```
UUU          UUU  VVV         VVV        111       RRRRRRRRRRR        000000000     MMM         MMM
UUU          UUU  VVV         VVV        111       RRRRRRRRRRRR       000000000     MMM         MMM
UUU          UUU  VVV         VVV        111       RRRRRRRRRRRR       000000000     MMM         MMM
UUU          UUU  VVV         VVV     111111       RRR         RRR  000         000  MMMMMMM   MMMMMMM
UUU          UUU  VVV         VVV     111111       RRR         RRR  000         000  MMMMMMM   MMMMMMM
UUU          UUU  VVV         VVV     111111       RRR         RRR  000         000  MMMMMMM   MMMMMMM
UUU          UUU  VVV         VVV        111       RRR         RRR  000         000  MMM  MMM  MMM
UUU          UUU  VVV         VVV        111       RRR         RRR  000         000  MMM  MMM  MMM
UUU          UUU  VVV         VVV        111       RRR         RRR  000         000  MMM  MMM  MMM
UUU          UUU  VVV         VVV        111       RRRRRRRRRRRR     000         000  MMM       MMM
UUU          UUU  VVV         VVV        111       RRRRRRRRRRRR     000         000  MMM       MMM
UUU          UUU  VVV         VVV        111       RRRRRRRRRRRR     000         000  MMM       MMM
UUU          UUU  VVV         VVV        111       RRR   RRR        000         000  MMM       MMM
UUU          UUU  VVV         VVV        111       RRR   RRR        000         000  MMM       MMM
UUU          UUU  VVV         VVV        111       RRR   RRR        000         000  MMM       MMM
UUU          UUU     VVV   VVV           111       RRR      RRR     000         000  MMM       MMM
UUU          UUU     VVV   VVV           111       RRR      RRR     000         000  MMM       MMM
UUU          UUU     VVV   VVV           111       RRR      RRR     000         000  MMM       MMM
UUUUUUUUUUUUUUUU        VVV            111111111    RRR         RRR    000000000     MMM       MMM
UUUUUUUUUUUUUUUU        VVV            111111111    RRR         RRR    000000000     MMM       MMM
UUUUUUUUUUUUUUUU        VVV            111111111    RRR         RRR    000000000     MMM       MMM
UUUUUUUUUUUUUUUU        VVV            111111111    RRR         RRR    000000000     MMM       MMM
```

```
BBBBBBBB      000000      000000    TTTTTTTTTT  DDDDDDDD    RRRRRRRR    UU      UU  VV      VV      11
BBBBBBBB      000000      000000    TTTTTTTTTT  DDDDDDDD    RRRRRRRR    UU      UU  VV      VV      11
BB      BB  00      00  00      00      TT      DD      DD  RR      RR  UU      UU  VV      VV    1111
BB      BB  00      00  00      00      TT      DD      DD  RR      RR  UU      UU  VV      VV    1111
BB      BB  00      00  00      00      TT      DD      DD  RR      RR  UU      UU  VV      VV      11
BB      BB  00      00  00      00      TT      DD      DD  RR      RR  UU      UU  VV      VV      11
BBBBBBBB    00      00  00      00      TT      DD      DD  RRRRRRRR    UU      UU  VV      VV      11
BBBBBBBB    00      00  00      00      TT      DD      DD  RRRRRRRR    UU      UU  VV      VV      11
BB      BB  00      00  00      00      TT      DD      DD  RR  RR      UU      UU  VV      VV      11
BB      BB  00      00  00      00      TT      DD      DD  RR  RR      UU      UU  VV      VV      11
BB      BB  00      00  00      00      TT      DD      DD  RR    RR    UU      UU    VV  VV        11          ....
BB      BB  00      00  00      00      TT      DD      DD  RR    RR    UU      UU    VV  VV        11          ....
BBBBBBBB      000000      000000        TT      DDDDDDDD    RR      RR  UUUUUUUUUU      VV        111111        ....
BBBBBBBB      000000      000000        TT      DDDDDDDD    RR      RR  UUUUUUUUUU      VV        111111        ....

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

Table of contents

```
00000001  0000      1         BOOT_UV1_SWITCH = 1             ; Build Micro-VAX I bootstrap emulator
00000001  0000      2         PQ              == 1
                    1         .NLIST  CND
                    3         .TITLE  BOOTDRUV1 - DISPATCHER FOR BOOTSTRAP I/O DRIVERS FOR MICRO-VAX I
                    8         .IDENT  'V03-008'
                    9
                   10 ;
                   11 ;*****************************************************************************
                   12 ;*                                                                           *
                   13 ;*  COPYRIGHT (c) 1978, 1980, 1982 BY                                        *
                   14 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                   15 ;*  ALL RIGHTS RESERVED.                                                     *
                   16 ;*                                                                           *
                   17 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                   18 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
                   19 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
                   20 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                   21 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
                   22 ;*  TRANSFERRED.                                                             *
                   23 ;*                                                                           *
                   24 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
                   25 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
                   26 ;*  CORPORATION.                                                             *
                   27 ;*                                                                           *
                   28 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
                   29 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                   30 ;*                                                                           *
                   31 ;*                                                                           *
                   32 ;*****************************************************************************
                   33 ;
                   34 ;++
                   35 ;
                   36 ;  FACILITY:
                   37 ;
                   38 ;       Minimal bootstrap driver for all VMS system disks.
                   39 ;
                   40 ; ENVIRONMENT:
                   41 ;
                   42 ;       Runs at IPL 31, kernel mode, memory management may be on or off,
                   43 ;       IS=1 (running on interrupt stack), code must be PIC.
                   44 ;
                   45 ; ABSTRACT:
                   46 ;
                   47 ;       This module contains a routine called BOO$QIO that handles I/O
                   48 ;       transfers to and from the VMS system disks.
                   49 ;
                   50 ; AUTHOR:
                   51 ;
                   52 ;       The VMS group
                   53 ;
                   54 ; REVISION HISTORY:
                   55 ;
                   56 ;       V03-008 KDM0070         Kathleen D. Morse       15-Aug-1983
                   57 ;               Create a Micro-VAX I version for use in VMBUVI.EXE.  It
                   58 ;               has no buffered datapath or map register logic.
                   59 ;
                   60 ;       V03-007 KDM0073         Kathleen D. Morse       22-Aug-1983
```

```
0000     61 ;                  Add EXE$GL_TENUSEC and EXE$GL_UBDELAY to the fixed
0000     62 ;                  data cells used by the bootstrap drivers.  Create
0000     63 ;                  BQO symbols for these data cells.
0000     64 ;
0000     65 ;          V03-006 TCM0004         Trudy C. Matthews       02-Aug-1983
0000     66 ;                  Add definition for EXE$GB_CPUDATA cell.
0000     67 ;
0000     68 ;          V03-005 KTA3059         Kerbey T. Altmann       21-Jun-1983
0000     69 ;                  Add entries for unit disconnect and boot device name -
0000     70 ;                  thus bumping VMB version number.
0000     71 ;
0000     72 ;          V03-004 RLRCPUDISP      Robert L. Rappaport     15-Jun-1983
0000     73 ;                  Recode CPUDISP macros to use new format.
0000     74 ;
0000     75 ;          V03-003 TCM0003         Trudy C. Matthews       23-Feb-1983
0000     76 ;                  Increment VMB version number to indicate adding RPB$L_BADPGS
0000     77 ;                  field.
0000     78 ;
0000     79 ;          V03-002 TCM0002         Trudy C. Matthews       05-Jan-1983
0000     80 ;                  Add 11/790-specific path to BOO$PURDPR.
0000     81 ;
0000     82 ;          V03-001 KTA0092         Kerbey T. Altmann       02-Apr-1982
0000     83 ;                  Bump the version number because of KTA0090.
0000     84 ;
0000     85 ;          V02-021 KTA0090         Kerbey T. Altmann       26-Mar-1982
0000     86 ;                  Add new cell to IOVEC to contain address of microcode
0000     87 ;                  required by a booting device.
0000     88 ;
0000     89 ;--
```

```
                        .SBTTL  Declarations
                ;
                ; MACRO LIBRARY CALLS
                ;

                        $BQODEF                         ; Define boot qio offsets
                        $BTDDEF                         ; Define boot device types
                        $IODEF                          ; DEFINE I/O FUNCTION CODES
                        $MBADEF                         ; DEFINE MASSBUS ADAPTER REGISTERS
                        $NDTDEF                         ; NEXUS device types
                        $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                        $PTEDEF                         ; DEFINE PAGE TABLE ENTRY FIELDS
                        $RPBDEF                         ; DEFINE RESTART PARAMETER BLOCK
                        $SSDEF                          ; DEFINE STATUS CODES
                        $UBADEF                         ; UNIBUS ADAPTER REGISTER DEFINITIONS
                        $UBIDEF                         ; 11/750 UNIBUS adapter regs.
                        $VADEF                          ; DEFINE VIRTUAL ADDRESS FIELDS

                ;
                ; MACROS
                ;

                ;
                ; LOCAL SYMBOLS
                ;

                        $DEFINI BDT                     ; Define Boot Driver Table offsets

0000            $DEF    BDT$L_CPUTYPE   .BLKW   1       ;  CPU type
0002            $DEF    BDT$L_DEVTYPE   .BLKW   1       ;  Boot RO device type
0004            $DEF    BDT$L_ACTION    .BLKL   1       ;  Action routine
0008            $DEF    BDT$L_SIZE      .BLKL   1       ;  Driver size
000C            $DEF    BDT$L_ADDR      .BLKL   1       ;  Driver address (offset)
0010            $DEF    BDT$L_ENTRY     .BLKL   1       ;  Driver entry (offset from address)
0014            $DEF    BDT$L_DRIVRNAME .BLKL   1       ;  Driver name (offset from address)
0018            $DEF    BDT$L_AUXDRNAME .BLKL   1       ;  Auxiliary driver name (offset)
001C            $DEF    BDT$L_UNIT_INIT .BLKL   1       ;  Driver unit init (offset from address)
0020            $DEF    BDT$L_UNIT_DISC .BLKL   1       ;  Driver unit disc (offset from address)
0024            $DEF    BDT$L_DEVNAME   .BLKL   1       ;  Device name (offset from address)
0028
00000028        BDT$K_LENGTH=.                          ;  Length of entry

                        $DEFEND BDT                     ; End of Boot Driver Table definitions
```

```
                     0000    136          .SBTTL  DRIVER FIXED DATA AREA
                     0000    137
                     0000    138
                     0000    139 ;        FIXED DATA CELLS FOR BOOTSTRAP DRIVER
                     0000    140 ;
                     0000    141
                00000000     142          .PSECT  BOOTDRIVR_1,LONG        ; CERTAIN DRIVERS REQUIRE ALIGNMENT!
                     0000    143
                     0000    144 BOO$AL_VECTOR::                          ; VECTOR TO BOOT DRIVER ENTRY POINTS
00000046'            0000    145          .LONG   BOO$QIO-BOO$AL_VECTOR   ; OFFSET TO BOOTSTRAP QIO ROUTINE
000000A6'            0004    146          .LONG   BOO$MAP-BOO$AL_VECTOR   ; OFFSET TO MAPPING ROUTINE
00000000'            0008    147          .LONG   BOO$SELECT-BOO$AL_VECTOR; OFFSET TO BOOTSTRAP I/O DRIVER
                     000C    148                                          ; INITIALLY SET TO ROUTINE WHICH
                     000C    149                                          ; SELECTS DRIVER
00000000             000C    150          .LONG   0                       ; OFFSET TO SYSTEM DISK DRIVER NAME
                     0010    151                                          ; (ASCIC STRING). SET UP BY BOOT DRIVER.
                     0010    152 ;
                     0010    153 ; The next two words are the version number and the version number check fields.
                     0010    154 ; (The second word is the ones complement of the first word.)  The version
                     0010    155 ; number should be incremented whenever the interface between VMB and the
                     0010    156 ; rest of the system changes.  Release 1.0 VMB did not contain these fields.
                     0010    157 ;
                     0010    158 ;       Version 2 - Boot driver passes system disk driver name to SYSBOOT
                     0010    159 ;       Version 3 - VMB build memory description vector into RPB
                     0010    160 ;       Version 4 - VMB BOOTDRIVR purges UBA buffered datapath, all drivers
                     0010    161 ;                   return to BOOTDRIVR with success/failure status
                     0010    162 ;       Version 5 - VMB passes an argument list to the secondary boot
                     0010    163 ;                   in AP.  FILEREAD cacheing is present.
                     0010    164 ;       Version 6 - VMB passes nexus device type of boot adapter in
                     0010    165 ;                   RPB$B_BOOTNDT.
                     0010    166 ;       Version 7 - BOO$AL_VECTOR now has new entry points for RESELECTing
                     0010    167 ;                   a driver and UNIT_INIT for a driver.  Also new info
                     0010    168 ;                   passed in the argument list.
                     0010    169 ;       Version 8 - BOO$AL_VECTOR now has a new cell: BQO$L_UCODE.
                     0010    170 ;       Version 9 - VMB passes number of bad memory pages found during
                     0010    171 ;                   bootstrap scan in RPB$L_BADPGS.
                     0010    172 ;       Version 10- BOO$AL_VECTOR has two new cells: UNIT_DISC and DEVNAME
                     0010    173 ;
                     0010    174 ;       Version 11- BOO$AL_VECTOR has two new cells:  TENUSEC and UBDELAY
                     0010    175 ;
                     0010    176          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$W_VERSION
FFF4 0008            0010    177          .WORD   11,^C<11>                   ; VERSION # AND VERSION # CHECK FIELD.
00000063'            0014    178          .LONG   BOO$RESELECT-BOO$AL_VECTOR ; Offset to set new driver
00000012'            0018    179          .LONG   BOO$MOVE-BOO$AL_VECTOR  ; Offset to routine to select and move
                     001C    180          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UNIT_INIT
00000000             001C    181          .LONG   0                       ; Offset to UNIT_INIT
                     0020    182          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_AUXDRNAME
00000000             0020    183          .LONG   0                       ; Offset to auxiliary driver name
                     0024    184                                          ;  second driver
                     0024    185          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_DIS
                     0024    186 BOO$GL_UMR_DIS::                        ; Number of map registers disabled
00000000             0024    187          .LONG   0
                     0028    188          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UCODE
                     0028    189 BOO$GL_UCODE::                          ; Address of microcode in memory
00000000             0028    190          .LONG   0
                     002C    191          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UNIT_DISC
00000000             002C    192          .LONG   0                       ; Offset to UNIT_DISC
```

```
                     0030   193          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_DEVNAME
00000000             0030   194          .LONG   0                       ; Offset to boot device name
                     0034   195          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_TMPL
                     0034   196 BOO$GL_UMR_TMPL::                        ; ONIBUS map register template
80000000             0034   197          .LONG   UBA$M_MAP_VALID         ;  Default is valid, no buff data path
                     0038   198          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_UMR_DP
                     0038   199 BOO$GB_UMR_DP::                          ; ONIBUS map register data path
                01   0038   200          .BYTE   1                       ;  Default is buffered #1
                     0039   201          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_CPUTYPE
                     0039   202 EXE$GB_CPUTYPE::                         ; Location to hold processor
                01   0039   203          .BYTE   1                       ;  identification code
                     003A   204          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_CPUDATA
                     003A   205 EXE$GB_CPUDATA::                         ; Location to hold contents of SID.
00000001             003A   206          .LONG   1
                     003E   207          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_TENUSEC
                     003E   208 EXE$GL_TENUSEC::                         ; Location to hold TIMEDWAIT delay count
00000001             003E   209          .LONG   1                       ;  Default is value needed for Micro-VAX
                     0042   210          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UBDELAY
                     0042   211 EXE$GL_UBDELAY::                         ; Location to hold TIMEDWAIT delay count
00000001             0042   212          .LONG   1                       ;  Default is value needed for Micro-VAX
```

```
                        0046   214          .SBTTL  BOO$QIO - BOOTSTRAP QIO ROUTINE
                        0046   215
                        0046   216 ;++
                        0046   217 ; FUNCTIONAL DESCRIPTION:
                        0046   218 ;
                        0046   219 ;       BOO$QIO PROVIDES THE DEVICE INDEPENDENT I/O INTERFACE FOR BOTH
                        0046   220 ;       READING AND WRITING THE BOOTSTRAP DEVICE.
                        0046   221 ;
                        0046   222 ; CALLING SEQUENCE:
                        0046   223 ;
                        0046   224 ;       CALLG   ARGLIST,BOO$QIO
                        0046   225 ;
                        0046   226 ; INPUT PARAMETERS:
                        0046   227 ;
                        0046   228 ;       BUF(AP)  - BUFFER ADDRESS
                        0046   229 ;       SIZE(AP) - SIZE OF BUFFER IN BYTES
                        0046   230 ;       LBN(AP)  - LOGICAL BLOCK NUMBER
                        0046   231 ;       FUNC(AP) - FUNCTION CODE
                        0046   232 ;                       ACCEPTS IO$_READLBLK AND IO$_WRITELBLK
                        0046   233 ;       MODE(AP) - ADDRESS INTERPRETATION MODE
                        0046   234 ;                       0 => PHYSICAL, 1 => VIRTUAL
                        0046   235 ;       RPB(AP) - ADDRESS OF RESTART PARAMETER BLOCK
                        0046   236 ;
                        0046   237 ; OUTPUT PARAMETERS:
                        0046   238 ;
                        0046   239 ;       R0 - COMPLETION STATUS CODE
                        0046   240 ;       R1 - TOTAL BYTES TRANSFERRED
                        0046   241 ;
                        0046   242 ;--
                        0046   243
                        0046   244 ;
                        0046   245 ; Offsets from AP to input arguments:
                        0046   246 ;
                        0046   247
                00000004 0046   248          BUF     = 4
                00000008 0046   249          SIZE    = 8
                0000000C 0046   250          LBN     = 12
                00000010 0046   251          FUNC    = 16
                00000014 0046   252          MODE    = 20
                00000018 0046   253          RPB     = 24
                        0046   254
                        0046   255 BOO$QIO::
                    0FFC 0046   256          .WORD   ^M<R2,R3,R4,R5,R6,R7,-  ; PRESERVE REGISTERS
                        0048   257                  R8,R9,R10,R11>
                        0048   258
                        0048   259 ;
                        0048   260 ; If mapping is enabled, the processor register RP$_MAPEN contains a 1.
                        0048   261 ; Otherwise, the register contains a 0. Use this value as an index to
                        0048   262 ; choose the appropriate address of the adapter's register space.
                        0048   263 ;
                        0048   264
          59  18 AC   D0  0048   265          MOVL    RPB(AP),R9              ; GET BASE ADDRESS OF RESTART PARAMETER BLK
                        004C   421 ;
                        004C   422 ; For the QBUS, if the transfer MODE is physical, the BUF address contains
                        004C   423 ; the physical address of a SIZE physically contiguous bytes. The Driver
                        004C   424 ; assumes this.
                        004C   425 ;
```

```
                                                004C   426 ; If the MODE is 1 then the BUF address is translated into two components:
                                                004C   427 ; A physical PTE base address and a "virtual" address relative to that PTE.
                                                004C   428 ;
                                                004C   429
                5A    04 AC    D0               004C   430          movl    buf(ap),r10             ; get basic buffer address
                5B    0C AC    D0               0050   431          movl    lbn(ap),r11             ; get LBN
                58    08 AC    3C               0054   432          movzwl  size(ap),r8             ; get the size
                         04    12               0058   433          bneq    10$                     ; if neg then size < 64K
             58    01    10    78               005A   434          ashl    #16,#1,r8               ; set 64K
                         52    D4               005E   435 10$:     clrl    r2                      ; assume no PTE base address
                   26 14 AC    E9               0060   436          blbc    mode(ap),30$            ; br if mode is physical
                52    50 A9    D0               0064   437          movl    rpb$l_svaspt(r9),r2     ; assume system address
                                                0068   438                                          ; address SYSPTE
                                                0068   439                                          ; This address is physical if
                                                0068   440                                          ; mapping is OFF and a VA if ON
                03 5A    1F    E2               0068   441          bbss    #va$v_system,r10,20$    ; br if system address
                                                006C   442                                          ; setting the bit is the key to the
                                                006C   443                                          ; QBUS that the transfer is mapped.
                   52    08    DB               006C   444          mfpr    #pr$_p0br,r2            ; Get the PHYA or VA of the P0 PT
                   50    38    DB               006F   445 20$:     mfpr    #pr$_mapen,r0           ; mapping enabled?
                         16    13               0072   446          beql    30$                     ; br if no, r2 is physical
                                                0074   447
                                                0074   448 ;
                                                0074   449 ; convert the R2 address which is system virtual to a physical address
                                                0074   450 ;
                                                0074   451
          50    52    15    09    EF           0074   452          extzv   #va$v_vpn,#va$s_vpn,r2,r0; get PFN from SYS VA
                50    50 B940    D0           0079   453          movl    @rpb$[_svaspt(r9)[r0],r0; get PTE for that page for SYS PTE
           50    FFE00000 8F    CA           007E   454          bicl    #^c<pte$m_pfn>,r0       ; isolate PFN
          52    17    09    50    F0           0085   455          insv    r0,#va$v_vpn,#va$s_vpn+2,r2; convert to PHYSICAL PTE base
                                                008A   457
                                                008A   458 ;
                                                008A   459 ; If it is a UNIBUS boot device, derive the address of the device's CSR.
                                                008A   460 ;
                                                008A   461
                                                008A   462          ASSUME  RPB$L_CSRVIR  EQ  RPB$L_CSRPHY+4
                                                008A   463
                50    38    DB               008A   464 30$:     MFPR    #PR$_MAPEN,R0           ; Check for mapping enabled
          57    54 A940    D0               008D   465          MOVL    RPB$[_CSRPHY(R9)[R0],R7 ; Get address of device's CSR
                                                0092   466
                                                0092   467 PUSH_RETRY:
                      0A    DD               0092   468          PUSHL   #10                     ; Push retry count on stack
                                                0094   469
                55    5B    D0               0094   470 10$:     MOVL    R11,R5                  ; Get a working copy of the block number
             50    34 A9    D0               0097   471          MOVL    RPB$L_IOVEC(R9),R0      ; Get address of boot vectors
                08 B040    16               009B   472          JSB     @BQO$[_SELECT(R0)[R0]   ; Call driver thru self-relative vector
                   03 50    E8               009F   483 100$:    BLBS    R0,200$                 ; Branch if success
                   EF 6E    F5               00A2   484 150$:    SOBGTR  (SP),10$                ; Retry if count > 0
                         04    00A5   485 200$:    RET                                     ; Return with final status in R0
```

```
                        00A6   487          .SBTTL  BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QIO
                        00A6   488
                        00A6   489 ;++
                        00A6   490 ; FUNCTIONAL DESCRIPTION:
                        00A6   491 ;       BOO$MAP IS CALLED TO INITIALIZE THE DATA BASE FOR BOO$QIO TO PERMIT
                        00A6   492 ;       IT TO FUNCTION WITH MEMORY MANAGEMENT ENABLED.  AN AREA OF SYSTEM
                        00A6   493 ;       PAGE TABLE MUST BE PROVIDED SO THAT THE CONFIGURATION REGISTERS AND
                        00A6   494 ;       UNIBUS I/O PAGE CAN BE MAPPED.
                        00A6   495 ;
                        00A6   496 ; CALLING SEQUENCE:
                        00A6   497 ;       CALLG   ARGLIST,BOO$MAP
                        00A6   498 ;
                        00A6   499 ; INPUT PARAMETERS:
                        00A6   500 ;       SVASPT(AP) - SYSTEM VIRTUAL ADDRESS OF THE SYSTEM PAGE TABLE
              00000004  00A6   501 ;       SVASPT  =    4
                        00A6   502 ;       VABASE(AP) - BASE VIRTUAL ADDRESS OF A 24 PAGE WINDOW TO MAP
                        00A6   503 ;                    THE ADAPTER CONFIGURATION REGISTERS AND UNIBUS
              00000008  00A6   504 ;       VABASE  =    8
                        00A6   505 ;                    I/O PAGE.
                        00A6   506 ;       RPB(AP) -    ADDRESS OF RESTART PARAMETER BLOCK (RPB) CONTAINING
                        00A6   507 ;                    BOOTSTRAP DEVICE DESCRIPTION.
              0000000C  00A6   508 ;       RPB     =    12
                        00A6   509 ;
                        00A6   510 ; OUTPUT PARAMETERS:
                        00A6   511 ;       NONE
                        00A6   512 ;
                        00A6   513 ;--
                        00A6   514
                00FC    00A6   515 BOO$MAP::        .WORD   ^M<R2,R3,R4,R5,R6,R7>   ;
        57    0C AC     D0     00A8   516          MOVL    RPB(AP),R7              ; GET BASE ADDRESS FOR RPB
        52    04 AC     D0     00AC   517          MOVL    SVASPT(AP),R2           ; GET BASE OF SPT
        50 A7    52     D0     00B0   518          MOVL    R2,RPB$L_SVASPT(R7)     ; AND SAVE IN DATA BASE
        53    08 AC     D0     00B4   519          MOVL    VABASE(AP),R3           ; GET VIRTUAL ADDRESS OF WINDOW
        60 A7    53     D0     00B8   520          MOVL    R3,RPB$L_ADPVIR(R7)     ; SET AS ADAPTER VIRTUAL ADDRESS
     50    53    15     09     00BC   525          EXTZV   #VA$V_VPN,#VA$S_VPN,R3,R0       ; GET BASE VIRTUAL PAGE
           51  6240     DE     00C1   526          MOVAL   (R2)[R0],R1             ; COMPUTE WORKING SPT POINTER
           55    10     D0     00C5   530          MOVL    #16,R5                  ; SET FOR 16 PAGES
54    54 A7  00001FFF 8F     CB     00C8   531          BICL3   #^X1FFF,RPB$L_CSRPHY(R7),R4     ; GET PHY ADDR OF I/O PAGE BASE
        54    54    17     9C     00D1   532          ROTL    #<32-9>,R4,R4           ; AND CONVERT TO PAGE NUMBER
                    0E     10     00D5   533          BSBB    FILLSPT                 ; STORE PTES INTO SPT
        50    54 A7    3C     00D7   534          MOVZWL  RPB$L_CSRPHY(R7),R0     ; GET I/O PAGE OFFSET
58 A7   FFFF3000 E043    9E     00DB   535          MOVAB   <^X1000-^XE000>(R0)[R3],RPB$L_CSRVIR(R7) ; SET VIRTUAL CSR ADDR
                    04     00E4   536          RET                             ;
                           00E5   537
                           00E5   538 ;++
                           00E5   539 ; FILLSPT
                           00E5   540 ;
                           00E5   541 ; INPUTS:
                           00E5   542 ;       R1 - POINTER TO CURRENT SPT ENTRY (UPDATED)
                           00E5   543 ;       R4 - PFN (UPDATED)
                           00E5   544 ;       R5 - COUNT OF PAGES TO FILL (UPDATED)
                           00E5   545 ;
                           00E5   546 FILLSPT:                                ;
81    54   90000000 8F     C9     00E5   547          BISL3   #<PTE$M_VALID!PTE$C_KW>,R4,(R1)+ ; STORE A PTE
                    54     D6     00ED   548          INCL    R4                      ; ADVANCE TO NEXT PFN
                 F3 55     F5     00EF   549          SOBGTR  R5,FILLSPT              ; STORE THEM ALL
                    05     00F2   550          RSB
```

```
                00F3    626          .ALIGN  LONG                       ; Alignment needed by some drivers!!!
000000F4        00F4    627 BOO$QIOSIZ=.-BOO$AL_VECTOR                  ; Size of boot QIO routine
                00F4    628
000000F4        00F4    629 BOO$DRIVER==.                               ; Start of boot driver (after
                00F4    630                                             ; it's been moved)
                00F4    631                                             ; NOTE:  Boot drivers must be in
                00F4    632                                             ; psect BOOTDRIVR_2
                00F4    633
            00000000    634          .PSECT  BOOTDRIVR_3
                0000    635
00000000        0000    636 BOO$DRIVER_TBL=.                            ; Boot driver table
                0000    637
            00000000    638          .PSECT  BOOTDRIVR_5
                0000    639
00000000        0000    640          .LONG   0                          ; End of boot driver table
                0004    641
            00000000    642          .PSECT  BOOTDRIVR_6
```

```
                     0000   644          .SBTTL  BOO$SELECT - Select boot driver
                     0000   645
                     0000   646 ;++
                     0000   647 ; FUNCTIONAL DESCRIPTION:
                     0000   648 ;
                     0000   649 ;       This routine is called the first time BOO$QIO calls a driver.
                     0000   650 ;       It searches the boot driver table to locate the proper driver.
                     0000   651 ;       The correct linkage is made in BOO$AL_VECTOR.
                     0000   652 ;       RPB$L_IOVECSZ is also stored with the size of BOO$QIO plus
                     0000   653 ;       the size of the driver.  The driver is then jumped to.
                     0000   654 ;
                     0000   655 ; CALLING SEQUENCE:
                     0000   656 ;
                     0000   657 ;       JSB     BOO$SELECT      (Actually called through self-relative
                     0000   658 ;                                vector in BOO$AL_VECTOR+BOO$L_SELECT)
                     0000   659 ;
                     0000   660 ; INPUT PARAMETERS:
                     0000   661 ;
                     0000   662 ;       R9      Address of the RPB
                     0000   663 ;
                     0000   664 ; OUTPUT PARAMETERS:
                     0000   665 ;
                     0000   666 ;       None
                     0000   667 ;
                     0000   668 ;--
                     0000   669
                     0000   670 BOO$SELECT:
007E 8F    BB        0000   671          PUSHR   #^M<R1,R2,R3,R4,R5,R6>
     5D    10        0004   672          BSBB    BOO$RESELECT                    ; Select the correct driver
007E 8F    BA        0006   673          POPR    #^M<R1,R2,R3,R4,R5,R6>
                     000A   674 ;
                     000A   675 ; Set up driver vector and jump to driver.
                     000A   676 ;
50     34 A9    D0   000A   677          MOVL    RPB$L_IOVEC(R9),R0              ; Get address of vectors
   08 B040    17   000E   678          JMP     @BOO$L_SELECT(R0)[R0]           ; Jump to driver
```

```
                     0012   680          .SBTTL  BOO$MOVE - Select and move boot driver
                     0012   681
                     0012   682 ;++
                     0012   683 ; FUNCTIONAL DESCRIPTION:
                     0012   684 ;
                     0012   685 ;       This routine is called after VMB is finished with a driver.
                     0012   686 ;       It searches the boot driver table to locate the proper driver.
                     0012   687 ;       The correct linkage is made in BOO$AL_VECTOR and driver moved.
                     0012   688 ;
                     0012   689 ; CALLING SEQUENCE:
                     0012   690 ;
                     0012   691 ;       JSB     BOO$MOVE        (Actually called through self-relative
                     0012   692 ;                                vector in BOO$AL_VECTOR+BQO$L_MOVE)
                     0012   693 ;
                     0012   694 ; INPUT PARAMETERS:
                     0012   695 ;
                     0012   696 ;       R9      Address of the RPB
                     0012   697 ;
                     0012   698 ; OUTPUT PARAMETERS:
                     0012   699 ;
                     0012   700 ;       None
                     0012   701 ;
                     0012   702 ;--
                     0012   703
                     0012   704 BOO$MOVE:
        00FE 8F   BB 0012   705          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7> ; Save registers
                4B   10 0016   706          BSBB    BOO$RESELECT            ; Select the correct driver
    56  0C B545   9E 0018   707          MOVAB   @BDT$L_ADDR(R5)[R5],R6  ; Address of current position
    54  00F4'CF   9E 001D   708          MOVAB   W^BOO$DRIVER,R4         ; Address of new position
    57  56   54   C3 0022   709          SUBL3   R4,R6,R7                ; Offset
                24   13 0026   710          BEQL    20$                     ; None, so don't move
64  66   08 A5   28 0028   711          MOVC3   BDT$L_SIZE(R5),(R6),(R4); Move driver
    54  0000'CF   9E 002D   712          MOVAB   W^BOO$AL_VECTOR,R4
      08 A4   57   C2 0032   713          SUBL2   R7,BQO$L_SELECT(R4)     ; Adjust offset
      0C A4   57   C2 0036   714          SUBL2   R7,BQO$L_DRIVRNAME(R4)
            20 A4   D5 003A   715          TSTL    BQO$L_AUXDRNAME(R4)     ; Is there one?
                04   13 003D   716          BEQL    10$                     ; No, don;t mess
      20 A4   57   C2 003F   717          SUBL2   R7,BQO$L_AUXDRNAME(R4)
            1C A4   D5 0043   718 10$:     TSTL    BQO$L_UNIT_INIT(R4)     ; Is there one?
                04   13 0046   719          BEQL    20$                     ; No, don;t mess
      1C A4   57   C2 0048   720          SUBL2   R7,BQO$L_UNIT_INIT(R4)
            2C A4   D5 004C   721 20$:     TSTL    BQO$L_UNIT_DISC(R4)     ; Is there one?
                04   13 004F   722          BEQL    30$                     ; No, don;t mess
      2C A4   57   C2 0051   723          SUBL2   R7,BQO$L_UNIT_DISC(R4)
            30 A4   D5 0055   724 30$:     TSTL    BQO$L_DEVNAME(R4)       ; Is there one?
                04   13 0058   725          BEQL    40$                     ; No, don;t mess
      30 A4   57   C2 005A   726          SUBL2   R7,BQO$L_DEVNAME(R4)
        00FE 8F   BA 005E   727 40$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7>
                05   0062   728          RSB
                     0063   729
                     0063   730 BOO$RESELECT:
    55  0000'CF   DE 0063   731          MOVAL   W^BOO$DRIVER_TBL,R5     ; Get address of boot driver table
      53  66 A9   9A 0068   732          MOVZBL  RPB$B_DEVTYP(R9),R3     ; Get value of boot device type
    54  0039'CF   9A 006C   733          MOVZBL  W^EXE$GB_CPUTYPE,R4     ; Get cpu type
    56  00F4'8F   3C 0071   734          MOVZWL  #<BOO$DRIVER-BOO$AL_VECTOR>,R6 ; Compute offset to driver table
                     0076   735 ;
                     0076   736 ; Determine if next driver in table is the correct one.
```

```
                     0076   737 ;
                     0076   738 10$:    CVTWL   BDT$L_CPUTYPE(R5),R0    ; Get cpu type from table
                  7B 13     0079   739          BEQL    400$            ; End of table
                  05 19     007B   740          BLSS    20$             ; Driver doesn't care about cpu type
               54 50 D1     007D   741          CMPL    R0,R4           ; Cpu types match?
                  17 12     0080   742          BNEQ    40$             ; No, try next driver
                     0082   743
            50 02 A5 32     0082   744 20$:    CVTWL   BDT$L_DEVTYPE(R5),R0    ; Get boot device type from table
                  05 19     0086   745          BLSS    30$             ; Driver doesn't care about device type
               53 50 D1     0088   746          CMPL    R0,R3           ; Device types match?
                  0C 12     008B   747          BNEQ    40$             ; No, try next driver
                     008D   748
            50 04 A5 D0     008D   749 30$:    MOVL    BDT$L_ACTION(R5),R0     ; Get action routine offset from table
                  0F 13     0091   750          BEQL    60$             ; No action routine, this is the driver
                6540 16     0093   751          JSB     (R5)[R0]        ; Call action routine
              09 50 E8      0096   752          BLBS    R0,60$          ; Branch if this is the driver
            56 08 A5 C0     0099   753 40$:    ADDL    BDT$L_SIZE(R5),R6       ; Account for this driver's size
               55 28 C0     009D   754          ADDL    #BDT$K_LENGTH,R5        ; Point to next driver entry
                  D4 11     00A0   755          BRB     10$             ; Try next driver
                     00A2   756 ;
                     00A2   757 ; Have the right driver.  R5 points to driver table entry. R6 contains
                     00A2   758 ; accumulated offset from IOVEC to the start of the driver.  Update
                     00A2   759 ; pertainent entries in the IOVEC.
                     00A2   760 ;
         54 0000'CF DE       00A2   761 60$:    MOVAL   W^BOO$AL_VECTOR,R4      ; Cover the vector
          000000F4 8F C1     00A7   762          ADDL3   #BOO$QIOSIZ,-           ; Add boot QIO size to
                  08 A5      00AD   763                  BDT$L_SIZE(R5),-        ;  driver size
                  38 A9      00AF   764                  RPB$L_IOVECSZ(R9)       ;  and store in RPB
               10 A5 56 C1   00B1   765          ADDL3   R6,BDT$L_ENTRY(R5),-    ; Calc offset to driver
                  08 A4      00B5   766                  BQO$L_SELECT(R4)        ;  entry point and store in vector
               14 A5 56 C1   00B7   767          ADDL3   R6,BDT$L_DRIVRNAME(R5),-; Calc offset to driver
                  0C A4      00BB   768                  BQO$L_DRIVRNAME(R4)     ;  name and store in vector
                  1C A4 D4   00BD   769          CLRL    BQO$L_UNIT_INIT(R4)     ; Assume none
               51 1C A5 D0   00C0   770          MOVL    BDT$L_UNIT_INIT(R5),R1  ; Pick up possible UNIT_INIT entry
                     05 13   00C4   771          BEQL    70$             ; None specified, default to a RET
         1C A4 51 56 C1      00C6   772          ADDL3   R6,R1,BQO$L_UNIT_INIT(R4) ;Calc offset to driver
                     00CB   773                                          ;  UNIT_INIT point and store in vector
                  20 A4 D4   00CB   774 70$:    CLRL    BQO$L_AUXDRNAME(R4)     ; Assume none
               51 18 A5 D0   00CE   775          MOVL    BDT$L_AUXDRNAME(R5),R1  ; Pick up possible driver name
                     05 13   00D2   776          BEQL    80$             ; None specified, default to a zero
         20 A4 51 56 C1      00D4   777          ADDL3   R6,R1,BQO$L_AUXDRNAME(R4) ; Calc offset to driver
                     00D9   778                                          ;  auxiliary name and store in vector
                  2C A4 D4   00D9   779 80$:    CLRL    BQO$L_UNIT_DISC(R4)     ; Assume none
               51 20 A5 D0   00DC   780          MOVL    BDT$L_UNIT_DISC(R5),R1  ; Pick up possible UNIT DISC entry
                     05 13   00E0   781          BEQL    90$             ; None specified, default to a zero
         2C A4 51 56 C1      00E2   782          ADDL3   R6,R1,BQO$L_UNIT_DISC(R4) ; Calc offset to driver
                     00E7   783                                          ;  UNIT_DISC point and store in vector
                  30 A4 D4   00E7   784 90$:    CLRL    BQO$L_DEVNAME(R4)       ; Assume none
               51 24 A5 D0   00EA   785          MOVL    BDT$L_DEVNAME(R5),R1    ; Pick up possible device name
                     05 13   00EE   786          BEQL    100$            ; None specified, default to a zero
         30 A4 51 56 C1      00F0   787          ADDL3   R6,R1,BQO$L_DEVNAME(R4) ; Calc offset to device
                     00F5   788                                          ;  name and store in vector
                     05     00F5   789 100$:    RSB
                     00F6   790
                     00F6   791 ;
                     00F6   792 ; No driver in the driver table accepted this QIO
                     00F6   793 ;
```

```
 00  00F6    794 400$:   HALT
     00F7    795
     00F7    796         .END
```

```
BDT$K_LENGTH                    = 00000028
BDT$L_ACTION                      00000004
BDT$L_ADDR                        0000000C
BDT$L_AUXDRNAME                   00000018
BDT$L_CPUTYPE                     00000000
BDT$L_DEVNAME                     00000024
BDT$L_DEVTYPE                     00000002
BDT$L_DRIVRNAME                   00000014
BDT$L_ENTRY                       00000010
BDT$L_SIZE                        00000008
BDT$L_UNIT_DISC                   00000020
BDT$L_UNIT_INIT                   0000001C
BOO$AC_VECTOR                     00000000 RG     02
BOO$DRIVER                      = 000000F4 RG     02
BOO$DRIVER_TBL                  = 00000000 R      03
BOO$GB_UMR_DP                     00000038 RG     02
BOO$GL_UCODE                      00000028 RG     02
BOO$GL_UMR_DIS                    00000024 RG     02
BOO$GL_UMR_TMPL                   00000034 RG     02
BOO$MAP                           000000A6 RG     02
BOO$MOVE                          00000012 R      05
BOO$QIO                           00000046 RG     02
BOO$QIOSIZ                      = 000000F4
BOO$RESELECT                      00000063 R      05
BOO$SELECT                        00000000 R      05
BOOT_UV1_SWITCH                 = 00000001
BQO$B_CPUTYPE                   = 00000039
BQO$B_UMR_DP                    = 00000038
BQO$L_AUXDRNAME                 = 00000020
BQO$L_CPUDATA                   = 0000003A
BQO$L_DEVNAME                   = 00000030
BQO$L_DRIVRNAME                 = 0000000C
BQO$L_SELECT                    = 00000008
BQO$L_TENUSEC                   = 0000003E
BQO$L_UBDELAY                   = 00000042
BQO$L_UCODE                     = 00000028
BQO$L_UMR_DIS                   = 00000024
BQO$L_UMR_TMPL                  = 00000034
BQO$L_UNIT_DISC                 = 0000002C
BQO$L_UNIT_INIT                 = 0000001C
BQO$W_VERSION                   = 00000010
BUF                             = 00000004
EXE$GB_CPUDATA                    0000003A RG     02
EXE$GB_CPUTYPE                    00000039 RG     02
EXE$GL_TENUSEC                    0000003E RG     02
EXE$GL_UBDELAY                    00000042 RG     02
FILLSPT                           000000E5 R      02
FUNC                            = 00000010
LBN                             = 0000000C
MODE                            = 00000014
PQ                              = 00000001   G
PR$_MAPEN                       = 00000038
PR$_P0BR                        = 00000008
PTE$C_KW                        = 10000000
PTE$M_PFN                       = 001FFFFF
PTE$M_VALID                     = 80000000
PUSH_RETRY                        00000092 R      02
RPB                             = 0000000C
RPB$B_DEVTYP                    = 00000066
RPB$L_ADPVIR                    = 00000060
RPB$L_CSRPHY                    = 00000054
RPB$L_CSRVIR                    = 00000058
RPB$L_IOVEC                     = 00000034
RPB$L_IOVECSZ                   = 00000038
RPB$L_SVASPT                    = 00000050
SIZE                            = 00000008
SVASPT                          = 00000004
UBA$M_MAP_VALID                 = 80000000
VA$S_VPN                        = 00000015
VA$V_SYSTEM                     = 0000001F
VA$V_VPN                        = 00000009
VABASE                          = 00000008
```

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	00000028 (40.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_1	000000F4 (244.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
BOOTDRIVR_3	00000000 (0.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_5	00000004 (4.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_6	000000F7 (247.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.06	00:00:00.36
Command processing	82	00:00:00.59	00:00:01.11
Pass 1	346	00:00:12.88	00:00:15.20
Symbol table sort	0	00:00:02.14	00:00:02.17
Pass 2	99	00:00:02.81	00:00:03.69
Symbol table output	8	00:00:00.08	00:00:00.08
Psect synopsis output	3	00:00:00.04	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	547	00:00:18.61	00:00:22.65

The working set limit was 1350 pages.
79271 bytes (155 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1343 non-local and 21 local symbols.
798 source lines were read in Pass 1, producing 19 object records in Pass 2.
21 pages of virtual memory were used to define 20 macros.

Macro library statistics

Macro library name	Macros defined
DISK$STARWORK03:[GAMACHE.UV1ROM.VMS]LIBUV1.ML	8
DISK$STARWORK03:[GAMACHE.UV1ROM.OBJ]VMB.MLB;3	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	16

1414 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MAC/LIS=LIS$:BOOTDRUV1/OBJ=OBJ$:BOOTDRUV1 VMS$:BOOUV1SWT+VMS$:BOOTDRIVR+OBJ$:VMB/LIB+VMS$:LIBUV1/LIB
```

LUV1ROM

PQBTDRIVR
LIS

QVSS
LIS

SEARCHMSG
LIS

VMBUVAX1
MAP

SETUSER
LIS

VMBUVAX1
LIS

BOOTDRUV1
LIS

FILERDUV1
LIS

BOOTIOUV1
LIS

CONIO
LIS

NETBOOT
LIS