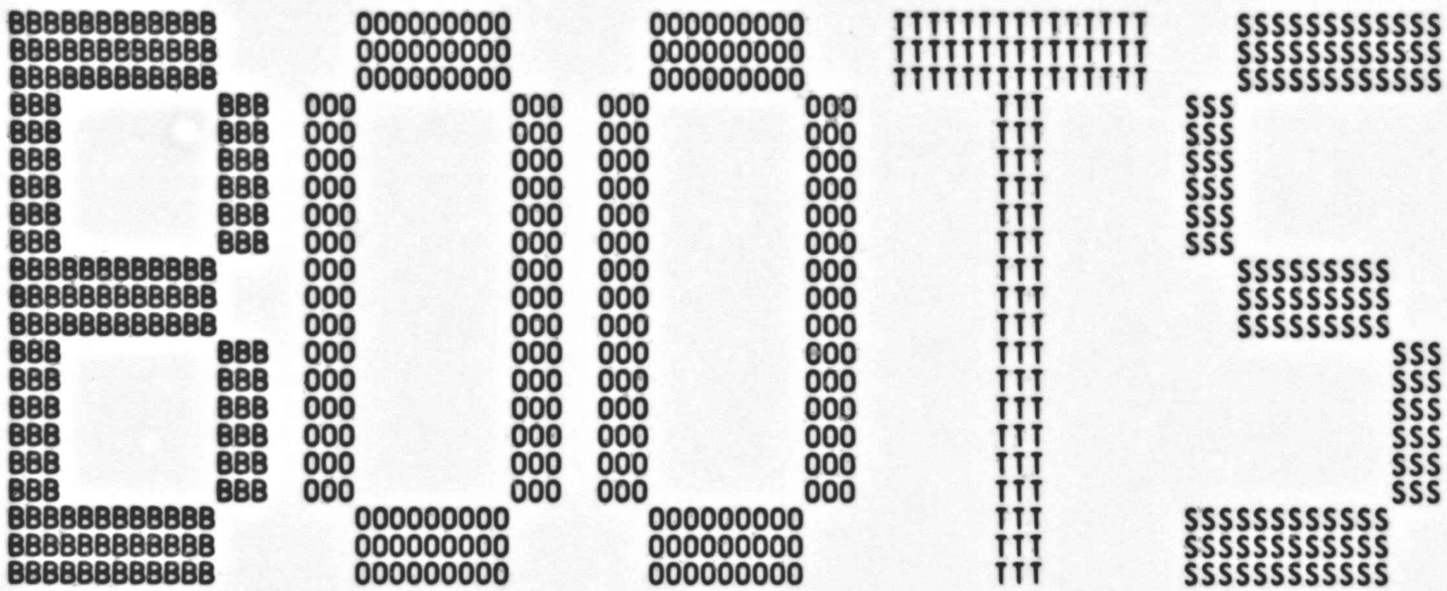
BBBBBBBBBBB OOOOOOOOO OOOOOOOOO TTTTTTTTTTTTTTT SSSSSSSSSSS
BBB BBB OOO OOO OOO OOO TTT SSS
BBBBBBBBBBB OOO OOO OOO OOO TTT SSSSSSSSS
BBB BBB OOO OOO OOO OOO TTT SSS
BBBBBBBBBBB OOOOOOOOO OOOOOOOOO TTT SSSSSSSSSSS

FILEID**BTMEM730

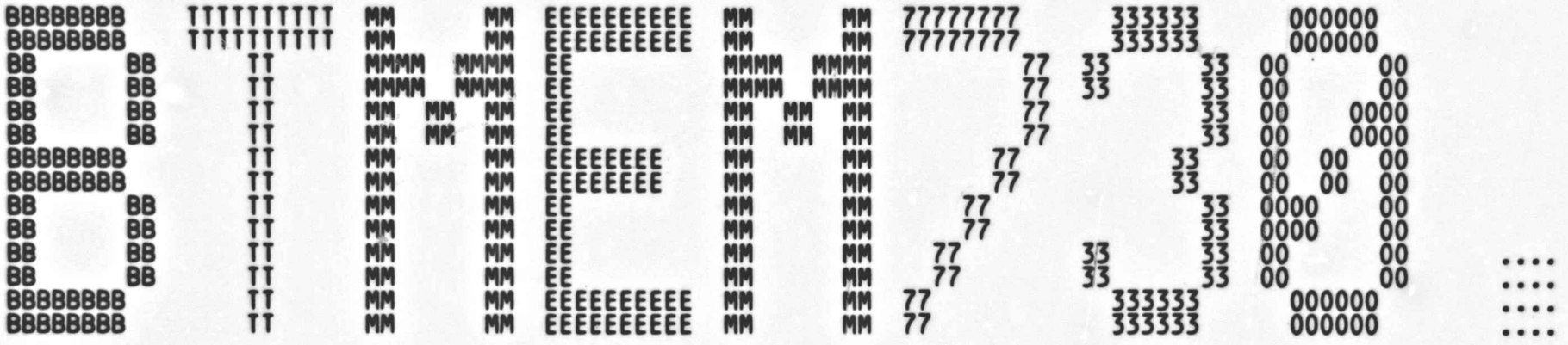

Table of contents

```
0000     1          .TITLE  BTMEM730 - Configure and Test 11/730 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 11/730 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 11/730 specific and
0000    44 ;       locates the memory controller on the system,
0000    45 ;       determines how much memory it controls, which pages of that
0000    46 ;       are present (and good).  Then the routine sets bits
0000    47 ;       in the PFN bitmap to represent each present (and good)
0000    48 ;       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 11/730 system bus are called slots, and are
0000     62 ;       identified by slot numbers 0-16.  The first slot is fixed and
0000     63 ;       is the memory controller.  The remaining 15 slots are floating.
0000     64 ;
0000     65 ;       The 11/730 currently supports 1 memory controller, located at
0000     66 ;       ^XF20000.  The 11/730 does not have interleaved memory.
0000     67 ;
0000     68 ;       Memory controller registers also contain the starting page
0000     69 ;       number / 128.  This routine determines where the end of memory
0000     70 ;       on a memory controller is by analyzing the memory present map
0000     71 ;       in the third memory controller register.
0000     72 ;
0000     73 ;
0000     74 ; OUTPUTS:
0000     75 ;
0000     76 ;       R7, R8, R11, and SP are preserved
0000     77 ;       All others (including AP and FP) are altered
0000     78 ;
0000     79 ; IMPLICIT OUTPUTS:
0000     80 ;
0000     81 ;       The PFN bitmap is modified to describe all of physical memory.
0000     82 ;
0000     83 ;       RPB$L_PFNCNT stores the number of pages of physical memory.
0000     84 ;
0000     85 ;       All single parity errors in memory are cleared.
0000     86 ;
0000     87 ;       RPB$B_CONFREG describes each NEXUS on the system bus with an
0000     88 ;       adapter type code.
0000     89 ;
0000     90 ; AUTHOR:
0000     91 ;
0000     92 ;       C. A. Samuelson, creation date 24-April-1981
0000     93 ;
0000     94 ; REVISION HISTORY:
0000     95 ;
0000     96 ;       V03-004 TCM0005         Trudy C. Matthews       27-Apr-1983
0000     97 ;               Change sense of CRDTEST flag from an enable to an inhibit; i.e.
0000     98 ;               remove pages with CRD errors by default.
0000     99 ;
0000    100 ;       V03-003 TCM0004         Trudy C. Matthews       9-Feb-1983
0000    101 ;               Fix bug in TCM0002.
0000    102 ;
0000    103 ;       V03-002 TCM0003         Trudy C. Matthews       26-Jan-1983
0000    104 ;               Add support for RPB$V_CRDTEST flag; report pages with CRD
0000    105 ;               errors as bad if this flag is set.
0000    106 ;
0000    107 ;       V03-001 KDM0078         Kathleen D. Morse       15-Mar-1982
0000    108 ;               Clear all MA780-specific boot flags.
0000    109 ;--
```

```
                        .SBTTL  Declarations

                        .DEFAULT DISPLACEMENT, WORD

                ;
                ; Macros to describe VMS data structures
                ;

                        $DMPDEF                         ; System dump file header definitions
                        $I0730DEF                       ; 11/730 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   136          .SBTTL  CHECKMEM_730, Identify 11/730 memory
                                0000   137
                                0000   138 ;++
                                0000   139 ;
                                0000   140 ; CHECKMEM_730, Locate and test memory for 11/730
                                0000   141 ;
                                0000   142 ;--
                                0000   143
                                0000   144 ;
                                0000   145 ; The table that follows identifies the adapter type codes of adapters
                                0000   146 ; located in fixed and floating 730 slots.
                                0000   147 ;
                                0000   148
                                0000   149 ADAP_TYPE_730:                          ; Adapter type table for 11/730
                             10 0000   150          .BYTE   NDT$_MEM16NI            ; Memory controller 0.
00'00'00'00'00'00'00'00'00'00'00'00'00' 0001   151          .BYTE   0 [15]                  ; Floating slot.
                    00'00'00'           000D
                                0010   152
                                0010   153
                                0010   154 ;
                                0010   155 ; Get address of slot assignment table.
                                0010   156 ;
                                0010   157
                                0010   158 CHECKMEM_730::                          ; Entry for 11/730
                  52   ED AF   9E  0010   159          MOVAB   ADAP_TYPE_730,R2        ; Get 11/730 fixed slot assignment
                                0014   160                                          ; table
                                0014   161 ;
                                0014   162 ; Start testing slot positions to find adapters. First save the stack
                                0014   163 ; position so it can be restored after a machine check.
                                0014   164 ;
                                0014   165
                                0014   166 INIT_SEARCH:                            ; Start searching for adapters
                     5C   10    9A  0014   167          MOVZBL  #IO730$AL_NNEX,AP       ; Set up nexus loop counter
                     5D   5E    D0  0017   168          MOVL    SP,FP                   ; Save current top of stack.
                                001A   169
                                001A   170 ;
                                001A   171 ; Set up the physical address of the 1st slot on the system bus and
                                001A   172 ; the address of the adapter type table.
                                001A   173 ;
                                001A   174
                  54   00F20000 9F  9E  001A   175          MOVAB   @#IO730$AL_IOBASE,R4    ; Get address of 1st slot
                                0021   176
                                0021   177 ;
                                0021   178 ; During this memory locate and test loop, the following registers are
                                0021   179 ; used:
                                0021   180 ;
                                0021   181 ;       R0      - the contents of the slot's configuration register;
                                0021   182 ;                 the 3rd memory controller register;
                                0021   183 ;       R1      - bit position within memory present map;
                                0021   184 ;       R2      - address of the next byte in the 730-specific adapter
                                0021   185 ;                 type table
                                0021   186 ;       R3      - the default adapter type for the current slot;
                                0021   187 ;                 number of last page on a controller
                                0021   188 ;       R4      - address of the configuration register at the current
                                0021   189 ;                 slot position
                                0021   190 ;       R5      - address of next byte in RPB adapter type table
                                0021   191 ;       R7      - address of the SCB
```

```
                                0021   192 ;       R9      - bit setting in memory present map;
                                0021   193 ;                 starting PFN on a controller (always 0)
                                0021   194 ;       R10     - address of the memory description list in RPB (pagcnt & pfn)
                                0021   195 ;       R11     - address of the RPB
                                0021   196 ;
                                0021   197 ; Initialize the RPB slot field to a zero and obtain the default adapter
                                0021   198 ; type for this slot. Then set up a machine check fault handler to gain
                                0021   199 ; control if the loop addresses a non-existent configuration register
                                0021   200 ; (an empty slot). Then read the slot's configuration register.
                                0021   201 ;
                                0021   202 ; Initialize R10 (RPB  memory descripter list pointer) for search loop
                                0021   203 ;
        5A    00BC CB   9E      0021   204         MOVAB   RPB$L_MEMDSC(R11),R10   ; Set pointer to memory description list
30 AB    00005800 8F   CA       0026   205         BICL    #<RPB$M_MPM ! RPB$M_USEMPM ! RPB$M_FINDMEM>, -
                                002E   206                 RPB$L_BOOTR5(R11)       ; Clear all MA780-specific boot flags
                                002E   207
                                002E   208
                                002E   209 TRY_NEXUS_730:                           ; Memory locate and test loop.
                     85   94    002E   210         CLRB    (R5)+                   ; Assume nothing on slot.
                53   82   9A    0030   211         MOVZBL  (R2)+,R3                ; Get default adapter type.
     04 A7    009D'CF   9E      0033   212         MOVAB   DO_NEXT_730+1,4(R7)     ; Set up fault handler (+1 for
                                0039   213                                         ; handler execution on the
                                0039   214                                         ; interrupt stack).
                50   64   D0    0039   215         MOVL    (R4),R0                 ; Read CR at current slot.
                                003C   216
                                003C   217 ;
                                003C   218 ; Execution continues here if the configuration register is present.
                                003C   219 ; Load the adapter type into the RPB field. Then, if the adapter type
                                003C   220 ; is a memory controller, proceed to test memory. Otherwise, move to
                                003C   221 ; the next SBI slot.
                                003C   222 ;
                                003C   223
                     53   D5    003C   224         TSTL    R3                      ; Is this a floating slot?
                     06   12    003E   225         BNEQ    FIXED_SLOT              ; Branch if not
  FF A5              50   90    0040   226         MOVB    R0,-1(R5)               ; Save type read from config register
                     04   11    0044   227         BRB     CHECK_TYPE              ; Check if memory controller
                                0046   228
                                0046   229 FIXED_SLOT:                            ; Slot is fixed assignment
  FF A5              53   90    0046   230         MOVB    R3,-1(R5)               ; Save fixed type
                                004A   231
                                004A   232 CHECK_TYPE:                            ; Check adapter type for memory
                                004A   233
10    FF A5              91    004A   234         CMPB    -1(R5),#NDT$_MEM16NI    ; Memory controller?
                     4C   12    004E   235         BNEQ    DO_NEXT_730             ; No, advance to next slot.
                                0050   236
                                0050   237 ;
                                0050   238 ; Memory controller found:
                                0050   239 ; Find out whether the memory addresses are within legal bounds by
                                0050   240 ; computing the number of pages on the controller.
                                0050   241 ; Confirm that the ending page number is less or equal than the maximum
                                0050   242 ; page number for this machine.
                                0050   243 ;
                                0050   244
50    08 A4              D0    0050   245         MOVL    8(R4),R0                ; Get starting address register.
                     53   D4    0054   246         CLRL    R3                      ; Start with zero 128K chunks.
                51   0F   D0    0056   247         MOVL    #15,R1                  ; Start with top bit in map.
                                0059   248
```

```
                                0059   249 10$:                                ; Bit extraction loop.
      59   50   01   51   EF    0059   250         EXTZV   R1,#1,R0,R9         ; Extract one bit from map.
                53   59   C0    005E   251         ADDL    R9,R3              ; Add it to 128K chunk total.
                     F5   51   F4    0061   252         SOBGEQ  R1,10$             ; Move to next bit until done.
           53   53        08   78    0064   253         ASHL    #8,R3,R3           ; Convert number of 128K byte
                                0068   254                                    ; chunks to number of pages.
              04   50        18   E1    0068   255         BBC     #24,R0,20$         ; Branch if 16K memory
           53      53        02   78    006C   256         ASHL    #2,R3,R3           ; Multiply by four for 64K chip memory
                    8A        53   D0    0070   257 20$:    MOVL    R3,(R10)+          ; Save # of pages in this memory
                                0073   258         ASSUME  DMP$V_TR EQ 24
                                0073   259         ASSUME  DMP$S_TR EQ 8
                       FF     AA   94    0073   260         CLRB    -1(R10)            ; Store TR # in mem descr (always 0)
                              8A   D4    0076   261         CLRL    (R10)+             ; Starting PFN for this memory (always 0)
                              59   D4    0078   262         CLRL    R9                 ; Starting address always 0
                                007A   263
                                007A   264 ;
                                007A   265 ; Before starting memory test, establish a page skipping handler for
                                007A   266 ; machine checks.
                                007A   267 ; Also, enable CRD error reporting if not inhibited by RPB BOOTR5 flag.
                                007A   268 ;
                                007A   269
           08 30 AB    10   E0    007A   270         BBS     #RPB$V_CRDTEST, -          ; Branch if CRD test inhibited.
                                007F   271                 RPB$L_BOOTR5(R11),30$      ;
   04 A4    10000000 8F   C8    007F   272         BISL    #^X10000000,4(R4)          ; Enable CRD reporting in memory CSR1.
                                0087   273 30$:
      04 A7    00D9'CF   9E    0087   274         MOVAB   PAGE_MCHECK_730+1,4(R7) ; Set page skipping handler (+1
                                008D   275                                    ; for interrupt stack).
                    52   DD    008D   276         PUSHL   R2                     ; Save address of fixed nexus table.
           52       00B7'CF   9E    008F   277         MOVAB   TEST_QUAD_730,R2       ; Routine to test one page
                       FF69'   30    0094   278         BSBW    BOO$TEST_MEM           ; Test the specified range of PFN's
                         04    BA    0097   279         POPR    #^M<R2>                ; Retrieve fixed nexus table address.
                         13    11    0099   280         BRB     ALL_DONE_730           ; Only one memory controller
                                009B   281
                                009B   282         .ALIGN  LONG                   ; Longword-aligned handler.
                                009C   283
                                009C   284 ;
                                009C   285 ; Fault handler for non-existent configuration register, or unreadable
                                009C   286 ; registers, or a non-memory controller slot device. Restore stack
                                009C   287 ; pointer, clear all errors, and try for another slot if any remain.
                                009C   288 ;
                                009C   289
                                009C   290 DO_NEXT_730:                         ; Skip to next slot.
            5E   5D   D0    009C   291         MOVL    FP,SP                  ; Restore stack pointer.
   26    FFFFFFFF 8F   DA    009F   292         MTPR    #-1,#PR$_MCESR         ; Clear any faults.
      54    2000 C4   9E    00A6   293         MOVAB   IO730$AL_PERNEX(R4),R4 ; Move to next slot.
              80 5C   F5    00AB   294         SOBGTR  AP,TRY_NEXUS_730       ; If still a slot, loop.
                                00AE   295
                                00AE   296 ;
                                00AE   297 ; Reestablish the normal machine check fault handler.
                                00AE   298 ;
                                00AE   299
                                00AE   300 ALL_DONE_730:
   04 A7    0001'CF   DE    00AE   301         MOVAL   UNEXP_MCHK+1,4(R7)     ; Reset SCB vector.
                    8A   D4    00B4   302         CLRL    (R10)+                 ; Indicate end of RPB memory descr list
                         05    00B6   303         RSB                            ; Return to main routine.
```

```
                                00B7   305 ;++
                                00B7   306 ;
                                00B7   307 ; Functional Description:
                                00B7   308 ;
                                00B7   309 ;       Test a page of 730 memory.
                                00B7   310 ;
                                00B7   311 ; Calling Sequence:
                                00B7   312 ;
                                00B7   313 ;       JSB     TEST_QUAD_730
                                00B7   314 ;
                                00B7   315 ; Inputs:
                                00B7   316 ;
                                00B7   317 ;       R0 = starting address to test
                                00B7   318 ;       R1 = Quad word iteration count (64)
                                00B7   319 ;       R11= Address of RPB
                                00B7   320 ;
                                00B7   321 ; Outputs:
                                00B7   322 ;
                                00B7   323 ;       Returns via RSB if the entire page is OK
                                00B7   324 ;       Error exit via Machine Check code to BOO$PAGE_MCHECK
                                00B7   325 ;
                                00B7   326 ;--
                                00B7   327
                                00B7   328 TEST_QUAD_730:                       ; Test 1 quadword at a time.
                          60    7C 00B7   329         CLRQ    (R0)              ; Clear a quadword.
                        80  80    D1 00B9   330         CMPL    (R0)+,(R0)+       ; Read both longwords, and
                                00BC   331                                       ; advance to next quadword.
                                00BC   332
                                00BC   333 ;
                                00BC   334 ; If no gross errors occur in the clear to the quadword or in the
                                00BC   335 ; subsequent read instruction, then execution continues below. Otherwise
                                00BC   336 ; execution goes to the fault handler.
                                00BC   337 ;
                                00BC   338
                             F8 51    F5 00BC   339         SOBGTR  R1,TEST_QUAD_730  ; Continue clearing unless done.
                             10    E1 00BF   340         BBC     #RPB$V_CRDTEST,-  ; Check if user inhibited removing
                          01 30 AB       00C1   341                 RPB$L_BOOTR5(R11),10$ ; pages with CRD errors.
                                05 00C4   342 5$:     RSB                       ; If so, return success.
                                00C5   343
                                00C5   344 ;
                                00C5   345 ; Check for CRD error.
                                00C5   346 ;
                                00C5   347 10$:
                          7E 00F20004 9F    D0 00C5   348         MOVL    @#IO730$AL_IOBASE+4,-(SP); Get memory CSR 1.
                          8E 40000000 8F    D3 00CC   349         BITL    #^X40000000,(SP)+ ; Check for CRD error.
                                EF    13 00D3   350         BEQL    5$                ; Branch if no CRD error occurred.
                                08    11 00D5   351         BRB     ERR_EXIT_730      ; CRD error bit is read-to-clear.
                                00D7   352
                                00D7   353         .ALIGN  LONG                    ; All handlers longword-aligned.
                                00D8   354
                                00D8   355 ;
                                00D8   356 ; Handler that gains control when a page has gross memory errors.
                                00D8   357 ;
                                00D8   358
                                00D8   359 PAGE_MCHECK_730:                     ; Handle machine check.
                          26 FFFFFFFF 8F    DA 00D8   360         MTPR    #-1,#PR$_MCESR    ; Clear error indicator.
                                00DF   361 ERR_EXIT_730:
```

```
                                                 I 7
FF1E'  31  00DF   362          BRW     BOO$PAGE_MCHECK          ; Exit to common machine check handler
           00E2   363          .END
```

```
ADAP_TYPE_730                   00000000 R      02
ALL_DONE_730                    000000AE R      02
BOO$PAGE_MCHECK                 ******** X      02
BOO$TEST_MEM                    ******** X      02
CHECKMEM_730                    00000010 RG     02
CHECK_TYPE                      0000004A R      02
DMP$S_TR                  =     00000008
DMP$V_TR                  =     00000018
DO_NEXT_730                     0000009C R      02
ERR_EXIT_730                    000000DF R      02
FIXED_SLOT                      00000046 R      02
INIT_SEARCH                     00000014 R      02
IO730$AL_IOBASE           =     00F20000
IO730$AL_NNEX             =     00000010
IO730$AL_PERNEX           =     00002000
NDT$_MEMT6NI              =     00000010
OP$_ACBD                  =     0000006F
OP$_ACBF                  =     0000004F
OP$_ACBG                  =     00004FFD
OP$_ACBH                  =     00006FFD
OP$_ADDD2                 =     00000060
OP$_ADDD3                 =     00000061
OP$_ADDF2                 =     00000040
OP$_ADDF3                 =     00000041
OP$_ADDG2                 =     000040FD
OP$_ADDG3                 =     000041FD
OP$_ADDH2                 =     000060FD
OP$_ADDH3                 =     000061FD
OP$_ADDP4                 =     00000020
OP$_ADDP6                 =     00000021
OP$_ASHP                  =     000000F8
OP$_CLRD                  =     0000007C
OP$_CLRF                  =     000000D4
OP$_CLRG                  =     0000007C
OP$_CLRH                  =     00007CFD
OP$_CMPD                  =     00000071
OP$_CMPF                  =     00000051
OP$_CMPG                  =     000051FD
OP$_CMPH                  =     000071FD
OP$_CMPP3                 =     00000035
OP$_CMPP4                 =     00000037
OP$_CRC                   =     0000000B
OP$_CVTBD                 =     0000006C
OP$_CVTBF                 =     0000004C
OP$_CVTBG                 =     00004CFD
OP$_CVTBH                 =     00006CFD
OP$_CVTDB                 =     00000068
OP$_CVTDF                 =     00000076
OP$_CVTDH                 =     000032FD
OP$_CVTDL                 =     0000006A
OP$_CVTDW                 =     00000069
OP$_CVTFB                 =     00000048
OP$_CVTFD                 =     00000056
OP$_CVTFG                 =     000099FD
OP$_CVTFH                 =     000098FD
OP$_CVTFL                 =     0000004A
OP$_CVTFW                 =     00000049
OP$_CVTGB                 =     000048FD
OP$_CVTGF                 =     000033FD
OP$_CVTGH                 =     000056FD
OP$_CVTGL                 =     00004AFD
OP$_CVTGW                 =     000049FD
OP$_CVTHB                 =     000068FD
OP$_CVTHD                 =     0000F7FD
OP$_CVTHF                 =     0000F6FD
OP$_CVTHG                 =     000076FD
OP$_CVTHL                 =     00006AFD
OP$_CVTHW                 =     000069FD
OP$_CVTLD                 =     0000006E
OP$_CVTLF                 =     0000004E
OP$_CVTLG                 =     00004EFD
OP$_CVTLH                 =     00006EFD
OP$_CVTLP                 =     000000F9
OP$_CVTPL                 =     00000036
OP$_CVTPS                 =     00000008
OP$_CVTPT                 =     00000024
OP$_CVTRDL                =     0000006B
OP$_CVTRFL                =     0000004B
OP$_CVTRGL                =     00004BFD
OP$_CVTRHL                =     00006BFD
OP$_CVTSP                 =     00000009
OP$_CVTTP                 =     00000026
OP$_CVTWD                 =     0000006D
OP$_CVTWF                 =     0000004D
OP$_CVTWG                 =     00004DFD
OP$_CVTWH                 =     00006DFD
OP$_DIVD2                 =     00000066
OP$_DIVD3                 =     00000067
OP$_DIVF2                 =     00000046
OP$_DIVF3                 =     00000047
OP$_DIVG2                 =     000046FD
OP$_DIVG3                 =     000047FD
OP$_DIVH2                 =     000066FD
OP$_DIVH3                 =     000067FD
OP$_DIVP                  =     00000027
OP$_EDITPC                =     00000038
OP$_EMODD                 =     00000074
OP$_EMODF                 =     00000054
OP$_EMODG                 =     000054FD
OP$_EMODH                 =     000074FD
OP$_MATCHC                =     00000039
OP$_MNEGD                 =     00000072
OP$_MNEGF                 =     00000052
OP$_MNEGG                 =     000052FD
OP$_MNEGH                 =     000072FD
OP$_MOVD                  =     00000070
OP$_MOVF                  =     00000050
OP$_MOVG                  =     000050FD
OP$_MOVH                  =     000070FD
OP$_MOVP                  =     00000034
OP$_MOVTC                 =     0000002E
OP$_MOVTUC                =     0000002F
OP$_MULD2                 =     00000064
OP$_MULD3                 =     00000065
```

```
OP$_MULF2                       =  00000044
OP$_MULF3                       =  00000045
OP$_MULG2                       =  000044FD
OP$_MULG3                       =  000045FD
OP$_MULH2                       =  000064FD
OP$_MULH3                       =  000065FD
OP$_MULP                        =  00000025
OP$_POLYD                       =  00000075
OP$_POLYF                       =  00000055
OP$_POLYG                       =  000055FD
OP$_POLYH                       =  000075FD
OP$_SCANC                       =  0000002A
OP$_SKPC                        =  0000003B
OP$_SPANC                       =  0000002B
OP$_SUBD2                       =  00000062
OP$_SUBD3                       =  00000063
OP$_SUBF2                       =  00000042
OP$_SUBF3                       =  00000043
OP$_SUBG2                       =  000042FD
OP$_SUBG3                       =  000043FD
OP$_SUBH2                       =  000062FD
OP$_SUBH3                       =  000063FD
OP$_SUBP4                       =  00000022
OP$_SUBP6                       =  00000023
OP$_TSTD                        =  00000073
OP$_TSTF                        =  00000053
OP$_TSTG                        =  000053FD
OP$_TSTH                        =  000073FD
PAGE_MCHECK_730                    000000D8 R     02
PR$_MCESR                       =  00000026
RPB$L_BOOTR5                    =  00000030
RPB$L_MEMDSC                    =  000000BC
RPB$M_FINDMEM                   =  00004000
RPB$M_MPM                       =  00000800
RPB$M_USEMPM                    =  00001000
RPB$V_CRDTEST                   =  00000010
TEST_QUAD_730                      000000B7 R     02
TRY_NEXUS_730                      0000002E 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	000000E2 (226.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.10	00:00:00.57
Command processing	111	00:00:00.77	00:00:03.52
Pass 1	412	00:00:11.53	00:00:28.82
Symbol table sort	0	00:00:00.88	00:00:00.96
Pass 2	75	00:00:03.78	00:00:12.34
Symbol table output	20	00:00:00.14	00:00:00.15
Psect synopsis output	1	00:00:00.03	00:00:00.36
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	656	00:00:17.24	00:00:46.73

The working set limit was 1500 pages.
51772 bytes (102 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 629 non-local and 5 local symbols.
3115 source lines were read in Pass 1, producing 13 object records in Pass 2.
141 pages of virtual memory were used to define 140 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
_255DUA28:[SYS.OBJ]LIB.MLB;1	3
_255DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	10

700 GETS were required to define 10 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:BTMEM730/OBJ=OBJ$:BTMEM730 MASD$:[EMULAT.SRC]MISSING/UPDATE=(MASD$:[EMULAT.ENH]MISSING)+MASD$:[BOOTS.SRC]BTMEM730/UPD
```

CONFIG
LIS

BTMEM855
LIS

BTMEM790
LIS

CONFIGURE
LIS

BOOTDEF
LIS

BOOTIO
LIS

CONFIGMN
LIS

BOOTDRIVR
LIS

BTMEM730
LIS

BTMEM750
LIS

BTMEM780
LIS

BOOTBLOCK
LIS