLE:
3017                 0022   165         .WORD   ^B0011000000010111      ;HALT,NOP,REI,RET,PROBER/W
FFBE                 0024   166         .WORD   ^B1111111110111110      ;BRANCHES
0000                 0026   167         .WORD   ^B0000000000000000      ;
0002                 0028   168         .WORD   ^B0000000000000010      ;BRW
0000                 002A   169         .WORD   ^B0000000000000000      ;
000A                 002C   170         .WORD   ^B0000000000001010      ;CMPF,TSTF
0000                 002E   171         .WORD   ^B0000000000000000      ;
000A                 0030   172         .WORD   ^B0000000000001010      ;CMPD,TSTD
```

```
0000  0032   173          .WORD   ^B0000000000000000      ;
0022  0034   174          .WORD   ^B0000000000100010      ;CMPB,TSTB
0000  0036   175          .WORD   ^B0000000000000000      ;
0322  0038   176          .WORD   ^B0000001100100010      ;CMPW,TSTW,BISPSW,BICPSW
0000  003A   177          .WORD   ^B0000000000000000      ;
0022  003C   178          .WORD   ^B0000000000100010      ;CMPL,TSTL
3303  003E   179          .WORD   ^B0011001100000011      ;BBS,BBC,BLBS,BLBC,CMPV,CMPZV
      0040   180
```

```
00000000   182          .PSECT  WIONONPAGED,QUAD,RD,WRT
     0000  183          .SBTTL  * MACHINE CHECK ENTRY POINT *
     0000  184
     0000  185  ;++
     0000  186  ; MCHECK730 -- NEBULA MACHINE CHECK
     0000  187  ;
     0000  188  ; FUNCTIONAL DESCRIPTION:
     0000  189  ;       All machine checks are vectored to this entry point.  By casing
     0000  190  ;       off of the machine check type code in the machine check logout
     0000  191  ;       stack, determine the recovery action (if any) appropriate for
     0000  192  ;       each error.
     0000  193  ;
     0000  194  ; FORMAT OF NEBULA'S MACHINE CHECK LOGOUT STACK:
     0000  195  ;       On entry to this machine check handler, the stack is set up as follows:
     0000  196  ;
     0000  197  ;               00(SP): # bytes in machine check log on stack (always 0C hex)
     0000  198  ;               04(SP): machine check type code
     0000  199  ;               08(SP): 1st machine check parameter
     0000  200  ;               0C(SP): 2nd machine check parameter
     0000  201  ;               10(SP): exception PC
     0000  202  ;               14(SP): exception PSL
     0000  203  ;
     0000  204  ;       The meanings of the third and fourth longwords depend on the
     0000  205  ;       machine check type code.
     0000  206  ;
     0000  207  ;       As soon as the machine check handler is invoked, it sets up the stack
     0000  208  ;       as follows:
     0000  209  ;       +-------------------------------+       :(SP)
     0000  210  ;       |                               |
     0000  211  ;       .                               .
     0000  212  ;       |       saved R0 - R5, AP       |
     0000  213  ;       .                               .
     0000  214  ;       |                               |
     0000  215  ;       +-------------------------------+
     0000  216  ;       | pointer to PC/PSL of exception|       :MC$L_PCPSLPTR(AP)
     0000  217  ;       +-------------------------------+
     0000  218  ;       |       recovery mask           |       :MC$L_RECOVMSK(AP)
     0000  219  ;       +-------------------------------+
     0000  220  ;       | # bytes pushed on logout stack| :(AP) :MC$L_BYTCNT(AP)
     0000  221  ;       +-------------------------------+
     0000  222  ;       |  machine check type code      |       :MC$L_TYPECODE(AP)
     0000  223  ;       +-------------------------------+
     0000  224  ;       |       1st parameter           |       :MC$L_P1(AP)
     0000  225  ;       +-------------------------------+
     0000  226  ;       |       2nd parameter           |       :MC$L_P2(AP)
     0000  227  ;       +-------------------------------+
     0000  228  ;       |       exception PC            |       :MC$L_PC(AP)
     0000  229  ;       +-------------------------------+
     0000  230  ;       |       exception PSL           |       :MC$L_PSL(AP)
     0000  231  ;       +-------------------------------+
     0000  232  ;
     0000  233  ;       AP will point to the beginning of the machine check log on the stack.
     0000  234  ;       2 longwords are immediately pushed on top of the machine check log, and
     0000  235  ;       are referenced as negative offsets from AP.  These two longwords are
     0000  236  ;       input arguments to EXE$MCHK_BUGCHK, a routine that is called to check
     0000  237  ;       for a user-declared machine check recovery block.  This routine is
     0000  238  ;       called immediately before bugchecking, and expects the mask and the
```

```
                                0000   239 ;       pointer to the exception PC/PSL to be right on top of the machine check
                                0000   240 ;       log on the stack.
                                0000   241 ;--
                                0000   242          .ALIGN  LONG
                                0000   243 EXE$INT58::                              ; These interrupts are other-processor-
                                0000   244 EXE$INT5C::                              ; specific and should never be
                                0000   245 EXE$INT60::                              ; seen on Nebula.
                                0000   246 EXE$UBAERR_INT::
                                0000   247 EXE$RH780_INT::
                          2C    11  0000   248          BRB     BAD_TYPE
                                0002   249
                                0002   250
                                0002   251          .ALIGN  LONG
                                0004   252 EXE$MCHK::                              ; Machine check handler.
                          02    DD  0004   253          PUSHL   #MCHK$M_MCK                 ; Mask signals machine check.
                       14 AE    DF  0006   254          PUSHAL  MC$L_PC+4(SP)               ; Push pointer to exception PC/PSL.
                    103F 8F     BB  0009   255          PUSHR   #^M<R0,R1,R2,R3,R4,R5,AP>   ; Working registers.
                5C  5E    24    C1  000D   256          ADDL3   #<9*4>,SP,AP                ; AP points to mchk log frame.
                                0011   257
                                0011   258          CASE    MC$L_TYPECODE(AP), -        ; Case on machine check code.
                                0011   259                  <MICRO_ERRORS, -            ; code  0
                                0011   260                  TB_PARITY, -                ;       1
                                0011   261                  BAD_TYPE, -                 ; code  2 should never be seen.
                                0011   262                  BAD_MEM_CSR, -              ;       3
                                0011   263                  NO_FAST_INT, -              ;       4
                                0011   264                  FPA_PARITY, -               ;       5
                                0011   265                  SPTE_READCHK, -             ;       6
                                0011   266                  RDATASUBS, -                ;       7
                                0011   267                  NX_MEM, -                   ;       8
                                0011   268                  UNALIGNED_IO, -             ;       9
                                0011   269                  UNK_IO_ADDR, -              ;       A
                                0011   270                  BAD_UB_ADDR>, -             ;       B
                                0011   271                  TYPE=B
                                002E   272
                                002E   273 BAD_TYPE:                                ; undefined exception
                       52   03    D0  002E   274          MOVL    #<MCHK$M_MCK! -             ; Type code for check for
                                0031   275                  MCHK$M_LOG>,R2              ; recovery block.
                          0163  30  0031   276          BSBW    LOG_MCHECK                  ; Log a machine check.
                                0034   277 ;
                                0034   278 ; Check to see if bad machine checks are occurring too rapidly.
                                0034   279 ;
                50 0008'CF    DE  0034   280          MOVAL   W^LAST_BADMCK,R0            ; Address of time stamp.
                   51   01    D0  0039   281          MOVL    #BADMCK_MIN,R1              ; Min allowable time between errors.
                          00E4  30  003C   282          BSBW    CHK_AND_RESUME              ; Try to resume.  Returns if
                                003F   283                                              ; unresumable opcode.
                                003F   284          BUG_CHECK  BADMCKCOD,FATAL          ; Bad machine check code.
```

```
                                                L 5
                        0043   286          .SBTTL  TRANSLATION BUFFER PARITY ERRORS
                        0043   287 ;++
                        0043   288 ; FUNCTIONAL DESCRIPTION:
                        0043   289 ;       Find and log the PTE in memory which is a correct copy of the
                        0043   290 ;       bad PTE in the translation buffer.
                        0043   291 ;       The microcode has already invalidated the translation buffer
                        0043   292 ;       entry that caused the parity error.
                        0043   293 ;       If TB errors are occurring too fast, bug check.
                        0043   294 ;       Else attempt to resume the faulting instruction.
                        0043   295 ;       If instruction not resumable, reflect exception/bugcheck.
                        0043   296 ;
                        0043   297 ; Machine check parameters:
                        0043   298 ;       MC$L_P1(AP): TB entry in error
                        0043   299 ;               <31>    - PTE valid bit
                        0043   300 ;               <30:27> - protection mask
                        0043   301 ;               <26>    - PTE modify bit
                        0043   302 ;               <25>    - TB valid bit
                        0043   303 ;               <23:0>  - PFN
                        0043   304 ;       MC$L_P2(AP): VA of reference whose PTE was in TB.
                        0043   305 ;       ************************************************************
                        0043   306 ;       *** NOTE *** This routine will look up the PTE corresponding to the
                        0043   307 ;       VA in MC$L_P2(AP) and subtitute the contents of the PTE for the VA
                        0043   308 ;       in the error log.
                        0043   309 ;       ************************************************************
                        0043   310 ;--
                        0043   311
                        0043   312 TB_PARITY:
     51 00000000'GF     D0   0043   313          MOVL    G^MMG$GL_SPTBASE,R1     ; R1 = address of system page table.
        52 0C AC        D0   004A   314          MOVL    MC$L_P2(AP),R2          ; R2 = VA of reference causing error.
     52 15 09           EF   004E   315          EXTZV   #VA$V_VPN,#VA$S_VPN, -  ; Get virtual page number.
           53                0052   316                  R2,R3                  ;
        06 52 1F           E1   0053   317          BBC     #VA$V_SYSTEM,R2, -      ; Branch if process virtual address.
                                0057   318                  P0P1SPACE              ;
                                0057   319 ;
                                0057   320 ; System virtual address.
                                0057   321 ;       System page tables are never paged.  Use VPN to get the address of
                                0057   322 ;       the PTE which maps the system virtual address.
                                0057   323 ;
                                0057   324 SYSTEMSPACE:
           53 6143              D0   0057   325          MOVL    (R1)[R3],R3             ; Get PTE to log from system page table.
                 31             11   005B   326          BRB     LOG_TBERR               ; Go log PTE.
                                005D   327
                                005D   328 ;
                                005D   329 ; P0 or P1 space address.
                                005D   330 ;       P0 and P1 page tables can be paged.  Use VPN to get the address of
                                005D   331 ;       the PTE which maps the P0 or P1 space address, then extract its VPN
                                005D   332 ;       to check that the PTE is resident (valid).
                                005D   333 ;
                                005D   334 P0P1SPACE:
     54 00000000'GF     D0   005D   335          MOVL    G^SCH$GL_CURPCB,R4      ; R4 = address of current process PCB.
        55 6C A4        D0   0064   336          MOVL    PCB$L_PHD(R4),R5        ; R5 = address of process header.
        08 52 1E        E1   0068   337          BBC     #VA$V_P1,R2,P0SPACE     ; Branch if P0 space.
                                006C   338 P1SPACE:
        53 00D0 D543     DE   006C   339          MOVAL   @PHD$L_P1BR(R5)[R3],R3  ; Get SVAPTE for P1 space VA.
                    06   11   0072   340          BRB     CHECK_PTE               ; Go check for valid PTE.
                                0074   341 P0SPACE:
        53 00C8 D543     DE   0074   342          MOVAL   @PHD$L_P0BR(R5)[R3],R3  ; Get SVAPTE for P0 space VA.
```

```
                                007A   343 ;
                                007A   344 ; Check for valid PTE.
                                007A   345 ;
                                007A   346 CHECK_PTE:
      53   15   09   EF          007A   347         EXTZV   #VA$V_VPN,#VA$S_VPN, -  ; Get VPN index into System Page Table.
                50              007E   348                 R3,R0                  ;
         51   6140   DE          007F   349         MOVAL   (R1)[R0],R1            ; Get address of SPTE for page table.
                61   D5          0083   350         TSTL    (R1)                   ; Is SPTE valid?
                05   14          0085   351         BGTR    PTE_NOT_RESIDENT       ; Branch if not.
           53   63   D0          0087   352         MOVL    (R3),R3                ; Get PTE.
                02   11          008A   353         BRB     LOG_TBERR              ; Go log PTE.
                                008C   354 ;
                                008C   355 ; The page table containing the PTE which matches the PTE in the translation
                                008C   356 ; buffer is not currently resident in memory.  Log a 0 as the memory PTE
                                008C   357 ; parameter.
                                008C   358 ;
                                008C   359 PTE_NOT_RESIDENT:
                53   D4          008C   360         CLRL    R3
                                008E   361
                                008E   362 ;
                                008E   363 ; Log Translation Buffer Parity error.  R3 contains the good PTE from memory.
                                008E   364 ;
                                008E   365 LOG_TBERR:
   0C   AC   53   D0          008E   366         MOVL    R3,MC$L_P2(AP)         ; Substitute PTE as P2 parameter.
      52   03   D0          0092   367         MOVL    #<MCHK$M_MCK! -        ; Type code for check for
                              0095   368                 MCHK$M_LOG>,R2         ; recovery block.
             00FF   30          0095   369         BSBW    LOG_MCHECK             ; Log the machine check.
                              0098   370
           50   1B   DB          0098   371         MFPR    #PR730$_TODR,R0        ; Get current time, in 10ms
                              009B   372                                        ; units.
52    50   0000'CF   C3          009B   373         SUBL3   W^EXE$GL_TB1OLD,R0,R2  ; How long since last error?
           0A   52   D1          00A1   374         CMPL    R2,#TB_THRESHOLD       ; Errors coming too fast?
                03   1A          00A4   375         BGTRU   10$                    ; No; continue.
             00DD   31          00A6   376         BRW     BUGCHECK               ; TB bad -- fatal error.
                              00A9   377 10$:
0004'CF   0000'CF   D0          00A9   378         MOVL    W^EXE$GL_TB1OLD, -     ; Save time of next-to-last
                              00B0   379                 W^EXE$GL_TB2OLD        ; TB error.
      0000'CF   50   D0          00B0   380         MOVL    R0,W^EXE$GL_TB1OLD     ; Save time of last TB error.
             007F   30          00B5   381         BSBW    RESUME                 ; Try to resume instruction.
                              00B8   382                                        ; Returns if unresumable opcode.
             0097   31          00B8   383         BRW     REFLECTCHK             ; Reflect exception/bugcheck.
```

```
                     00BB    385          .SBTTL  MICROCODE DETECTED ERRORS
                     00BB    386 ;++
                     00BB    387 ; The following machine checks are caused by microcode-detected
                     00BB    388 ; inconsistencies in the hardware.
                     00BB    389 ;
                     00BB    390 ; Machine check parameters:
                     00BB    391 ;       if MC$L_P1(AP) = 0: No other information available
                     00BB    392 ;       if MC$L_P1(AP) = 2: Unable to write back PTE<M> bit
                     00BB    393 ;       if MC$L_P1(AP) = 3: Bad 8085 interrupt
                     00BB    394 ;       MC$L_P2(AP) is always 0
                     00BB    395 ;--
                     00BB    396
                     00BB    397 MICRO_ERRORS:
                     00BB    398
                     00BB    399          .SBTTL  MEMORY REFERENCE ERRORS
                     00BB    400 ;++
                     00BB    401 ; These machine checks are caused by memory reference errors.
                     00BB    402 ;
                     00BB    403 ; Illegal format of memory CSR.
                     00BB    404 ;       MC$L_P1(AP): Virtual address referenced
                     00BB    405 ;       MC$L_P2(AP): Bad CSR value
                     00BB    406 ;
                     00BB    407 ; Unaligned or non-longword reference to I/O space.
                     00BB    408 ;       MC$L_P1(AP): Physical address referenced
                     00BB    409 ;       MC$L_P2(AP): 0
                     00BB    410 ;
                     00BB    411 ; Illegal UNIBUS reference.
                     00BB    412 ;       MC$L_P1(AP): physical address referenced
                     00BB    413 ;       MC$L_P2(AP): 0
                     00BB    414 ;
                     00BB    415 ; Hard memory error on SPTE read.
                     00BB    416 ;       MC$L_P1(AP): physical address of SPTE
                     00BB    417 ;       MC$L_P2(AP): error syndrome bits
                     00BB    418 ;--
                     00BB    419
                     00BB    420 BAD_MEM_CSR:                            ; Illegal format of memory CSR.
                     00BB    421 UNALIGNED_IO:                           ; Unaligned reference to I/O space, or
                     00BB    422                                         ; non-longword ref to I/O space.
                     00BB    423 BAD_UB_ADDR:                            ; Unaligned or longword ref
                     00BB    424                                         ; to UNIBUS space.
                     00BB    425 SPTE_READCHK:                           ; Hard memory error on SPTE read.
 52    03    D0      00BB    426          MOVL    #<MCHK$M_MCK! -        ; Type code for recovery block
                     00BE    427                  MCHK$M_LOG>,R2         ; check.
    00D6    30       00BE    428          BSBW    LOG_MCHECK             ; Log the machine check.
    008E    31       00C1    429          BRW     REFLECTCHK             ; Reflect exception/bugcheck,
                     00C4    430                                         ; based on current process mode.
```

```
                        00C4    432          .SBTTL  NON-EXISTENT MEMORY
                        00C4    433 ;++
                        00C4    434 ; Machine Checks due to non-existent memory or non-existent I/O space addresses
                        00C4    435 ; may have their own specific recovery block.  A recovery block may prevent
                        00C4    436 ; logging of an NXM machine check, and/or it may prevent bugchecking because
                        00C4    437 ; of an NXM machine check.
                        00C4    438 ;
                        00C4    439 ; Non-existent memory.
                        00C4    440 ;       MC$L_P1(AP): physical address referenced
                        00C4    441 ;       MC$L_P2(AP): 0
                        00C4    442 ;
                        00C4    443 ; Illegal I/O space address.
                        00C4    444 ;       MC$L_P1(AP): physical address referenced
                        00C4    445 ;       MC$L_P2(AP): 0
                        00C4    446 ;--
                        00C4    447 NX_MEM.                                     ; Reference to non-existent memory.
                        00C4    448 UNR_IO_ADDR:                                ; Unrecognized I/O space address.
     52    07    D0     00C4    449          MOVL    #<MCHK$M_LOG! -            ; Type code for checking for
                        00C7    450                  MCHK$M_MCK! -              ; recovery block.
                        00C7    451                  MCHK$M_NEXM>,R2            ;
        00CD    30      00C7    452          BSBW    LOG_MCHECK                 ; Log the machire check.
                        00CA    453
  FC AC    04    C8     00CA    454          BISL    #MCHK$M_NEXM, -            ; Indicate NXM in recovery mask on the
                        00CE    455                  MC$L_RECOVMSK(AP)          ; stack.
        0081    31      00CE    456          BRW     REFLECTCHK                 ; Reflect exception/bugcheck.
                        00D1    457
```

```
                                00D1   459          .SBTTL  ILLEGAL INTERRUPTS
                                00D1   460 ;++
                                00D1   461 ; Since this error occurs asynchronously, it doesn't make sense
                                00D1   462 ; to decide what action to take based on the current process (as
                                00D1   463 ; REFLECTCHK does).  Log the error and try to resume.
                                00D1   464 ;
                                00D1   465 ; Machine check parameters:
                                00D1   466 ;       MC$L_P1(AP): 0
                                00D1   467 ;       MC$L_P2(AP): 0
                                00D1   468 ;--
                                00D1   469
                                00D1   470 NO_FAST_INT:                         ; Fast interrupt without support.
          52    03      D0      00D1   471          MOVL    #<MCHK$M_MCK! -     ; Type code for recovery block
                                00D4   472                  MCHK$M_LOG>,R2      ; check.
             00C0       30      00D4   473          BSBW    LOG_MCHECK          ; Log the machine check.
     50  000C'CF       DE      00D7   474          MOVAL   W^LAST_BADINT,R0    ; Get address of bad int time stamp.
          51    01      D0      00DC   475          MOVL    #BADINT_MIN,R1      ; Min time between bad interrupts.
             0041       30      00DF   476          BSBW    CHK_AND_RESUME      ; If errors not occurring too rapidly,
                                00E2   477                                      ; try to resume; returns if attempt fails.
             00A1       31      00E2   478          BRW     BUGCHECK            ; Bugcheck.
```

```
                     00E5    480          .SBTTL  FPA PARITY
                     00E5    481 ;++
                     00E5    482 ; Floating Point Accelerator Parity Error
                     00E5    483 ;
                     00E5    484 ;       Retry the instruction.  The subroutine CHK_AND_RESUME checks to see if
                     00E5    485 ;       errors are recurring too rapidly.
                     00E5    486 ;
                     00E5    487 ; Machine check parameters:
                     00E5    488 ;       MC$L_P1(AP):
                     00E5    489 ;               <31:3>  - UNPREDICTABLE
                     00E5    490 ;               <2>     - Group 1 parity bit
                     00E5    491 ;               <1>     - Group 0 parity bit
                     00E5    492 ;               <0>     - Error summary bit
                     00E5    493 ;       MC$L_P2(AP): 0
                     00E5    494 ;--
                     00E5    495
                     00E5    496 FPA_PARITY:
     52    03    D0  00E5    497          MOVL    #<MCHK$M_MCK! -          ; Type code for recovery
                     00E8    498                  MCHK$M_LOG>,R2           ; block check.
        00AC    30   00E8    499          BSBW    LOG_MCHECK               ; Log the machine check.
50   0010'CF    DE   00EB    500          MOVAL   W^LAST_FPAPARITY,R0      ; Time of last FPA parity error.
     51    01    D0  00F0    501          MOVL    #FPAPARITY_MIN,R1        ; Error threshhold.
        002D    30   00F3    502          BSBW    CHK_AND_RESUME           ; If errors not occurring too rapidly,
                     00F6    503                                           ; try to resume.  Return if opcode is
                     00F6    504                                           ; unresumable.
        0059    31   00F6    505          BRW     REFLECTCHK               ; Reflect exception/bugcheck.
```

```
                        00F9   507          .SBTT   RDATASUBS
                        00F9   508
                        00F9   509 ;++
                        00F9   510 ;       UNCORRECTABLE ECC ERRORS -- READ DATA SUBSTITUTE
                        00F9   511 ;
                        00F9   512 ; Since this memory error could not be corrected by the hardware, the
                        00F9   513 ; physical memory page is unusable.  Mark the page bad, and reflect exception/
                        00F9   514 ; bugcheck.
                        00F9   515 ;
                        00F9   516 ; Machine check parameters:
                        00F9   517 ;       MC$L_P1(AP): physical address of reference
                        00F9   518 ;       MC$L_P2(AP): error syndrome bits
                        00F9   519 ;--
                        00F9   520          .ENABL  LSB
                        00F9   521 RDATASUBS:                               ; Read data substitute.
                        00F9   522 ;
                        00F9   523 ; Mark page bad.
                        00F9   524 ;
     50   08 AC   F7 8F   78  00F9   525          ASHL    #-9,MC$L_P1(AP),R0      ; Get PFN of bad memory location.
       00000000'GF    50   D1  00FF   526          CMPL    R0,G^MMG$GL_MAXPFN      ; PFN data base for this page?
                      0C   1A  0106   527          BGTRU   10$                     ; No, cannot mark page bad.
       51    00000000'GF   DE  0108   528          MOVAL   G^PFN$AB_TYPE, R1       ; Get address of PFN TYPE array.
            00 B140   20   88  010F   529          BISB2   #PFN$M_BADPAG, -        ; Mark page bad.
                        0114   530                  @(R1)[R0]                       ;
                        0114   531 10$:
                        0114   532 ;
                        0114   533 ; Log a machine check and a memory error.
                        0114   534 ;
                52    03   D0  0114   535          MOVL    #<MCHK$M_MCK! -         ; Type code for recovery block
                        0117   536                  MCHK$M_LOG>,R2          ; check.
                 007D   30  0117   537          BSBW    LOG_MCHECK              ; Log the machine check.
           53    08   9A  011A   538          MOVZBL  #EMB$K_HE,R3            ; Error type code for logging.
                 00B7   30  011D   539          BSBW    EXE$LOGMEM              ; Log a hard memory error.
                 002F   31  0120   540          BRW     REFLECTCHK              ; Reflect exception/bugcheck.
                        0123   541          .DSABL  LSB
```

```
                        0123   543          .SBTTL  * EXITS FROM MACHINE CHECK ROUTINES *
                        0123   544          .SBTTL  CHK_AND_RESUME
                        0123   545 ;++
                        0123   546 ; CHK_AND_RESUME
                        0123   547 ;
                        0123   548 ; FUNCTIONAL DESCRIPTION:
                        0123   549 ;       Called to check time that this error last occurred.
                        0123   550 ;       CHK_AND_RESUME has three possible exit paths:
                        0123   551 ;               (1) If errors are occurring too rapidly, BUGCHECK.
                        0123   552 ;       Else fall through to RESUME:
                        0123   553 ;               (2) If opcode is unresumable, RSB.
                        0123   554 ;               (3) Else resume: clear stack and REI to retry the instruction.
                        0123   555 ; INPUTS:
                        0123   556 ;       R0: Address of longword which contains time error last occurred.
                        0123   557 ;       R1: Minimum time that must have elapsed since the last error.
                        0123   558 ;
                        0123   559 ; IMPLICIT INPUTS:
                        0123   560 ;        PR730$_TODR
                        0123   561 ;
                        0123   562 ; OUTPUTS:
                        0123   563 ;       The longword pointed to by R0 is updated with the time of this error.
                        0123   564 ;--
                        0123   565 CHK_AND_RESUME:
     52    1B    DB     0123   566          MFPR    #PR730$_TODR,R2         ; Get current time in R2.
53   52    60    C3     0126   567          SUBL3   (R0),R2,R3              ; How long since last error?
     51    53    D1     012A   568          CMPL    R3,R1                   ; Compare against minimum threshhold.
           05    1A     012D   569          BGTRU   10$                     ; Br if enough time has elapsed.
           8E    D5     012F   570          TSTL    (SP)+                   ; Else clear return address from stack.
         0052    31     0131   571          BRW     BUGCHECK                ; Errors recurring too fast; bugcheck.
                        0134   572 10$:
     60    52    D0     0134   573          MOVL    R2,(R0)                 ; Save time of latest error.
                        0137   574                                          ; Fall through to RESUME.
```

```
                                0137   576          .SBTTL  RESUME
                                0137   577 ;++
                                0137   578 ; RESUME
                                0137   579 ;
                                0137   580 ; FUNCTIONAL DESCRIPTION:
                                0137   581 ;       Try to resume faulting instruction.
                                0137   582 ;       Instruction can be resumed if its corresponding bit in RESUME_TABLE
                                0137   583 ;       is set.  Resumable instructions have the common characteristic that
                                0137   584 ;       they can be guaranteed to re-execute correctly.
                                0137   585 ;
                                0137   586 ; CALLING SEQUENCE:
                                0137   587 ;       RESUME is called as a subroutine, or entered as the back-end of the
                                0137   588 ;       CHK_AND_RESUME subroutine. It only returns to its caller,
                                0137   589 ;       however, if instruction could not be resumed.  If it was successful,
                                0137   590 ;       it REIs to the resumable instruction.
                                0137   591 ;
                                0137   592 ; IMPLICIT INPUTS:
                                0137   593 ;       MC$L_PC(AP): the address of the faulting instruction
                                0137   594 ;       RESUME_TABLE: a table that maps one bit to each VAX-11 opcode
                                0137   595 ;--
                                0137   596
                                0137   597 RESUME:
     55    10 BC    9A          0137   598          MOVZBL  @MC$L_PC(AP),R5         ; Get faulting opcode.
01 0022'CF   55    E0           013B   599          BBS     R5,W^RESUME_TABLE,10$   ; Branch if resumable.
                   05           0141   600          RSB                             ; Unable to resume; return to
                                0142   601                                          ; caller.
                                0142   602 ;
                                0142   603 ; Instruction can be resumed.  Retry the opcode.
                                0142   604 ;
                                0142   605 10$:
          8E       D5          0142   606          TSTL    (SP)+                   ; Pop return addr from stack.
  103F  8F       BA           0144   607          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ; Restore registers.
  5E    08       C0           0148   608          ADDL    #<2*4>,SP               ; Remove inputs for recovery blk check.
  5E    8E       C0           014B   609          ADDL    (SP)+,SP                ; Clear mck log from stack.
  26    00       DA           014E   610          MTPR    #0,#PR730$_MCESR        ; Clear "mcheck in progress" flag.
                   02           0151   611          REI                             ; Retry faulting instruction.
```

```
                                0152   613          .SBTTL  REFLCTCHK
                                0152   614 ;++
                                0152   615 ;       REFLECT MACHINE CHECK TO USER
                                0152   616 ;
                                0152   617 ; This code is entered if the machine check was fatal.  It determines
                                0152   618 ; if it was just fatal to the process which caused it (current process
                                0152   619 ; is in USER or SUPER mode), or if it was fatal to the entire system
                                0152   620 ; (current process is in EXEC or KERNEL mode).
                                0152   621 ;
                                0152   622 ; If current process is in USER or SUPER mode,
                                0152   623 ;       set up an exception on user's stack and REI to it
                                0152   624 ; If current process is in EXEC or KERNEL mode,
                                0152   625 ;       issue a fatal bugcheck.
                                0152   626 ; CALLING SEQUENCE:
                                0152   627 ;       BRB/W -- NOTHING EXTRA CAN BE ON THE STACK!!
                                0152   628 ;
                                0152   629 ; STACK CONTENTS:
                                0152   630 ;       00(SP): saved R0,R1,R2,R3,R4,R5,AP
                                0152   631 ;       1C(SP): 2 longword inputs for recovery block check
                                0152   632 ;       24(SP): (also AP) machine check log -- 1st longword is a byte count.
                                0152   633 ;--
                                0152   634 REFLECTCHK:                                ; Reflect exception according
                                0152   635                                           ; to current access mode.
03 14 AC    19    E0            0152   636          BBS     #PSL$V_CURMOD+1, -       ; Branch if USER or SUPER.
                                0157   637                  MC$L_PSL(AP),10$         ;
                        002C    31    0157   638          BRW     BUGCHECK                 ; EXEC or KERNEL; bugcheck.
                                015A   639
                                015A   640 10$:                                       ; SUPER or USER; exception.
                    50    00    DB    015A   641          MFPR    #PR$_KSP,R0              ; Get kernel stack pointer.
                70    10 AC     7D    015D   642          MOVQ    MC$L_PC(AP),-(R0)        ; Push PC,PSL on kernel stack.
                    00    50    DA    0161   643          MTPR    R0,#PR$_KSP              ; Replace new kernel stack ptr.
                    103F  8F    BA    0164   644          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ; Restore registers.
                    5E    08    C0    0168   645          ADDL    #<2*4>,SP                ; Pop inputs for recovery block check.
                    5E    8E    C0    016B   646          ADDL    (SP)+,SP                 ; Pop mck log from stack.
                                016E   647 ;
                                016E   648 ; Set up an exception stack for current process.
                                016E   649 ; The faulting PC,PSL pair are still on the interrupt stack.  Alter
                                016E   650 ; them to look as if an exception has occurred.
                                016E   651 ;
        6E    00000000'GF     9E    016E   652          MOVAB   G^EXE$MCHECK,(SP)        ; Replace exception PC.
04 AE   04 AE     02    18    EF    0175   653          EXTZV   #PSL$V_CURMOD, -         ; Zero exception PSL, except
                                017C   654                  #PSL$S_CURMOD, -         ; for current access mode.
                                017C   655                  4(SP),4(SP)
     04 AE   04 AE     16    9C    017C   656          ROTL    #PSL$V_PRVMOD, -         ; Create a PSL of current mode
                                0182   657                  4(SP),4(SP)              ; kernel, correct previous
                                0182   658                                           ; mode, and IPL 0.
                    26    00    DA    0182   659          MTPR    #0,#PR730$_MCESR         ; Clear "mcheck in progress" flag.
                                02    0185   660          REI                              ; Go to exception handler.
                                0186   661
```

```
                     0186   663          .SBTTL  BUGCHECK
                     0186   664 ;++
                     0186   665 ; If user has declared a recovery block, transfer control to it.
                     0186   666 ; Else issue a fatal bugcheck.
                     0186   667 ;
                     0186   668 ; CALLING SEQUENCE:
                     0186   669 ;       BRB/W -- NOTHING EXTRA CAN BE ON THE STACK!!!
                     0186   670 ;
                     0186   671 ; STACK CONTENTS ON ENTRY:
                     0186   672 ;       00(SP): saved R0,R1,R2,R3,R4,R5,AP
                     0186   673 ;       1C(SP): 2 longword inputs for recovery block check
                     0186   674 ;       24(SP): (also AP) machine check log
                     0186   675 ;--
                     0186   676 BUGCHECK:
          103F 8F    BA  0186   677          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ; Restore registers.
                     018A   678 ;
                     018A   679 ; A fatal bugcheck is now inevitable unless a user has declared a machine
                     018A   680 ; check recovery block.
                     018A   681 ;
                     018A   682
             26 00   DA  018A   683          MTPR    #0,#PR730$_MCESR        ; ******* temporary!! **********
00000000'GF   16   018D   684          JSB     G^EXE$MCHK_BUGCHK       ; If return, no recovery block.
                     0193   685          BUG_CHECK -                     ; Issue fatal bugcheck.
                     0193   686                  MACHINECHK,FATAL
```

```
                     0197   688          .SBTTL  * LOGGING ROUTINES FOR MACHINE CHECKS *
                     0197   689 ;++
                     0197   690 ; LOG_MCHECK -- format inputs to LOGGER
                     0197   691 ;
                     0197   692 ; INPUTS:
                     0197   693 ;       R2: a mask which specifies the type of error (hence, the type
                     0197   694 ;               of recovery block to check for)
                     0197   695 ;
                     0197   696 ; IMPLICIT INPUTS:
                     0197   697 ;       (AP): points to machine check log on stack
                     0197   698 ;
                     0197   699 ; OUTPUTS:
                     0197   700 ;       Error is formatted and logged in system error log.
                     0197   701 ;--
                     0197   702
                     0197   703 LOG_MCHECK:
                     0197   704 ;
                     0197   705 ; Test if a machine check recovery block that specifies no error
                     0197   706 ; logging is in effect.
                     0197   707 ;
  51    10 AC    DE  0197   708          MOVAL   MC$L_PC(AP),R1          ; R1 points to PC,PSL of abort.
00000000'GF    16  019B   709          JSB     G^EXE$MCHK_TEST         ; Logging inhibited?
      01 50    E9  01A1   710          BLBC    R0,10$                  ; Branch if no.
               05  01A4   711          RSB                             ; Else return.
                   01A5   712 10$:                                     ; Set up inputs to LOGGER.
00000000'GF    D6  01A5   713          INCL    G^EXE$GL_MCHKERRS       ; Bump machine check error count
      53    02   D0  01AB   714          MOVL    #EMB$K_MC,R3            ; Use Machine Check type code.
54    08    6C   C1  01AE   715          ADDL3   MC$L_BYTCNT(AP), -      ; Size of data to log: machine check
                   01B2   716                  #<2*4>,R4               ; stack + PC,PSL.
  55    04 AC    DE  01B2   717          MOVAL   MC$L_TYPECODE(AP),R5    ; Address of data to log.
                   01B6   718 ;++
                   01B6   719 ; LOGGER - release error data to error logger
                   01B6   720 ;
                   01B6   721 ; INPUTS:
                   01B6   722 ;       R3: error type
                   01B6   723 ;       R4: number of bytes to log
                   01B6   724 ;       R5: address of information to be logged
                   01B6   725 ;
                   01B6   726 ; OUTPUTS:
                   01B6   727 ;       Error log is inserted into error log buffer.
                   01B6   728 ;       If no error log buffer, return with error status in R0.
                   01B6   729 ;       R0-R5 destroyed.
                   01B6   730 ;--
                   01B6   731
                   01B6   732 LOGGER:
  51    54    10   C1  01B6   733          ADDL3   #EMB$B_MC_SUMCOD,R4,R1  ; Add space for log header.
   00000000'GF    16  01BA   734          JSB     G^ERL$ALLOCEMB          ; Get error logging buffer.
         13 50    E9  01C0   735          BLBC    R0, 20$                 ; Br if failed to get buffer.
            52    DD  01C3   736          PUSHL   R2                      ; Save buffer addr on stack.
    04 A2   53    B0  01C5   737          MOVW    R3,EMB$W_MC_ENTRY(R2)   ; Set entry type.
10 A2   65    54    28  01C9   738          MOVC3   R4,(R5),EMB$B_MC_SUMCOD(R2)     ; transfer info to log
            04    BA  01CE   739          POPR    #^M<R2>                 ; Retreive buffer address.
   00000000'GF    16  01D0   740          JSB     G^ERL$RELEASEMB         ; Give buffer to logger.
                   01D6   741 20$:
               05  01D6   742          RSB
```

```
                          01D7   744 ;++
                          01D7   745 ; EXE$LOGMEM -- log memory Control and Status Registers
                          01D7   746 ;
                          01D7   747 ; FUNCTIONAL DESCRIPTION:
                          01D7   748 ;       EXE$LOGMEM is called to log memory CSRs.  If called with R3 =
                          01D7   749 ;       EMB$K_SE (log a soft memory error), look at the memory CSRs to see
                          01D7   750 ;       if the CRD (soft error) bit was set; if not, don't log the CSRs.
                          01D7   751 ;       If "too many" CRD errors have been logged recently, also don't log
                          01D7   752 ;       the CSRs.
                          01D7   753 ;
                          01D7   754 ;       NOTE: The error bits in Nebula's memory CSRs are read-to-clear.
                          01D7   755 ;
                          01D7   756 ; INPUTS:
                          01D7   757 ;       R3: errorlog type code
                          01D7   758 ;
                          01D7   759 ; IMPLICIT INPUTS:
                          01D7   760 ;       The first longword of @MMG$GL_SBICONF contains the virtual address of
                          01D7   761 ;       the first memory controller CSR.
                          01D7   762 ;
                          01D7   763 ; OUTPUTS:
                          01D7   764 ;       Create entry in errorlog buffer containing the three memory controller
                          01D7   765 ;       Control and Status Registers.
                          01D7   766 ;       CRD error logging may be disabled.
                          01D7   767 ;       R0: low bit signals success/failure
                          01D7   768 ;       R1-R5 destroyed.
                          01D7   769 ;--
                          01D7   770 EXE$LOGMEM:
51   00000000'GF    D0    01D7   771         MOVL    G^MMG$GL_SBICONF,R1      ; Get address of SBICONF array.
       51   61     D0    01DE   772         MOVL    (R1),R1                  ; Get VA of 1st memory controller CSR.
                          01E1   773         DSBINT  #31,R0                   ; Block out all interrupts.
         08 A1     DD    01E7   774         PUSHL   MEM$L_CSR2(R1)           ; Push memory controller registers
         04 A1     DD    01EA   775         PUSHL   MEM$L_CSR1(R1)           ; on the stack.  NOTE: memory error
            61     DD    01ED   776         PUSHL   MEM$L_CSR0(R1)           ; bits are read-to-clear.
                          01EF   777         ENBINT  R0                       ; Restore IPL.
      06   53     91    01F2   778         CMPB    R3,#EMB$K_SE             ; Looking for CRD errors?
            18     12    01F5   779         BNEQ    LOG_CSRS                 ; No.  Unconditionally log CSRs.
22 04 AE    1E     E1    01F7   780         BBC     #CSR1$V_CRD, -           ; If CRD error bit not set,
                          01FC   781                 MEM$L_CSR1(SP),NOLOG     ; do not log soft memory error.
                          01FC   782 ;
                          01FC   783 ; A CRD error occurred.  Count it, and if we haven't logged a lot of CRD errors
                          01FC   784 ; recently, log it.
                          01FC   785 ;
 00000000'GF    D6    01FC   786         INCL    G^EXE$GL_MEMERRS         ; Bump memory error counter
     001C'CF    96    0202   787         INCB    W^ECC$AB_MEMERR          ; Count # of CRD errors LOGGED recently.
06   001C'CF    91    0206   788         CMPB    W^ECC$AB_MEMERR, -       ; Already logged enough CRD errors
                          020B   789                 #CRDLOGMAX           ; recently?
            11     1A    020B   790         BGTRU   NOLOG                    ; Yes.  Skip the logging.
            06     11    020D   791         BRB     LOGNOCNT                 ; Don't bump count again.
                          020F   792
                          020F   793 LOG_CSRS:
 00000000'GF    D6    020F   794         INCL    G^EXE$GL_MEMERRS         ; Bump memory error counter
                          0215   795 LOGNOCNT:
      54   0C     9A    0215   796         MOVZBL  #<4*3>,R4                ; Size of errorlog buffer.
      55   5E     D0    0218   797         MOVL    SP,R5                    ; Point to error log buffer.
         FF98     30    021B   798         BSBW    LOGGER                   ; Log memory CSRs.
                          021E   799 NOLOG:
      5E   0C     C0    021E   800         ADDL    #<4*3>,SP                ; Pop CSRs off stack.
```

```
05  0221    801          RSB
```

```
                                0222   803          .SBTTL  ECC$REENABLE -- TIMER CALL FROM SYSTEM CLOCK ROUTINE
                                0222   804 ;++
                                0222   805 ; ECC$REENABLE -- TIMER CALL FROM SYSTEM CLOCK ROUTINE
                                0222   806 ;
                                0222   807 ; FUNCTIONAL DESCRIPTION:
                                0222   808 ;       This routine periodically scans memory controller CSRs for
                                0222   809 ;       CRD errors.  CRD errors are normally reported by interrupt,
                                0222   810 ;       but even when CRD interrupts are turned off this routine will
                                0222   811 ;       still scan memory controller CSRs periodically, to report a
                                0222   812 ;       representative sample of CRD errors.
                                0222   813 ;
                                0222   814 ;       Also, check if it is time to reenable CRD interrupts.
                                0222   815 ;       CRD interrupts may have been disabled by the CRD interrupt handler,
                                0222   816 ;       EXE$LOGCRD, if it determines that "too many" interrupts are being
                                0222   817 ;       received.
                                0222   818 ;
                                0222   819 ; INPUTS:
                                0222   820 ;       NONE
                                0222   821 ;
                                0222   822 ; IMPLICIT OUTPUTS:
                                0222   823 ;       If a CRD error is found the memory controller CSRs will be logged.
                                0222   824 ;       CRD (Corrected Read Data) interrupts may be enabled for all
                                0222   825 ;       memory controllers.
                                0222   826 ;--
                                0222   827
                                0222   828 ECC$REENABLE::
        001A'CF    B7           0222   829          DECW    W^ECC$GW_CRDWATCH       ; Time to scan for CRD errors?
             0E    14           0226   830          BGTR    REENAB_SCAN             ; Branch if no.
             3F    BB           0228   831          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save working registers.
    001A'CF    3C    B0         022A   832          MOVW    #CRDWATCHTIME, -        ; Reset scan timer.
                                022F   833                  W^ECC$GW_CRDWATCH       ;
          53   06    3C         022F   834          MOVZWL  #EMB$K_SE,R3            ; Test for CRD error,
               A3    10         0232   835          BSBB    EXE$LOGMEM              ; and log memory CSRs if found.
               3F    BA         0234   836          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                                0236   837 ;
                                0236   838 ; If any CRD errors were found, the memory controller CSRs were logged.
                                0236   839 ; Now check to see if its time to enable CRD interrupts.  CRD interrupts are
                                0236   840 ; enabled periodically, whether or not they were disabled by EXE$LOGCRD.
                                0236   841 ;
                                0236   842 REENAB_SCAN:
        0018'CF    B7           0236   843          DECW    W^ECC$GW_REENAB         ; Has reenable time elapsed?
             2D    14           023A   844          BGTR    20$                     ; Branch if no.
  0018'CF    0384 8F    B0      023C   845          MOVW    #REENABTIME, -          ; Yes.  Reset reenable timer.
                                0243   846                  W^ECC$GW_REENAB         ;
        001C'CF    94           0243   847          CLRB    W^ECC$AB_MEMERR         ; Reset CRD log counter.
        001D'CF    94           0247   848          CLRB    W^ECC$AB_CRDCNT         ; Reset CRD interrupt counter.
 00000000'GF    00'  E1         024B   849          BBC     S^#EXE$V_CRDENABL, -    ; Br if SYSGEN parameter does
               16                0252   850                  G^EXE$GL_FLAGS,20$      ; not specify CRD interrupts.
               51   DD           0253   851          PUSHL   R1                      ; Save working register.
  51   00000000'GF    D0         0255   852          MOVL    G^MMG$GL_SBICONF,R1     ; Get address of CONFREG array.
            51   61    D0         025C   853          MOVL    (R1),R1                 ; Get VA of 1st memory controller CSR.
04 A1    10000000 8F    C8      025F   854          BISL    #CSR1$M_CRDENAB, -      ; Reenable CRD interrupts.
                                0267   855                  MEM$L_CSR1(R1)          ;
               02   BA           0267   856          POPR    #^M<R1>
                     05         0269   857 20$:     RSB                             ; Return.
```

```
                                026A   859          .SBTTL  EXE$LOGCRD -- CORRECTED MEMORY DATA INTERRUPTS
                                026A   860 ;++
                                026A   861 ; This routine is called when a CRD -- Corrected Read Data -- interrupt
                                026A   862 ; is received from a memory controller., Log all interrupts, and
                                026A   863 ; continue.  If too many CRD interrupts are logged, turn off CRD interrupts.
                                026A   864 ;--
                                026A   865          .ALIGN  LONG
                                026C   866 EXE$LOGCRD::
                                026C   867 EXE$INT54::
                          3F    BB    026C   868          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save working registers.
                       53   06    3C    026E   869          MOVZWL  #EMB$K_SE,R3            ; Soft memory error.
                          FF63    30    0271   870          BSBW    EXE$LOGMEM              ; Log a memory error.
                      001E'CF    D6    0274   871          INCL    W^MMG$L_CRDCNT          ; Count total CRD interrupts.
                      001D'CF    96    0278   872          INCB    W^ECC$AB_CRDCNT         ; Count recent CRD interrupts.
                03    001D'CF    91    027C   873          CMPB    W^ECC$AB_CRDCNT, -      ; More than enough CRD interrupts
                                0281   874                  #CRDINTMAX              ; lately?
                          12    1B    0281   875          BLEQU   10$                     ; No, do not disable CRD interrupts.
              51    00000000'GF    D0    0283   876          MOVL    G^MMG$GL_SBICONF,R1     ; Get address of CONFREG array.
                    51    61    D0    028A   877          MOVL    (R1),R1                 ; Get VA of 1st memory controller CSR.
        04 A1    10000000 8F    CA    028D   878          BICL    #CSR1$M_CRDENAB, -      ; Disable CRD interrupts.
                                0295   879                  MEM$L_CSR1(R1)
                                0295   880 10$:
                          3F    BA    0295   881          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                          02    0297   882          REI                             ; Return from interrupt.
                                0298   883
                                0298   884
                                0298   885          .SBTTL  END OF MODULE
                                0298   886          .END
```

```
BADINT_MIN                =  00000001
BADMCK_MIN                =  00000001
BAD_MEM_CSR                  000000BB  R        03
BAD_TYPE                     0000002E  R        03
BAD_UB_ADDR                  000000BB  R        03
BUG$_BADMCKCOD               ********    X      03
BUG$_MACHINECHK              ********    X      03
BUGCHECK                     00000186  R        03
CHECK_PTE                    0000007A  R        03
CHK_AND_RESUME               00000123  R        03
CRDINTMAX                 =  00000003
CRDLOGMAX                 =  00000006
CRDWATCHTIME              =  0000003C
CSR1$M_CRDENAB            =  10000000
CSR1$V_CRD                =  0000001E
ECC$AB_CRDCNT                0000001D  R        02
ECC$AB_MEMERR                0000001C  R        02
ECC$GW_CRDWATCH              0000001A  R        02
ECC$GW_REENAB                00000C18  R        02
ECC$REENABLE                 00000222  RG       03
EMB$B_MC_SUMCOD           =  00000010
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$GL_FLAGS                 ********    X      03
EXE$GL_MCHKERRS              ********    X      03
EXE$GL_MEMERRS               ********    X      03
EXE$GL_TB1OLD                00000000  RG       02
EXE$GL_TB2OLD                00000004  RG       02
EXE$INT54                    0000026C  RG       03
EXE$INT58                    00000000  RG       03
EXE$INT5C                    00000000  RG       03
EXE$INT60                    00000000  RG       03
EXE$LOGCRD                   0000026C  RG       03
EXE$LOGMEM                   000001D7  R        03
EXE$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$RH780_INT                00000000  RG       03
EXE$UBAERR_INT               00000000  RG       03
EXE$V_CRDENABL               ********    X      03
FPAPARITY_MIN             =  00000001
FPA_PARITY                   000000E5  R        03
LAST_BADINT                  0000000C  R        02
LAST_BADMCK                  00000008  R        02
LAST_FPAPARITY               00000010  R        02
LAST_RDS                     00000014  R        02
LOGGER                       000001B6  R        03
LOGNOCNT                     00000215  R        03
LOG_CSRS                     0000020F  R        03
LOG_MCHECK                   00000197  R        03
LOG_TBERR                    0000008E  R        03
MC$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
MEM$L_CSR0                   00000000
MEM$L_CSR1                   00000004
MEM$L_CSR2                   00000008
MICRO_ERRORS                 000000BB  R        03
MMG$GL_MAXPFN                ********    X      03
MMG$GL_SBICONF               ********    X      03
MMG$GL_SPTBASE               ********    X      03
MMG$L_CRDCNT                 0000C01E  R        02
NOLOG                        0000021E  R        03
NO_FAST_INT                  000000D1  R        03
NXM_EM                       000000C4  R        03
POP1SPACE                    0000005D  R        03
POSPACE                      00000074  R        03
P1SPACE                      0000006C  R        03
PCB$L_PHD                 =  0000006C
PFN$AB_TYPE                  ********    X      03
PFN$M_BADPAG              =  00000020
PHD$L_P0BR                =  000000C8
PHD$L_P1BR                =  000000D0
PR$_IPL                   =  00000012
PR$_KSP                   =  00000000
PR730$_MCESR              =  00000026
PR730$_TODR               =  0000001B
PSL$S_CURMOD              =  00000002
PSL$V_CURMOD              =  00000018
PSL$V_PRVMOD              =  00000016
PTE_NOT_RESIDENT             0000008C  R        03
RDATASUBS                    000000F9  R        03
RDS_MIN                   =  00000001
REENABTIME                =  00000384
REENAB_SCAN                  00000236  R        03
REFLECTCHK                   00000152  R        03
RESUME                       00000137  R        03
RESUME_TABLE                 00000022  R        02
SCH$GL_CURPCB                ********      X    03
SPTE_READCHK                 000000BB  R        03
SYSTEMSPACE                  00000057  R        03
TB_PARITY                    00000043  R        03
TB_THRESHOLD              =  0000000A
UNALIGNED_IO                 000000BB  R        03
UNK_IO_ADDR                  000000C4  R        03
VA$S_VPN                  =  00000015
VA$V_P1                   =  0000001E
VA$V_SYSTEM               =  0000001F
VA$V_VPN                  =  00000009
```

### 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 | 00000040 ( 64.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD |
| WIONONPAGED | 00000298 ( 664.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD |

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 29 | 00:00:00.06 | 00:00:01.46 |
| Command processing | 121 | 00:00:00.46 | 00:00:03.03 |
| Pass 1 | 299 | 00:00:06.53 | 00:00:22.57 |
| Symbol table sort | 0 | 00:00:00.85 | 00:00:02.20 |
| Pass 2 | 166 | 00:00:01.73 | 00:00:07.71 |
| Symbol table output | 13 | 00:00:00.07 | 00:00:00.26 |
| 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 | 633 | 00:00:09.72 | 00:00:37.25 |

The working set limit was 1500 pages.
53475 bytes (105 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 843 non-local and 11 local symbols.
886 source lines were read in Pass 1, producing 18 object records in Pass 2.
32 pages of virtual memory were used to define 31 macros.

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| _$255$DUA28:[SYSLOA.OBJ]MC.MLB;1 | 5 |
| _$255$DUA28:[SYS.OBJ]LIB.MLB;1 | 17 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 6 |
| TOTALS (all libraries) | 28 |

958 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:MCHECK730/OBJ=OBJ$:MCHECK730 MSRC$:MCHECK730/UPDATE=(ENH$:MCHECK730)+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