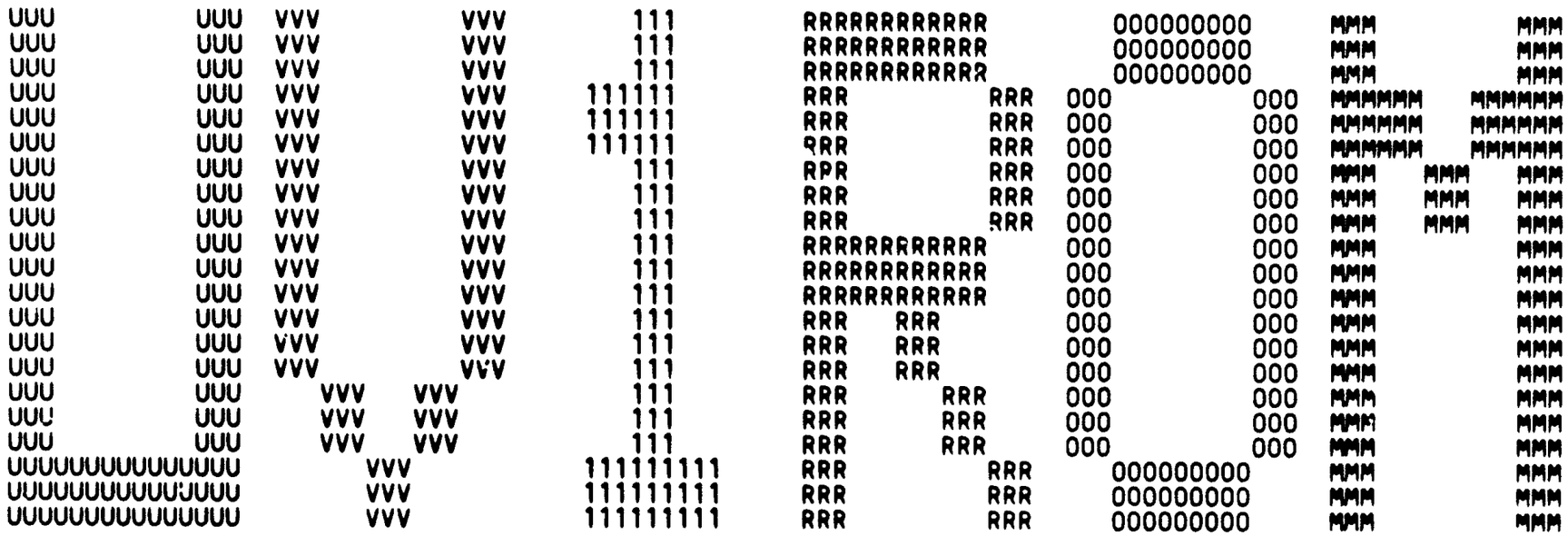
UUU UUU VVV VVV 111 RRRRRRRRRRR 000000000 MMM MMM
UUU UUU VVV VVV 111 RRRRRRRRRRR 000000000 MMM MMM
UUU UUU VVV VVV 111 RRRRRRRRRRR 000000000 MMM MMM
UUU UUU VVV VVV 111111 RRR RRR 000 000 MMMMMM MMMMMM
UUU UUU VVV VVV 111111 RRR RRR 000 000 MMMMMM MMMMMM
UUU UUU VVV VVV 111111 RRR RRR 000 000 MMMMMM MMMMMM
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 RRRRRRRRRRR 000 000 MMM MMM
UUU UUU VVV VVV 111 RRRRRRRRRRR 000 000 MMM MMM
UUU UUU VVV VVV 111 RRRRRRRRRRR 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
UUUUUUUUUUUUUUU VVV 111111111 RRR RRR 000000000 MMM MMM
UUUUUUUUUUUUUUU VVV 111111111 RRR RRR 000000000 MMM MMM
UUUUUUUUUUUUUUU VVV 111111111 RRR RRR 000000000 MMM MMM

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


LL              IIIIII      SSSSSSSS
LL              IIII I      SSSSSSSS
LL                I.      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
**NEW**   00000001  0000    .1          PQ              = 1
-1                  0000     1          .NLIST  CND
                    0000     3          .TITLE  BOOTDRUV1 - DISPATCHER FOR BOOTSTRAP I/O DRIVERS FOR MICRO-VAX I
                    0000     8          .IDENT  'V03-008'
                    0000     9
                    0000    10 ;
                    0000    11 ;*****************************************************************************
                    0000    12 ;*                                                                           *
                    0000    13 ;*  COPYRIGHT (c) 1978, 1980, 1982 BY                                        *
                    0000    14 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                    0000    15 ;*  ALL RIGHTS RESERVED.                                                     *
                    0000    16 ;*                                                                           *
                    0000    17 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                    0000    18 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                    0000    19 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
                    0000    20 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                    0000    21 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
                    0000    22 ;*  TRANSFERRED.                                                             *
                    0000    23 ;*                                                                           *
                    0000    24 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE      *
                    0000    25 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                    0000    26 ;*  CORPORATION.                                                             *
                    0000    27 ;*                                                                           *
                    0000    28 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                    0000    29 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                    0000    30 ;*                                                                           *
                    0000    31 ;*                                                                           *
                    0000    32 ;*****************************************************************************
                    0000    33 ;
                    0000    34 ;++
                    0000    35 ;
                    0000    36 ;  FACILITY:
                    0000    37 ;
                    0000    38 ;       Minimal bootstrap driver for all VMS system disks.
                    0000    39 ;
                    0000    40 ; ENVIRONMENT:
                    0000    41 ;
                    0000    42 ;       Runs at IPL 31, kernel mode, memory management may be on or off,
                    0000    43 ;       IS=1 (running on interrupt stack), code must be PIC.
                    0000    44 ;
                    0000    45 ; ABSTRACT:
                    0000    46 ;
                    0000    47 ;       This module contains a routine called BOO$QIO that handles I/O
                    0000    48 ;       transfers to and from the VMS system disks.
                    0000    49 ;
                    0000    50 ; AUTHOR:
                    0000    51 ;
                    0000    52 ;       The VMS group
                    0000    53 ;
                    0000    54 ; REVISION HISTORY:
                    0000    55 ;
                    0000    56 ;       V03-008 KDM0070         Kathleen D. Morse       15-Aug-1983
                    0000    57 ;               Create a Micro-VAX I version for use in VMBUVT.EXE.  It
                    0000    58 ;               has no buffered datapath or map register logic.
                    0000    59 ;
                    0000    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 KTA0093.
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 ;--
```

```
                        0000     91          .SBTTL  Declarations
                        0000     92
                        0000     93 ;
                        0000     94 ; MACRO LIBRARY CALLS
                        0000     95 ;
                        0000     96
                        0000     97          $BQODEF                         ; Define boot qio offsets
                        0000     98          $BTDDEF                         ; Define boot device types
                        0000     99          $IODEF                          ; DEFINE I/O FUNCTION CODES
                        0000    100          $MBADEF                         ; DEFINE MASSBUS ADAPTER REGISTERS
                        0000    101          $NDTDEF                         ; NEXUS device types
                        0000    102          $PRDEF                          ; DEFINE PROCESSOR REGISTERS
                        0000    103          $PTEDEF                         ; DEFINE PAGE TABLE ENTRY FIELDS
                        0000    104          $RPBDEF                         ; DEFINE RESTART PARAMETER BLOCK
                        0000    105          $SSDEF                          ; DEFINE STATUS CODES
                        0000    106          $UBADEF                         ; UNIBUS ADAPTER REGISTER DEFINITIONS
                        0000    107          $UBIDEF                         ; 11/750 UNIBUS adapter regs.
                        0000    108          $VADEF                          ; DEFINE VIRTUAL ADDRESS FIELDS
                        0000    109
                        0000    110 ;
                        0000    111 ; MACROS
                        0000    112 ;
                        0000    113
                        0000    114 ;
                        0000    115 ; LOCAL SYMBOLS
                        0000    116 ;
                        0000    117
                        0000    118          $DEFINI BDT                     ; Define Boot Driver Table offsets
                        0000    119
                        0000    120 $DEF     BDT$L_CPUTYPE    .BLKW   1      ;  CPU type
                        0002    121 $DEF     BDT$L_DEVTYPE    .BLKW   1      ;  Boot R0 device type
                        0004    122 $DEF     BDT$L_ACTION     .BLKL   1      ;  Action routine
                        0008    123 $DEF     BDT$L_SIZE       .BLKL   1      ;  Driver size
                        000C    124 $DEF     BDT$L_ADDR       .BLKL   1      ;  Driver address (offset)
                        0010    125 $DEF     BDT$L_ENTRY      .BLKL   1      ;  Driver entry (offset from address)
                        0014    126 $DEF     BDT$L_DRIVRNAME  .BLKL   1      ;  Driver name (offset from address)
                        0018    127 $DEF     BDT$L_AUXDRNAME  .BLKL   1      ;  Auxiliary driver name (offset)
                        001C    128 $DEF     BDT$L_UNIT_INIT  .BLKL   1      ;  Driver unit init (offset from address)
                        0020    129 $DEF     BDT$L_UNIT_DISC  .BLKL   1      ;  Driver unit disc (offset from address)
                        0024    130 $DEF     BDT$L_DEVNAME    .BLKL   1      ;  Device name (offset from address)
                        0028    131
00000028                0028    132 BDT$K_LENGTH=.                           ;  Length of entry
                        0028    133
                        0028    134          $DEFEND BDT                     ; End of Boot Driver Table definitions
```

```
                                                H 13
                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$AL_VECTOR; OFFSET TO BOOTSTRAP I/O DRIVER
                000C   148                                          ; INITIALLY SET TO POUTINE WHICH
                000C   149                                          ; SELECTS DRIVER
00000000        000C   150          .LONG   0                       ; OFFSET TO SYSTEM DISK DRIVER NAME
                0010   151                                          ; (ASCIC STRING). SET UP BY BOOT DRIVER.
                0010   152 ;
                0010   153 ; The next two words are the version number and the version number check fields.
                0010   154 ; (The second word is the ones complement of the first word.)  The version
                0010   155 ; number should be incremented whenever the interface between VMB and the
                0010   156 ; rest of the system changes.  Release 1.0 VMB did not contain these fields.
                0010   157 ;
                0010   158 ;       Version 2 - Boot driver passes system disk driver name to SYSBOOT
                0010   159 ;       Version 3 - VMB build memory description vector into RPB
                0010   160 ;       Version 4 - VMB BOOTDRIVR purges UBA buffered datapath, all drivers
                0010   161 ;                   return to BOOTDRIVR with success/failure status
                0010   162 ;       Version 5 - VMB passes an argument list to the secondary boot
                0010   163 ;                   in AP.  FILEREAD cacheing is present.
                0010   164 ;       Version 6 - VMB passes nexus device type of boot adapter in
                0010   165 ;                   RPB$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: BOO$L_UCODE.
                0010   170 ;       Version 9 - VMB passes number of bad memory pages found during
                0010   171 ;                   bootstrap scan in RPB$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 ;       Version 12- RPB$W_BOOTNDT is defined, high byte of this word must
                0010   177 ;                   be cleared in SYSBOOT for versions of VMB less than 12.
                0010   178 ;
                0010   179 ;       Version 13- RPB$B_CTRLLTR is defined; SYSBOOT must clear this field
                0010   180 ;                   for older versions of VMB.
                0010   181 ;
                0010   182          ASSUME  <.-BOO$AL_VECTOR> EQ BOO$W_VERSION
    FFF2 000D   0010   183          .WORD   13,^C<13>                   ; VERSION # AND VERSION # CHECK FIELD.
 00000063'      0014   184          .LONG   BOO$RESELECT-BOO$AL_VECTOR ; Offset to set new driver
 00000012'      0018   185          .LONG   BOO$MOVE-BOO$AL_VECTOR  ; Offset to routine to select and move
                001C   186          ASSUME  <.-BOO$AL_VECTOR> EQ BOO$L_UNIT_INIT
 00000000       001C   187          .LONG   0                       ; Offset to UNIT_INIT
                0020   188          ASSUME  <.-BOO$AL_VECTOR> EQ BOO$L_AUXDRNAME
 00000000       0020   189          .LONG   0                           ; Offset to auxiliary driver name
                0024   190                                              ;  second driver
                0024   191          ASSUME  <.-BOO$AL_VECTOR> EQ BOO$L_UMR_DIS
                0024   192 BOO$GL_UMR_DIS::                           ; Number of map registers disabled
```

```
                                                I 13

00000000  0024   193          .LONG   0
          0028   194          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UCODE
          0028   195 BOO$GL_UCODE::                              ; Address of microcode in memory
00000000  0028   196          .LONG   0
          002C   197          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UNIT_DISC
00000000  002C   198          .LONG   0                          ; Offset to UNIT_DISC
          0030   199          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_DEVNAME
00000000  0030   200          .LONG   0                          ; Offset to boot device name
          0034   201          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_TMPL
          0034   202 BOO$GL_UMR_TMPL::                           ; UNIBUS map register template
80000000  0034   203          .LONG   UBA$M_MAP_VALID            ;  Default is valid, no buff data path
          0038   204          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_UMR_DP
          0038   205 BOO$GB_UMR_DP::                            ; UNIBUS map register data path
       01 0038   206          .BYTE   1                          ;  Default is Buffered #1
          0039   207          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_CPUTYPE
          0039   208 EXE$GB_CPUTYPE::                           ; Location to hold processor
       01 0039   209          .BYTE   1                          ;  identification code
          003A   210          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_CPUDATA
          003A   211 EXE$GB_CPUDATA::                           ; Location to hold contents of SID.
00000001  003A   212          .LONG   1
          003E   213          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_TENUSEC
          003E   214 EXE$GL_TENUSEC::                           ; Location to hold TIMEDWAIT delay count
00000001  003E   215          .LONG   1                          ;  Default is value needed for Micro-VAX
          0042   216          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UBDELAY
          0042   217 EXE$GL_UBDELAY::                           ; Location to hold TIMEDWAIT delay count
00000001  0042   218          .LONG   1                          ;  Default is value needed for Micro-VAX
```

```
                        0046   220          .SBTTL  BOO$QIO - BOOTSTRAP QIO ROUTINE
                        0046   221
                        0046   222 ;++
                        0046   223 ; FUNCTIONAL DESCRIPTION:
                        0046   224 ;
                        0046   225 ;       BOO$QIO PROVIDES THE DEVICE INDEPENDENT I/O INTERFACE FOR BOTH
                        0046   226 ;       READING AND WRITING THE BOOTSTRAP DEVICE.
                        0046   227 ;
                        0046   228 ; CALLING SEQUENCE:
                        0046   229 ;
                        0046   230 ;       CALLG   ARGLIST,BOO$QIO
                        0046   231 ;
                        0046   232 ; INPUT PARAMETERS:
                        0046   233 ;
                        0046   234 ;       BUF(AP)  - BUFFER ADDRESS
                        0046   235 ;       SIZE(AP) - SIZE OF BUFFER IN BYTES
                        0046   236 ;       LBN(AP)  - LOGICAL BLOCK NUMBER
                        0046   237 ;       FUNC(AP) - FUNCTION CODE
                        0046   238 ;                  ACCEPTS IO$_READLBLK AND IO$_WRITELBLK
                        0046   239 ;       MODE(AP) - ADDRESS INTERPRETATION MODE
                        0046   240 ;                  0 => PHYSICAL, 1 => VIRTUAL
                        0046   241 ;       RPB(AP) - ADDRESS OF RESTART PARAMETER BLOCK
                        0046   242 ;
                        0046   243 ; OUTPUT PARAMETERS:
                        0046   244 ;
                        0046   245 ;       R0 - COMPLETION STATUS CODE
                        0046   246 ;       R1 - TOTAL BYTES TRANSFERRED
                        0046   247 ;
                        0046   248 ;--
                        0046   249
                        0046   250 ;
                        0046   251 ; Offsets from AP to input arguments:
                        0046   252 ;
                        0046   253
                00000004 0046   254          BUF     = 4
                00000008 0046   255          SIZE    = 8
                0000000C 0046   256          LBN     = 12
                00000010 0046   257          FUNC    = 16
                00000014 0046   258          MODE    = 20
                00000018 0046   259          RPB     = 24
                        0046   260
                        0046   261 BOO$QIO::
                    0FFC 0046   262          .WORD   ^M<R2,R3,R4,R5,R6,R7,-  ; PRESERVE REGISTERS
                        0048   263                  R8,R9,R10,R11>
                        0048   264
                        0048   265 ;
                        0048   266 ; If mapping is enabled, the processor register RP$_MAPEN contains a 1.
                        0048   267 ; Otherwise, the register contains a 0. Use this value as an index to
                        0048   268 ; choose the appropriate address of the adapter's register space.
                        0048   269 ;
                        0048   270
          59   18 AC   D0 0048   271          MOVL    RPB(AP),R9              ; GET BASE ADDRESS OF RESTART PARAMETER BLK
                        004C   427 ;
                        004C   428 ; For the QBUS, if the transfer MODE is physical, the BUF address contains
                        004C   429 ; the physical address of a SIZE physically contiguous bytes. The Driver
                        004C   430 ; assumes this.
                        004C   431 ;
```

```
                                004C   432 ; If the MODE is 1 then the BUF address is translated into two components:
                                004C   433 ; A physical PTE base address and a "virtual" address relative to that PTE.
                                004C   434 ;
                                004C   435
            5A    04 AC    D0   004C   436          movl    buf(ap),r10             ; get basic buffer address
            5B    0C AC    D0   0050   437          movl    lbn(ap),r11             ; get LBN
            58    08 AC    3C   0054   438          movzwl  size(ap),r8             ; get the size
                     04    12   0058   439          bneq    10$                     ; if neq then size < 64K
         58    01    10    78   005A   440          ashl    #16,#1,r8               ; set 64K
                     52    D4   005E   441 10$:     clrl    r2                      ; assume no PTE base address
               26 14 AC    E9   0060   442          blbc    mode(ap),30$            ; br if mode is physical
            52    50 A9    D0   0064   443          movl    rpb$l_svaspt(r9),r2     ; assume system address
                                0068   444                                          ; address SYSPTE
                                0068   445                                          ; This address is physical if
                                0068   446                                          ; mapping is OFF and a VA if ON
            03 5A    1F    E2   0068   447          bbss    #va$v_system,r10,20$    ; br if system address
                                006C   448                                          ; setting the bit is the key to the
                                006C   449                                          ; QBUS that the transfer is mapped.
               52    08    DB   006C   450          mfpr    #pr$_p0br,r2            ; Get the PHYA or VA of the P0 PT
               50    38    DB   006F   451 20$:     mfpr    #pr$_mapen,r0           ; mapping enabled?
                     16    13   0072   452          beql    30$                     ; br if no, r2 is physical
                                0074   453
                                0074   454 ;
                                0074   455 ; convert the R2 address which is system virtual to a physical address
                                0074   456 ;
                                0074   457
      50    52    15    09    EF   0074   458          extzv   #va$v_vpn,#va$s_vpn,r2,r0; get PFN from SYS VA
            50    50 B940     D0   0079   459          movl    @rpb$[_svaspt(r9)[r0],r0; get PTE for that page for SYS PTE
       50    FFE00000 8F      CA   007E   460          bicl    #^c<pte$m_pfn>,r0       ; isolate PFN
      52    17    09    50    F0   0085   461          insv    r0,#va$v_vpn,#va$s_vpn+2,r2; convert to PHYSICAL PTE base
                                008A   463
                                008A   464 ;
                                008A   465 ; If it is a UNIBUS boot device, derive the address of the device's CSR.
                                008A   466 ;
                                008A   467
                                008A   468          ASSUME  RPB$L_CSRVIR  EQ  RPB$L_CSRPHY+4
                                008A   469
               50    38    DB   008A   470 30$:     MFPR    #PR$_MAPEN,R0           ; Check for mapping enabled
      57    54 A940     D0   008D   471          MOVL    RPB$[_CSRPHY(R9)[R0],R7 ; Get address of device's CSR
                                0092   472
                                0092   473 PUSH_RETRY:
                     0A    DD   0092   474          PUSHL   #10                     ; Push retry count on stack
                                0094   475
               55    5B    D0   0094   476 10$:     MOVL    R11,R5                  ; Get a working copy of the block number
         50    34 A9     D0   0097   477          MOVL    RPB$L_IOVEC(R9),R0      ; Get address of boot vectors
               08 B040     16   009B   478          JSB     @BOO$[_SELECT(R0)[R0]   ; Call driver thru self-relative vector
                  03 50    E8   009F   489 100$:    BLBS    R0,200$                 ; Branch if success
                  EF 6E    F5   00A2   490 150$:    SOBGTR  (SP),10$                ; Retry if count > 0
                     04    00   00A5   491 200$:    RET                             ; Return with final status in R0
```

```
                                00A6   493          .SBTTL  BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QIO
                                00A6   494
                                00A6   495 ;++
                                00A6   496 ; FUNCTIONAL DESCRIPTION:
                                00A6   497 ;       BOO$MAP IS CALLED TO INITIALIZE THE DATA BASE FOR BOO$QIO TO PERMIT
                                00A6   498 ;       IT TO FUNCTION WITH MEMORY MANAGEMENT ENABLED.  AN AREA OF SYSTEM
                                00A6   499 ;       PAGE TABLE MUST BE PROVIDED SO THAT THE CONFIGURATION REGISTERS AND
                                00A6   500 ;       UNIBUS I/O PAGE CAN BE MAPPED.
                                00A6   501 ;
                                00A6   502 ; CALLING SEQUENCE:
                                00A6   503 ;       CALLG   ARGLIST,BOO$MAP
                                00A6   504 ;
                                00A6   505 ; INPUT PARAMETERS:
                                00A6   506 ;       SVASPT(AP) - SYSTEM VIRTUAL ADDRESS OF THE SYSTEM PAGE TABLE
                00000004        00A6   507 ;       SVASPT  =    4
                                00A6   508 ;       VABASE(AP) - BASE VIRTUAL ADDRESS OF A 24 PAGE WINDOW TO MAP
                                00A6   509 ;                    THE ADAPTER CONFIGURATION REGISTERS AND UNIBUS
                00000008        00A6   510 ;       VABASE  =    8
                                00A6   511 ;                    I/O PAGE.
                                00A6   512 ;       RPB(AP) -    ADDRESS OF RESTART PARAMETER BLOCK (RPB) CONTAINING
                                00A6   513 ;                    BOOTSTRAP DEVICE DESCRIPTION.
                0000000C        00A6   514 ;       RPB     =    12
                                00A6   515 ;
                                00A6   516 ; OUTPUT PARAMETERS:
                                00A6   517 ;       NONE
                                00A6   518 ;
                                00A6   519 ;--
                                00A6   520
                          00FC  00A6   521 BOO$MAP::      .WORD   ^M<R2,R3,R4,R5,R6,R7>   ;
                    57    0C AC   D0   00A8   522         MOVL    RPB(AP),R7              ; GET BASE ADDRESS FOR RPB
                    52    04 AC   D0   00AC   523         MOVL    SVASPT(AP),R2           ; GET BASE OF SPT
                    50 A7    52   D0   00B0   524         MOVL    R2,RPB$L_SVASPT(R7)     ; AND SAVE IN DATA BASE
                    53    08 AC   D0   00B4   525         MOVL    VABASE(AP),R3           ; GET VIRTUAL ADDRESS OF WINDOW
                    60 A7    53   D0   00B8   526         MOVL    R3,RPB$L_ADPVIR(R7)     ; SET AS ADAPTER VIRTUAL ADDRESS
          50    53    15    09   EF   00BC   531         EXTZV   #VA$V_VPN,#VA$S_VPN,R3,R0       ; GET BASE VIRTUAL PAGE
                51    6240   DE   00C1   532         MOVAL   (R2)[R0],R1             ; COMPUTE WORKING SPT POINTER
                55    10     D0   00C5   536         MOVL    #16,R5                  ; SET FOR 16 PAGES
54    54 A7   00001FFF 8F   CB   00C8   537         BICL3   #^X1FFF,RPB$L_CSRPHY(R7),R4     ; GET PHY ADDR OF I/O PAGE BASE
              54    54    17   9C   00D1   538         ROTL    #<32-9>,R4,R4           ; AND CONVERT TO PAGE NUMBER
                    0E    10   00D5   539         BSBB    FILLSPT                 ; STORE PTES INTO SPT
                50    54 A7   3C   00D7   540         MOVZWL  RPB$L_CSRPHY(R7),R0     ; GET I/O PAGE OFFSET
   58 A7    FFFF3000 E043   9E   00DB   541         MOVAB   <^X1000-^XE000>(R0)[R3],RPB$L_CSRVIR(R7) ; SET VIRTUAL CSR ADDR
                    04     10   00E4   542         RET                             ;
                           00E5   543
                           00E5   544 ;++
                           00E5   545 ; FILLSPT
                           00E5   546 ;
                           00E5   547 ; INPUTS:
                           00E5   548 ;       R1 - POINTER TO CURRENT SPT ENTRY (UPDATED)
                           00E5   549 ;       R4 - PFN (UPDATED)
                           00E5   550 ;       R5 - COUNT OF PAGES TO FILL (UPDATED)
                           00E5   551 ;
                           00E5   552 FILLSPT:                                     ;
   81    54    90000000 8F   C9   00E5   553         BISL3   #<PTE$M_VALID!PTE$C_KW>,R4,(R1)+ ; STORE A PTE
                     54    D6   00ED   554         INCL    R4                      ; ADVANCE TO NEXT PFN
                  F3 55    F5   00EF   555         SOBGTR  R5,FILLSPT              ; STORE THEM ALL
                           05   00F2   556         RSB
```

```
                00F3   632          .ALIGN  LONG                    ; Alignment needed by some drivers!!!
000000F4        00F4   633 BOO$QIOSIZ=.-BOO$AL_VECTOR               ; Size of boot QIO routine
                00F4   634
000000F4        00F4   635 BOO$DRIVER==.                            ; Start of boot driver (after
                00F4   636                                          ; it's been moved)
                00F4   637                                          ; NOTE:  Boot drivers must be in
                00F4   638                                          ; psect BOOTDRIVR_2
                00F4   639
            00000000   640          .PSECT  BOOTDRIVR_3
                0000   641
00000000        0000   642 BOO$DRIVER_TBL=.                         ; Boot driver table
                0000   643
            00000000   644          .PSECT  BOOTDRIVR_5
                0000   645
00000000        0000   646          .LONG   0                       ; End of boot driver table
                0000   647
            00000000   648          .PSECT  BOOTDRIVR_6
```

```
                     0000   650          .SBTTL  BOO$SELECT - Select boot driver
                     0000   651
                     0000   652 ;++
                     0000   653 ; FUNCTIONAL DESCRIPTION:
                     0000   654 ;
                     0000   655 ;       This routine is called the first time BOO$QIO calls a driver.
                     0000   656 ;       It searches the boot driver table to locate the proper driver.
                     0000   657 ;       The correct linkage is made in BOO$AL_VECTOR.
                     0000   658 ;       RPB$L_IOVECSZ is also stored with the size of BOO$QIO plus
                     0000   659 ;       the size of the driver.  The driver is then jumped to.
                     0000   660 ;
                     0000   661 ; CALLING SEQUENCE:
                     0000   662 ;
                     0000   663 ;       JSB     BOO$SELECT      (Actually called through self-relative
                     0000   664 ;                                vector in BOO$AL_VECTOR+BOO$L_SELECT)
                     0000   665 ;
                     0000   666 ; INPUT PARAMETERS:
                     0000   667 ;
                     0000   668 ;       R9      Address of the RPB
                     0000   669 ;
                     0000   670 ; OUTPUT PARAMETERS:
                     0000   671 ;
                     0000   672 ;       None
                     0000   673 ;
                     0000   674 ;--
                     0000   675
                     0000   676 BOO$SELECT:
007E 8F    BB        0000   677          PUSHR   #^M<R1,R2,R3,R4,R5,R6>
        5D   10      0004   678          BSBB    BOO$RESELECT                    ; Select the correct driver
007E 8F    BA        0006   679          POPR    #^M<R1,R2,R3,R4,R5,R6>
                     000A   680 ;
                     000A   681 ; Set up driver vector and jump to driver.
                     000A   682 ;
    50    34 A9      D0     000A   683          MOVL    RPB$L_IOVEC(R9),R0          ; Get address of vectors
       08 B040      17     000E   684          JMP     @BOO$L_SELECT(R0)[R0]       ; Jump to driver
```

```
                                0012   686          .SBTTL  BOO$MOVE - Select and move boot driver
                                0012   687
                                0012   688 ;++
                                0012   689 ; FUNCTIONAL DESCRIPTION:
                                0012   690 ;
                                0012   691 ;       This routine is called after VMB is finished with a driver.
                                0012   692 ;       It searches the boot driver table to locate the proper driver.
                                0012   693 ;       The correct linkage is made in BOO$AL_VECTOR and driver moved.
                                0012   694 ;
                                0012   695 ; CALLING SEQUENCE:
                                0012   696 ;
                                0012   697 ;       JSB     BOO$MOVE        (Actually called through self-relative
                                0012   698 ;                                vector in BOO$AL_VECTOR+BOO$L_MOVE)
                                0012   699 ;
                                0012   700 ; INPUT PARAMETERS:
                                0012   701 ;
                                0012   702 ;       R9      Address of the RPB
                                0012   703 ;
                                0012   704 ; OUTPUT PARAMETERS:
                                0012   705 ;
                                0012   706 ;       None
                                0012   707 ;
                                0012   708 ;--
                                0012   709
                                0012   710 BOO$MOVE:
          00FE 8F    BB         0012   711          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7> ; Save registers
                4B   10         0016   712          BSBB    BOO$RESELECT            ; Select the correct driver
     56   0C B545   9E         0018   713          MOVAB   @BDT$L_ADDR(R5)[R5],R6  ; Address of current position
     54   00F4'CF   9E         001D   714          MOVAB   W^BOO$DRIVER,R4         ; Address of new position
     57   56   54   C3         0022   715          SUBL3   R4,R6,R7                ; Offset
                24   13         0026   716          BEQL    20$                     ; None, so don't move
64   66   08 A5    28         0028   717          MOVC3   BDT$L_SIZE(R5),(R6),(R4); Move driver
     54   00D0'CF   9E         002D   718          MOVAB   W^BOO$AL_VECTOR,R4
   08 A4   57   C2         0032   719          SUBL2   R7,BOO$L_SELECT(R4)      ; Adjust offset
   0C A4   57   C2         0036   720          SUBL2   R7,BOO$L_DRIVRNAME(R4)
        20 A4   D5         003A   721          TSTL    BOO$L_AUXDRNAME(R4)     ; Is there one?
                04   13         003D   722          BEQL    10$                     ; No, don't mess
   20 A4   57   C2         003F   723          SUBL2   R7,BOO$L_AUXDRNAME(R4)
        1C A4   D5         0043   724 10$:     TSTL    BOO$L_UNIT_INIT(R4)     ; Is there one?
                04   13         0046   725          BEQL    20$                     ; No, don't mess
   1C A4   57   C2         0048   726          SUBL2   R7,BOO$L_UNIT_INIT(R4)
        2C A4   D5         004C   727 20$:     TSTL    BOO$L_UNIT_DISC(R4)     ; Is there one?
                04   13         004F   728          BEQL    30$                     ; No, don't mess
   2C A4   57   C2         0051   729          SUBL2   R7,BOO$L_UNIT_DISC(R4)
        30 A4   D5         0055   730 30$:     TSTL    BOO$L_DEVNAME(R4)       ; Is there one?
                04   13         0058   731          BEQL    40$                     ; No, don't mess
   30 A4   57   C2         005A   732          SUBL2   R7,BOO$L_DEVNAME(R4)
      00FE 8F    BA         005E   733 40$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7>
                05   0062   734          RSB
                       0063   735
                       0063   736 BOO$RESELECT:
55   0000'CF    DE         0063   737          MOVAL   W^BOO$DRIVER_TBL,R5     ; Get address of boot driver table
  53   66 A9    9A         0068   738          MOVZBL  RPB$B_DEVTYP(R9),R3     ; Get value of boot device type
54   0039'CF    9A         006C   739          MOVZBL  W^EXE$GB_CPUTYPE,R4     ; Get cpu type
56   00F4'8F    3C         0071   740          MOVZWL  #<BOO$DRIVER-BOO$AL_VECTOR>,R6 ; Compute offset to driver table
                       0076   741 ;
                       0076   742 ; Determine if next driver in table is the correct one.
```

```
                                0076   743 ;
                     50   65    32 0076   744 10$:    CVTWL   BDT$L_CPUTYPE(R5),R0     ; Get cpu type from table
                          7B    13 0079   745         BEQL    400$                     ; End of table
                          05    19 007B   746         BLSS    20$                      ; Driver doesn't care about cpu type
                     54   50    D1 007D   747         CMPL    R0,R4                    ; Cpu types match?
                          17    12 0080   748         BNEQ    40$                      ; No, try next driver
                                0082   749
                50   02 A5    32 0082   750 20$:    CVTWL   BDT$L_DEVTYPE(R5),R0     ; Get boot device type from table
                          05    19 0086   751         BLSS    30$                      ; Driver doesn't care about device type
                     53   50    D1 0088   752         CMPL    R0,R3                    ; Device types match?
                          0C    12 008B   753         BNEQ    40$                      ; No, try next driver
                                008D   754
                50   04 A5    D0 008D   755 30$:    MOVL    BDT$L_ACTION(R5),R0      ; Get action routine offset from table
                          0F    13 0091   756         BEQL    60$                      ; No action routine, this is the driver
                        6540    16 0093   757         JSB     (R5)[R0]                 ; Call action routine
                     09 50      E8 0096   758         BLBS    R0,60$                   ; Branch if this is the driver
                56   08 A5    C0 0099   759 40$:    ADDL    BDT$L_SIZE(R5),R6        ; Account for this driver's size
                     55   28   C0 009D   760         ADDL    #BDT$R_LENGTH,R5         ; Point to next driver entry
                          D4    11 00A0   761         BRB     10$                      ; Try next driver
                                00A2   762 ;
                                00A2   763 ; Have the right driver.  R5 points to driver table entry. R6 contains
                                00A2   764 ; accumulated offset from IOVEC to the start of the driver.  Update
                                00A2   765 ; pertainent entries in the IOVEC.
                                00A2   766 ;
          54   0000'CF          DE 00A2   767 60$:    MOVAL   W^BOO$AL_VECTOR,R4       ; Cover the vector
           000000F4 8F          C1 00A7   768         ADDL3   #BOO$QIOSIZ,-            ; Add boot QIO size to
                 08 A5             00AD   769                 BDT$L_SIZE(R5),-         ;  driver size
                 38 A9             00AF   770                 RPB$L_IOVECSZ(R9)        ;  and store in RPB
              10 A5    56    C1 00B1   771         ADDL3   R6,BDT$L_ENTRY(R5),-     ; Calc offset to driver
                 08 A4             00B5   772                 BQO$L_SELECT(R4)         ;  entry point and store in vector
              14 A5    56    C1 00B7   773         ADDL3   R6,BDT$L_DRIVRNAME(R5),- ; Calc offset to driver
                 0C A4             00BB   774                 BQO$L_DRIVRNAME(R4)      ;  name and store in vector
                 1C A4    D4 00BD   775         CLRL    BQO$L_UNIT_INIT(R4)      ; Assume none
                51   1C A5    D0 00C0   776         MOVL    BDT$L_UNIT_INIT(R5),R1   ; Pick up possible UNIT_INIT entry
                          05    13 00C4   777         BEQL    70$                      ; None specified, default to a RET
          1C A4    51    56    C1 00C6   778         ADDL3   R6,R1,BQO$L_UNIT_INIT(R4) ;Calc offset to driver
                                00CB   779                                          ;  UNIT_INIT point and store in vector
                 20 A4    D4 00CB   780 70$:    CLRL    BQO$L_AUXDRNAME(R4)      ; Assume none
                51   18 A5    D0 00CE   781         MOVL    BDT$L_AUXDRNAME(R5),R1   ; Pick up possible driver name
                          05    13 00D2   782         BEQL    80$                      ; None specified, default to a zero
          20 A4    51    56    C1 00D4   783         ADDL3   R6,R1,BQO$L_AUXDRNAME(R4) ; Calc offset to driver
                                00D9   784                                          ;  auxiliary name and store in vector
                 2C A4    D4 00D9   785 80$:    CLRL    BQO$L_UNIT_DISC(R4)      ; Assume none
                51   20 A5    D0 00DC   786         MOVL    BDT$L_UNIT_DISC(R5),R1   ; Pick up possible UNIT DISC entry
                          05    13 00E0   787         BEQL    90$                      ; None specified, default to a zero
          2C A4    51    56    C1 00E2   788         ADDL3   R6,R1,BQO$L_UNIT_DISC(R4) ; Calc offset to driver
                                00E7   789                                          ;  UNIT_DISC point and store in vector
                 30 A4    D4 00E7   790 90$:    CLRL    BQO$L_DEVNAME(R4)        ; Assume none
                51   24 A5    D0 00EA   791         MOVL    BDT$L_DEVNAME(R5),R1     ; Pick up possible device name
                          05    13 00EE   792         BEQL    100$                     ; None specified, default to a zero
          30 A4    51    56    C1 00F0   793         ADDL3   R6,R1,BQO$L_DEVNAME(R4)  ; Calc offset to device
                                00F5   794                                          ;  name and store in vector
                          05    00F5   795 100$:    RSB
                                00F6   796
                                00F6   797 ;
                                00F6   798 ; No driver in the driver table accepted this QIO
                                00F6   799 ;
```

```
00  00F6   800 400$:    HALT
    00F7   801
    00F7   802          .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
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                    = 000C0054
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	99	00:00:00.20	00:00:01.68
Command processing	129	00:00:00.81	00:00:06.38
Pass 1	413	00:00:15.10	00:00:47.28
Symbol table sort	0	00:00:02.16	00:00:05.90
Pass 2	122	00:00:02.91	00:00:09.32
Symbol table output	9	00:00:00.12	00:00:01.02
Psect synopsis output	3	00:00:00.04	00:00:00.14
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	777	00:00:21.34	00:01:11.72

The working set limit was 1950 pages.
79539 bytes (156 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1348 non-local and 21 local symbols.
804 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
_255DUA18:[UV1ROM.OBJ]LIBUV1.MLB;1	8
_255DUA18:[UV1ROM.OBJ]VMB.MLB;1	1
_255DUA18:[SYSLIB]STARLET.MLB;3	7
TOTALS (all libraries)	16

1419 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:BOOTDRUV1P/OBJ=OBJ$:BOOTDRUV1P MSRC$:BOOUV1SWT/UPDATE=(BUG$:BOOUV1SWT)+MSRC$:BOOTDRIVRP/UPDATE=(BUG$:BOOTDRIVRP)+LIB$

0451 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

UAFPARSE
LIS

BOOTDRUVIP
LIS

VMBUVAX.P
LIS

UVI.ROM

VMBUVAXIP
MAP

CONIO
LIS

UVMS

REMOVE
COM