D 12

```
UUU         UUU  VVV         VVV        111        RRRRRRRRRRRR        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  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        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
UUUUUUUUUUUUUUU        VVV           111111111     RRR         RRR     000000000     MMM         MMM
UUUUUUUUUUUUUUU        VVV           111111111     RRR         RRR     000000000     MMM         MMM
UUUUUUUUUUUUUUU        VVV           111111111     RRR         RRR     000000000     MMM         MMM
```

BO
Ta

```
**FILE**ID**VMBUVAX1P                                    C 15



VV      VV  MM      MM  BBBBBBBB    UU      UU  VV      VV    AAAAAA    XX      XX      11      PPPPPPPP
VV      VV  MM      MM  BBBBBBBB    UU      UU  VV      VV    AAAAAA    XX      XX      11      PPPPPPPP
VV      VV  MMMM  MMMM  BB      BB  UU      UU  VV      VV  AA      AA  XX      XX    1111      PP      PP
VV      VV  MMMM  MMMM  BB      BB  UU      UU  VV      VV  AA      AA  XX      XX    1111      PP      PP
VV      VV  MM  MM  MM  BB      BB  UU      UU  VV      VV  AA      AA    XX  XX        11      PP      PP
VV      VV  MM  MM  MM  BB      BB  UU      UU  VV      VV  AA      AA    XX  XX        11      PP      PP
VV      VV  MM      MM  BBBBBBBB    UU      UU  VV      VV  AA      AA      XX          11      PPPPPPPP
VV      VV  MM      MM  BBBBBBBB    UU      UU  VV      VV  AA      AA      XX          11      PPPPPPPP
VV      VV  MM      MM  BB      BB  UU      UU  VV      VV  AAAAAAAAAA    XX  XX        11      PP
VV      VV  MM      MM  BB      BB  UU      UU  VV      VV  AAAAAAAAAA    XX  XX        11      PP
  VV  VV    MM      MM  BB      BB  UU      UU    VV  VV    AA      AA  XX      XX      11      PP              ....
  VV  VV    MM      MM  BB      BB  UU      UU    VV  VV    AA      AA  XX      XX      11      PP              ....
    VV      MM      MM  BBBBBBBB    UUUUUUUUUU      VV      AA      AA  XX      XX    111111    PP              ....
    VV      MM      MM  BBBBBBBB    UUUUUUUUUU      VV      AA      AA  XX      XX    111111    PP              ....

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
```

VM
VC

Table of contents

```
0000          .title  VMB_MICROVAX_I
0000          .ident /V04.0-06/
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  Copyright (c) 1984                                                       *
0000    ;*  by DIGITAL Equipment Corporation, Maynard, Mass.                         *
0000    ;*                                                                           *
0000    ;*  This software is furnished under a license and may be used and  copied   *
0000    ;*  only  in  accordance  with  the  terms  of  such  license and with the   *
0000    ;*  inclusion of the above copyright notice.  This software or  any  other   *
0000    ;*  copies  thereof may not be provided or otherwise made available to any   *
0000    ;*  other person.  No title to and ownership of  the  software  is  hereby   *
0000    ;*  transferred.                                                             *
0000    ;*                                                                           *
0000    ;*  The information in this software is subject to change  without  notice   *
0000    ;*  and  should  not  be  construed  as  a commitment by DIGITAL Equipment   *
0000    ;*  Corporation.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL assumes no responsibility for the use or  reliability  of  its   *
0000    ;*  software on equipment which is not supplied by DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
0000    ; Facility:
0000    ;
0000    ;       Bootstrap for the MicroVAX I.
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module contains a modified version of the VMB which resides
0000    ;       on ROM in the MicroVAX I, and is intended to be used as an
0000    ;       intermediate bootstrap which is not restricted by the size of the
0000    ;       ROM and which allows the MicroVAX-I system to boot from floating-CSR
0000    ;       devices.  Should the ROM in the MicroVAX I ever become large enough
0000    ;       to encompass the new features contained herein, this intermediate
0000    ;       VMB should work equally as well as the primary VMB.
0000    ;
0000    ;       Once assembled and linked, this bootstrap procedure must reside
0000    ;       on a device with a fixed CSR under the name "sysboot.exe".  When
0000    ;       the user boots from this disk, this bootstrap program prompts the
0000    ;       user for the name of the device he really wants to boot from.
0000    ;       The P which is tacked onto the end of the name of this and some
0000    ;       other modules stands for "prompting version."
0000    ;
0000    ;       For information about patching the intermediate VMB, see
0000    ;       sys$update:vmbuvax1.com.
0000    ;
0000    ; Author: R. Heinen
0000    ;
0000    ; Date: July 1983
0000    ;
0000    ; Modifications:
0000    ;
0000    ;       V4.0-06 DGB0107         Donald G. Blair         19-DEC-1984
0000    ;                       Change address offsets so they are relative
0000    ;                       to the end of VMB code rather than the start
```

```
          0000  ;                       of the RPB so the intermediate VMB will work
          0000  ;                       on old MicroVAX I ROM's.
          0000  ;
          0000  ;       V4.0-05 DGB0105         Donald G. Blair         6-DEC-1984
          0000  ;                       Since the intermediate VMB pushes SYSBOOT out
          0000  ;                       past the end of the 64K bytes of tested memory,
          0000  ;                       test the extra pages for memory errors.
          0000  ;
          0000  ;       V4.0-04 DGB0104         Donald G. Blair         25-NOV-1984
          0000  ;                       Fix intermediate VMB so it doesn't try to re-load
          0000  ;                       the SCB in the middle of its own code.
          0000  ;
          0000  ;       V4.0-03 DGB0100         Donald G. Blair         15-NOV-1984
          0000  ;                       Move patch_device_name to the beginning of
          0000  ;                       the program to minimize possibility of its address
          0000  ;                       changing.
          0000  ;
          0000  ;       V4.0-02 DGB0094         Donald G. Blair         2-NOV-1984
          0000  ;                       Add DA and DJ as device name synonyms for DU.
          0000  ;                       Set up controller letter for boot device so it does
          0000  ;                       not change between boot time and system startup.
          0000  ;
          0000  ;       V4.0-01 JES0001         Jack Speight            September 1984
          0000  ;                       Added support to boot any device using Rod Gamache's
          0000  ;                       VMBUVAX2 floating CSR calculations.
          0000  ;
          0000  ;       V1.0-01 WHM0001         Bill Matthews           July 1984
          0000  ;                       Added support for using VMB as a secondary bootstrap.
          0000  ;                       Added support for the RL02 boot driver.
          0000  ;--
          0000
00000001  0000          vmb$secondary == 1
          0000
          0000          $bdtdef                         ;define boot driver descriptor
          0000          $bqodef                         ;define boot driver offsets
          0000          $btddef                         ;define boot device types
          0000          $ihddef                         ;define VMS image header
          0000          $iodef                          ;define I/O function codes
          0000          $ipldef                         ;ipl's
          0000          $ndtdef                         ;define adapter types
          0000          $prdef                          ;processor registers
          0000          $pruv1def                       ;processor registers for MicroVAX I
          0000          $rpbdef                         ;RPB
          0000          $ssdef                          ;define VMS status codes
          0000          $vmbargdef                      ,define VMB arguments
          0000
          0000
          0000  ;
          0000  ; define some new btd symbols
          0000  ;
          0000
00000008  0000          btd$k_prom = 8
00000060  0000          btd$k_qna = 96
          0000
          0000  ;**************************************************************************
          0000
          0000  ;
          0000  ; define a macro to define boot driver names etc.
          0000  ;
          0000
          0000          .macro boot_device name,h_unit,pcsr,type,rtn,rank=0,modulo=0,max_ctrl=0,?l1
          0000          .iif gt <modulo-^xff>, .error ; maximum value for modulo is ^xff
          0000          .iif gt <max_ctrl-^xff>, .error ; maximum value for max_ctrl is ^xff
          0000  l1:     .asciz  /name/
          0000          .byte   h_unit
          0000          .byte   type
          0000          .byte   rank
          0000          .byte   modulo
          0000          .byte   max_ctrl
          0000          .byte   0
          0000          .long   pcsr+phy_a_io_space
          0000          .long   rtn-l1
          0000          .endm
          0000
          0000  ;
          0000  ; define macros to aid with error message printing
          0000  ;
          0000
          0000          .macro  fatal_message   code
          0000          .if     nb,code
          0000          movzwl  #ss$_'code,r0
          0000          .endc
          0000          brw     fatal_error
          0000          .endm
          0000
          0000          .macro  msg_def mname,txt
          0000          .word   ss$_'mname
          0000          .if     nb,<txt>
          0000          .word   a_'mname-.
          0000          .save_psect
          0000          .psect  $$$$10boot,byte
          0000  last_msg = .
          0000  a_'mname:
          0000          .asciz  \txt \
          0000          .restore_psect
          0000          .iff
          0000          .word   last_msg-.
          0000          .endc
          0000          .endm
          0000
          0000  ;
          0000  ; define local data structure offsets
          0000  ;
          0000
          0000  ;
          0000  ; define boot device desc structure
          0000  ;
          0000
          0000          $defini bd
00000004  0000  bd_l_name:      .blkl   1
00000005  0004  bd_b_high_unit: .blkb   1
00000006  0005  bd_b_type:      .blkb   1
00000007  0006  bd_b_rank:      .blkb   1
00000008  0007  bd_b_modulo:    .blkb   1
```

```
00000009  0008  bd_b_max_ctrl:  .blkb   1
0000000A  0009  bd_b_spare:     .blkb   1
0000000E  000A  bd_a_csr:       .blkl   1
00000012  000E  bd_a_routine:   .blkl   1
          0012  bd_s_bd:
          0012          $defend bd
          0000  ;
          0000  ; define local data constants
          0000  ;
00000007  0000  MAX_CTRLRS              = 7
          0000  ;
          0000  ; define local data constants
          0000  ;
          0000
20000000  0000  phy_a_io_space          = ^x20000000    ;physical address of I/O space
          0000
          0000  ;
          0000  ; define extents
          0000  ;
          0000
00002000  0000  k_max_memory_pages      = 8192          ;max number of pages
0000007F  0000  k_max_io_pages          = 127           ;max pages in one I/O transfer
          0000
          0000  ;
          0000  ; define addresses to be used to locate sections of memory.
          0000  ;
          0000
00000000  0000  k_scb_addr              = ^x0           ; offset from vmb_end to start of scb
00000400  0000  k_pfn_map_addr          = ^x400         ; offset from vmb_end to start of
          0000                                          ;    pfn map.
00000E00  0000  k_next_boot_addr        = ^x0e00        ; offset from vmb_end to start of
          0000                                          ;    next bootstrap
00018000  0000  k_max_boot_len          = ^x18000       ; maximum length in bytes of the entire
          0000                                          ; bootstrap, including the rpb, primary
          0000                                          ; vmb, pfn bitmaps, intermediate vmb,
          0000                                          ; sysboot and other miscellaneous
          0000                                          ; intervening pieces.
          0000
          0000  ;
          0000  ; define MicroVAX I machine check codes used here
          0000  ;
          0000
00000001  0000  k_parity.error          = 1
00000002  0000  k_bus.timeout           = 2
          0000
          0000  ;
          0000  ; define scb vectors used here
          0000  ;
          0000
00000004  0000  scb_a_mcheck            = 4
00000060  0000  scb_a_write_timeout     = ^x60
0000002C  0000  scb_a_breakpoint        = ^x2c
00000028  0000  scb_a_trace_trap        = ^x28
          0000
          0000  ;
          0000  ; define bits in MicroVAX I switch pack
          0000  ;
```

```
                0000
        00000006 0000    switch_v_QVSS              = 6              ; 1 if normal, 0 if QVSS
        00000007 0000    switch_v_disk_boot         = 7              ; 1 if normal, 0 if disable disk search
                0000
                0000    ;
                0000    ; define MSV-11 Memory controller values
                0000    ;
                0000
        20001440 0000    msv11_csr_base = ^x1440 + phy_a_io_space
        00000001 0000    msv11_csr_parity_enable = 1
                0000
                0000    ;
                0000    ; define led values
                0000    ;
                0000
        00000F0D 0000    led_memory_ok = ^xf0d
        00000F0E 0000    led_boot_inprogress = ^xf0e
        00000F0F 0000    led_transfer_control = ^xf0f
                0000
                0000    ;
                0000    ; define console halt code
                0000    ;
                0000
        00000F05 0000    console_halt = ^xf05
```

```
                                     0000         .sbttl  read write data
                                     0000
                                 00000000         .psect  $$$$04boot,long
                                     0000
                                     0000   ;
                                     0000   ; patch_device_name is used by the intermediate vmb to determine the
                                     0000   ; boot device name.  If the customer has patched a device name into
                                     0000   ; this location using the SYS$UPDATE:VMBUVAX1.COM command procedure,
                                     0000   ; we boot from this device.  Otherwise, we prompt the console device
                                     0000   ; to find out the name of the system disk.
                                     0000
                                     0000   ;*****************************************************************
                                     0000   ;*                                                               *
                                     0000   ;*                          WARNING                              *
                                     0000   ;*                                                               *
                                     0000   ;* The address of patch_device_name is hard-coded into the       *
                                     0000   ;* sys$update:vmbuvax1.com command procedure.  If you cause      *
                                     0000   ;* the address of this location to change, you will break       *
                                     0000   ;* the command procedure.                                        *
                                     0000   ;*                                                               *
                                     0000   ;*****************************************************************
                                     0000   patch_device_name::
                        00000004     0000           .BLKL   1
                                     0004
                                     0004   ;
                                     0004   ; strings used for file opens
                                     0004   ;
                                     0004
                                     0004   vmsfile:                                    ;Name of standard secondary
58 45 53 59 53 2E 30 53 59 53 5B 00' 0004           .ASCIC  /[SYSO.SYSEXE]SYSBOOT.EXE/ ;bootstrap image file.
58 45 2E 54 4F 4F 42 53 59 53 5D 45  0010
                                  45 001C
                                  18 0004
                                     001D
                                     001D   diagfile:                                   ;Name of standard diagnostic
41 4D 53 59 53 2E 30 53 59 53 5B 00' 001D           .ASCIC  /[SYSO.SYSMAINT]DIAGBOOT.EXE/ ;secondary bootstrap image.
54 4F 4F 42 47 41 49 44 5D 54 4E 49  0029
                        45 58 45 2E  0035
                                  1B 001D
                                     0039
                                     0039   nameprompt:                                 ;Prompt string for secondary
20 3A 65 6C 69 66 74 6F 6F 42 0A 0D  0039           .ASCIZ  <13><10>/Bootfile: /    ;boot file name.
                                  00 0045
                                     0046
                                     0046   devnameprompt:
63 69 76 65 44 20 74 6F 6F 42 0A 0D  0046           .ASCIZ  <13><10>/Boot Device: / ;Prompt string for boot device name
                         00 20 3A 65 0052
                                     0056
                                     0056   ;
                                     0056   ; define boot device priority lists
                                     0056   ;
                                     0056
                                     0056   synonym_device_list: ; synonyms for DU
                                     0056
                                     0056           boot_device     DAA,3,<^X1468>,btd$k_uda,disk_boot,26
                                     0068           boot_device     DJA,3,<^X1468>,btd$k_uda,disk_boot,26
                                     007A
```

```
                             007A  303 boot_device_list:
                             007A  304
                             007A  305         boot_device     DUA,3,<^X1468>,btd$k_uda,disk_boot,26
                             008C  306         boot_device     DLA,3,<^X1900>,btd$k_dl,disk_boot,14
                             009E  307
                             009E  308 no_disk_boot_device_list:
                             009E  309
                             009E  310         boot_device     PRA,0,<^x0000>,btd$k_prom,prom_boot
                             00B0  311         boot_device     XQA,0,<^X1920>,btd$k_qna,network_boot,,<^x10>,1
                       0000  00C2  312         .word   0                       ;implant a zero name
                             00C4  313
                             00C4  314 ;
                             00C4  315 ; define text to correspond to ss$_ values.
                             00C4  316 ;
                             00C4  317
                             00C4  318 message_header:
52 45 2D 46 2D 54 4F 4F 42 25 0A 0D  00C4  319         .asciz  <13><10>/%BOOT-F-ERROR, /
            00 20 2C 52 4F 52  00D0
                             00D6  320
                             00D6  321 message_base:
                             00D6  322
                             00D6  323 ;
                             00D6  324 ; define some ss$_ codes that are only used here
                             00D6  325 ;
                             00D6  326
                   00008000  00D6  327 ss$_memerr = ^x8000
                   00008008  00D6  328 ss$_scbint = ^x8008
                   00008010  00D6  329 ss$_2ndint = ^x8010
                   00008018  00D6  330 ss$_norom  = ^x8018
                             00D6  331
                             00D6  332 msg_def nosuchdev,<None of the bootable devices contain a program image>
                             00DA  333 msg_def devassign,<Device is not present>
                             00DE  334 msg_def nosuchfile,<Program image not found>
                             00E2  335 msg_def filestruct,<Invalid boot device file structure>
                             00E6  336         msg_def badchksum
                             00EA  337         msg_def badfilehdr
                             00EE  338         msg_def badirectory
                             00F2  339 msg_def filnotcntg,<Invalid program image format>
                             00F6  340 msg_def endoffile
                             00FA  341 msg_def badfilename,<Invalid filename>
                             00FE  342 msg_def bufferovf,<Program image does not fit in available memory>
                             0102  343 msg_def ctrlerr,<Boot device I/O error>
                             0106  344 msg_def devinact,<Failed to initialize boot device>
                             010A  345 msg_def devoffline,<Device is offline>
                             010E  346 msg_def memerr,<Memory initialization error>
                             0112  347 msg_def scbint,<Unexpected SCB exception or machine check>
                             0116  348 msg_def 2ndint,<Unexpected exception after starting program image>
                             011A  349 msg_def norom,<No valid ROM image found>
                             011E  350 msg_def nosuchnode,<No response from load server>
                       0000  0122  351         .word   0                       ;terminate list
                             0124  352
                             0124  353 ;
                             0124  354 ; writable data
                             0124  355 ;
                             0124  356         .ALIGN  LONG
                             0124  357
                             0124  358 ;
```

```
              0124  ; Parameter list handed from primary boot to secondary boot
              0124  ; The first location contains the argument count.  It is intended
              0124  ; that the secondary boot will know what is in the list based on
              0124  ; the argument count and the VMB version number.  This means that
              0124  ; new information should be placed at new offsets even if older
              0124  ; stuff becomes obsolete.  The VMB version number can be used to
              0124  ; totally change the argument meanings if necessary.
              0124  ;
              0124
              0124  second_param:
    00000128  0124          fil$gq_cache     == .+vmb$q_filecache ;FILEREAD cache descriptor
    00000148  0124          boo$gb_systemid == .+vmb$b_systemid  ;SCS system id
    0000000E  0124          .long   <vmb$c_argbytcnt-45/4    ;Size of argument list
              0128          .rept   vmb$c_argbytcnt-4        ;Reserve space for the arguments
              0128          .byte   0
          00  0128          .endr
              0160
              0160  file_cache_desc:                         ;saved cache desc
    00000000  0160          .long   0                        ;to re-init the cache after error
    00000000  0164          .long   0
              0168
              0168  ;
              0168  ; address of the RPB as a global
              0168  ;
              0168
              0168  boo$gl_rpbbase::
    00000000  0168          .long   0
              016C
              016C  ;
              016C  ; machine check support
              016C  ;
              016C
              016C  machine_check_continue:                  ;contains 0 or that address to
    00000170  016C          .blkl   1                        ;transfer to after a machine check
              0170
              0170  ;
              0170  ; error device name
              0170  ;
              0170
              0170  boot_device_name:
    00000000  0170          .long   0
          00  0174          .byte   0
              0175
              0175  ;
              0175  ; floating device modulo table
              0175  ;       (modulo value -1)
              0175
              0175  modulo_tbl:
          07  0175          .BYTE   ^x07            ; DJ11 (rank = 1)
          0F  0176          .BYTE   ^x0f            ; DH11 (rank = 2)
          07  0177          .BYTE   ^x07            ; DQ11 (rank = 3)
          07  0178          .BYTE   ^x07            ; DU11 (rank = 4)
          07  0179          .BYTE   ^x07            ; DUP11 (rank = 5)
          07  017A          .BYTE   ^x07            ; LK11A (rank = 6)
          07  017B          .BYTE   ^x07            ; DMC11/DMR11 (rank = 7)
          07  017C          .BYTE   ^x07            ; DZ11 (rank = 8)
          07  017D          .BYTE   ^x07            ; KMC11 (rank = 9)
```

```
read write data

07  017E   416         .BYTE   ^x07              ; LPP11 (rank = 10)
07  017F   417         .BYTE   ^x07              ; VMV21 (rank = 11)
0F  0180   418         .BYTE   ^x0f              ; VMV31 (rank = 12)
07  0181   419         .BYTE   ^x07              ; DWR70 (rank = 13)
07  0182   420         .BYTE   ^x07              ; RL11 (rank = 14)
0F  0183   421         .BYTE   ^x0f              ; LPA11-K (rank = 15)
07  0184   422         .BYTE   ^x07              ; KW11-C (rank = 16)
07  0185   423         .BYTE   ^x07              ; RESERVED (rank = 17)
07  0186   424         .BYTE   ^x07              ; RX11 (rank = 18)
07  0187   425         .BYTE   ^x07              ; DR11-W (rank = 19)
07  0188   426         .BYTE   ^x07              ; DR11-B (rank = 20)
07  0189   427         .BYTE   ^x07              ; DMP11 (rank = 21)
07  018A   428         .BYTE   ^x07              ; DPV11 (rank = 22)
07  018B   429         .BYTE   ^x07              ; ISB11 (rank = 23)
0F  018C   430         .BYTE   ^x0f              ; DMV11 (rank = 24)
07  018D   431         .BYTE   ^x07              ; DEUNA (rank = 25)
03  018E   432         .BYTE   ^x03              ; UDA50 (rank = 26)
00  018F   433         .BYTE   0
    0190   434         .align  long
    0190   435
```

```
0190            .sbttl  boot code
0190    ;++
0190    ; ROM_START
0190    ;
0190    ; functional description:
0190    ;
0190    ; This code is entered after the MicroVAX I microcode has completed its
0190    ; restart/boot/halt sequence. It runs at IPL 31, in Kernel mode on the
0190    ; interrupt stack. The action is to initialize an RPB, setup a bitmap of
0190    ; useable memory pages and load the next part of the system boot based on
0190    ; the input flag settings.
0190    ;
0190    ; If the inputs include a specific boot device name that device and only
0190    ; that device is booted. On the otherhand, if no specific boot device
0190    ; is specified then a priority ordered sequence of boot devices is tried.
0190    ; (In this case R0 will be 0 or contain all blanks.
0190    ;
0190    ; As follows:
0190    ;
0190    ;       DU units 0,1,2,3
0190    ;       Other disks
0190    ;       ROM (See below for an explanation of how the ROM is found.)
0190    ;       QNA
0190    ;
0190    ;       If none of these devices provide a bootstrap then a message is
0190    ;       displayed followed by a HALT.
0190    ;
0190    ; ROM systems are recognized by the boot memory search. A ROM system must
0190    ; be aligned on a 4KB boundary and contain a foot print which is the same
0190    ; as the second part of the boot block described below.
0190    ;
0190    ; If the boot is from a mass storage device then for each valid volume
0190    ; that is found, the volume is searched as a Files-11 volume and then
0190    ; the secondary boot image is found. If the volume is not a Files-11 volume
0190    ; then block 0 of the volume is read and checked to see if it meets the
0190    ; standard for the boot block format. If not, the volume is not used and the
0190    ; next volume is tried unless a specific device was specified by the user.
0190    ;
0190    ; The boot block format is:
0190    ;
0190    ;                           +-------+-------+-------+-------+
0190    ;       BB+0:               !      1!      n!      any value!
0190    ;                           +-------+-------+-------+-------+
0190    ;                           ! low LBN       ! High LBN      !
0190    ;                           +-------+-------+-------+-------+
0190    ;
0190    ; This second part is used for both the boot block and the ROM system.
0190    ;
0190    ;                           +-------+-------+-------+-------+
0190    ;       BB+(2*n)+0:         !  Chk  !      k!       18(Hex)!
0190    ;                           +-------+-------+-------+-------+
0190    ;                           ! any value, most likely 0      !
0190    ;                           +-------+-------+-------+-------+
0190    ;       BB+(2*n)+8:         ! size in blocks of the image   !
0190    ;                           +-------+-------+-------+-------+
0190    ;       BB+(2*n)+12:        ! load offset                   !
0190    ;                           +-------+-------+-------+-------+
0190 ;        BB+(2*n)+16:    | offset into image to start    |
0190 ;                        +-------+-------+-------+-------+
0190 ;        BB+(2*n)+20:    | sum of the previous three LW's|
0190 ;                        +-------+-------+-------+-------+
0190 ;
0190 ; The input bits in R5 can contain a bit that disables the Files-11 search.
0190 ;
0190 ; If a Files-11 boot is done then the file booted is either:
0190 ;
0190 ;        SYSBOOT.EXE - default
0190 ;        DIAGBOOT.EXE - R5 bit setting
0190 ;        solicited from the console
0190 ;
0190 ; Details of how the memory look and the register settings when the
0190 ; secondary bootstraps are entered are documented where the exits occur.
0190 ;
0190 ; inputs:
0190 ;
0190 ;        r0 = boot device name in ASCII or 0 if none specified
0190 ;        r1 = switch pack settings 1 is "ON", 0 is "OFF"
0190 ;
0190 ;                Bit     Meaning
0190 ;                ---     -------
0190 ;
0190 ;                7       Enable disk search during bootstrap
0190 ;
0190 ;                6       1 if VT100/VT200 console, 0 if QVSS video option
0190 ;
0190 ;                4-5     Halt action
0190 ;                3       Console Break enabled
0190 ;
0190 ;                2       Reserved
0190 ;
0190 ;                0-1     Console baud rate
0190 ;
0190 ;        R5 = software boot control flags from the /N boot command qualifier.
0190 ;
0190 ;        The following bits are used by this boot ROM code:
0190 ;
0190 ;                Bit     Meaning
0190 ;                ---     -------
0190 ;
0190 ;                 3      RPB$V_BBLOCK.
0190 ;                        If set, the attempt to Files-11 boot is skipped
0190 ;                        and only the boot block type boot is done.
0190 ;
0190 ;                 4      RPB$V_DIAG.
0190 ;                        Diagnostic boot. Secondary bootstrap is image
0190 ;                        called [SYSMAINT]DIAGBOOT.EXE.
0190 ;
0190 ;                 6      RPB$V_HEADER.
0190 ;                        Image header. Takes the transfer address of the
0190 ;                        secondary bootstrap image from that file's
0190 ;                        image header. If RPB$V_HEADER is not set,
0190 ;                        transfers control to the first byte of the
0190 ;                        secondary boot file.
0190 ;
```

```
0190   ;        8       RPB$V_SOLICT.
0190   ;                File name. Prompt for the name of a
0190   ;                secondary bootstrap file.
0190   ;
0190   ;        9       RPB$V_HALT.
0190   ;                Halt before transfer. Executes a HALT
0190   ;                instruction before transferring control to the
0190   ;                secondary bootstrap.
0190   ;
0190   ;        <31:28> RPB$V_TOPSYS
0190   ;                Specifies the top level directory number for system
0190   ;                disks with multiple systems
0190   ;
0190   ;    The following bits are NOT used by this boot ROM code:
0190   ;
0190   ;        Bit     Meaning
0190   ;        ---     -------
0190   ;
0190   ;         0      RPB$V_CONV.
0190   ;                Conversational boot. At various points in the
0190   ;                system boot procedure, the bootstrap code
0190   ;                solicits parameters and other input from the
0190   ;                console terminal. If the DIAG is also on, then
0190   ;                the diagnostic supervisor should enter "MENU"
0190   ;                mode and prompt user for devices to test.
0190   ;
0190   ;         1      RPB$V_DEBUG.
0190   ;                Debug. If this flag is set, VMS maps the code
0190   ;                for the XDELTA debugger into the system page
0190   ;                tables of the running system.
0190   ;
0190   ;         2      RPB$V_INIBPT.
0190   ;                Initial breakpoint. If RPB$V_DEBUG is set, VMS
0190   ;                executes a BPT instruction immediately after
0190   ;                enabling mapping.
0190   ;
0190   ;         5      RPB$V_BOOBPT.
0190   ;                Bootstrap breakpoint. Stops the primary
0190   ;                and secondary bootstraps with a breakpoint
0190   ;                instruction before testing memory.
0190   ;
0190   ;         7      RPB$V_NOTEST.
0190   ;                Memory test inhibit. Sets a bit in the PFN bit
0190   ;                map for each page of memory present. Does not
0190   ;                test the memory.
0190   ;
0190   ;        10      RPB$V_NOPFND.
0190   ;                No PFN deletion (not implemented; intended to
0190   ;                tell VMB not to read a file from the boot device
0190   ;                that identifies bad or reserved memory pages,
0190   ;                so that VMB does not mark these pages as valid
0190   ;                in the PFN bitmap).
0190   ;
0190   ;        11      RPB$V_MPM.
0190   ;                Specifies that multi-port memory is to be used
0190   ;                for the total exec memory requirement.  No local
0190   ;                memory is to be used.  This is for tightly-coupled
```

```
0190  ;                       multi-processing.  If the DIAG is also on, then
0190  ;                       the diagnostic supervisor enters "AUTOTEST" mode.
0190  ;
0190  ;                 12      RPB$V_USEMPM.
0190  ;                         Specifies that multi-port memory should be used in
0190  ;                         addition to local memory, as though both were one
0190  ;                         single pool of pages.
0190  ;
0190  ;                 13      RPB$V_MEMTEST
0190  ;                         Specifies that a more extensive algorithm be used
0190  ;                         when testing main memory for hardware uncorrectable
0190  ;                         (RDS) errors.
0190  ;
0190  ;                 14      RPB$V_FINDMEM
0190  ;                         Requests use of MA780 memory if MS780 is insufficient
0190  ;                         for booting.  Used for 11/782 installations.
0190  ;
0190  ;       r10 = original PC
0190  ;       r11 = original PSL
0190  ;       AP = halt code
0190  ;       SP = address of 64K memory block + 200 hex
0190  ;
0190  ; implicit inputs:
0190  ;
0190  ;       IPL is 31, interrupt stack.
0190  ;       The first instruction of this code is at SP.
0190  ;
0190  ;       All of the system's memory controllers have been
0190  ;       initialized to have parity error detect ON. This
0190  ;       means that the 64K memory block that contains this
0190  ;       code has correct parity. The cache is enabled and
0190  ;       will continue to be enabled throughout.
0190  ;
0190  ; When the secondary bootstrap code gains control memory will look like:
0190  ;
0190  ; 0       +-------------------------------------+
0190  ;         + RPB                                 +
0190  ; 200     +-------------------------------------+
0190  ;         + 8K of Boot Code                     +
0190  ;         + boot driver preamble starts at 200  +
0190  ; 4200    +-------------------------------------+ (PR$_SCBB value)
0190  ;         + 2 Pages of SCB                      +
0190  ; 4600    +-------------------------------------+
0190  ;         + 2 Pages of PFN Bit Map described by +
0190  ;         + RPB fields                          +
0190  ; 4A00    +-------------------------------------+
0190  ;         + available for stack (3 Pages)       +
0190  ; 5000    +-------------------------------------+
0190  ;         + Secondary boot code image           +
0190  ;         .
0190  ;         .
0190  ;         .
0190  ;
0190  ; The register contents when control is passed to the secondary
0190  ; bootstrap are:
0190  ;
0190  ;       R11 = base address of RPB
```

```
    0190  ;        AP = address of the secondary boot parameter block alla VMB
    0190  ;        SP = current stack pointer
    0190  ;        PR$_SCBB = SCB address
    0190  ;
    0190  ; If the intermediate VMB is being used, when the real secondary
    0190  ; bootstrap (e.g. SYSBOOT, DIAGBOOT) begin execution, memory is organized
    0190  ; as below.  Note that we use more than the 64Kbytes of tested memory
    0190  ; that has been allocated for us.  If a memory error is found in the
    0190  ; spillover area, we report an error and halt.  Note also that the "physical"
    0190  ; addresses given below are relative to the beginning of the 64Kbytes
    0190  ; of tested memory.
    0190  ;
    0190  ;        0       +--------------------------------------+
    0190  ;                + RPB                                  +
    0190  ;        200     +--------------------------------------+
    0190  ;                + 8K of Boot Code - VMBUVAX1.EXE       +
    0190  ;                + boot driver preamble starts at 200   +
    0190  ;        4200    +--------------------------------------+ (PR$_SCBB value)
    0190  ;                + 2 Pages of SCB                       +
    0190  ;        4600    +--------------------------------------+
    0190  ;                + 2 Pages of PFN Bit Map described by  +
    0190  ;                + RPB fields                           +
    0190  ;        4A00    +--------------------------------------+
    0190  ;                + available for stack (3 Pages)        +
    0190  ;        5000    +--------------------------------------+
    0190  ;                + VMBUVAX1P.EXE                        +
    0190  ;                +                                      +
    0190  ; vmb_end + k_scb_addr: ---------------------------------+ (PR$_SCBB value)
    0190  ;                + 2 Pages of SCB                       +
    0190  ; vmb_end + k_pfn_map_addr: -----------------------------+
    0190  ;                + 2 Pages of PFN Bit Map described by  +
    0190  ;                + RPB fields                           +
    0190  ;                +--------------------------------------+
    0190  ;                + available for stack (3 Pages)        +
    0190  ; vmb_end + k_next_boot_addr: ---------------------------+
    0190  ;                + Secondary bootstrap                  +
    0190  ;                + (at this writing, SYSBOOT or other   +
    0190  ;                + 2ndary bootstrap begin at AC00)      +
    0190  ;                +--------------------------------------+
    0190  ;                + Room for expansion                   +
    0190  ; k_max_boot_len: +-------------------------------------+
    0190  ;
    0190  ;--
    0190
    0190  ;
    0190  ; create a label to point to the end of VMB.
    0190  ;
    0190
00000000          .psect  ___ZZZVMB_END,page
    0000  VMB_END::
    0000
    0000  ;
    0000  ; the label rom_base is (and must remain) the first location in the boot ROM
    0000  ;
    0000
00000000          .psect  $$$$00boot,long
    0000  ROM_BASE:
```

```
                   018D'  31  0000   722           brw     rom_start                    ;transfer control to actual code
         83 82 81 80 06       0003   723           .byte   6,^x80,^x81,^x82,^x83        ;footprint
                              0008   724
                          00000190   725           .psect  $$$$04boot,long              ;
                              0190   726
                              0190   727 ROM_START:
                              0190   728           .default        displacement,word
```

```
                             0190        .sbttl  SCB initialization and XDELTA breakpoint
                             0190
             26    0F    DA  0190        mtpr    #^xf,#pr$_mcesr          ;reset any machine checks
                             0193
                             0193  ;
                             0193  ; setup SCB for the duration of this execution
                             0193  ;
                             0193
          0400'CF        9E  0193  10$:  movab   vmb_end + k_scb_addr +- ;address scb plus two pages
                57           0197                  ^x400,r7
          59   FF 8F     9A  0198        movzbl  #255,r9                  ;setloop count DIV 2
      77  0881'CF        DE  019C  17$:  moval   unfielded_scb_int+1,-(r7) ;address general error routine
             F8 59       F4  01A1        sobgeq  r9,17$                   ;continue in loop
             C5 AF       D4  01A4        clrl    machine_check_continue ;init machine check continue address
          0735'CF        9E  01A7        movab   machine_check_detect+1,- ;init machine check vector
             04 A7           01AB                scb_a_mcheck(r7)         ;
          08DD'CF        9E  01AD        movab   write_timeout_int+1,- ;init write timeout vector
             60 A7           01B1                scb_a_write_timeout(r7);
             11    57    DA  01B3        mtpr    r7,#pr$_scbb             ;insert scb address in PR
                             01B6
                             01B6  ;
                             01B6  ; Read the system identification processor register to discover which
                             01B6  ; kind of VAX is to be booted.
                             01B6  ;
                             01B6
             58    3E    DB  01B6        mfpr    #pr$_sid,r8              ;Read the CPU identification
                             01B9                                         ;processor register.
      58   58   E8 8F    78  01B9        ashl    #-pr$v_sid_type,r8,r8    ;Get CPU identification code.
        0000'CF    58    90  01BE        movb    r8,exe$gb_cputype        ;Save processor code globally
                             01C3                                         ;in boot driver desc table
                             01C3
                             01C3  ;
                             01C3  ; If we are LINKED as a SECONDARY the following reference
                             01C3  ; will be non-zero.
                             01C3  ;
                             01C3        .WEAK   VMB$SECONDARY
                   01    D5  01C3        TSTL    #VMB$SECONDARY           ; are we pretending to be a secondary?
                   0C    13  01C5        BEQL    60$                      ; br if no, continue
          51   20 AB     D0  01C7        MOVL    RPB$L_BOOTR1(R11),R1     ; else, set up boot registers
          52   24 AB     7D  01CB        MOVQ    RPB$L_BOOTR2(R11),R2     ;      ...
          54   2C AB     7D  01CF        MOVQ    RPB$L_BOOTR4(R11),R4     ;      ...
                             01D3
                             01D3  60$:
                             01D3  ;
                             01D3  ; Copy boot r1 thru boot r5 from primary bootstrap
                             01D3  ;
          51   20 AB     D0  01D3        movl    rpb$l_bootr1(r11),r1     ;use same boot r1 as rom VMB
          52   24 AB     D0  01D7        movl    rpb$l_bootr2(r11),r2     ;use same boot r2 as rom VMB
          53   28 AB     D0  01DB        movl    rpb$l_bootr3(r11),r3     ;use same boot r3 as rom VMB
          54   2C AB     D0  01DF        movl    rpb$l_bootr4(r11),r4     ;use same boot r4 as rom VMB.
          55   30 AB     D0  01E3        movl    rpb$l_bootr5(r11),r5     ;use same boot r5 as rom VMB
                             01E7  ;
                             01E7  ; If the DEBUG flag is defined (meaning that XDELTA has been linked
                             01E7  ; with this primary bootstrap), set up 2 XDELTA handlers in the SCB --
                             01E7  ; one for breakpoints and one for tbit traps. Then initialize the
                             01E7  ; XDELTA breakpoint table, allocate 3 pages of stack, and, if requested,
                             01E7  ; execute a breakpoint before proceeding with the bootstrap.
```

```
                                  01E7  ;
                                  01E7
                                  01E7          .weak   xdt$breakpoint
                                  01E7          .weak   xdt$trace_trap
                                  01E7          .weak   xdt$initial_break
            00000000'8F      D5   01E7          tstl    #xdt$breakpoint              ; Test if XDELTA is linked
                     23      13   01ED          beql    noxdt                        ; Br if not
      2C A7    0001'CF       9E   01EF          movab   xdt$breakpoint+1,scb_a_breakpoint(r7) ;Set up BPT handler.
      28 A7    0001'CF       9E   01F5          movab   xdt$trace_trap+1,scb_a_trace_trap(r7) ;Set up TBIT handler.
    0000'CF    020E'CF       9E   01FB          movab   ini$brk,xdt$initial_break ;Store the initial breakpoint.
               56   5E       D0   0202          movl    sp,r6                        ;Save current top of stack.
               0E00'CF       9E   0205          movab   vmb_end + -                  ;address a stack
                     5E           0209                    k_next_boot_addr,sp
          01 55     05       E1   020A          bbc     #rpb$v_boobpt,r5,nobrk  ;If no BPT was requested in the
                                  020E                                           ;boot flags, just proceed.
                                  020E
                                  020E  ;
                                  020E  ; Initial breakpoint.
                                  020E  ;
                                  020E  ; Current register status is as follows:
                                  020E  ;
                                  020E  ;       R0-R5   - initial input values
                                  020E  ;       R6      - SP value at start of ROM code
                                  020E  ;       R7      - address of the SCB
                                  020E  ;       R8      - processor identification code
                                  020E  ;       R9      - destroyed
                                  020E  ;       R10-FP  - initial input values
                                  020E  ;       SP      - address of a 3-page stack
                                  020E  ;
                                  020E  ; Code following the breakpoint is going to restore SP to its original
                                  020E  ; value. If you want to modify SP in XDELTA, modify R6 instead.
                                  020E  ;
                                  020E
                                  020E ini$brk::                                     ;Debugging breakpoint.
                                  020E
                             03   020E          bpt                                  ;Stop in XDELTA.
                                  020F
                                  020F NOBRK:                                        ;Proceed with bootstrapping.
               5E    56      D0   020F          movl    r6,sp                        ;restore stack pointer
                                  0212 NOXDT:
```

```
                              0212        .sbttl  rpb initialization
                              0212  ;
                              0212  ; initialize and address the RPB
                              0212  ;
                              0212
                56   5B   D0  0212              movl    r11,r6                     ;address rpb with temp reg
             1C A6   50   7D  0215              movq    r0,rpb$l_bootr0(r6)        ;save registers
             24 A6   52   7D  0219              movq    r2,rpb$l_bootr2(r6)        ;
             2C A6   54   7D  021D              movq    r4,rpb$l_bootr4(r6)        ;
                              0221  ;
                              0221  ; To solicit a boot device name, call a device-independent subroutine that
                              0221  ; writes a prompt string to the console terminal, and then reads the
                              0221  ; user typed boot device name.
                              0221  ;
                              0221
                     01   D5  0221              tstl    #VMB$SECONDARY             ;are we pretending to be a secondary?
                     1E   13  0223              beql    10$                        ;br if no, continue
     1C A6    FDD7 CF     D0  0225              movl    patch_device_name,rpb$l_bootr0(r6)  ;save patch device name
                     16   12  022B              bneq    10$                        ;a device? then br, no device continue
                  20 A6   DD  022D              pushl   rpb$l_bootr1(r6)           ;Pass options switch settings
                  68 A6   9F  0230              pushab  rpb$t_file(r6)             ;Set address of input buffer.
                     05   DD  0233              pushl   #5                         ;Set maximum character count.
                FE0D CF   9F  0235              pushab  devnameprompt              ;Set address of prompt string.
          0000'CF    04   FB  0239              calls   #4,boo$readprompt          ;Prompt and read string.
                  69 A6   D0  023E              movl    rpb$t_file+1(r6),-         ;save device name as boot r0
                  1C A6       0241                      rpb$l_bootr0(r6)
                              0243
                              0243  10$:
             10 A6   5A   7D  0243              movq    r10,rpb$l_haltpc(r6)       ;save halt PC and PSL
                66   56   D0  0247              movl    r6,rpb$l_base(r6)          ;address of RPB
           FF19 CF   56   D0  024A              movl    r6,boo$gl_rpbbase          ;also globally
             18 A6   5C   D0  024F              movl    ap,rpb$l_haltcode(r6)      ;save halt code
                  04 A6   D4  0253              clrl    rpb$l_restart(r6)          ;init header fields
                  0C A6   D4  0256              clrl    rpb$l_rstrtflg(r6)         ;
             08 A6   01   CE  0259              mnegl   #1,rpb$l_chksum(r6)        ;
     34 A6    0000'CF     9E  025D              movab   boo$al_vector,rpb$l_iovec(r6) ;insert address of driver
     00       6E     00   2C  0263              movc5   #0,(sp),#0,-               ;init remainder of RPB
     38 A6    00D1   8F       0267                      #rpb$c_length-rpb$l_iovecsz,rpb$l_iovecsz(r6)
                5B   56   D0  026C              movl    r6,r11                     ;set future RPB address
           00B0 CB   57   D0  026F              movl    r7,rpb$l_scbb(r11)         ;save scbb address in RPB
           0090 CB   28   90  0274              movb    #ndt$_ub0,rpb$b_confreg(r11) ;one Qbus on Micro-VAX I.
           00A1 CB   28   B0  0279              movw    #ndt$_ub0,rpb$w_bootndt(r11) ;Pretend this is UNIBUS.
                              027E
                              027E  ;
                              027E  ; init the secondary bootstrap parameter block
                              027E  ;
                              027E
        5C    FEA2 CF     9E  027E              movab   second_param,ap            ;load its base address
             0C AC   01   CE  0283              mnegl   #1,vmb$l_lo_pfn(ap)        ;set pfn data
             10 AC   01   CE  0287              mnegl   #1,vmb$l_hi_pfn(ap)        ;set pfn data
                              028B
                              028B  ;
                              028B  ; address larger stack and setup free memory pointer
                              028B  ;
                              028B
           0E00'CF        9E  028B  15$:        movab   vmb_end + -                ;address target for I/O
                     5E       028F                          k_next_boot_addr,sp    ;and create a three page stack
        5A   5E   D0  0290    885         movl    sp,r10               ;copy to address of free memory
```

```
                                 0293          .sbttl  memory initialization
                                 0293    ;
                                 0293    ; initialize parity memory
                                 0293    ;
                                 0293    ; allocate and init RPB PFN bit map
                                 0293    ;
                                 0293
      44 AB    01    0A     78   0293          ashl    #10,#1,rpb$q_pfnmap(r11) ;set size to 1024 bytes
            0400'CF         9E   0298          movab   vmb_end+k_pfn_map_addr,- ;set up pfn desc addr
                  48 AB          029C             rpb$q_pfnmap+4(r11)
      00    6E    00        2C   029E          movc5   #0,(sp),#0,-             ; init to zeroes
     0400'CF    44 AB            02A2             rpb$q_pfnmap(r11),vmb_end+k_pfn_map_addr
     FEBF CF    C3'AF       9E   02A7          movab   b^5$,machine_check_continue ;setup continue address
     59    00001FFF 8F      D0   02AD          movl    #k_max_memory_pages-1,r9 ;set page number of last memory page
        07 20 AB    06      E0   02B4          bbs     #switch_v_qvss,rpb$l_bootr1(r11),3$;if set console not QVSS
     59    00000200 8F      C2   02B9          subl    #512,r9                  ;else - last 256K used for QVSS video
                                 02C0    3$:   setipl  #^x1d-1                  ;lower IPL to allow write timeout
                                 02C3
                                 02C3    ;
                                 02C3    ; sweep all of memory to set parity and establish bit map
                                 02C3    ;
                                 02C3
     FEA2 CF    037E'CF     9E   02C3    5$:   movab   nxm_memory,machine_check_continue ;enable machine check
                      56    7C   02CA          clrq    r6                       ;set first physical address
                                 02CC                                           ;zero page count
      58    00BC CB         9E   02CC          movab   rpb$l_memdsc(r11),r8     ;address first memory descriptor
                      68    D4   02D1          clrl    (r8)                     ;init page count field
         04 A8    01        CE   02D3          mnegl   #1,4(r8)                 ;set very low PFN
                                 02D7
                                 02D7    ;
                                 02D7    ; write 0's to a page
                                 02D7    ;
                                 02D7
                                 02D7    page_boundary:
                                 02D7
      56    57    09        78   02D7          ashl    #9,r7,r6                 ;compute page address
                                 02DB
                                 02DB    ;
                                 02DB    ; don't write 0's in the 64k where this code is
                                 02DB    ;
                                 02DB
            56    5B        D1   02DB          cmpl    r11,r6                   ;compare addresses to find out
                  42        12   02DE          bneq    20$                      ;br if not in the 64KB of good memory
               0C AC        D5   02E0          tstl    vmb$l_lo_pfn(ap)         ;low pfn inited?
                  04        18   02E3          bgeq    10$                      ;br if yes
         0C AC    57        D0   02E5          movl    r7,vmb$l_lo_pfn(ap)      ;insert base of area as lowest PFN
               04 A8        D5   02E9    10$:  tstl    4(r8)                    ;memory desc PFN set yet?
                  04        18   02EC          bgeq    15$                      ;br if yes
         04 A8    57        D0   02EE          movl    r7,4(r8)                 ;insert this low PFN
            50    01        CE   02F2    15$:  mnegl   #1,r0                    ;set all bits in a register
 0400'CF    20    57    50  F0   02F5          insv    r0,r7,#32,vmb_end + k_pfn_map_addr      ;enable 128 pages
 0404'CF    20    57    50  F0   02FC          insv    r0,r7,#32,vmb_end + k_pfn_map_addr+4    ;of good memory
 0408'CF    20    57    50  F0   0303          insv    r0,r7,#32,vmb_end + k_pfn_map_addr+8    ;
 040C'CF    20    57    50  F0   030A          insv    r0,r7,#32,vmb_end + k_pfn_map_addr+12   ;
         50    7F 8F        9A   0311          movzbl  #127,r0                  ;load page count in r0
          57    6740        9E   0315          movab   (r7)[r0],r7              ;adjust PFN to last page tested
         4C AB    50        C0   0319          addl    r0,rpb$l_pfncnt(r11)     ;adjust good page count < 8000
```

```
           68    50   C0  031D    944           addl    r0,(r8)                   ;adjust desc size
                 3D   11  0320    945           brb     40$                       ;continue in common
                          0322    946
                          0322    947 ;
                          0322    948 ; write and then read a single memory page
                          0322    949 ;
                          0322    950
                          0322    951 ;
                          0322    952 ; loop to check for correct parity and verify contents
                          0322    953 ;
                          0322    954
                 66   7C  0322    955 20$:      clrq    (r6)                      ;write memory to zero
           50    02   D0  0324    956           movl    #k_bus.timeout,r0         ;initialize for error exit
           51    86   7D  0327    957           movq    (r6)+,r1                  ;read memory back for parity detect
                          032A    958                                             ;first read checks for page present
                          032A    959                                             ;skip out if not present to nxm_memory
                          032A    960                                             ;skip out on parity error
                 52   12  032A    961           bneq    nxm_memory                ;if eql then correct read back
                          032C    962                                             ;if neq then odd case of PROM
                          032C    963
                          032C    964 ;
                          032C    965 ; check for page cross
                          032C    966 ;
                          032C    967
     56    01FF  8F   B3  032C    968           bitw    #^x1ff,r6                 ;page cross?
                 EF   12  0331    969           bneq    20$                       ;br if no, keep going
                          0333    970
                          0333    971 ;
                          0333    972 ; try to write a non-zero value and verify that it can be done
                          0333    973 ;
                          0333    974 ; This is to detect pages in ROM's that are all zeros
                          0333    975 ;
                          0333    976
OFFFFFFC E6      3F   B0  0333    977           movw    #63,<1@28>-4(r6)          ;write memory, page is present
                          033A    978                                             ;with don't cache bit
OFFFFFFC E6      3F   B1  033A    979           cmpw    #63,<1@28>-4(r6)          ;read back correct?
                 38   12  0341    980           bneq    nxm_memory                ;if neq then some non RAM memory
            FC A6     B4  0343    981           clrw    -4(r6)                    ;reset to zero
                          0346    982
                          0346    983 ;
                          0346    984 ; page is written - parity appears correct
                          0346    985 ;
                          0346    986
     01    57    01   F0  0346    987 30$:      insv    #1,r7,#1,-                ;insert bit in PFN map
          0400'CF         034A    988                     vmb_end + k_pfn_map_addr
            0C AC     D5  034D    989           tstl    vmb$l_lo_pfn(ap)          ;low pfn inited?
                 04   18  0350    990           bgeq    35$                       ;br if yes
       0C AC     57   D0  0352    991           movl    r7,vmb$l_lo_pfn(ap)       ;insert lowest PFN
            04 A8     D5  0356    992 35$:      tstl    4(r8)                     ;memory desc PFN set yet?
                 04   18  0359    993           bgeq    40$                       ;br if yes
       04 A8     57   D0  035B    994           movl    r7,4(r8)                  ;insert this low PFN
       10 AC     57   D0  035F    995 40$:      movl    r7,vmc$l_hi_pfn(ap)       ;insert highest
                 68   D6  0363    996           incl    (r8)                      ;count in current memory desc
            4C AB     D6  0365    997           incl    rpb$l_pfncmt(r11)         ;count as good page
                          0368    998
                          0368    999 ;
                          0368   1000 ; come here to move to next page
```

```
                             0368  ;
                             0368
                             0368  next_page:
                             0368
FF69 57    01    59    F1    0368          acbl    r9,#1,r7,page_boundary  ; continue until end of memory
                             036E
                             036E  ;
                             036E  ; restore IPL and setup SCB for booting
                             036E  ;
                             036E
                             036E          setipl  #ipl$_power             ;reset IPL
         FDF7 CF       D4    0371          clrl    machine_check_continue  ;reset machine check continue addr
 23    00000F0D 8F     DA    0375          mtpr    #Led_memory_ok,#pr$_txdb; set lights
                 52    11    037C          brb     begin_boot
                             037E
                             037E  ;
                             037E  ; come here when a page does not exist
                             037E  ;
                             037E
                             037E  nxm_memory:
                             037E
                             037E  ;
                             037E  ; the primary bootstrap and SYSBOOT were all intended to fit within
                             037E  ; 64 K bytes of memory which are tested and determined to be error-free.
                             037E  ; however, when the secondary bootstrap is used, the bootstrap procedures
                             037E  ; collectively take up more than the 64K bytes of tested memory.  As a
                             037E  ; stopgap measure, if we find an error in the untested portion of memory,
                             037E  ; we treat it as a fatal error.
                             037E  ;
       00000001'EF     D5    037E          tstl    vmb$secondary           ; primary or secondary bootstrap?
                 11    13    0384          beql    5$                      ; br if primary
 51    00018000 EB     9E    0386          movab   k_max_boot_len(r11),r1  ; r1 <- address of end of sysboot
  51    51    F7 8F    78    038D          ashl    #-9,r1,r1               ; convert r1 to pfn
           51    57    D1    0392          cmpl    r7,r1                   ; is memory error in range of
                             0395                                          ; the bootstrap code?
                 31    19    0395          blss    fatal_memory_error      ; br if so
                             0397
           50    01    D1    0397  5$:     cmpl    #k_parity.error,r0      ;expected error?
                 12    12    039A          bneq    20$                     ;parity is ok
                             039C
                             039C  ;
                             039C  ; reset the memory controllers to clear parity error
                             039C  ;
                             039C
           51    0F    D0    039C          movl    #15,r1                  ;set loop count of controllers
 52    20001440 8F     D0    039F          movl    #msv11_csr_base,r2      ;address base of controlller CSR's
           82    01    B0    03A6  10$:    movw    #msv11_csr_parity_enable,(r2)+;blast all possible CSR's
              FA 51    F4    03A9          sobgeq  r1,10$                  ;continue until done
                 BA    11    03AC          brb     next_page               ;no check for PROM needed on parity
                             03AE                                          ;problem
                             03AE  ;
                             03AE  ; bus timeout means non existant memory on a read, the page is not present
                             03AE  ;
                             03AE  ; But, page could be PROM memory
                             03AE  ;
                             03AE
           50    02    D1    03AE  20$:    cmpl    #k_bus.timeout,r0       ;expected error?
```

```
              15    12  03B1          bneq    fatal_memory_error       ;br if unexpected
                        03B3
                        03B3   ;
                        03B3   ; record holes in memory via descriptors
                        03B3   ;
                        03B3
              68    D5  03B3          tstl    (r8)                     ;this desc in use?
              B1    13  03B5          beql    next_page                ;br if no, don't move to next
  50    00FC  CB    9E  03B7          movab   <rpb$c_nmemdsc*rpb$c_memdscsiz>+rpb$l_memdsc(r11),r0
        58    50    D1  03BC          cmpl    r0,r8                    ;overrun area?
              A7    12  03BF          bneq    next_page                ;br if yes
        58    08    C0  03C1          addl    #rpb$c_memdscsiz,r8      ;address next memory desc
              68    D4  03C4          clrl    (r8)                     ;set count to zero
              A0    11  03C6          brb     next_page                ;
                        03C8
                        03C8   ;
                        03C8   ; memory initialization error
                        03C8   ;
                        03C8
                        03C8   fatal_memory_error:
                        03C8
                        03C8          fatal_message   memerr
```

```
                         03D0  ;++
                         03D0  ; begin_boot - start the booting process
                         03D0  ;
                         03D0  ; functional description:
                         03D0  ;
                         03D0  ; This sequence is entered after the RPB and PFN bitmap are set up.
                         03D0  ;
                         03D0  ; The process of selecting a boot device and type of boot operation starts
                         03D0  ; here.
                         03D0  ;
                         03D0  ; inputs:
                         03D0  ;
                         03D0  ;       r11 = address of the RPB
                         03D0  ;       ap = address of the secondary parameter block
                         03D0  ;
                         03D0  ;--
                         03D0
                         03D0  begin_boot:
                         03D0
                         03D0  ;
                         03D0  ; If the "solicit for secondary bootstrap file" flag is not set,
                         03D0  ; just use a predefined file specification.
                         03D0  ;
                         03D0
             03    E0    03D0          bbs     #rpb$v_bblock,-          ;br if not files-11 boot
          30 AB          03D2                  rpb$l_bootr5(r11),-      ;
             43          03D4                  25$                      ;
             08    E1    03D5          bbc     #rpb$v_solict,-          ;If "solicit" flag is not
          30 AB          03D7                  rpb$l_bootr5(r11),-      ;set, just use a default file
             13          03D9                  10$                      ;specification.
                         03DA
                         03DA  ;
                         03DA  ; To solicit a file name, call a device-independent subroutine that
                         03DA  ; writes a prompt string to the console terminal, and then reads the
                         03DA  ; user typed file name. All device specifications are ignored.
                         03DA  ;
                         03DA
          20 AB    DD    03DA          pushl   rpb$l_bootr1(r11)        ;Pass options switch settings
          68 AB    9F    03DD          pushab  rpb$t_file(r11)          ;Set address of input buffer.
             27    DD    03E0          pushl   #39                      ;Set maximum character count.
        FC53 CF    9F    03E2          pushab  nameprompt               ;Set address of prompt string.
   0000'CF   04    FB    03E6          calls   #4,boo$readprompt        ;Prompt and read string.
             2B    11    03EB          brb     25$                      ;Go try to read the file.
                         03ED
                         03ED  ;
                         03ED  ; If the solicit boot flag was not set, use a default file name string.
                         03ED  ; Usually, this file name is [SYSEXE]SYSBOOT.EXE. However, if the
                         03ED  ; diagnostic boot flag is set, the file name is [SYSMAINT]DIAGBOOT.EXE.
                         03ED  ;
                         03ED
   57   FC13 CF    9E    03ED  10$:    movab   vmsfile,r7               ;Assume SYSBOOT.EXE.
             04    E1    03F2          bbc     #rpb$v_diag,-            ;If the diagnostic flag is not
          30 AB          03F4                  rpb$l_bootr5(r11),-      ;set, SYSBOOT is correct.
             05          03F6                  15$
   57   FC22 CF    9E    03F7          movab   diagfile,r7              ;Otherwise, use predefined
                         03FC                                           ;name of diagnostic boot.
                         03FC
```

```
                                     03FC  ;
                                     03FC  ; Copy the file name to the RPB.
                                     03FC  ;
                                     03FC
                   50    67    9A    03FC  15$:    movzbl  (r7),r0                   ;Size of name string
                         50    D6    03FF          incl    r0                        ;Include the byte count character
          68 AB    67    50    28    0401          movc3   r0,(r7),rpb$t_file(r11)   ;Move name into RPB
                   04    1C    EF    0406          extzv   #rpb$v_topsys,#rpb$s_topsys,-
             50     30 AB            0409                  rpb$l_bootr5(r11),r0      ;Value of 0-F means top level
                                     040C                                            ;system directory "SYS0" - "SYSF"
                   09    50    D1    040C          cmpl    r0,#9                     ;0 - 9 ?
                         03    15    040F          bleq    20$                       ;Branch if yes
                   50    07    C0    0411          addl    #<<^A/A/>-<^A/9/>-1>,r0   ;Add bias to make A - F
             6D AB       50    80    0414  20$:    addb    r0,rpb$t_file+5(r11)      ;Form "SYSn"
                                     0418
                                     0418  ;
                                     0418  ; extract and stabilize device name info
                                     0418  ;
                                     0418
               57     FC3A CF  9E    0418  25$:    movab   synonym_device_list,r7    ;address descriptor list
  7E     1C AB   80A0A0A0 8F   CB    041D          bicl3   #^x80A0A0A0,rpb$l_bootr0(r11),-(sp) ;make name uppercase
                                     0426                                            ;remove possible parity bit
                   20    6E    91    0426          cmpb    (sp),#^a/ /               ;special non-name?
                         13    14    0429          bgtr    35$                       ;br if gtr then specific device
                                     042B
                                     042B  ;
                                     042B  ; non-specific device name
                                     042B  ;
                                     042B
               57     FC4B CF  9E    042B          movab   boot_device_list,r7       ; skip checking synonym device names
                         6E    D4    0430          clrl    (sp)                      ;specify non name
                         07    E0    0432          bbs     #switch_v_disk_boot,-     ;br if entire list is to be searched
                 10 20 AB            0434                  rpb$l_bootr1(r11),40$     ;
               57     FC63 CF  9E    0437          movab   no_disk_boot_device_list,r7;address alternate descriptor list
                         09    11    043C          brb     40$                       ;continue
                                     043E
                                     043E  ;
                                     043E  ; specific device name
                                     043E  ;
                                     043E
             FD2D CF     6E    D0    043E  35$:    movl    (sp),boot_device_name     ;save specified name
               03 AE     30    82    0443          subb    #^a/0/,3(sp)              ;reduce unit number
                                     0447
                                     0447  ;
                                     0447  ; start with first entry in boot device list and try each one until a
                                     0447  ; boot occurs or the list is empty
                                     0447  ;
                                     0447
                  67 AB        94    0447  40$:    clrb    rpb$b_slave(r11)          ;no slave or
                  64 AB        B4    044A          clrw    rpb$w_unit(r11)           ;unit info
             5A   0E00'CF      9E    044D          movab   vmb_end + k_next_boot_addr,r10 ;set nominal load address
          66 AB      05 A7     90    0452          movb    bd_b_type(r7),rpb$b_devtyp(r11) ;load device type
                         6E    95    0457          tstb    (sp)                      ;special non-name?
                         24    13    0459          beql    45$                       ;br if yes, no specific device
        67    6E    18   00    ED    045B          cmpzv   #0,#24,(sp),bd_l_name(r7); compare three characters for equal
                         0A    13    0460          beql    43$                       ;br if no match
                                     0462          assume  bd_b_modulo  EQ   bd_b_rank+1
```

```
              06 A7    B5 0462            tstw    bd_b_rank(r7)                 ;other controllers supported?
                 41    13 0465            beql    50$                           ;Br if no, continue
           67    6E    B1 0467            cmpw    (SP),bd_l_name(r7)            ;is this the right device?
                 3C    12 046A            bneq    50$
      04 A7   03 AE    91 046C      43$:  cmpb    3(sp),bd_b_high_unit(r7)  ;unit in range?
                 35    1A 0471            bgtru   50$                           ;br if no
      04 A7   03 AE    90 0473            movb    3(sp),bd_b_high_unit(r7);boot specific unit only
      64 AB   03 AE    9B 0478            movzbw  3(sp),rpb$w_unit(r11)         ;
                 05    11 047D            brb     47$
         FCEC CF 67    D0 047F      45$:  movl    bd_l_name(r7),boot_device_name;build error device name
FCE8 CF   30  64 AB    81 0484      47$:  addb3   rpb$w_unit(r11),#^a/0/,boot_device_name+3;
          0E B747      16 048B            jsb     abd_a_routine(r7)[r7]         ;use device specific routine
              39 50    E8 048F            blbs    r0,100$                       ;br if success, transfer control
         32 50    01   E1 0492            bbc     #1,r0,55$                     ;severe or fatal error?
                          0496                                                  ;br if fatal error
                 6E    95 0496            tstb    (sp)                          ;special non-name?
                 2E    12 0498            bneq    55$                           ;br if specific device
                          049A            ;
                          049A            ; search for next controller of this type
                          049A            ;
              02 AE    96 049A            incb    2(SP)                         ;bump the controller number (char)
   51     02 AE   41 8F  83 049D          subb3   #^a/A/,2(SP),r1               ;get controller number (value)
              07 51    91 04A3            cmpb    r1,#max_ctrlrs                ;have we done all controllers?
                 9F    15 04A6            bleq    40$
              57 12    C0 04A8      50$:  addl2   #bd_s_bd,r7                   ;try next device
   23      00000F0D 8F DA 04AB            mtpr    #led_memory_ok,#pr$_txdb; tell operator
                 67    B5 04B2            tstw    bd_l_name(r7)                 ;is there another?
                 91    12 04B4            bneq    40$                           ;continue if not end of list
                          04B6
                          04B6      ;
                          04B6      ; device data base search done without a match or valid boot device
                          04B6      ;
                          04B6
      50     0848 8F   3C 04B6            movzwl  #ss$_devassign,r0             ;list end, specific name error
                 6E    95 04BB            tstb    (sp)                          ;special non-name?
                 09    12 04BD            bneq    55$                           ;br if specific device
      50     0908 8F   3C 04BF            movzwl  #ss$_nosuchdev,r0             ;list end, generic error
           FCA8 CF     D4 04C4            clrl    boot_device_name              ;no specific name
                          04C8      55$:  fatal_message                         ;issue error in r0
                          04CB
                          04CB      ;
                          04CB      ; secondary image in place, transfer control to it
                          04CB      ;
                          04CB
              40 8F    83 04CB     100$:  subb3   #^a/A/-1,-              ; Controller letter A->1, B->2, etc.
   0108 CB   FCA1 CF      04CE                    boot_device_name+2,rpb$b_ctrlltr(r11)
   5E      0E00'CF     9E 04D4            movab   vmb_end + k_next_boot_addr,sp ;load fresh sp
                          04D9
                          04D9      ;
                          04D9      ; restore SCBB values
                          04D9      ;
                          04D9
   23    00000F0F 8F   DA 04D9            mtpr    #led_transfer_control,#pr$_txdb; tell user
           0400'CF     9E 04E0            movab   vmb_end + k_scb_addr +- ; address scb plus two pages
                57        04E4                      ^x400,r7
       59     FF 8F    9A 04E5            movzbl  #255,r9                       ;setloop count DIV 2
      77   0889'CF     DE C4E9     110$:  moval   secondary_scb_int+1,-(r7) ;address general error routine
```

```
                    F8 59    F4  04EE   1252             sobgeq  r9,110$                      ;continue in loop
                                 04F1   1253
                                 04F1   1254 ;
                                 04F1   1255 ; recompute size of bitmap
                                 04F1   1256 ;
                                 04F1   1257
  50    10 AC    FD 8F    78  04F1   1258             ashl    #-3,vmb$l_hi_pfn(ap),r0 ;get last valid PFN
     44 AB    50    01    C1  04F7   1259             addl3   #1,r0,rpb$q_pfnmap(r11) ;set size of map in bytes
        14 AC    44 AB    7D  04FC   1260             movq    rpb$q_pfnmap(r11),vmb$q_pfnmap(ap);copy pfnmap desc
                                 0501   1261
                                 0501   1262 ;
                                 0501   1263 ; halt system prior to entering secondary boot if requested
                                 0501   1264 ;
                                 0501   1265
                       09    E1  0501   1266             bbc     #rpb$v_halt,-                ;If boot flags don't call for
                    30 AB        0503   1267                     rpb$l_bootr5(r11),-          ;halt, just transfer to new
                       07        0505   1268                     120$                         ;bootstrap image.
   23      00000F05 8F    DA  0506   1269             mtpr    #console_halt,#pr$_txdb ;Otherwise, HALT.
                       65    17  050D   1270 120$:      jmp     (r5)                         ;Execute JUMP.
```

```
                         050F                 .sbttl  specific device boot subroutines
                         050F  ;++
                         050F  ; prom_boot
                         050F  ;
                         050F  ; functional description:
                         050F  ;
                         050F  ; This routine tries to boot from a PROM system image that may be in
                         050F  ; memory. Each bad page 16KB boundary is tested to see if it is readable.
                         050F  ;
                         050F  ; inputs:
                         050F  ;
                         050F  ;          r7 = address of the internal boot device description
                         050F  ;          r10 = address of the secondary boot's memory
                         050F  ;          r11 = RPB address
                         050F  ;          ap = address of the secondary parameter block
                         050F  ;
                         050F  ; outputs:
                         050F  ;
                         050F  ;          r0 = ss$_norom - no rom present, severe error
                         050F  ;
                         050F  ;          or
                         050F  ;
                         050F  ;          r0 = 1 if success
                         050F  ;          r5 = transfer address
                         050F  ;
                         050F  ;          r7,r11 are preserved
                         050F  ;--
                         050F
                         050F  prom_boot:
                         050F
                         050F  ;
                         050F  ; cycle up through memory
                         050F  ;
                         050F
FC57 CF    35'AF     9E  050F           movab   b^20$,machine_check_continue;implant for read timeout
              53     D4  0515           clrl    r3                      ;initial page address
18 0400'CF    53     E0  0517  10$:     bbs     r3,vmb_end + k_pfn_map_addr,20$ ;br if that boundary is not bad
  51    53    09     78  051D           ashl    #9,r3,r1                ;compute address
        18    61     B1  0521           cmpw    (r1),#^x18              ;try to read that memory
                         0524                                           ;may machine check
              0F     12  0524           bneq    20$                     ;br if not key
            0367     30  0526           bsbw    verify_boot_block       ;verify the boot block
           09 50     E9  0529           blbc    r0,20$                  ;br if not correct
                         052C
                         052C  ;
                         052C  ; PROM found, boot from it
                         052C  ;
                         052C
55    10 A1   51     C1  052C           addl3   r1,16(r1),r5            ;compute starting address
                         0531
                         0531  ;
                         0531  ; reset machine state
                         0531  ;
                         0531
           34 AB     7C  0531           clrq    rpb$l_iovec(r11)        ;no driver
                     05  0534           rsb                             ;done
                         0535
```

```
                                          0535  ;
                                          0535  ; move onto next 16KB boundary
                                          0535  ;
                                          0535
 FFD8 53    20    00001FFF 8F       F1    0535  20$:     acbl    #k_max_memory_pages-1,#32,r3,10$; continue until done
            50        801A 8F       3C    053F           movzwl  #ss$_norom!2,r0            ;set severe error code
                      FC24 CF       D4    0544           clrl    machine_check_continue  ;
                                    05    0548           rsb
```

```
                         0549  ;++
                         0549  ; network_boot
                         0549  ;
                         0549  ; functional description:
                         0549  ;
                         0549  ; This routine tries to boot from a network device
                         0549  ;
                         0549  ; inputs:
                         0549  ;
                         0549  ;       r7 = address of the internal boot device description
                         0549  ;       r10 = address of the secondary boot's memory
                         0549  ;       r11 = RPB address
                         0549  ;       ap = address of the secondary parameter block
                         0549  ;
                         0549  ; outputs:
                         0549  ;
                         0549  ;       r0 = 1 if success
                         0549  ;       r5 = transfer address
                         0549  ;
                         0549  ;       or
                         0549  ;
                         0549  ;       r0 = ss$_nosuchdev      - CSR does not exist - severe
                         0549  ;          = ss$_bufferovf      - secondary bootstrap does not fit - fatal
                         0549  ;          = ss$_devinact       - device could not be inited - fatal
                         0549  ;          = ss$_ctrlerr        - I/O error during operation - fatal
                         0549  ;          = ss$_devoffline     - device is offline - severe
                         0549  ;
                         0549  ;       r7,r11 are preserved
                         0549  ;--
                         0549
                         0549  network_boot:
                         0549
           0277   30     0549          bsbw    validate_csr            ;test CSR of device
                         054C                                          ;return implies success
                         054C  ;
                         054C  ; boot via the Ethernet
                         054C  ;
                         054C
        52    7E  DE     054C          moval   -(sp),r2                ;address target for transfer address
           68 AB  9F     054F          pushab  rpb$t_file(r11)         ;address to store node name
           28 AB  9F     0552          pushab  rpb$l_bootr3(r11)       ;address to store node address
              62  9F     0555          pushab  (r2)                    ;address to store transfer address
         0400'CF  9F     0557          pushab  vmb_end+k_pfn_map_addr  ;address of bit map
         0E00'CF  9F     055B          pushab  vmb_end + k_next_boot_addr ;buffer space
   53    2E00'CF  9E     055F          movab   vmb_end + k_next_boot_addr+<16*512>,r3 ;image load address
              63  9F     0564          pushab  (r3)                    ;image load address
 0000'CF      06  FB     0566          calls   #6,boo$downline_load    ;try QNA boot
   55     53  8E  C1     056B          addl3   (sp)+,r3,r5             ;compute transfer address
           03 50  E9     056F          blbc    r0,10$                  ;br if not success
           34 AB  7C     0572          clrq    rpb$l_iovec(r11)        ;no driver
                  05     0575  10$:    rsb                             ;done
```

```
                         0576  ;++
                         0576  ; disk_boot
                         0576  ;
                         0576  ; functional description:
                         0576  ;
                         0576  ; This routine tries to boot from a disk device
                         0576  ;
                         0576  ; inputs:
                         0576  ;
                         0576  ;          r7 = address of the internal boot device description
                         0576  ;          r10 = address of the secondary boot's memory
                         0576  ;          r11 = RPB address
                         0576  ;          ap = address of the secondary parameter block
                         0576  ;
                         0576  ; outputs:
                         0576  ;
                         0576  ;          r0 = 1 if success
                         0576  ;          r5 = transfer address
                         0576  ;
                         0576  ;          or
                         0576  ;
                         0576  ;          r0 = ss$_nosuchdev       - CSR does not exist - severe
                         0576  ;             = ss$_nosuchfile      - file is not on the volume - fatal
                         0576  ;             = ss$_filnotcntg      - boot file is not contiguous - fatal
                         0576  ;             = ss$_bufferovf       - secondary bootstrap does not fit - fatal
                         0576  ;             = ss$_devinact        - device could not be inited - fatal
                         0576  ;             = ss$_ctrlerr         - I/O error during operation - fatal
                         0576  ;             = ss$_devoffline      - device is offline - severe
                         0576  ;
                         0576  ;          r7,r11 are preserved
                         0576  ;--
                         0576
                         0576  disk_boot:
                         0576
           024A   30     0576          bsbw    validate_csr              ;check CSR and return if success
                         0579
                         0579  ;
                         0579  ; move and initialize the disk driver
                         0579  ;
                         0579
   52    34 AB    D0     0579          movl    rpb$l_iovec(r11),r2       ;address boot driver
      59    5B    D0     057D          movl    r11,r9                    ; load addr of rpb
      18 B242     16     0580          jsb     @bqo$l_move(r2)[r2]       ;call move code
                         0584
                         0584  ;
                         0584  ; try low to high units, removable first, non-removable second
                         0584  ;
                         0584  ; Build a mask with two sets of 8 bits. The first 8 bits are the available
                         0584  ; "soft" disk units and the second 8 are the available "hard". The mask
                         0584  ; starts with rpb$w_unit to bd_b_high_unit set in each set.
                         0584  ;
                         0584
            58    D4     0584          clrl    r8                        ;no units to search
   56    64 AB    3C     0586          movzwl  rpb$w_unit(r11),r6        ;build the basic mask
   51    04 A7    9A     058A          movzbl  bd_b_high_unit(r7),r1     ;get high
58    51    56    C3     058E          subl3   r6,r1,r8                  ;number of units
            58    D6     0592          incl    r8                        ;plus 1
```

```
          58  01  58  78 0594          ashl    r8,#1,r8                ;form mask
                  58  D7 0598          decl    r8                      ;
          58  58  56  78 059A          ashl    r6,r8,r8                ;move to correct bit pos
      58  08  08  58  F0 059E          insv    r8,#8,#8,r8             ;duplicate mask
              51  56  91 05A3          cmpb    r6,r1                   ;high = low - one unit?
                  02  12 05A6          bneq    10$                     ;br if yes, enter search
                  58  94 05A8          clrb    r8                      ;no soft disk search
                         05AA
                         05AA  ;
                         05AA  ; select a unit from the mask
                         05AA  ;
                         05AA
      56  58  10  00  EA 05AA  10$:    ffs     #0,#16,r8,r6            ;get the unit number
       64 AB  56  08  AB 05AF          bicw3   #^x8,r6,rpb$w_unit(r11) ;set unit number, less mask flag
 FBB8 CF   64 AB  30  81 05B4          addb3   #^a/0/,rpb$w_unit(r11),boot_device_name+3; new unit in name
                         05BB
                         05BB  ;
                         05BB  ; now, init that unit on the controller
                         05BB  ;
                         05BB
          52   34 AB  D0 05BB          movl    rpb$l_iovec(r11),r2     ;address boot driver
          51   1C A2  D0 05BF          movl    bqo$l_unit_init(r2),r1  ;Pick up device init routine
                  20  13 05C3          beql    30$                     ;None
                         05C5
                         05C5  ;
                         05C5  ; init the controller and a specific unit
                         05C5  ;
                         05C5  ; it is OK for the unit to be offline but not for the controller to fail
                         05C5  ;
                         05C5
            6241  6C  FA 05C5          callg   (ap),(r2)[r1]           ;do any necessary unit init
               09 50  E8 05C9          blbs    r0,20$                  ;br if unit is online
                         05CC
                         05CC  ;
                         05CC  ; If the unit is not online, it is a fatal error if the controller failed.
                         05CC  ;
                         05CC
        0084 8F   50  B1 05CC          cmpw    r0,#ss$_devoffline      ;offline?
                  36  12 05D1          bneq    50$                     ;br if no, more fatal error
                  21  11 05D3          brb     35$                     ;continue with next unit
                         05D5
                         05D5  ;
                         05D5  ; controller is up, unit is online, make removable, non-removable tests
                         05D5  ;
                         05D5  ; success from the online is:
                         05D5  ;
                         05D5  ;       #1 unit is online, can't detect hard or soft
                         05D5  ;       #9 unit is online, hard disk
                         05D5  ;       #25 unit is online, soft disk
                         05D5  ;
                         05D5
          0C 56   03  E0 05D5  20$:    bbs     #3,r6,30$               ;br if hard disk mask, try unit
                         05D9
                         05D9  ;
                         05D9  ; looking for a soft disk - can the controller can tell?
                         05D9  ;
                         05D9
      06 50    03    E1  05D9  1504          bbc     #3,r0,25$                ;br if not detectable soft or hard
                         05DD  1505
                         05DD  1506 ;
                         05DD  1507 ; looking for a soft disk and the controller can tell
                         05DD  1508 ;
                         05DD  1509
      04 50    04    E0  05DD  1510          bbs     #4,r0,30$                ;br if soft disk flag set, try unit
               1B    11  05E1  1511          brb     40$                      ;continue in common
                         05E3  1512
                         05E3  1513 ;
                         05E3  1514 ; since the controller can't tell, shut off tests in soft mask
                         05E3  1515 ; but do this unit anyway
                         05E3  1516 ;
                         05E3  1517
               58    94  05E3  1518 25$:     clrb    r8                       ;no more soft disk tests
                         05E5  1519
                         05E5  1520 ;
                         05E5  1521 ; try a boot of this unit
                         05E5  1522 ;
                         05E5  1523
       03C0 8F       BB  05E5  1524 30$:     pushr   #^m<r6,r7,r8,r9>         ;save context values
            1F       10  05E9  1525          bsbb    boot_disk_unit           ;try this unit
       03C0 8F       BA  05EB  1526          popr    #^m<r6,r7,r8,r9>         ;restore context values
         17 50       E8  05EF  1527          blbs    r0,50$                   ;br if success
      13 50    01    E1  05F2  1528          bbc     #1,r0,50$                ;br if fatal error
                         05F6  1529                                           ;continue if just severe error
  51    56     08    C1  05F6  1530 35$:     addl3   #8,r6,r1                 ;clear bit in both masks
      00 58    51    E5  05FA  1531          bbcc    r1,r8,40$                ;hard mask or greater
      00 58    56    E5  05FE  1532 40$:     bbcc    r6,r8,45$                ;and soft or hard
               58    B5  0602  1533 45$:     tstw    r8                       ;more units?
               A4    12  0604  1534          bneq    10$                      ;br if yes
                         0606  1535                                           ;exit with error status of last unit
         50    02    88  0606  1536          bisb    #1@1,r0                  ;make error semi-success
                         0609  1537
                         0609  1538 ;
                         0609  1539 ; fixup name with real booted device and transfer control
                         0609  1540 ;
                         0609  1541
                     05  0609  1542 50$:     rsb
```

```
                                 060A         .sbttl  boot a specific disk unit routine
                                 060A   ;++
                                 060A   ; boot_disk_unit
                                 060A   ;
                                 060A   ; functional description:
                                 060A   ;
                                 060A   ; This routine tries a boot of a particular disk unit. The device and
                                 060A   ; driver are present and verified. This routine is used for each unit on
                                 060A   ; which a boot is to be tried. RPB$B_UNIT contains the unit information.
                                 060A   ;
                                 060A   ; inputs:
                                 060A   ;
                                 060A   ;       r9 = rpb address
                                 060A   ;       r10 = address  of the secondary boot's memory
                                 060A   ;       r11 = RPB address
                                 060A   ;       ap = address of the secondary parameter block
                                 060A   ;
                                 060A   ; outputs:
                                 060A   ;
                                 060A   ;       r0 = ss$_success
                                 060A   ;       r5 = transfer address
                                 060A   ;
                                 060A   ;       or
                                 060A   ;
                                 060A   ;       r0 = ss$_nosuchdev        - CSR does not exits
                                 060A   ;          = ss$_nosuchfile       - file is not on the volume
                                 060A   ;          = ss$_filnotcntg       - boot file is not contiguous
                                 060A   ;          = ss$_bufferovf        - secondary bootstrap does not fit
                                 060A   ;          = ss$_devinact         - device could not be inited
                                 060A   ;          = ss$_ctrlerr          - I/O error during operation
                                 060A   ;          = ss$_devoffline       - device is offline
                                 060A   ;
                                 060A   ;       r10 and r11 are preserved.
                                 060A   ;--
                                 060A
                                 060A   boot_disk_unit:
                                 060A
                                 060A   ;
                                 060A   ; do forced boot block boot
                                 060A   ;
                                 060A   ; If RPB$V_BBLOCK is set then read LBN 0 and transfer control to the
                                 060A   ; block.
                                 060A   ;
                                 060A
                 03      E0      060A         bbs     #rpb$v_bblock,-             ;br if direct boot block boot
            63 30 AB             060C                 rpb$l_bootr5(r11),80$      ;
                                 060F
                                 060F   ;
                                 060F   ; init the file read cache if this is a FILES-11 boot
                                 060F   ;
                                 060F
FB12 CF    FB4D CF       7D      060F         movq    file_cache_desc,fil$gq_cache ;reload the descriptor
                 11      12      0616         bneq    10$                          ;br if done
              F9E5'      30      0618         bsbw    boo$cache_alloc              ;allocate the cache
      50   8000 8F       3C      061B         movzwl  #ss$_memerr,r0               ;assume no memory
FB39 CF    FB04 CF       7D      0620         movq    fil$gq_cache,file_cache_desc ;save the descriptor
                 48      13      0627         beql    75$                          ;br if cache not allocated
```

```
                      0629
                      0629 ;
                      0629 ; Call a device-independent routine, FIL$OPENFILE to locate the named
                      0629 ; file on the disk.
                      0629 ;
                      0629 ; the cache open is where the drive is mounted so it can fail if there is
                      0629 ; no physical volume
                      0629 ;
                      0629
         F9D4'   30   0629 10$:     bsbw    boo$cache_open          ;Open the FILEREAD cache
         23 50   E9   062C          blbc    r0,55$                  ;br if error
         69 AB   9F   062F 15$:     pushab  rpb$t_file+1(r11)       ;Address of file name string.
    7E   68 AB   9A   0632          movzbl  rpb$t_file(r11),-(sp)   ;Character count of file name.
            7E   D4   0636          clrl    -(sp)                   ;Allocate scratch for channel
                      0638                                          ;and get adr of scratch storage
         3C AB   DF   0638          pushal  rpb$l_fillbn(r11)       ;RPB fields that receive file
                      063B                                          ;statistics during OPEN.
            6A   DF   063B          pushal  (R10)                   ;File header buffer at end of
                      063D                                          ;memory.
       0200 CA   DF   063D          pushal  512(R10)                ;Index file header buffer at
                      0641                                          ;end of memory.
         10 AE   DF   0641          pushal  16(sp)                  ;Address in file name desc.
         10 AE   DF   0644          pushal  16(sp)                  ;Address of phony channel.
 0000'CF    05   FB   0647          calls   #5,fil$openfile         ;Call FILREAD to locate file.
      5E    0C   C0   064C          addl2   #12,sp                  ;Clean up scratch space
         5B 50   E8   064F          blbs    r0,boot_file            ;Branch on success.
                      0652
                      0652 ;
                      0652 ; the volume is not a files-11 volume, try boot block booting, if the error
                      0652 ; related to a file structure problem
                      0652 ;
                      0652
 08C0 8F    50   B1   0652 55$:     cmpw    r0,#ss$_filestruct      ;test for file structure error code
            19   13   0657          beql    80$                     ;br if that's what it is
 0810 8F    50   B1   0659          cmpw    r0,#ss$_badfilehdr      ;test for file structure error code
            12   13   065E          beql    80$                     ;br if that's what it is
 0828 8F    50   B1   0660          cmpw    r0,#ss$_badirectory     ;test for file structure error code
            0B   13   0665          beql    80$                     ;br if that's what it is
 0808 8F    50   B1   0667          cmpw    r0,#ss$_badchksum       ;test for file structure error code
            04   13   066C          beql    80$                     ;br if that's not what it is
      50    02   C8   066E          bisl    #1@1,r0                 ;make non-fatal
                 05   0671 75$:     rsb                             ;and go back to caller
                      0672
                      0672 ;
                      0672 ; read LBN 0 as boot block
                      0672 ;
                      0672
            58   D4   0672 80$:     clrl    r8                      ;block to read
      59    01   D0   0674          movl    #1,r9                   ;size to read
      56    5A   D0   0677          movl    r10,r6                  ;Start of free memory
          0080   30   067A          bsbw    readfile                ;read the block to R10
         F1 50   E9   067D          blbc    r0,75$                  ;br if error
                      0680
                      0680 ;
                      0680 ; validate the boot block
                      0680 ;
                      0680
```

```
   50    08C2 8F   3C  0680               movzwl  #ss$_filestruct!2,r0    ;set error code, semi-success
     52    02 AA   9A  0685               movzbl  2(r10),r2               ;get offset to secondary id field
     01    03 AA   91  0689               cmpb    3(r10),#1               ;next field a BR instruction
              E2   12  068D               bneq    75$                     ;br if no
      51   6A42    3E  068F               movaw   (r10)[r2],r1            ;address next field
                       0693                                               ;this must be in the same page!
            01FA   30  0693               bsbw    verify_boot_block       ;check boot block
           D8 50   E9  0696               blbc    r0,75$                  ;br if not a valid block
58     04 AA   10  9C  0699               rotl    #16,4(r10),r8           ;get secondary image LBN
       59   08 A1  D0  069E               movl    8(r1),r9                ;get image size
       5A   0C A1  C0  06A2               addl    12(r1),r10              ;compute load address
55     10 A1   5A  C1  06A6               addl3   r10,16(r1),r5           ;compute transfer address
              42   11  06AB               brb     readin_boot             ;boot block is valid, read file
                       06AD
                       06AD  ;
                       06AD  ; File was located successfully. Make sure that the file is contiguous.
                       06AD  ; The file statistics block is the following:
                       06AD  ;
                       06AD  ;          +----------------------+
                       06AD  ;          !     starting LBN     !  (0 if file not contiguous)
                       06AD  ;          +----------------------+
                       06AD  ;          !    size in blocks    !
                       06AD  ;          +----------------------+
                       06AD  ;
                       06AD
                       06AD  boot_file:                                   ;Test for contiguity.
   58      3C AB   7D  06AD               movq    rpb$l_fillbn(r11),r8    ;Get file statistics.
              58   D5  06B1               tstl    r8                      ;Contiguous file?
              06   12  C6B3               bneq    60$                     ;Yes, continue.
   50    02AC 8F   3C  06B5               movzwl  #ss$_filnotcntg,r0      ;search fatal error
                   05  06BA               rsb
                       06BB
                       06BB  ;
                       06BB  ; If the software boot control flags indicate that that transfer
                       06BB  ; address of the secondary bootstrap is stored in the image file's
                       06BB  ; header block, read that header block. Otherwise, assume that the
                       06BB  ; transfer address is simply the 1st byte in the image file.
                       06BB  ;
                       06BB
         55    5A  D0  06BB  60$:         movl    r10,r5                  ;Assume no special transfer address.
               06  E1  06BE               bbc     #rpb$v_header,-         ;If no header requested,
            30 AB      06C0                       rpb$l_bootr5(r11),-     ;then just branch past header
               2C      06C2                       readin_boot             ;reading code.
         56    5A  D0  06C3               movl    r10,r6                  ;Start of free memory
         59    01  D0  06C6               movl    #1,r9                   ,Header is always only 1 block.
               32  10  06C9               bsbb    readfile                ;Read header block.
            5E 50  E9  06CB               blbc    r0,no_fit               ;br if error
   58      3C AB   7D  06CE               movq    rpb$l_fillbn(r11),r8    ;R8 = 1st LBN, R9 = block count
         52    59  7D  06D2               movq    r9,r2                   ;R2 = block count, R3 = hdr adr
           F928'   30  06D5               bsbw    boo$image_att           ;Get image attributes
                       06D8
                       06D8  ;
                       06D8  ; R1 = image header block count
                       06D8  ; R2 = size of file in blocks excluding symbol table and patch text
                       06D8  ;
                       06D8
   00A0 CB     51  D0  06D8               movl    r1,rpb$b_hdrpgcnt(r11)  ;Store image header block count
```

```
    59   52   51   C3  06DD          subl3   r1,r2,r9                ;Blocks in image after header block(s)
         58   51   C0  06E1          addl    r1,r8                   ;LBN of first block beyond headr block
    51   02   AA   3C  06E4          movzwl  ihd$w_activoff(r10),r1  ;Get offset to image
                       06E8                                          ;activation data in header.
         51   5A   C0  06E8          addl    r10,r1                  ;Form transfer vector address.
    55      614A   9E  06EB          movab   (r1)[r10],r5            ;Get transfer address.
                       06EF
                       06EF  ;
                       06EF  ; Now read in the file. If the file is too large for the remaining
                       06EF  ; memory space, see if the required additional pages are usable.
                       06EF  ; If they are, use them.  If not issue a fatal diagnostic and HALT.
                       06EF  ;
                       06EF  ; Registers set up now are the following:
                       06EF  ;
                       06EF  ;       R5      - transfer address
                       06EF  ;       R8      - starting LBN of file (after header)
                       06EF  ;       R9      - size of file in blocks
                       06EF  ;       R10     - address of 1st byte in free memory
                       06EF  ;       R11     - address of the RPB
                       06EF  ;       AP      - secondary boot argument list
                       06EF  ;
                       06EF
                       06EF  readin_boot:
                       06EF
 14 AC    44 AB    7D  06EF          movq    rpb$q_pfnmap(r11),vmb$q_pfnmap(ap);setup bitmap desc
         56   5A   D0  06F4          movl    r10,r6                  ;buffer for read
                       06F7
                       06F7  ;
                       06F7  ; Will the desired number of blocks fit in the space remaining in the
                       06F7  ; pre-tested 64kb of memory?  If not, check that the additional pages
                       06F7  ; required are usable.  If they are, then read it all, otherwise quit.
                       06F7  ;
                       06F7
            01BE   30  06F7          bsbw    verify_image_memory     ;verify pages for image
           2F 50   E9  06FA          blbc    r0,no_fit               ;br if error
                       06FD
                       06FD  ;
                       06FD  ; Now read the secondary boot code into memory
                       06FD  ;
                       06FD  ;       Calls the device-independent bootstrap QIO routine to read
                       06FD  ;       a file. Divides the file into pieces as large as possible, so
                       06FD  ;       that the read is a small number (like 1) of DMA transfers.
                       06FD  ;
                       06FD  ; Registers:
                       06FD  ;
                       06FD  ;       R5      - secondary boot transfer address
                       06FD  ;       R6      - buffer address
                       06FD  ;       R8      - logical block number (LBN)
                       06FD  ;       R9      - number of blocks in file
                       06FD  ;
                       06FD
                       06FD  readfile:                               ;Read file into memory.
    57    7F 8F    9A  06FD          movzbl  #k_max_io_pages,r7      ;Assume maximum transfer size.
         59   57   D1  0701          cmpl    r7,r9                   ;Minimize with file size.
              03   15  0704          bleq    10$                     ;Branch if file larger than
                       0706                                          ;maximum transfer size.
         57   59   D0  0706          movl    r9,r7                   ;Set to remaining file size.
```

```
                               0709  10$:                                        ;Push arguments for QIO.
                     5B   DD   0709            pushl   r11                       ;Push phony channel number.
                     00   DD   070B            pushl   #0                        ;Physical read mode.
               7E    21   3C   070D            movzwl  #io$_readlblk,-(sp)       ;Read logical block function.
                     58   DD   0710            pushl   r8                        ;Starting LBN.
         7E    57    09   9C   0712            rotl    #9,r7,-(sp)               ;Transfer size in bytes.
                     56   DD   0716            pushl   r6                        ;Buffer address
           56    04  AE   C0   0718            addl    4(sp),r6                  ;Update buffer address.
                58   57   C0   071C            addl    r7,r8                     ;Update LBN.
           0000'CF   06   FB   071F            calls   #6,boo$qio                ;Call a bootstrap QIO routine.
                05   50   E9   0724            blbc    r0,30$                    ;Continue on success.
                59   57   C2   0727            subl    r7,r9                     ;Decrement blocks remaining.
                     D1   14   072A            bgtr    readfile                  ;Continue if not done.
                               072C
                               072C  ;
                               072C  ;         R0      - status
                               072C  ;         R5      - secondary boot transfer address
                               072C  ;         R6      - buffer address updated past last byte read
                               072C  ;         R8      - LBN updated to block after last block read
                               072C  ;         R9      - blocks in file (reduced to number not read)
                               072C  ;
                               072C
                               072C  30$:
                          05   072C  no_fit: rsb                                 ;Return to caller when done.
```

```
        072D              .sbttl  scb interrupt routines
        072D        ;++
        072D        ; ignore_scb_int
        072D        ;
        072D        ; functional description:
        072D        ;
        072D        ; This sequence runs via an SCB vectored interrupt.
        072D        ;
        072D        ; inputs:
        072D        ;
        072D        ;         none
        072D        ;
        072D        ; outputs:
        072D        ;
        072D        ;         none
        072D        ;--
        072D
        072D              .align  long
        0730
        0730        ignore_scb_int:
    02  0730              rei                                    ;
```

```
                                  0731  ;++
                                  0731  ; machine_check_detect
                                  0731  ;
                                  0731  ; functional description:
                                  0731  ;
                                  0731  ; This sequence runs when it is enabled in the machine check vector.
                                  0731  ; The action is to alter the return address to a value in r1 and continue.
                                  0731  ;
                                  0731  ; inputs:
                                  0731  ;
                                  0731  ;       machine check stack
                                  0731  ;       machine_check_continue  = address of the continuation code or 0
                                  0731  ;
                                  0731  ; outputs:
                                  0731  ;
                                  0731  ;       r0 = machine check code
                                  0731  ;--
                                  0731
                                  0731          .align  long
                                  0734
                                  0734  machine_check_detect:
                                  0734
     26    000000FF 8F    DA      0734          mtpr    #^xff,#pr$_mcesr        ;clear machine check error
                FA2D CF    D5      073B          tstl    machine_check_continue  ;change return PC?
                     0D    13      073F          beql    10$                     ;if eql then no, unexpected
          50     04 AE    D0      0741          movl    4(sp),r0                ;load reason
                5E   8E    C0      0745          addl    (sp)+,sp                ;pop stack
        6E      FA20 CF    D0      0748          movl    machine_check_continue,(sp) ;actually change return PC
                     02            074D          rei                             ;continue
                                  074E  10$:    fatal_message   scbint
```

```
                        0756  ;++
                        0756  ; fatal_error
                        0756  ;
                        0756  ; functional description:
                        0756  ;
                        0756  ; This routine is entered when a fatal error is to be displayed.
                        0756  ; The input code is a standard ss$_ value and it is matched to a text
                        0756  ; string by scanning a table of longword entries. The first work of the
                        0756  ; longword is the low word of the ss$ code and the next word is the
                        0756  ; displacement to the message text.
                        0756  ;
                        0756  ; inputs:
                        0756  ;
                        0756  ;       r0 = internal error code
                        0756  ;
                        0756  ; outputs:
                        0756  ;
                        0756  ;       The boot is abandoned, the registers are restored to
                        0756  ;       reflect the initial contents and the system is halted.
                        0756  ;--
                        0756
                        0756  fatal_error:
                        0756
 5B    FA0E CF     D0   0756          movl    boo$gl_rpbbase,r11          ;r11 <- addr of rpb
       FA0D CF     D4   075B          clrl    machine_check_continue      ;disable error continue
 51    F973 CF     9E   075F          movab   message_base,r1             ;address message desc
       50   03     CA   0764          bicl    #3,r0                       ;remove severity bits
       50   81     B1   0767  10$:    cmpw    (r1)+,r0                    ;compare code
            06     13   076A          beql    15$                         ;br if found
            81     B5   076C          tstw    (r1)+                       ;advance and test for zero offset?
            F7     12   076E          bneq    10$                         ;continue in not found
            2C     11   0770          brb     20$                         ;if list end then no message
       50   61     32   0772  15$:    cvtwl   (r1),r0                     ;fetch displacement from cell
          20 AB    DD   0775          pushl   rpb$l_bootr1(r11)           ;Pass options switch settings
            7E     7C   0778          clrq    -(sp)                       ;no read data
          6140     9F   077A          pushab  (r1)[r0]                    ;address of message text
                        077D
                        077D  ;
                        077D  ; output the header part followed by the input code's message
                        077D  ;
                        077D
          20 AB    DD   077D          pushl   rpb$l_bootr1(r11)           ;Pass options switch settings
            7E     7C   0780          clrq    -(sp)                       ;setup header
       F93E CF     9F   0782          pushab  message_header              ;
 0000'CF    04     FB   0786          calls   #4,boo$readprompt           ;output header
 0000'CF    04     FB   078B          calls   #4,boo$readprompt           ;output message
          20 AB    DD   0790          pushl   rpb$l_bootr1(r11)           ;Pass options switch settings
            7E     7C   0793          clrq    -(sp)                       ;output device name
       F9D7 CF     9F   0795          pushab  boot_device_name
 0000'CF    04     FB   0799          calls   #4,boo$readprompt           ;
                        079E
                        079E  ;
                        079E  ; reload the input registers
                        079E  ;
                        079E
       5E   6B     D0   079E  20$:    movl    rpb$l_base(r11),sp          ;load sp
 5E    0200 CE     9E   07A1          movab   ^x200(sp),sp                ;
```

```
             50    1C AB    7D  07A6  1907          movq    rpb$L_bootr0(r11),r0    ;load r0,r1
             52    24 AB    7D  07AA  1908          movq    rpb$L_bootr2(r11),r2    ;load r2,r3
             54    2C AB    7D  07AE  1909          movq    rpb$L_bootr4(r11),r4    ;load r4,r5
             5C    18 AB    D0  07B2  1910          movl    rpb$L_haltcode(r11),ap  ;load halt code
             5A    10 AB    7D  07B6  1911          movq    rob$L_haltpc(r11),r10   ;restore PC,PSL
                                07BA  1912
                                07BA  1913 ;
                                07BA  1914 ; halt system, continue will restart the boot
                                07BA  1915 ;
                                07BA  1916
       23    00000F05 8F    DA  07BA  1917 25$:     mtpr    #console_halt,#pr$_txdb ;halt processor
                      F7    11  07C1  1918          brb     25$                     ;
```

```
                              07C3  ;++
                              07C3  ; validate_csr - test for present CSR
                              07C3  ;
                              07C3  ; functional description:
                              07C3  ;
                              07C3  ; This routine tests for a device CSR and returns to the caller's caller
                              07C3  ; if the CSR is not present. The CSR address is calculated from the base
                              07C3  ; CSR address and the controller number.
                              07C3  ;
                              07C3  ; inputs:
                              07C3  ;
                              07C3  ;       r7 = boot device descriptor address
                              07C3  ;       r11 = rpb address
                              07C3  ;
                              07C3  ; outputs:
                              07C3  ;
                              07C3  ;       return to caller implies that the device is present
                              07C3  ;       return to caller's caller with r0 = ss$_devassign+2
                              07C3  ;
                              07C3  ;       The RPB$L_PHYCSR value is filled in.
                              07C3  ;
                              07C3  ;       r0,r1 are destroyed
                              07C3  ;--
                              07C3
                              07C3  validate_csr:
                              07C3
           51    0A A7    D0  07C3          movl    bd_a_csr(r7),r1              ; assume fixed CSR address
 50   F9A5 CF    41 8F    83  07C7          subb3   #^a/A/,boot_device_name+2,r0 ; get boot controller number
                    05    13  07CE          beql    20$                          ; br if controller zero
                    23    10  07D0          bsbb    float_csr                    ; else, find floating CSR address
                 12 50    E9  07D2          blbc    r0,70$                       ; br if error
      F991 CF    DF'AF    9E  07D5  20$:    movab   b^60$,machine_check_continue ; change machine check addr
                    50    D4  07DB          clrl    r0                           ; set present flag
                    61    B5  07DD          tstw    (r1)                         ; test if CSR is present
           F989 CF        D4  07DF  60$:    clrl    machine_check_continue       ; zap machine check address
                    50    D5  07E3          tstl    r0                           ; CSR present?
                    09    13  07E5          beql    80$                          ; br in yes, continue
      50   084A 8F        3C  07E7  70$:    movzwl  #ss$_devassign!2,r0          ; set error but semi-success
                 51 8ED0      07EC          popl    r1                           ; pop return to caller
                    05        07EF          rsb
                              07F0          ;
                              07F0          ; success, save CSR address (r1).
                              07F0          ;
       54 AB     51       D0  07F0  80$:    movl    r1,rpb$l_csrphy(r11)         ; save CSR address
                          05  07F4          rsb
                          07F5          .sbttl  calculate floating CSR address
                          07F5   ;++
                          07F5   ; float_csr
                          07F5   ;
                          07F5   ; functional description:
                          07F5   ;
                          07F5   ;          This routine will take the rank of a given device and
                          07F5   ;          float the CSR's to find the corresonding controller.
                          07F5   ;
                          07F5   ;          The modulo for the device is non-zero if controllers are
                          07F5   ;          consecutive from the first in I/O space.  Else, the rank is
                          07F5   ;          non-zero and the device CSR address "floats" with other
                          07F5   ;          devices in the machine.
                          07F5   ;
                          07F5   ; inputs:
                          07F5   ;
                          07F5   ;          r0      - controller number in low byte (non-zero)
                          07F5   ;          r1      - CSR address for first controller of device
                          07F5   ;          r7      - boot device descriptor address
                          07F5   ;
                          07F5   ; outputs:
                          07F5   ;
                          07F5   ;          r0      - true or false
                          07F5   ;          r1      - CSR address, if success
                          07F5   ;
                          07F5   ;          All other registers are preserved
                          07F5   ;
                          07F5   ;
                          07F5   ; --
                          07F5
                          07F5   float_csr:
                3C    BB  07F5          pushr   #^m<r2,r3,r4,r5>              ; save registers
          54    50    9A  07F7          movzbl  r0,r4                         ; save controller number
   53     07    A7    9A  07FA          movzbl  bd_b_modulo(r7),r3            ; get modulo value for device
                4A    12  07FE          bneq    40$                           ; br if present, find controller
   55     06    A7    9A  0800   10$:   movzbl  bd_b_rank(r7),r5              ; get rank of device
                55    D7  0804          decl    r5                            ; Minus one
                5B    15  0806          bleq    80$                           ; br if bad, return error
          07    54    D1  0808          cmpl    r4,#max_ctrlrs                ; is controller number reasonalbe?
                59    1A  080B          bgtru   100$                          ; br if no, return error
F959 CF    6A'AF      9E  080D          movab   b^120$,machine_check_continue ; change machine check addr
   52   F95E CF       9E  0813          movab   modulo_tbl,r2                 ; get device CSR modulo table
51     20000008 8F    D0  0818          movl    #phy_a_io_space+8,r1          ; get start CSR address
                          081F
                          081F          ;
                          081F          ; at this point:
                          081F          ;
                          081F          ;       r1      - physical address of CSR for first floating device
                          081F          ;       r2      - address of device modulo table
                          081F          ;       r3      - scratch
                          081F          ;       r4      - controller number
                          081F          ;       r5      - rank (non-zero value)
                          081F
                61    B5  081F   20$:   tstw    (r1)                          ; is CSR address present?
                          0821
                55    D5  0821          tstl    r5                            ; is rank now zero
                11    13  0823          beql    25$                           ; Br if yes, continue
```

```
             53    62  9A  0825       movzbl  (r2),r3                 ; get device's modulo value
                   3C  13  0828       beql    100$                    ; error, if end of table
        50   53    01  C1  082A       addl3   #1,r3,r0                ; skip to next CSR set
             51    50  C0  082E       addl    r0,r1                   ;       ...
             51    53  CA  0831       bicl    r3,r1                   ; and round down
                   E9  11  0834       brb     20$                     ; loop if we have not reached our device
                           0836
                           0836 25$:  ;
                           0836       ; rank is now zero, r1 is where the first controller for
                           0836       ; our device should be.
                           0836       ;
                           0836
                   53  D6  0836       incl    r3                      ; round up modulo value
   F92E CF      63'AF  9E  0838       movab   b^80$,machine_check_continue ; new exception handler
                   05  11  083E       brb     35$                     ; get into loop to find right controller
             51    53  C0  0840 30$:  addl    r3,r1                   ; skip to next controller
                   61  B5  0843       tstw    (r1)                    ; is CSR address present?
             F8    54  F5  0845 35$:  sobgtr  r4,30$                  ; loop if more
                   12  11  0848       brb     60$
                           084A
                           084A 40$:  ;
                           084A       ; modulo is non-zero calculate where our controller must be.
                           084A       ;
                           084A       ; at this point:
                           084A       ;
                           084A       ;          r1       - physical address of CSR for first fixed device
                           084A       ;          r3       - modulo value
                           084A       ;          r4       - controller number
                           084A       ;          r7       - boot device descriptor address
                           084A       ;
                           084A       ;
     08 A7         54  91  084A       cmpb    r4,bd_b_max_ctrl(r7)    ; is controller # in range?
                   06  15  084E       bleq    50$                     ; br if yes, continue
        54      08 A7  C2  0850       subl    bd_b_max_ctrl(r7),r4    ; remove fixed one's from list
                   AA  11  0854       brb     10$                     ; now find floating device
             54    53  C4  0856 50$:  mull    r3,r4                   ; compute controller offset
             51    54  C0  0859       addl    r4,r1                   ; and adjust CSR address
           F90C    CF  D4  085C 60$:  clrl    machine_check_continue  ; reset machine check handler
             50    01  9A  0860       movzbl  #1,r0                   ; return status
                   3C  BA  0863 80$:  popr    #^m<r2,r3,r4,r5>        ; restore registers
                       05  0865       rsb
                           0866
                           0866 100$: ;
                           0866       ; no modulo value in table, we went past end!
                           0866       ;
                   50  D4  0866       clrl    r0                      ; return failure
                   F9  11  0868       brb     80$
                           086A
                           086A       ;
                           086A 120$: ; no CSR address present, move to next device in modulo table
                           086A       ;
                           086A
                   55  D7  086A       decl    r5                      ; count down rank
                   52  D6  086C       incl    r2                      ; skip to next modulo value
             53    62  9A  086E       movzbl  (r2),r3                 ; get device's modulo value
                   F3  13  0871       beql    100$                    ; error, if end of table
        50   53    01  C1  0873       addl3   #1,r3,r0                ; skip to next CSR set
```

```
                 51   50   C0   0877   2080        addl    r0,r1                    ;      ...
                 51   53   CA   087A   2081        bicl    r3,r1                    ; and round down
                      A0   11   087D   2082        brb     20$                      ; continue
                                087F   2083
```

```
087F    ;++
087F    ; unfielded_scb_int
087F    ; secondary_scb_int
087F    ;
087F    ; functional description:
087F    ;
087F    ; This routine is executed if an unwanted SCB interrupt occurs during
087F    ; booting. An error message is displayed and the system is halted.
087F    ;
087F    ; inputs:
087F    ;
087F    ;       scb interrupt stack
087F    ;
087F    ; outputs:
087F    ;
087F    ;       none
087F    ;--
087F
087F            .align  long
0880
0880    unfielded_scb_int:
0880
0880            fatal_message   scbint
0888
0888    secondary_scb_int:
0888
0888            fatal_message   2ndint
```

```
                               0890  ;++
                               0890  ; verify_boot_block
                               0890  ;
                               0890  ; functional description:
                               0890  ;
                               0890  ; This routine verifyes a small memory section as a boot block descriptor.
                               0890  ; It is used to verify a disk boot block or a ROM id block.
                               0890  ;
                               0890  ;
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;       BB+0:                          |   1   |   n   |   any value   |
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;                                      | Low LBN       | High LBN      |
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;       BB+(2*n)+0:                    |  Chk  |   k   |       18(Hex)|
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;                                      | any value, most likely 0      |
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;       BB+(2*n)+8:                    | size in blocks of the image   |
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;       BB+(2*n)+12:                   | offset to load                |
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;       BB+(2*n)+16:                   | offset into image to start    |
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;       BB+(2*n)+20:                   | sum of the previous three LW's|
                               0890  ;                                      +-------+-------+---------------+
                               0890  ;
                               0890  ; inputs:
                               0890  ;
                               0890  ;       r1 = address of the block
                               0890  ;
                               0890  ; outputs:
                               0890  ;
                               0890  ;       r0 = true or false
                               0890  ;       r1 = original address
                               0890  ;
                               0890  ;       r2 is destroyed
                               0890  ;--
                               0890
                               0890  verify_boot_block:
                               0890
                   50    D4    0890          clrl    r0                      ;assume not a valid block
             18    61    B1    0892          cmpw    (r1),#^x18              .VAX instruction set id?
                   20    12    0895          bneq    10$                     ;br if no
   52     02 A1    18    81    0897          addb3   #^x18,2(r1),r2          ;get optional value
                52 52    92    089C          mcomb   r2,r2                   ;ones's complement it
          03 A1    52    91    089F          cmpb    r2,3(r1)                ;check check sum byte
                   12    12    08A3          bneq    10$                     ;continue if no match
 52    0C A1    08 A1    C1    08A5          addl3   8(r1),12(r1),r2         ;check other words
             52 10 A1    C0    08AB          addl    16(r1),r2               ;get augment to load address
          14 A1    52    D1    08AF          cmpl    r2,20(r1)               ;match?
                   02    12    08B3          bneq    10$                     ;br if no
                   50    D6    08B5          incl    r0                      ;success
                         05    08B7  10$:    rsb
```

```
                                 08B8  2170 ; ++
                                 08B8  2171 ; verify_image_memory
                                 08B8  2172 ;
                                 08B8  2173 ; functional description:
                                 08B8  2174 ;
                                 08B8  2175 ; This routine checks for n contiguous pages from the established load
                                 08B8  2176 ; address.
                                 08B8  2177 ;
                                 08B8  2178 ; inputs:
                                 08B8  2179 ;
                                 08B8  2180 ;        r9 = desired page count
                                 08B8  2181 ;        r10 = target load address
                                 08B8  2182 ;        r11 = address of the RPB
                                 08B8  2183 ;        ap = boot argument list
                                 08B8  2184 ;
                                 08B8  2185
                                 08B8  2186 verify_image_memory:
                                 08B8  2187
   50   0601 8F     3C  08B8  2188          movzwl  #ss$_bufferovf,r0          ;set error code
   52     5B   17   9C  08BD  2189          rotl    #<32=9>,r11,r2             ;PFN for RPB
     52     7F A2   DE  08C1  2190          moval   127(r2),r2                 ;Last PFN guaranteed to be good
   51     5A   17   9C  08C5  2191          rotl    #<32-9>,r10,r1             ;Starting PFN for read
          51   59   C0  08C9  2192          addl    r9,r1                      ;Last+1 PFN needed to be good
               05   11  08CC  2193          brb     30$                        ;Zero or more iterations
 07 18 BC      52   E1  08CE  2194 10$:     bbc     r2,@vmb$q_pfnmap+4(ap),40$ ;Branch if cannot
                        08D3  2195                                             ;read the entire secondary boot
    F7 52      51   F2  08D3  2196 30$:     aoblss  r1,r2,10$                  ;Check the next page
       50      01   D0  08D7  2197          movl    #1,r0                      ;correct
                    05  08DA  2198 40$:     rsb
```

```
                         08DB  ;++
                         08DB  ; write_timeout
                         08DB  ;
                         08DB  ; functional description:
                         08DB  ;
                         08DB  ; This sequence runs when a write timeout interrupt occurs.
                         08DB  ;
                         08DB  ; inputs:
                         08DB  ;
                         08DB  ;        PC/PSL are on the stack
                         08DB  ;        machine_check_continue = address to continue at or 0
                         08DB  ;
                         08DB  ; outputs:
                         08DB  ;
                         08DB  ;        r0 = error code
                         08DB  ;--
                         08DB
                         08DB
                         08DB           .align  long
                         08DC
                         08DC  write_timeout_int:
                         08DC
6E    F88C CF     D0     08DC           movl    machine_check_continue,(sp) ;reset PC
            9D    13     08E1           beql    unfielded_scb_int          ;unexpected error if no continue addr
      50    02    D0     08E3           movl    #k_bus.timeout,r0          ;set code
                  02     08E6           rei                                ;done
                         08E7
                         08E7           .end    ROM_START
```

```
A_2NDINT                        0000017E R     03     DEVNAMEPROMPT                   00000046 R     02
A_BADFILENAME                   000000A8 R     03     DIAGFILE                        0000001D R     02
A_BUFFEROVF                     000000BA R     03     DISK_BOOT                       00000576 R     02
A_CTRLERR                       000000EA R     03     EXE$GB_CPUTYPE                  ********   X   02
A_DEVASSIGN                     00000036 R     03     FATAL_ERROR                     00000756 R     02
A_DEVINACT                      00000101 R     03     FATAL_MEMORY_ERROR              000003C8 R     02
A_DEVOFFLINE                    00000123 R     03     FIL$GQ_CACHE                  = 00000128 RG    02
A_FILESTRUCT                    00000066 R     C3     FIL$OPENFILE                    ********   X   02
A_FILNOTCNTG                    0000008A R     03     FILE_CACHE_DESC                 00000160 R     02
A_MEMERR                        00000136 R     03     FLOAT_CSR                       000007F5 R     02
A_NOROM                         000001B1 R     03     IGNORE_SCB_INT                  00000730 R     02
A_NOSUCHDEV                     00000000 R     03     IHD$W_ACTIVOFF                = 00000002
A_NOSUCHFILE                    0000004D R     03     INI$BRK                         0000020E RG    02
A_NOSUCHNODE                    000001CB R     03     IO$_READLBLK                  = 00000021
A_SCBINT                        00000153 R     03     IPL$_POWER                    = 0000001F
BDT$L_ACTION                    00000004              K_BUS.TIMEOUT                 = 00000002
BDT$L_ADDR                      0000000C              K_MAX_BOOT_LEN                = 00018000
BDT$L_AUXDRNAME                 00000018              K_MAX_IO_PAGES                = 0000007F
BDT$L_CPUTYPE                   00000000              K_MAX_MEMORY_PAGES            = 00002000
BDT$L_DEVNAME                   00000024              K_NEXT_BOOT_ADDR              = 00000E00
BDT$L_DEVTYPE                   00000002              K_PARITY.ERROR                = 00000001
BDT$L_DRIVRNAME                 00000014              K_PFN_MAP_ADDR                = 00000400
BDT$L_ENTRY                     00000010              K_SCB_ADDR                    = 00000000
BDT$L_SIZE                      00000008              LAST_MSG                      = 000001CB R     03
BDT$L_UNIT_DISC                 00000020              LED_BOOT_INPROGRESS           = 00000F0E
BDT$L_UNIT_INIT                 0000001C              LED_MEMORY_OK                 = 00000F0D
BD_A_CSR                        0000000A              LED_TRANSFER_CONTROL          = 00000F0F
BD_A_ROUTINE                    0000000E              MACHINE_CHECK_CONTINUE          0000016C R     02
BD_B_HIGH_UNIT                  00000004              MACHINE_CHECK_DETECT            00000734 R     02
BD_B_MAX_CTRL                   00000008              MAX_CTRLRS                    = 00000007
BD_B_MODULO                     00000007              MESSAGE_BASE                    000000D6 R     02
BD_B_RANK                       00000006              MESSAGE_HEADER                  000000C4 R     02
BD_B_SPARE                      00000009              MODULO_TBL                      00000175 R     02
BD_B_TYPE                       00000005              MSV11_CSR_BASE                = 20001440
BD_L_NAME                       00000000              MSV11_CSR_PARITY_ENABLE       = 00000001
BD_S_BD                         00000012              NAMEPROMPT                      00000039 R     02
BEGIN_BOOT                      000003D0 R     02     NDT$_UBO                      = 00000028
BOO$AL_VECTOR                   ********   X   02     NETWORK_BOOT                    00000549 R     02
BOO$CACHE_ALLOC                 ********   X   02     NEXT_PAGE                       00000368 R     02
BOO$CACHE_OPEN                  ********   X   02     NOBRK                           0000020F R     02
BOO$DOWNLINE_LOAD               ********   X   02     NOXDT                           00000212 R     02
BOO$GB_SYSTEMID               = 00000148 RG    02     NO_DISK_BOOT_DEVICE_LIST        0000009E R     02
BOO$GL_RPBBASE                  00000168 RG    02     NO_FIT                          0000072C R     02
BOO$IMAGE_ATT                   ********   X   02     NXM_MEMORY                      0000037E R     02
BOO$QIO                         ********   X   02     PAGE_BOUNDARY                   000002D7 R     02
BOO$READPROMPT                  ********   X   02     PATCH_DEVICE_NAME               00000000 RG    02
BOOT_DEVICE_LIST                0000007A R     02     PHY_A_IO_SPACE                = 20000000
BOOT_DEVICE_NAME                00000170 R     02     PR$V_SID_TYPE                 = 00000018
BOOT_DISK_UNIT                  0000060A R     02     PR$_IPL                       = 00000012
BOOT_FILE                       000006AD R     02     PR$_MCESR                     = 00000026
BQO$C_MOVE                    = 00000018              PR$_SCBB                      = 00000011
BQO$L_UNIT_INIT               = 0000001C              PR$_SID                       = 0000003E
BTD$K_DL                      = 00000002              PR$_TXDB                      = 00000023
BTD$K_PROM                    = 00000008              PROM_BOOT                       0000050F R     02
BTD$K_QNA                     = 00000060              READFILE                        000006FD R     02
BTD$K_UDA                     = 00000011              READIN_BOOT                     000006EF R     02
CONSOLE_HALT                  = 00000F05              ROM_BASE                        00000000 R     05
```

```
ROM_START                        00000190 R     02
RPB$B_CONFREG                  = 00000090
RPB$B_CTRLLTR                  = 00000108
RPB$B_DEVTYP                   = 00000066
RPB$B_HDRPGCNT                 = 000000A0
RPB$B_SLAVE                    = 00000067
RPB$C_LENGTH                   = 00000109
RPB$C_MEMDSCSIZ                = 00000008
RPB$C_NMEMDSC                  = 00000008
RPB$L_BASE                     = 00000000
RPB$L_BOOTR0                   = 0000001C
RPB$L_BOOTR1                   = 00000020
RPB$L_BOOTR2                   = 00000024
RPB$L_BOOTR3                   = 00000028
RPB$L_BOOTR4                   = 0000002C
RPB$L_BOOTR5                   = 00000030
RPB$L_CHKSUM                   = 00000008
RPB$L_CSRPHY                   = 00000054
RPB$L_FILLBN                   = 0000003C
RPB$L_HALTCODE                 = 00000018
RPB$L_HALTPC                   = 00000010
RPB$L_IOVEC                    = 00000034
RPB$L_IOVECSZ                  = 00000038
RPB$L_MEMDSC                   = 000000BC
RPB$L_PFNCNT                   = 0000004C
RPB$L_RESTART                  = 00000004
RPB$L_RSTRTFLG                 = 0000000C
RPB$L_SCBB                     = 000000B0
RPB$Q_PFNMAP                   = 00000044
RPB$S_TOPSYS                   = 00000004
RPB$T_FILE                     = 00000068
RPB$V_BBLOCK                   = 00000003
RPB$V_BOOBPT                   = 00000005
RPB$V_DIAG                     = 00000004
RPB$V_HALT                     = 00000009
RPB$V_HEADER                   = 00000006
RPB$V_SOLICT                   = 00000008
RPB$V_TOPSYS                   = 0000001C
RPB$W_BOOTNDT                  = 000000A1
RPB$W_UNIT                     = 00000064
SCB_A_BREAKPOINT               = 0000002C
SCB_A_MCHECK                   = 00000004
SCB_A_TRACE_TRAP               = 00000028
SCB_A_WRITE_TIMEOUT            = 00000060
SECONDARY_SCB_INT                00000888 R     02
SECOND_PARAM                     00000124 R     02
SS$_2NDINT                     = 00008010
SS$_BADCHKSUM                  = 00000808
SS$_BADFILEHDR                 = 00000810
SS$_BADFILENAME                = 00000818
SS$_BADIRECTORY                = 00000828
SS$_BUFFEROVF                  = 00000601
SS$_CTRLERR                    = 00000054
SS$_DEVASSIGN                  = 00000848
SS$_DEVINACT                   = 000020D4
SS$_DEVOFFLINE                 = 00000084
SS$_ENDOFFILE                  = 00000870

SS$_FILESTRUCT                 = 000008C0
SS$_FILNOTCNTG                 = 000002AC
SS$_MEMERR                     = 00008000
SS$_NOROM                      = 00008018
SS$_NOSUCHDEV                  = 00000908
SS$_NOSUCHFILE                 = 00000910
SS$_NOSUCHNODE                 = 0000028C
SS$_SCBINT                     = 00008008
SWITCH_V_DISK_BOOT             = 00000007
SWITCH_V_QVSS                  = 00000006
SYNONYM_DEVICE_LIST              00000056 R     02
UNFIELDED_SCB_INT                00000880 R     02
VALIDATE_CSR                     000007C3 R     02
VERIFY_BOOT_BLOCK                00000890 R     02
VERIFY_IMAGE_MEMORY              000008B8 R     02
VMB$B_SYSTEMID                   00000024
VMB$C_ARGBYTCNT                  0000003C
VMB$L_CI_HIPFN                   00000030
VMB$L_FLAGS                      0000002C
VMB$L_HI_PFN                     00000010
VMB$L_LO_PFN                     0000000C
VMB$Q_FILECACHE                  00000004
VMB$Q_NODENAME                   00000034
VMB$Q_PFNMAP                     00000014
VMB$Q_UCODE                      0000001C
VMB$SECONDARY                  = 00000001W G
VMB_END                          00000000 RG    04
VMSFILE                          00000004 R     02
WRITE_TIMEOUT_INT                000008DC R     02
XDT$BREAKPOINT                   ********W GX   02
XDT$INITIAL_BREAK                ********W GX   02
XDT$TRACE_TRAP                   ********W GX   02
```

```
                                                +----------------+
                                                ! Psect synopsis !
                                                +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           0000003C  (   60.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$$04BOOT                      000008E7  ( 2279.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$$10BOOT                      000001E9  (  489.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
___ZZZVMB_END                   00000000  (    0.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC PAGE
$$$$00BOOT                      00000008  (    8.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                            +------------------------+
                                            ! Performance indicators !
                                            +------------------------+

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                     100    00:00:00.21    00:00:01.80
Command processing                 126    00:00:00.65    00:00:04.78
Pass 1                             426    00:00:17.00    00:00:50.50
Symbol table sort                    0    00:00:02.09    00:00:02.62
Pass 2                             373    00:00:05.41    00:00:16.58
Symbol table output                 27    00:00:00.18    00:00:00.30
Psect synopsis output                4    00:00:00.04    00:00:00.04
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals              1058    00:00:25.58    00:01:16.62

The working set limit was 2100 pages.
92033 bytes (180 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1206 non-local and 79 local symbols.
2227 source lines were read in Pass 1, producing 26 object records in Pass 2.
25 pages of virtual memory were used to define 24 macros.

                                          +--------------------------+
                                          ! Macro library statistics !
                                          +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA18:[UV1RCM.OBJ]LIBUV1.MLB;1                   6
_$255$DUA18:[UV1ROM.OBJ]VMB.MLB;1                      4
_$255$CUA18:[SYSLIB]STARLET.MLB;3                      7
TOTALS (all libraries)                                17

1220 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:VMBUVAX1P/OBJ=OBJ$:VMBUVAX1P MSRC$:VMBUVAX1P/UPDATE=(BUG$:VMBUVAX1P)+LIB$:VMB/LIB+LIB$:LIBUV1/LIB
```

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

UAFPARSE
LIS

BOOTDRUV1P
LIS

VMBUVAX1P
LIS

UV1ROM

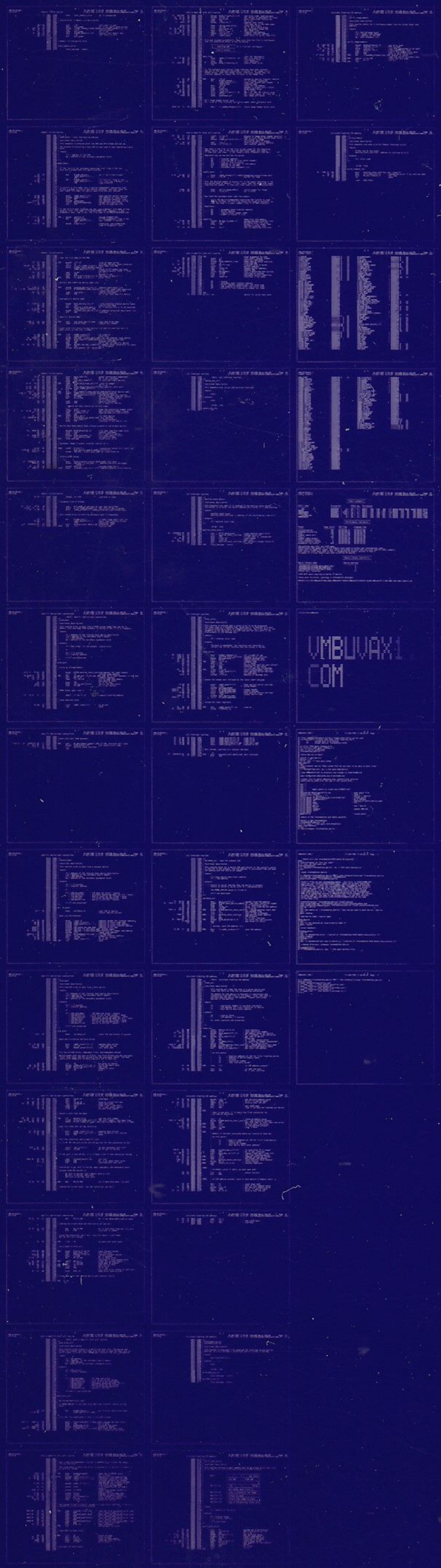
VMBUVAX1P
MAP

CONIO
LIS

UVMS

REMOVE
COM

0452 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD



VMBUVAX1
COM