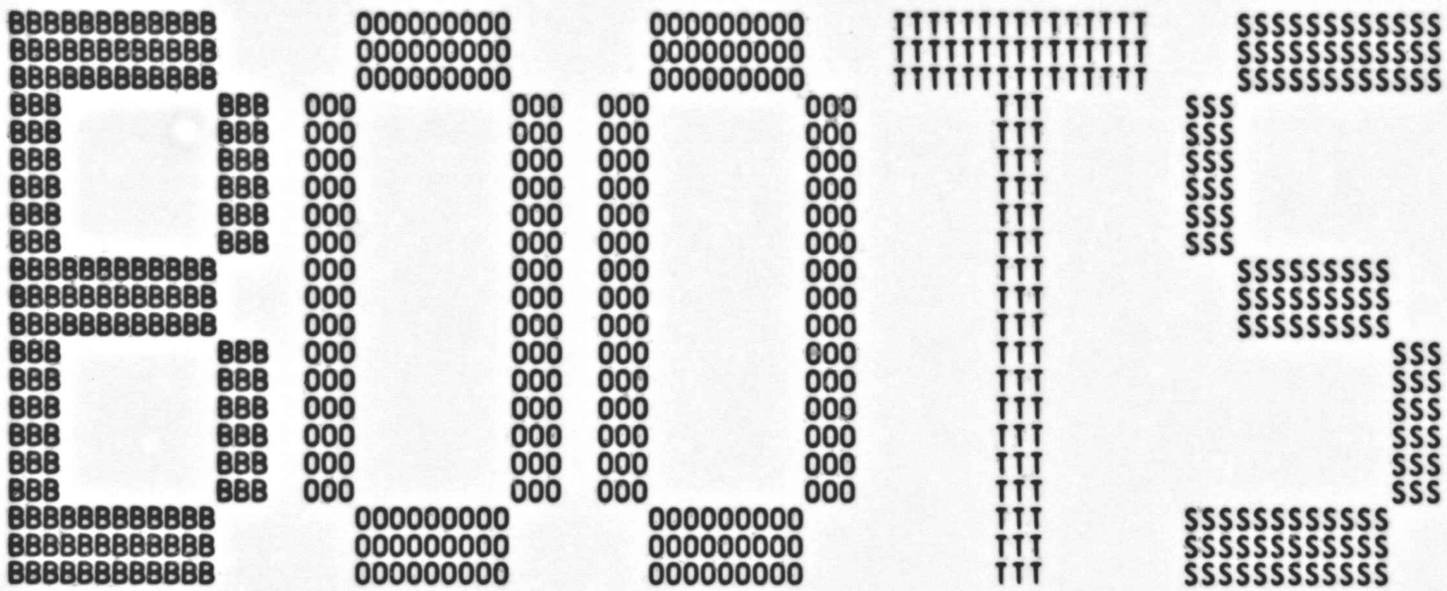
BBBBBBBBBBB OOOOOOOOO OOOOOOOOO TTTTTTTTTTTTTTT SSSSSSSSSSS
BBB BBB OOO OOO OOO OOO TTT SSS
BBBBBBBBBBB OOO OOO OOO OOO TTT SSSSSSSSS
BBB BBB OOO OOO OOO OOO TTT SSS
BBBBBBBBBBB OOOOOOOOO OOOOOOOOO TTT SSSSSSSSSSS

```
BBBBBBBB    TTTTTTTTTT  MM      MM  EEEEEEEEEE  MM      MM  77777777      888888      000000
BBBBBBBB    TTTTTTTTTT  MM      MM  EEEEEEEEEE  MM      MM  77777777      888888      000000
BB      BB      TT      MMMM  MMMM  EE          MMMM  MMMM          77  88      88  00      00
BB      BB      TT      MMMM  MMMM  EE          MMMM  MMMM          77  88      88  00      00
BB      BB      TT      MM  MM  MM  EE          MM  MM  MM          77  88      88  00    0000
BB      BB      TT      MM  MM  MM  EE          MM  MM  MM          77  88      88  00    0000
BBBBBBBB        TT      MM      MM  EEEEEEEE    MM      MM        77    888888    00  00  00
BBBBBBBB        TT      MM      MM  EEEEEEEE    MM      MM        77    888888    00  00  00
BB      BB      TT      MM      MM  EE          MM      MM      77      88      88  0000    00
BB      BB      TT      MM      MM  EE          MM      MM      77      88      88  0000    00
BB      BB      TT      MM      MM  EE          MM      MM    77        88      88  00      00
BB      BB      TT      MM      MM  EE          MM      MM    77        88      88  00      00
BBBBBBBB        TT      MM      MM  EEEEEEEEEE  MM      MM  77          888888      000000
BBBBBBBB        TT      MM      MM  EEEEEEEEEE  MM      MM  77          888888      000000

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  BTMEM780 - Configure and Test 11/780 Memory
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:
0000    31 ;
0000    32 ;       Linked with VMB.EXE - part of the
0000    33 ;       bootstrap module for VAX 11/780 hardware.
0000    34 ;
0000    35 ; ENVIRONMENT:
0000    36 ;
0000    37 ;       Runs at IPL 31, kernel mode, memory management is OFF, IS=1
0000    38 ;       (running on interrupt stack), and code must be PIC.
0000    39 ;
0000    40 ;
0000    41 ; FUNCTIONAL DESCRIPTION:
0000    42 ;
0000    43 ;       This routine is 11/780 specific and
0000    44 ;       determines how many memory controllers are on the system,
0000    45 ;       where they are, how much memory they control, which pages
0000    46 ;       of that memory are present (and good). Then the routines
0000    47 ;       set bits in the PFN bitmap to represent each present (and
0000    48 ;       good) page of memory.
0000    49 ;
0000    50 ;       As a side effect, the routines store the type of adapter located
0000    51 ;       at each bus slot in the RPB.
0000    52 ;
0000    53 ; INPUTS:
0000    54 ;
0000    55 ;       R5      - address of 1st RPB configuration code field
0000    56 ;       R7      - address of the SCB
0000    57 ;       R11     - address of the RPB
```

```
0000     58 ;
0000     59 ; IMPLICIT INPUTS:
0000     60 ;
0000     61 ;       The positions on the 11/780 system bus are called NEXUSes, and
0000     62 ;       are identified by TR numbers 0-15. Although conventions place
0000     63 ;       memory controllers at fixed TR numbers, the software must
0000     64 ;       identify the adapter code in each NEXUS device's configuration
0000     65 ;       register to decide what the adapter or controller is.
0000     66 ;
0000     67 ;       If a NEXUS has a readable configuration register, the register
0000     68 ;       contains the adapter code in the lower byte. If the
0000     69 ;       configuration register is of a memory controller, the adapter
0000     70 ;       code matches an entry in the table MEMTYP (see table in code
0000     71 ;       below).
0000     72 ;
0000     73 ;       Memory controller registers also contain
0000     74 ;
0000     75 ;               the size of the memory in either 128-page or 2048-page units
0000     76 ;               the starting page number / 128
0000     77 ;               a bit indicating if the memory is externally interleaved or not
0000     78 ;
0000     79 ; OUTPUTS:
0000     80 ;
0000     81 ;       R7, R8, R11, and SP are preserved
0000     82 ;       All others (including AP and FP) are altered
0000     83 ;
0000     84 ; IMPLICIT OUTPUTS:
0000     85 ;
0000     86 ;       The PFN bitmap is modified to describe all of physical memory.
0000     87 ;
0000     88 ;       RPB$L_PFNCNT stores the number of pages of physical memory.
0000     89 ;
0000     90 ;       All single parity errors in memory are cleared.
0000     91 ;
0000     92 ;       RPB$B_CONFREG describes each NEXUS on the system bus with an
0000     93 ;       adapter type code.
0000     94 ;
0000     95 ; AUTHOR:
0000     96 ;
0000     97 ;       C. A. Samuelson, creation date 24-April-1981
0000     98 ;
0000     99 ; REVISION HISTORY:
0000    100 ;
0000    101 ;       V03-004 RNG0004         Rod Gamache             07-Jul-1983
0000    102 ;               Add support for MS780-H memory arrays.
0000    103 ;
0000    104 ;       V03-003 TCM0003         Trudy C. Matthews       27-Apr-1983
0000    105 ;               Change sense of CRDTEST flag from an enable to an inhibit, so
0000    106 ;               that pages with CRD errors are removed by default.
0000    107 ;
0000    108 ;       V03-002 TCM0002         Trudy C. Matthews       26-Jan-1983
0000    109 ;               Add support for RPB$V_CRDTEST flag that specifies that
0000    110 ;               pages with CRD errors be removed during the memory test.
0000    111 ;
0000    112 ;       V03-001 KDM0078         Kathleen D. Morse       15-Mar-1982
0000    113 ;               Add RPB$V_FINDMEM flag and logic, that allows the
0000    114 ;               11/782 installation to use MA780 memory instead of
```

```
0000    115 ;                   MS780 memory.
0000    116 ;
0000    117 ;       V02-003 PHL0012         Peter H. Lipman         04-Aug-1981
0000    118 ;                   Change PSECT to put this code into an area that VMB
0000    119 ;                   can overlay when it reads in the secondary boot.
0000    120 ;
0000    121 ;       V02-002 PHL0011         Peter H. Lipman         8-Jul-1981
0000    122 ;                   Use new common test memory loop routine BOO$TEST_MEM.
0000    123 ;
0000    124 ;                   Fix test for MS780C controller to use right mask.
0000    125 ;
0000    126 ;                   Eliminate use of CPU dependent MAX_PGS parameter.
0000    127 ;
0000    128 ;       V02-001 TCM0001         Trudy C. Matthews       2-Jul-1981
0000    129 ;                   Add support for new 11/780 memory controller, MS780E.
0000    130 ;
0000    131 ;
0000    132 ;--
```

```
                        .SBTTL  Declarations

                        .DEFAULT DISPLACEMENT, WORD

                ;
                ; Macros to describe VMS data structures
                ;

                        $DMPDEF                         ; System dump file header definitions
                        $IO780DEF                       ; 11/780 definitions
                        $MPMDEF                         ; Multiport memory register definitions
                        $NDTDEF                         ; Nexus device types
                        $PRDEF                          ; Processor registers
                        $RPBDEF                         ; Restart parameter block

                ;
                ; Macros
                ;

                        .MACRO  ERROR,STR               ; Outputs an error string to the
                        BSBW    ERROUT                  ; console terminal.
                        .ASCIZ  STR
                        .ENDM   ERROR

                        .PSECT  YBTMEM,LONG
```

```
                        0000   160          .SBTTL  CHECKMEM_780, Identify 11/780 memory
                        0000   161
                        0000   162 ;++
                        0000   163 ;
                        0000   164 ; CHECKMEM_780, Locate and test memory for 11/780
                        0000   165 ;
                        0000   166 ;--
                        0000   167
                        0000   168 ;
                        0000   169 ; The table below describes the 4 valid types of memory setups on
                        0000   170 ; the 11/780.
                        0000   171 ;
                        0000   172
                        0000   173 MEMTYP:
                     08 0000   174          .BYTE   NDT$_MEM4NI             ; 4K - not interleaved
                     09 0001   175          .BYTE   NDT$_MEM4I              ; 4K - interleaved
                     10 0002   176          .BYTE   NDT$_MEM16NI            ; 16K - not interleaved
                     11 0003   177          .BYTE   NDT$_MEM16I             ; 16K - interleaved
                     68 0004   178          .BYTE   NDT$_MEM64NIL           ; 64K - non-interleaved lower
                     69 0005   179          .BYTE   NDT$_MEM64EIL           ; 64K - externally interleaved
                     6A 0006   180          .BYTE   NDT$_MEM64NIU           ; 64K - non-interleaved upper
                     6B 0007   181          .BYTE   NDT$_MEM64EIU           ; 64K - externally interleaved
                     6C 0008   182          .BYTE   NDT$_MEM64I             ; 64K - internally interleaved
                     70 0009   183          .BYTE   NDT$_MEM256NIL          ; 256K - non-interleaved lower
                     71 000A   184          .BYTE   NDT$_MEM256EIL          ; 256K - externally interleaved
                     72 000B   185          .BYTE   NDT$_MEM256NIU          ; 256K - non-interleaved upper
                     73 000C   186          .BYTE   NDT$_MEM256EIU          ; 256K - externally interleaved
                     74 000D   187          .BYTE   NDT$_MEM256I            ; 256K - internally interleaved
                        000E   188
               0000000E 000E   189 MEMTYPCNT       = .-MEMTYP              ; Number of memory types.
                        000E   190
                        000E   191 SHRMEMTYP:
                     40 000E   192          .BYTE   NDT$_MPM0               ; Multiport memory (port #0)
                     41 000F   193          .BYTE   NDT$_MPM1               ; Multiport memory (port #1)
                     42 0010   194          .BYTE   NDT$_MPM2               ; Multiport memory (port #2)
                     43 0011   195          .BYTE   NDT$_MPM3               ; Multiport memory (port #3)
               00000004 0012   196 SHRMEMTYPCNT    = .-SHRMEMTYP
                        0012   197
                        0012   198 BASE_MA780:                             ; Base at which MA780's are placed
               00000000 0012   199          .LONG   0                       ;
                        0016   200 TOTAL_MA780:                           ; Total MA780 memory in bytes
               00000000 0016   201          .LONG   0                       ;
                        001A   202 MEM_CURADR:                            ; Array of current memory adr ranges
               0000005A 001A   203          .BLKB   <8*8>                   ; Max of 8 memory controllers allowed
                        005A   204 MEM_FINALADR:                          ; Array of desired memory adr ranges
               0000009A 005A   205          .BLKB   <8*8>                   ; Max of 8 memory controllers allowed
                        009A   206 MEM_TR_MTYPE:                          ; Array of memory TRs and corresp type
               000000AA 009A   207          .BLKB   <2*8>                   ; Max of 8 memory controllers allowed
                        00AA   208
                        00AA   209 ;
                        00AA   210 ; Start testing NEXUSes to find adapters. First, save the stack position
                        00AA   211 ; so it can be restored after a machine check.
                        00AA   212 ;
                        00AA   213
                        00AA   214 CHECKMEM_780::
                        00AA   215
               5C 10 D0 00AA   216          MOVL    #I0780$AL_NNEX,AP      ; Set up NEXUS loop counter.
```

```
                 5D   5E   D0   00AD   217          MOVL    SP,FP                   ; Save current top of stack.
                                00B0   218
                                00B0   219 ;
                                00B0   220 ; Set up the physical address of the 1st NEXUS on the system bus.
                                00B0   221 ;
                                00B0   222
                 54   01   1D   9C   00B0   223          ROTL    #29,#1,R4               ; 1st NEXUS is at ^X20000000.
                                00B4   224
                                00B4   225 ;
                                00B4   226 ; Point to first entry in the memory description list in the RPB.
                                00B4   227 ;
                 5A   00BC CB   9E   00B4   228          MOVAB   RPB$L_MEMDSC(R11),R10   ; Set pointer to memory description list
                                00B9   229
                                00B9   230 ;
                                00B9   231 ; During this memory locate and test loop, the following registers are
                                00B9   232 ; used:
                                00B9   233 ;
                                00B9   234 ;       R2      - value read from the configuration register
                                00B9   235 ;       R3      - maximum number of pages possible
                                00B9   236 ;                 - number of pages in the next controller
                                00B9   237 ;       R4      - address of the configuration register on the current
                                00B9   238 ;                 NEXUS
                                00B9   239 ;       R5      - address of next byte in RPB adapter type table
                                00B9   240 ;       R7      - address of the SCB
                                00B9   241 ;       R9      - starting page number for the next controller
                                00B9   242 ;       R10     - address of the memory description list in RPB (pagcnt & pfn)
                                00B9   243 ;       R11     - address of the RPB
                                00B9   244 ;       AP      - nexus loop counter
                                00B9   245 ;       FP      - saved SP; used to restore stack position after machine check
                                00B9   246 ;
                                00B9   247 ; Set up a machine check fault handler to gain control if the loop
                                00B9   248 ; addresses a non-existent configuration register (an empty NEXUS) or
                                00B9   249 ; gets a serious parity error on a page. Then read the configuration
                                00B9   250 ; register.
                                00B9   251 ;
                                00B9   252
                                00B9   253 TRY_NEXUS_780:                               ; Memory locate and test loop.
                       85   94   00B9   254          CLRB    (R5)+                   ; Assume nothing on NEXUS.
                04 A7  017D'CF   9E   00BB   255          MOVAB   DO_NEXT_780+1,4(R7)     ; Set up fault handler (+1
                                00C1   256                                             ; forces exception on the
                                00C1   257                                             ; interrupt stack).
                       52   64   D0   00C1   258          MOVL    (R4),R2                 ; Get CR at current NEXUS.
                                00C4   259
                                00C4   260 ;
                                00C4   261 ; Execution continues here if the configuration register is present.
                                00C4   262 ; Test to see what kind of adapter or memory controller this is.
                                00C4   263 ;
                                00C4   264
                    FF A5   52   90   00C4   265          MOVB    R2,-1(R5)               ; Store adapter code in RPB.
                    30 AB   0B   E0   00C8   266          BBS     #RPB$V_MPM,RPB$L_BOOTR5(R11),-
                            57        00CC   267                  CHK_SHRMEM              ; Br if multi-proc boot
             FF2D CF    0E   52   3A   00CD   268          LOCC    R2,#MEMTYPCNT,MEMTYP    ; Memory controller here?
                            4F   13   00D3   269          BEQL    CHK_SHRMEM              ; No.  Go check for shared memory.
                                00D5   270
                                00D5   271 ;
                                00D5   272 ; Find out whether the memory addresses are within legal bounds. If
                                00D5   273 ; so, compute the starting page number and the number of pages on the
```

```
                                00D5   274 ; controller.
                                00D5   275 ;
                                00D5   276
          50    04 A4    D0   00D5   277          MOVL    4(R4),R0                ; Get starting address register.
    59    50    0D   0F    EF   00D9   278          EXTZV   #15,#13,R0,R9           ; Starting page number/128
    59    59    07   78    00DE   279          ASHL    #7,R9,R9                ; Multiply by 128 = page number.
    53    52    06   09    EF   00E2   280          EXTZV   #9,#6,R2,R3             ; Get size in 128 or 2048 page units.
                     53    D6   00E7   281          INCL    R3                      ; Convert size-1 to size.
          50    07   D0    00E9   282          MOVL    #7,R0                   ; Assume 128 page units (MS780C).
          52    E0 8F    93   00EC   283          BITB    #^XE0,R2                ; MS780C controller?
                     03    13   00F0   284          BEQL    30$                     ; Yes, continue.
          50    0B   D0    00F2   285          MOVL    #11,R0                  ; Else use 2048 page units (MS780E).
    53    53    50   78    00F5   286 30$:     ASHL    R0,R3,R3                ; Multiply to find # of pages present.
                                00F9   287          ASSUME  DMP$V_TR EQ 24
                                00F9   288          ASSUME  DMP$S_TR EQ 8
                                00F9   289 ;
                                00F9   290 ; VAX 11/780 Interleaved Memory (controller = MS780C)
                                00F9   291 ;
                                00F9   292 ; When two memories are interleaved, they must be an even-odd pair of
                                00F9   293 ; TR numbers.  Both memories have the same starting physical address.  VMB
                                00F9   294 ; doubles the size of the first memory and checks all the pages as if
                                00F9   295 ; they belonged to the first controller.  When the second controller
                                00F9   296 ; is scanned, VMB realizes that it has already checked the corresponding
                                00F9   297 ; memory pages and skips to the code that finds the next CSR.  In the RPB,
                                00F9   298 ; there is only one memory descriptor initialized.  It contains the even
                                00F9   299 ; numbered TR #, double the memory page count, and the common base PFN.
                                00F9   300 ;
                                00F9   301 ; VAX 11/780 Interleaved Memory (controller = MS780E)
                                00F9   302 ;
                                00F9   303 ; The above paragraph also applies to the MS780E controller, with one
                                00F9   304 ; exception:  the MS780E controller stores the entire size of the interleaved
                                00F9   305 ; memory in the memory size field of the configuration register, not half the
                                00F9   306 ; size.  Also, the MS780E supports two forms of interleaving: external and
                                00F9   307 ; internal.  The internal interleaving requires no special VMS action: the
                                00F9   308 ; code path taken is the same as for non-interleaved memory.  MS780E external
                                00F9   309 ; interleaving shares a code path with MS780C interleaved memories.
                                00F9   310 ;
                          0C    52   E9   00F9   311          BLBC    R2,10$                  ; Branch if no interleaving.
                          7D    5C   E8   00FC   312          BLBS    AP,DO_NEXT_780          ; Br if  2nd of interleaved TR# pair.
                52        E0 8F    93   00FF   313          BITB    #^XE0,R2                ; MS780C controller?
                               03    12   0103   314          BNEQ    10$                     ; No; MS780E already has correct size.
                          53   53    C0   0105   315          ADDL    R3,R3                   ; MS780C requires page count doubled.
                          8A   53    D0   0108   316 10$:     MOVL    R3,(R10)+               ; Save # of pages in this memory
FF AA     54              04   0D    EF   010B   317          EXTZV   #13,#4,R4,-1(R10)       ; Save TR number for this memory
                          8A   59    D0   0111   318          MOVL    R9,(R10)+               ; Save starting PFN for this memory
                                0114   319 TEST_780:
     04 A7          0425'CF    9E   0114   320          MOVAB   PAGE_MCHECK_780+1,4(R7) ; Set page skipping handler (+1
                                011A   321                                          ; for on interrupt stack).
          52        0402'CF    9E   011A   322          MOVAB   TEST_QUAD_780,R2        ; Routine to test one page
                       FEDE'   30   011F   323          BSBW    BOO$TEST_MEM            ; Test the specified range of PFN's
                               58    11   0122   324          BRB     DO_NEXT_780
                                0124   325 CHK_SHRMEM:
   FEE4 CF          04    52   3A   0124   326          LOCC    R2,#SHRMEMTYPCNT,SHRMEMTYP ; Is this a shared memory?
                               50    13   012A   327          BEQL    DO_NEXT_780             ; No.  Go look for another NEXUS.
                                012C   328
    64      00400000 8F    C8   012C   329          BISL    #^X00400000,MPM$L_CSR(R4) ; Clear power-up bit
 04 A4      FF000000 8F    C8   0133   330          BISL    #^XFF000000,MPM$L_CR(R4)  ; Clear error bits
```

```
                08 A4    D000C000 8F    C8  013B  331          BISL    #^XD000C000,MPM$L_SR(R4)  ; Clear error bits
                10 A4    80000000 8F    C8  0143  332          BISL    #^X80000000,MPM$L_ERR(R4) ; Clear error bits
                           53    0C A4    D0  014B  333          MOVL    MPM$L_INV(R4),R3          ; Get Invalidation Control Register
                   59      53    0B    14  EF  014F  334          EXTZV   #MPM$V_INV_STADR,#MPM$S_INV_STADR,R3,R9 ; Starting address
                   59      59    09    78  0154  335          ASHL    #9,R9,R9                  ; Convert to starting PFN
                   53      53    03    10  EF  0158  336          EXTZV   #MPM$V_INV_MEMSZ,#MPM$S_INV_MEMSZ,R3,R3 ; Array size
                                       53  D6  015D  337          INCL    R3                        ; 0 = one board, so add one to count
                           53    53    09  78  015F  338          ASHL    #9,R3,R3                  ; Multiply by 512 to compute # of pages
                                 8A    53  D0  0163  339          MOVL    R3,(R10)+                 ; Save # of pages in this memory
                                             0166  340          ASSUME  DMP$V_TR EQ 24
                                             0166  341          ASSUME  DMP$S_TR EQ 8
             FF AA      54      04    0D  EF  0166  342          EXTZV   #13,#4,R4,-1(R10)         ; Save TR number for this memory
                                 8A    59  D0  016C  343          MOVL    R9,(R10)+                 ; Save starting PFN for this memory
                     00001800 8F        D3  016F  344          BITL    #<RPB$M_MPM ! RPB$M_USEMPM>,- ; If multi-proc boot or MA780
                           30 AB              0175  345                  RPB$L_BOOTR5(R11)         ; memory used as local memory,
                              9B    12        0177  346          BNEQ    TEST_780                  ; then go include it in bitmap.
                              01    11        0179  347          BRB     DO_NEXT_780               ; Branch around bytes left by .ALIGN
                                              017B  348
                                              017B  349          .ALIGN  LONG                      ; Longword-aligned handler.
                                              017C  350
                                              017C  351 ;
                                              017C  352 ; Fault handler for non-existent configuration register, or unreadable
                                              017C  353 ; registers, or a non-memory controller NEXUS device. Restore stack
                                              017C  354 ; pointer, clear all errors, and try for another NEXUS if any remain.
                                              017C  355 ;
                                              017C  356
                                              017C  357 DO_NEXT_780:                              ; Skip to next NEXUS.
                           5E    5D    D0    017C  358          MOVL    FP,SP                     ; Restore stack pointer.
                           30    00    DA    017F  359          MTPR    #0,#PR$_SBIFS             ; Clear any faults.
                     54    2000  C4    9E    0182  360          MOVAB   IO780$AL_PERNEX(R4),R4    ; Move to next NEXUS.
                              1E 5C    F5    0187  361          SOBGTR  AP,NEXT_NEXUS_780         ; If still a NEXUS, loop.
                                              018A  362
                                              018A  363 ;
                                              018A  364
                                              018A  365 ; Check if there is sufficient memory to boot VMS.
                                              018A  366 ;
                                              018A  367
                        10 30 AB    0E    E5  018A  368          BBCC    #RPB$V_FINDMEM,RPB$L_BOOTR5(R11),10$ ; Only check if requested
                52   00BC CB      18    00    EF  018F  369          EXTZV   #0,#24,RPB$L_MEMDSC(R11),R2 ; Get # pages in first memory
                      00000400 8F      52    D1  0196  370          CMPL    R2,#^X400                 ; Is there at least 512K?
                                       0C    1F  019D  371          BLSSU   FIND_MEM                  ; Go try to find MA780 memory instead
                                              019F  372 ;
                                              019F  373 ; Reestablish the normal machine check fault handler.
                                              019F  374 ;
                                              019F  375 10$:
                        04 A7    0001'CF    DE  019F  376          MOVAL   UNEXP_MCHK+1,4(R7)        ; Reset SCB vector.
                                       8A    D4  01A5  377          CLRL    (R10)+                    ; Indicate end of RPB memory descr list
                                       05    01A7  378          RSB                               ; Return to main routine.
                                              01A8  379
                                              01A8  380 ;
                                              01A8  381 ; Extra label and branch here to loop back through the NEXUS testing
                                              01A8  382 ; code.
                                              01A8  383 ;
                                              01A8  384
                                              01A8  385 NEXT_NEXUS_780:                           ; Try the next NEXUS.
                                    FF0E      31  01A8  386          BRW     TRY_NEXUS_780             ; Branch to top of loop.
```

```
                                01AB   388          .SBTTL  FIND_MEM - Locate MA780 memory and use it instead of MS780
                                01AB   389 ;
                                01AB   390 ; This code assumes that VMB will be entirely loaded into one memory
                                01AB   391 ; and not across memory controllers, except for the case of interleaved
                                01AB   392 ; local memory.
                                01AB   393 ;
                                01AB   394 FIND_MEM:
          57    FE6B CF    9E   01AB   395          MOVAB   MEM_CURADR,R7           ; Adr of old phys adr range array
          58    FEA6 CF    9E   01B0   396          MOVAB   MEM_FINALADR,R8         ; Adr of new phys adr range array
          59    FEE1 CF    9E   01B5   397          MOVAB   MEM_TR_MTYPE,R9         ; Adr of memory TR #s and corresp type
          5A    00BC CB    9E   01BA   398          MOVAB   RPB$L_MEMDSC(R11),R10   ; Adr of mem dsc array (all controllers)
          55    0090 CB    9E   01BF   399          MOVAB   RPB$B_CONFREG(R11),R5   ; Adr of nexus type codes
                     52      7C   01C4   400          CLRQ    R2                      ; Init total MA memory and local memory
                     56      D4   01C6   401          CLRL    R6                      ; Init starting MA adr to 0
                                01C8   402
                                01C8   403 10$:
    54    8A    FF000000 8F    CB   01C8   404          BICL3   #^XFF000000,(R10)+,R4   ; Get count of pages in this memory
                     55      13   01D0   405          BEQL    40$                     ; Br if no more memories
                        50 FF AA    9A   01D2   406          MOVZBL  -1(R10),R0              ; Get TR # for this memory
                           89   50    9B   01D6   407          MOVZBW  R0,(R9)+                ; Remember TR # and assume local memory
                        67   8A 09    78   01D9   408          ASHL    #9,(R10)+,(R7)          ; Compute old starting phys byte adr
                        54   54 09    78   01DD   409          ASHL    #9,R4,R4                ; Compute count of bytes in this memory
                  04 A7    67   54    C1   01E1   410          ADDL3   R4,(R7),4(R7)           ; Compute old ending phys byte adr
                              04 A7    D7   01E6   411          DECL    4(R7)                   ; Make old ending adr exact to byte
                     50    6540      9A   01E9   412          MOVZBL  (R5)[R0],R0             ; Get NEXUS type for this memory
               FE0D CF     0E   50    3A   01ED   413          LOCC    R0,#MEMTYPCNT,MEMTYP    ; Is this local memory?
                           18   13    F3   01F3   414          BEQL    20$                     ; Br on no, assume it is MA780 memory
                        68   52    D0   01F5   415          MOVL    R2,(R8)                 ; Set new starting phys byte adr
                  04 A8    68   54    C1   01F8   416          ADDL3   R4,(R8),4(R8)           ; Compute new ending phys byte adr
                           52   54    C0   01FD   417          ADDL    R4,R2                   ; Add mem size to local mem accum
                        04 A7   56    D1   0200   418          CMPL    R6,4(R7)                ; Does local memory extend beyond 8MB?
                           16   14    0204   419          BGTR    30$                     ; Br on no, local mem < 8MB
                  56    04 A7    01   C1   0206   420          ADDL3   #1,4(R7),R6             ; Get next adr after all local memory
                           0F   11    020B   421          BRB     30$                     ; Continue in common code
                                020D   422 20$:
                     FF A9    01   90   020D   423          MOVB    #1,-1(R9)               ; Set MA780 indicator
                        68   53    D0   0211   424          MOVL    R3,(R8)                 ; Set new starting phys byte adr
                  04 A8    68   54    C1   0214   425          ADDL3   R4,(R8),4(R8)           ; Compute new ending phys byte adr
                           53   54    C0   0219   426          ADDL    R4,R3                   ; Add mem size to MA780 mem accum
                                021C   427 30$:
                              04 A8    D7   021C   428          DECL    4(R8)                   ; Make new ending adr exact to byte
                        57   08    C0   021F   429          ADDL    #8,R7                   ; Point to next old range
                        58   08    C0   0222   430          ADDL    #8,R8                   ; Point to next new range
                           A1   11    0225   431          BRB     10$                     ; Loop to do next memory
                                0227   432 40$:
               FDEA CF     53   D0   0227   433          MOVL    R3,TOTAL_MA780          ; Remember total bytes of MA780 memory
                           67   7C    022C   434          CLRQ    (R7)                    ; End the old range array
                           68   7C    022E   435          CLRQ    (R8)                    ; End the new range array
                           69   B4    0230   436          CLRW    (R9)                    ; End the TR # and type list
                                0232   437 ;
                                0232   438 ; Now check if all MA780 memory is where it should be for an 11/782.
                                0232   439 ; In other words, it starts at physical address 0 and each controller
                                0232   440 ; is contiguous to the previous one.  Local memory will be at some
                                0232   441 ; higher physical address than all MA780 memory.
                                0232   442 ;
          57    FDE4 CF    9E   0232   443          MOVAB   MEM_CURADR,R7           ; Adr of old phys adr range array
          58    FE1F CF    9E   0237   444          MOVAB   MEM_FINALADR,R8         ; Adr of new phys adr range array
```

```
                59   FE5A CF    9E   023C   445          MOVAB   MEM_TR_MTYPE,R9          ; Adr of memory TR #s and corresp type
                                      0241   446 50$:
                          69    95   0241   447          TSTB    (R9)                     ; Any more memories?
                          1B    13   0243   448          BEQL    70$                      ; Br if no more memories
                      0C 01     A9   E9   0245   449          BLBC    1(R9),60$                ; Br if this is local memory
                       68       67   D1   0249   450          CMPL    (R7),(R8)                ; Is current phys adr the desired adr?
                          15    12   024C   451          BNEQ    80$                      ; Br if memory phys adrs are not set up
                04 A8     04   A7   D1   024E   452          CMPL    4(R7),4(R8)              ; Is current phys adr the desired adr?
                          0E    12   0253   453          BNEQ    80$                      ; Br if memory phys adrs are not set up
                                      0255   454 60$:
                          57    08   C0   0255   455          ADDL    #8,R7                    ; Point to next old range
                          58    08   C0   0258   456          ADDL    #8,R8                    ; Point to next new range
                          59    02   C0   025B   457          ADDL    #2,R9                    ; Point to next memory TR and type
                          E1    11   025E   458          BRB     50$                      ; Continue with next memory
                                      0260   459 70$:
                        0159    31   0260   460          BRW     MA780_AT_0               ; MA780 memory adrs already set ok
                                      0263   461 ;
                                      0263   462 ; Now set all the MA780 starting addresses to be contiguous,
                                      0263   463 ; above all local memory.  Use the maximum of 8MB and what is currently
                                      0263   464 ; being used for local memory.
                                      0263   465 ;
                                      0263   466 80$:
                59   FDF3 CF    9E   0263   467          MOVAB   MEM_FINALADR,R8          ; Adr of new phys adr range array
                59   FE2E CF    9E   0268   468          MOVAB   MEM_TR_MTYPE,R9          ; Adr of memory TR #s and corresp type
                                      026D   469 90$:
                          53    69   9A   026D   470          MOVZBL  (R9),R3                  ; Get TR # for this memory
                          24    13   0270   471          BEQL    110$                     ; Br if no more memories
                      18 01     A9   E9   0272   472          BLBC    1(R9),100$               ; Br if this is local memory
                54        68    56   C1   0276   473          ADDL3   R6,(R8),R4               ; Get starting adr for this memory
                54        54    02   78   027A   474          ASHL    #2,R4,R4                 ; Compute starting adr bits for MA780
                                54   D6   027E   475          INCL    R4                       ; Set enable bit for setting start adr
                53        53    0D   78   0280   476          ASHL    #13,R3,R3                ; Compute TR offset part of contlr adr
          53     2000000C    8F   C8   0284   477          BISL    #^X2000000C,R3           ; Get controller register address
                          63    54   D0   028B   478          MOVL    R4,(R3)                  ; Set new starting adr for this memory
                                      028E   479 100$:
                          59    02   C0   028E   480          ADDL    #2,R9                    ; Point to next memory TR #
                          58    08   C0   0291   481          ADDL    #8,R8                    ; Point to next new range
                                D7   11   0294   482          BRB     90$                      ; Continue for next mem
                                      0296   483 ;
                                      0296   484 ; Now move VMB into the MA780 memory at the lowest starting address.
                                      0296   485 ; Then continue executing VMB in that memory so that the local memory
                                      0296   486 ; starting addresses may be changed.
                                      0296   487 ;
                                      0296   488 110$:
                FD77 CF              56   D0   0296   489          MOVL    R6,BASE_MA780            ; Remember this base address
                0000'CF              01   D0   029B   490          MOVL    #1,CONTINUE_INDEX        ; Set continuation code index
                50    0000'CF        9E   02A0   491          MOVAB   START_BOOT,R0            ; Get physical addr of start of VMB
           50    00000200    8F      C2   02A5   492          SUBL    #^X200,R0                ; Include the RPB
        66    60    0200'8F         28   02AC   493          MOVC3   #BOOTHIGH+^X200,(R0),(R6) ; Move VMB/RPB into different memory
58      56    00000200    8F      C1   02B2   494          ADDL3   #^X200,R6,R8             ; Remember base address
                50    1C    AB   7D   02BA   495          MOVQ    RPB$L_BOOTR0(R11),R0     ; Reset register R0-R1 to boot value
                52    24    AB   7D   02BE   496          MOVQ    RPB$L_BOOTR2(R11),R2     ; Reset register R2-R3 to boot value
                54    2C    AB   7D   02C2   497          MOVQ    RPB$L_BOOTR4(R11),R4     ; Reset register R4-R5 to boot value
                5C    18    AB   D0   02C6   498          MOVL    RPB$L_HALTCODE(R11),AP   ; Reset register AP to boot value
                5A    10    AB   D0   02CA   499          MOVL    RPB$L_HALTPC(R11),R10    ; Reset register R10 to boot value
                5B    14    AB   D0   02CE   500          MOVL    RPB$L_HALTPSL(R11),R11   ; Reset register R11 to boot value
                   5E         58   D0   02D2   501          MOVL    R8,SP                    ; Reset register SP to point to RPB+200
```

```
                0000'C8   17   02D5   502          JMP     START_BOOT_1(R8)        ; Continue executing VMB in other memory
                                02D9   503
                                02D9   504 CONT1_PATH::
                                02D9   505
                                02D9   506 ;
                                02D9   507 ; Now move all local memory into the next 8MB above the MA780 memory.
                                02D9   508 ;
     56   FD32 CF    FD39 CF   C1   02D9   509          ADDL3   TOTAL_MA780,BASE_MA780,R6 ; Get base adr for local memory
          58   FD75 CF   9E   02E1   510          MOVAB   MEM_FINALADR,R8         ; Adr of new phys adr range array
          59   FDB0 CF   9E   02E6   511          MOVAB   MEM_TR_MTYPE,R9         ; Adr of memory TR #s and corresp type
                                02EB   512 10$:
                53   69   9A   02EB   513          MOVZBL  (R9),R3                 ; Get TR # for this memory
                     5C   13   02EE   514          BEQL    40$                     ; Br if no more memories
                50 01 A9   E8   02F0   515          BLBS    1(R9),30$               ; Br if this is MA780 memory
          54    68   56   C1   02F4   516          ADDL3   R6,(R8),R4              ; Get starting adr for this memory
     54   54    FF   8F   78   02F8   517          ASHL    #-1,R4,R4               ; Get right bits to write into register
          53    53   0D   78   02FD   518          ASHL    #13,R3,R3               ; Compute TR offset part of contlr adr
     53    20000004   8F   C8   0301   519          BISL    #^X20000004,R3          ; Get controller register address
          54   4000   8F   A8   0308   520          BISW    #^X4000,R4              ; Set enable write to starting adr bit
               63   54   D0   030D   521          MOVL    R4,(R3)                 ; Set new starting adr for this memory
               53   04   8A   0310   522          BICB    #4,R3                   ; Get adr of interleaved bits
               52   63   D0   0313   523          MOVL    (R3),R2                 ; Branch if this controller is
                 24 52   E9   0316   524          BLBC    R2,20$                  ; not interleaved
     51    04 A8   68   C3   0319   525          SUBL3   (R8),4(R8),R1           ; Get # bytes of interleaved memory
                    51   D6   031E   526          INCL    R1                      ; Round up to a page boundary
     51   51    FE   8F   78   0320   527          ASHL    #-2,R1,R1               ; Calculate base adr for second half
               51   54   C0   0325   528          ADDL    R4,R1                   ; of this memory
55   53   00002004   8F   C1   0328   529          ADDL3   #^X2004,R3,R5           ; Get address of second controller
               65   51   D0   0330   530          MOVL    R1,(R5)                 ; Set new starting adr for this memory
               55   04   8A   0333   531          BICB    #4,R5                   ; Get adr of interleaved bits
          65   00000100   8F   D0   0336   532          MOVL    #^X100,(R5)             ; Turn off interleaving
                                033D   533 20$:
          63   00000100   8F   D0   033D   534          MOVL    #^X100,(R3)             ; Turn off interleaving
                                0344   535 30$:
                    59   02   C0   0344   536          ADDL    #2,R9                   ; Point to next memory TR #
                    58   08   C0   0347   537          ADDL    #8,R8                   ; Point to next new range
                         9F   11   034A   538          BRB     10$                     ; Continue for next mem
                                034C   539 ;
                                034C   540 ; Now move VMB back into local memory, which is way up high now.
                                034C   541 ;
                                034C   542 40$:
               0000'CF   02   D0   034C   543          MOVL    #2,CONTINUE_INDEX       ; Set continuation code index
               50   0000'CF   9E   0351   544          MOVAB   START_BOOT,R0           ; Get physical addr of start of VMB
          50    00000200   8F   C2   0356   545          SUBL    #^X200,R0               ; Include the RPB
     66   60    0200'8F   28   035D   546          MOVC3   #BOOTHIGH+^X200,(R0),(R6) ; Move VMB/RPB into different memory
58   56    00000200   8F   C1   0363   547          ADDL3   #^X200,R6,R8            ; Remember base address
               50    1C AB   7D   036B   548          MOVQ    RPB$L_BOOTR0(R11),R0    ; Reset register R0-R1 to boot value
               52    24 AB   7D   036F   549          MOVQ    RPB$L_BOOTR2(R11),R2    ; Reset register R2-R3 to boot value
               54    2C AB   7D   0373   550          MOVQ    RPB$L_BOOTR4(R11),R4    ; Reset register R4-R5 to boot value
               5C    18 AB   D0   0377   551          MOVL    RPB$L_HALTCODE(R11),AP  ; Reset register AP to boot value
               5A    10 AB   D0   037B   552          MOVL    RPB$L_HALTPC(R11),R10   ; Reset register R10 to boot value
               5B    14 AB   D0   037F   553          MOVL    RPB$L_HALTPSL(R11),R11  ; Reset register R11 to boot value
                  5E   58   D0   0383   554          MOVL    R8,SP                   ; Reset register SP to point to RPB+200
                  0000'C8   17   0386   555          JMP     START_BOOT_1(R8)        ; Continue executing VMB in other memory
                                038A   556
                                038A   557 CONT2_PATH::
                                038A   558
```

```
                                 038A    559 ;
                                 038A    560 ; Now set MA780 memory to be contiguous, starting at physical address 0.
                                 038A    561 ;
            58    FCCC CF    9E  038A    562         MOVAB   MEM_FINALADR,R8         ; Adr of new phys adr range array
            59    FD07 CF    9E  038F    563         MOVAB   MEM_TR_MTYPE,R9         ; Adr of memory TR #s and corresp type
                                 0394    564 10$:
                 53    69    9A  0394    565         MOVZBL  (R9),R3                 ; Get TR # for this memory
                       23    13  0397    566         BEQL    MA780_AT_0              ; Br if no more memories
                    17 01    A9  E9  0399    567         BLBC    1(R9),20$               ; Br if this is local memory
                      54    68  D0  039D    568         MOVL    (R8),R4                 ; Get starting adr for this memory
            54        54    02  78  03A0    569         ASHL    #2,R4,R4                ; Compute starting adr bits for MA780
                            54  D6  03A4    570         INCL    R4                      ; Set enable bit for setting start adr
            53        53    0D  78  03A6    571         ASHL    #13,R3,R3               ; Compute TR offset part of contlr adr
         53     2000000C    8F  C8  03AA    572         BISL    #^X2000000C,R3          ; Get controller register address
                      63    54  D0  03B1    573         MOVL    R4,(R3)                 ; Set new starting adr for this memory
                                 03B4    574 20$:
                      59    02  C0  03B4    575         ADDL    #2,R9                   ; Point to next memory TR #
                      58    08  C0  03B7    576         ADDL    #8,R8                   ; Point to next new range
                            D8  11  03BA    577         BRB     10$                     ; Continue for next mem
                                 03BC    578 ;
                                 03BC    579 ; Now move VMB into the MA780 memory starting at physical address 0.
                                 03BC    580 ; This should be the correct memory to boot from, so re-execute
                                 03BC    581 ; VMB entirely over again now.
                                 03BC    582 ;
                                 03BC    583 MA780_AT_0:
            50    0000'CF    9E  03BC    584         MOVAB   START_BOOT,R0           ; Get physical addr of start of VMB
         50     00000200    8F  C2  03C1    585         SUBL    #^X200,R0               ; Include the RPB
00000000 9F    60    0200'8F    28  03C8    586         MOVC3   #BOOTHIGH+^X200,(R0),@#0 ; Move VMB/RPB into different memory
              50      1C AB    7D  03D2    587         MOVQ    RPB$L_BOOTR0(R11),R0    ; Reset register R0-R1 to boot value
              52      24 AB    7D  03D6    588         MOVQ    RPB$L_BOOTR2(R11),R2    ; Reset register R2-R3 to boot value
              54      2C AB    7D  03DA    589         MOVQ    RPB$L_BOOTR4(R11),R4    ; Reset register R4-R5 to boot value
            55      0800 8F    A8  03DE    590         BISW    #RPB$M_MPM,R5           ; Use MA780 memory only
              5C      18 AB    D0  03E3    591         MOVL    RPB$L_HALTCODE(R11),AP  ; Reset register AP to boot value
              5A      10 AB    D0  03E7    592         MOVL    RPB$L_HALTPC(R11),R10   ; Reset register R10 to boot value
              5B      14 AB    D0  03EB    593         MOVL    RPB$L_HALTPSL(R11),R11  ; Reset register R11 to boot value
         5E   00000200    8F    D0  03EF    594         MOVL    #^X200,SP               ; Set register SP to adr of RPB+200
              00000000    9F    D4  03F6    595         CLRL    @#00                    ; Set physical adr of RPB in RPB
              00000200'9F    17  03FC    596         JMP     @#START_BOOT+^X200      ; Continue executing VMB in other memory
```

```
                     0402    598          .SBTTL  TEST_QUAD_780 - Test a quadword of memory
                     0402    599 ;++
                     0402    600 ;
                     0402    601 ; Functional Description:
                     0402    602 ;
                     0402    603 ;       Test a page of 780 memory,
                     0402    604 ;
                     0402    605 ; Calling Sequence:
                     0402    606 ;
                     0402    607 ;       JSB     TEST_QUAD_780
                     0402    608 ;
                     0402    609 ; Inputs:
                     0402    610 ;
                     0402    611 ;       R0 = starting address to test
                     0402    612 ;       R1 = Quad word iteration count (64)
                     0402    613 ;       R11= Address of RPB
                     0402    614 ;
                     0402    615 ; Outputs:
                     0402    616 ;
                     0402    617 ;       Returns via RSB if the entire page is OK
                     0402    618 ;       Error exit via Machine Check code to BOO$PAGE_MCHECK
                     0402    619 ;
                     0402    620 ;--
                     0402    621
                     0402    622 TEST_QUAD_780:                          ; Test 1 quadword at a time.
          36   50   DA 0402    623          MTPR    R0,#PR$_SBIQC          ; Clear a quadword.
          80   80   D1 0405    624          CMPL    (R0)+,(R0)+            ; Read both longwords, and
                     0408    625                                         ; advance to next quadword.
                     0408    626
                     0408    627 ;
                     0408    628 ; If no gross errors occur in the clear to the quadword or in the
                     0408    629 ; subsequent read instruction, then execution continues below. Otherwise
                     0408    630 ; execution goes to the fault handler -- PAGE_MCHECK_780.
                     0408    631 ;
                     0408    632
             F7 51   F5 0408    633          SOBGTR  R1,TEST_QUAD_780       ; Continue clearing unless done.
       01 30 AB   10   E1 040B    634          BBC     #RPB$V_CRDTEST,-       ; Branch if CRD test not inhibited.
                     0410    635                  RPB$L_BOOTR5(R11),10$  ;
                     0410    636 5$:      RSB                            ; No CRD test; return.
                     0411    637 ;
                     0411    638 ; Check to see if there were any CRD errors on this page.
                     0411    639 ;
                     0411    640 10$:
             7E   34   DB 0411    641          MFPR    #PR$_SBIER,-(SP)       ; Get SBI error register.
             34   6E   DA 0414    642          MTPR    (SP),#PR$_SBIER        ; Clear errors just in case.
    8E   00004000 8F   D3 0417    643          BITL    #^X4000,(SP)+          ; Check for CRD error.
                F0   13 041E    644          BEQL    5$                     ; Branch if no CRD errors occurred.
                05   11 0420    645          BRB     ERR_EXIT_780           ; Else take error exit.
                     0422    646
                     0422    647
                     0422    648          .ALIGN  LONG                   ; All handlers longword-aligned.
                     0424    649 ;
                     0424    650 ; Handler that gains control when a page has gross memory errors.
                     0424    651 ;
                     0424    652
                     0424    653 PAGE_MCHECK_780:                        ; Skip current page.
             30   00   DA 0424    654          MTPR    #0,#PR$_SBIFS          ; Clear error indicator.
```

```
                0427   655 ERR_EXIT_780:
FBD6'  31       0427   656          BRW     BOO$PAGE_MCHECK          ; Back to common handler
                042A   657          .END
```

```
BASE_MA780                      00000012 R      02
BOO$PAGE_MCHECK                 ********   X    02
BOO$TEST_MEM                    ********   X    02
BOOTHIGH                        ********   X    02
CHECKMEM_780                    000000AA RG     02
CHK_SHRMEM                      00000124 R      02
CONT1_PATH                      000002D9 RG     02
CONT2_PATH                      0000038A RG     02
CONTINUE_INDEX                  ********   X    02
DMP$S_TR                    =   00000008
DMP$V_TR                    =   00000018
DO_NEXT_780                     0000017C R      02
ERR_EXIT_780                    00000427 R      02
FIND_MEM                        000001AB R      02
IO780$AL_NNEX               =   00000010
IO780$AL_PERNEX             =   00002000
MA780_AT_0                      000003BC R      02
MEMTYP                          00000000 R      02
MEMTYPCNT                   =   0000000E
MEM_CURADR                      0000001A R      02
MEM_FINALADR                    0000005A R      02
MEM_TR_MTYPE                    0000009A R      02
MPM$L_CR                    =   00000004
MPM$L_CSR                   =   00000000
MPM$L_ERR                   =   00000010
MPM$L_INV                   =   0000000C
MPM$L_SR                    =   00000008
MPM$S_INV_MEMSZ             =   00000003
MPM$S_INV_STADR             =   0000000B
MPM$V_INV_MEMSZ             =   00000010
MPM$V_INV_STADR             =   00000014
NDT$_MEM16I                 =   00000011
NDT$_MEM16NI                =   00000010
NDT$_MEM256EIL              =   00000071
NDT$_MEM256EIU              =   00000073
NDT$_MEM256I                =   00000074
NDT$_MEM256NIL              =   00000070
NDT$_MEM256NIU              =   00000072
NDT$_MEM4I                  =   00000009
NDT$_MEM4NI                 =   00000008
NDT$_MEM64EIL               =   00000069
NDT$_MEM64EIU               =   0000006B
NDT$_MEM64I                 =   0000006C
NDT$_MEM64NIL               =   00000068
NDT$_MEM64NIU               =   0000006A
NDT$_MPM0                   =   00000040
NDT$_MPM1                   =   00000041
NDT$_MPM2                   =   00000042
NDT$_MPM3                   =   00000043
NEXT_NEXUS_780                  000001A8 R      02
PAGE_MCHECR_780                 00000424 R      02
PR$_SBIER                   =   00000034
PR$_SBIFS                   =   00000030
PR$_SBIQC                   =   00000036
RPB$B_CONFREG               =   00000090
RPB$L_BOOTR0                =   0000001C
RPB$L_BOOTR2                =   00000024
RPB$L_BOOTR4                =   0000002C
RPB$L_BOOTR5                =   00000030
RPB$L_HALTCODE              =   00000018
RPB$L_HALTPC                =   00000010
RPB$L_HALTPSL               =   00000014
RPB$L_MEMDSC                =   000000BC
RPB$M_MPM                   =   00000800
RPB$M_USEMPM                =   00001000
RPB$V_CRDTEST               =   00000010
RPB$V_FINDMEM               =   0000000E
RPB$V_MPM                   =   0000000B
SHRMEMTYP                       0000000E R      02
SHRMEMTYPCNT                =   00000004
START_BOOT                      ********   X    02
START_BOOT_1                    ********   X    02
TEST_780                        00000114 R      02
TEST_QUAD_780                   00000402 R      02
TOTAL_MA780                     00000016 R      02
TRY_NEXUS_780                   000000B9 R      02
UNEXP_MCHK                      ********   X    02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YBTMEM	0000042A (1066.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.09	00:00:00.90
Command processing	132	00:00:00.65	00:00:03.45
Pass 1	216	00:00:05.05	00:00:12.36
Symbol table sort	0	00:00:00.51	00:00:00.71
Pass 2	134	00:00:01.69	00:00:03.96
Symbol table output	10	00:00:00.08	00:00:00.07
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	532	00:00:08.10	00:00:21.50

The working set limit was 1350 pages.
29561 bytes (58 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 379 non-local and 22 local symbols.
657 source lines were read in Pass 1, producing 15 object records in Pass 2.
15 pages of virtual memory were used to define 14 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
_255DUA28:[SYS.OBJ]LIB.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	10

435 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:BTMEM780/OBJ=OBJ$:BTMEM780 MSRC$:BTMEM780/UPDATE=(ENH$:BTMEM780)+EXECML$/LIB+LIB$:BOOTS.MLB/LIB
```

CONFIG
LIS

BTMEM855
LIS

BTMEM790
LIS

CONFIGURE
LIS

BOOTDEF
LIS

BOOTIO
LIS

CONFIGMN
LIS

BOOTDRIVR
LIS

BTMEM730
LIS

BTMEM750
LIS

BTMEM780
LIS

BOOTBLOCK
LIS