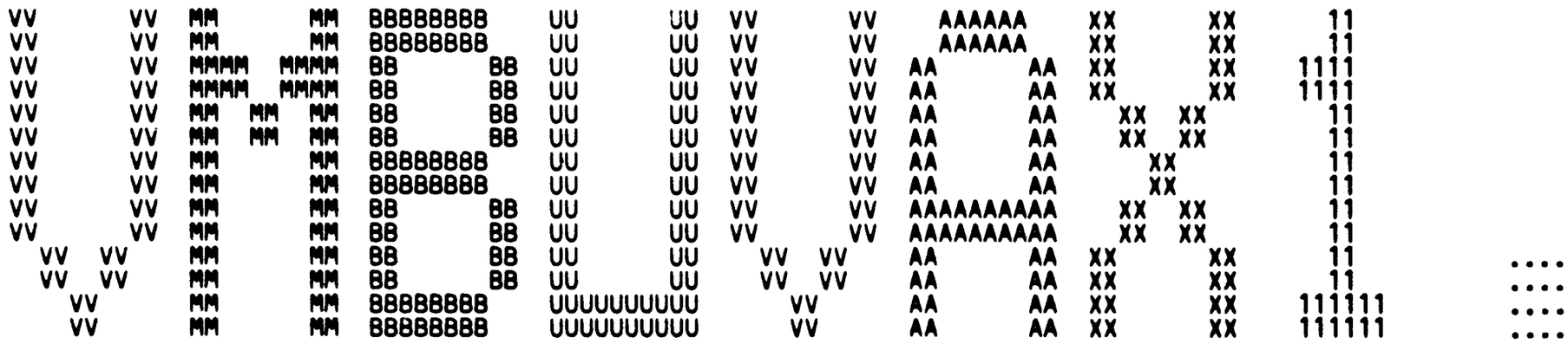
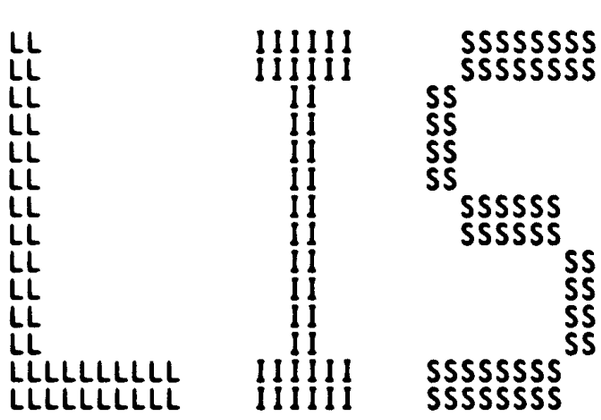
```
UUU          UUU  VVV         VVV        111       RRRRRRRRRRR        000000000     MMM         MMM
UUU          UUU  VVV         VVV        111       RRRRRRRRRRRR       000000000     MMM         MMM
UUU          UUU  VVV         VVV        111       RRRRRRRRRRRR       000000000     MMM         MMM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


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


LL IIIIII SSSSSSS
LL IIIIII SSSSSSS
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 SSSSSSS
LLLLLLLLLL IIIIII SSSSSSS

```
0000      1          .title  VMB_MICROVAX_I
0000      2          .ident /V01.0-00/
0000      3 ;
0000      4 ;*****************************************************************************
0000      5 ;*                                                                           *
0000      6 ;*  Copyright (c) 1984                                                       *
0000      7 ;*  by DIGITAL Equipment Corporation, Maynard, Mass.                         *
0000      8 ;*                                                                           *
0000      9 ;*  This software is furnished under a license and may be used and  copied   *
0000     10 ;*  only  in  accordance  with  the  terms  of  such  license and with the   *
0000     11 ;*  inclusion of the above copyright notice.  This software or  any  other   *
0000     12 ;*  copies  thereof may not be provided or otherwise made available to any   *
0000     13 ;*  other person.  No title to and ownership of  the  software  is  hereby   *
0000     14 ;*  transferred.                                                             *
0000     15 ;*                                                                           *
0000     16 ;*  The information in this software is subject to change  without  notice   *
0000     17 ;*  and  should  not  be  construed  as  a commitment by DIGITAL Equipment   *
0000     18 ;*  Corporation.                                                             *
0000     19 ;*                                                                           *
0000     20 ;*  DIGITAL assumes no responsibility for the use or  reliability  of  its   *
0000     21 ;*  software on equipment which is not supplied by DIGITAL.                  *
0000     22 ;*                                                                           *
0000     23 ;*****************************************************************************
0000     24 ;
0000     25 ; Facility:
0000     26 ;
0000     27 ;       Bootstrap ROM code for the MicroVAX I.
0000     28 ;
0000     29 ; Abstract:
0000     30 ;
0000     31 ;       This module contains a hybrid bootstrap for VMS, ELN, diagnostics
0000     32 ;       and other systems that is a combination of the other VAX hardware
0000     33 ;       boot functions and VMB's traditional function. It was edited down
0000     34 ;       from the VMS VMB and then enhanced to perform the priority ordered
0000     35 ;       boot.
0000     36 ;
0000     37 ; Author: R. Heinen
0000     38 ;
0000     39 ; Date: July 1983
0000     40 ;--
0000     41
0000     42          $bdtdef                         ;define boot driver descriptor
0000     43          $bqodef                         ;define boot driver offsets
0000     44          $btddef                         ;define boot device types
0000     45          $ihddef                         ;define VMS image header
0000     46          $iodef                          ;define I/O function codes
0000     47          $ipldef                         ;ipl's
0000     48          $ndtdef                         ;define adapter types
0000     49          $prdef                          ;processor registers
0000     50          $pruvldef                       ;processor registers for MicroVAX I
0000     51          $rpbdef                         ;RPB
0000     52          $ssdef                          ;define VMS status codes
0000     53          $vmbargdef                      ;define VMB arguments
0000     54
0000     55
0000     56 ;
0000     57 ; define some new btd symbols
```

```
                     0000     58 ;
                     0000     59
00000008             0000     60          btd$k_prom = 8
00000060             0000     61          btd$k_qna = 96
                     0000     62
                     0000     63 ;*****************************************************************************
                     0000     64
                     0000     65 ;
                     0000     66 ; define a macro to define boot driver names etc.
                     0000     67 ;
                     0000     68
                     0000     69          .macro boot_device name,h_unit,pcsr,type,rtn,?l1
                     0000     70 l1:      .asciz  /name/
                     0000     71          .byte   h_unit
                     0000     72          .byte   type
                     0000     73          .long   pcsr+phy_a_io_space
                     0000     74          .long   rtn-l1
                     0000     75          .endm
                     0000     76
                     0000     77 ;
                     0000     78 ; define macros to aid with error message printing
                     0000     79 ;
                     0000     80
                     0000     81          .macro  fatal_message   code
                     0000     82          .if     nb,code
                     0000     83          movzwl  #ss$_'code,r0
                     0000     84          .endc
                     0000     85          brw     fatal_error
                     0000     86          .endm
                     0000     87
                     0000     88          .macro  msg_def mname,txt
                     0000     89          .word   ss$_'mname
                     0000     90          .if     nb,<txt>
                     0000     91          .word   a_'mname-.
                     0000     92          .save_psect
                     0000     93          .psect  $$$$10boot,byte
                     0000     94 last_msg = .
                     0000     95 a_'mname:
                     0000     96          .asciz  \txt \
                     0000     97          .restore_psect
                     0000     98          .iff
                     0000     99          .word   last_msg-.
                     0000    100          .endc
                     0000    101          .endm
                     0000    102
                     0000    103 ;
                     0000    104 ; define local data structure offsets
                     0000    105 ;
                     0000    106
                     0000    107 ;
                     0000    108 ; define boot device desc structure
                     0000    109 ;
                     0000    110
                     0000    111          $defini bd
00000004             0000    112 bd_l_name:      .blkl   1
00000005             0004    113 bd_b_high_unit: .blkb   1
00000006             0005    114 bd_b_type:      .blkb   1
```

```
0000000A  0006   115 bd_a_csr:       .blkl   1
0000000E  000A   116 bd_a_routine:   .blkl   1
          000E   117 bd_s_bd:
          000E   118         $defend bd
          0000   119
          0000   120 ;
          0000   121 ; define local data constants
          0000   122 ;
          0000   123
20000000  0000   124 phy_a_io_space          = ^x20000000    ;physical address of I/O space
          0000   125
          0000   126 ;
          0000   127 ; define extents
          0000   128 ;
          0000   129
00002000  0000   130 k_max_memory_pages      = 8192          ;max number of pages
0000007F  0000   131 k_max_io_pages          = 127           ;max pages in one I/O transfer
          0000   132
          0000   133 ;
          0000   134 ; define addresses in 64K segment
          0000   135 ;
          0000   136
00000000  0000   137 k_rpb_addr              = 0
00000200  0000   138 k_rom_code_addr         = ^x200         ;allow for 16K
00004200  0000   139 k_scb_addr              = ^x4200        ;
00004600  0000   140 k_pfn_map_addr          = ^x4600        ;2 pages
          0000   141                                       ;stack area of three pages
00005000  0000   142 k_secondary_boot_addr   = ^x5000
          0000   143
          0000   144 ;
          0000   145 ; define MicroVAX I machine check codes used here
          0000   146 ;
          0000   147
00000001  0000   148 k_parity.error          = 1
00000002  0000   149 k_bus.timeout           = 2
          0000   150
          0000   151 ;
          0000   152 ; define scb vectors used here
          0000   153 ;
          0000   154
00000004  0000   155 scb_a_mcheck            = 4
00000060  0000   156 scb_a_write_timeout     = ^x60
0000002C  0000   157 scb_a_breakpoint        = ^x2c
00000028  0000   158 scb_a_trace_trap        = ^x28
          0000   159
          0000   160 ;
          0000   161 ; define bits in MicroVAX I switch pack
          0000   162 ;
          0000   163
00000006  0000   164 switch_v_QVSS           = 6             ; 1 if normal, 0 if QVSS
00000007  0000   165 switch_v_disk_boot      = 7             ; 1 if normal, 0 if disable disk search
          0000   166
          0000   167 ;
          0000   168 ; define MSV-11 Memory controller values
          0000   169 ;
          0000   170
20001440  0000   171 msv11_csr_base = ^x1440 + phy_a_io_space
```

```
00000001   0000   172 msv11_csr_parity_enable = 1
           0000   173
           0000   174 ;
           0000   175 ; define led values
           0000   176 ;
           0000   177
00000F0D   0000   178 led_memory_ok = ^xf0d
00000F0E   0000   179 led_boot_inprogress = ^xf0e
00000F0F   0000   180 led_transfer_control = ^xf0f
           0000   181
           0000   182 ;
           0000   183 ; define console halt code
           0000   184 ;
           0000   185
00000F05   0000   186 console_halt = ^xf05
```

```
                                read write data
                                    0000   188          .sbttl  read write data
                                    0000   189
                                00000000   190          .psect  $$$$04boot,long
                                    0000   191
                                    0000   192 ;
                                    0000   193 ; strings used for file opens
                                    0000   194 ;
                                    0000   195
                                    0000   196 vmsfil :                                 ;Name of standard secondary
58 45 53 59 53 2E 30 53 59 53 5B 00'0000   197          .ASCIC  /[SYSO.SYSEXE]SYSBOOT.EXE/ ;bootstrap image file.
58 45 2E 54 4F 4F 42 53 59 53 5D 45 000C
                                   45 0018
                                   18 0000
                                    0019   198
                                    0019   199 diagfile:                                 ;Name of standard diagnostic
41 4D 53 59 53 2E 30 53 59 53 5B 00'0019   200          .ASCIC  /[SYSO.SYSMAINT]DIAGBOOT.EXE/ ;secondary bootstrap image.
54 4F 4F 42 47 41 49 44 5D 54 4E 49 0025
                        45 58 45 2E 0031
                                   1B 0019
                                    0035   201
                                    0035   202 nameprompt:                                ;Prompt string for secondary
00 3A 65 6C 69 66 74 6F 6F 42 0A 0D 0035   203          .ASCIZ  <13><10>/Bootfile:/        ;boot file name.
                                    0041   204
                                    0041   205 ;
                                    0041   206 ; define two boot device priority lists
                                    0041   207 ;
                                    0041   208
                                    0041   209 boot_device_list:
                                    0041   210
                                    0041   211          boot_device     DUA,3,<^X1468>,btd$k_uda,disk_boot
                                    004F   212
                                    004F   213 no_disk_boot_device_list:
                                    004F   214
                                    004F   215          boot_device     PRA,0,<^x0000>,btd$k_prom,prom_boot
                                    005D   216          boot_device     XQA,0,<^X1920>,btd$k_qna,network_boot
                                0000 006B  217          .word   0                          ;implant a zero name
                                    006D   218
                                    006D   219 ;
                                    006D   220 ; define text to correspond to ss$_ values.
                                    006D   221 ;
                                    006D   222
                                    006D   223 message_header:
52 45 2D 46 2D 54 4F 4F 42 25 0A 0D 006D   224          .asciz  <13><10>/%BOOT-F-ERROR, /
                     00 20 2C 52 4F 52 0079
                                    007F   225
                                    007F   226 message_base:
                                    007F   227
                                    007F   228 ;
                                    007F   229 ; define some ss$_ codes that are only used here
                                    007F   230 ;
                                    007F   231
                            00008000 007F  232 ss$_memerr = ^x8000
                            00008008 007F  233 ss$_scbint = ^x8008
                            00008010 007F  234 ss$_2ndint = ^x8010
                            00008018 007F  235 ss$_norom  = ^x8018
                                    007F   236
                                    007F   237 msg_def nosuchdev,<None of the bootable devices contain a program image>
```

```
                                read write data
                                0083   238 msg_def devassign,<Device is not present>
                                0087   239 msg_def nosuchfile,<Program image not found>
                                008B   240 msg_def filestruct,<Invalid boot device file structure>
                                008F   241         msg_def badchksum
                                0093   242         msg_def badfilehdr
                                0097   243         msg_def badirectory
                                009B   244 msg_def filnotcntg,<Invalid program image format>
                                009F   245 msg_def endoffile
                                00A3   246 msg_def badfilename,<Invalid filename>
                                00A7   247 msg_def bufferovf,<Program image does not fit in available memory>
                                00AB   248 msg_def ctrlerr,<Boot device I/O error>
                                00AF   249 msg_def devinact,<Failed to initialize boot device>
                                00B3   250 msg_def devoffline,<Device is offline>
                                00B7   251 msg_def memerr,<Memory initialization error>
                                00BB   252 msg_def scbint,<Unexpected SCB exception or machine check>
                                00BF   253 msg_def 2ndint,<Unexpected exception after starting program image>
                                00C3   254 msg_def norom,<No valid ROM image found>
                                00C7   255 msg_def nosuchnode,<No response from load server>
                          0000  00CB   256         .word   0                       ;terminate list
                                00CD   257
                                00CD   258 ;
                                00CD   259 ; writable data
                                00CD   260 ;
                                00CD   261         .ALIGN  LONG
                                00D0   262
                                00D0   263 ;
                                00D0   264 ; Parameter list handed from primary boot to secondary boot
                                00D0   265 ; The first location contains the argument count.  It is intended
                                00D0   266 ; that the secondary boot will know what is in the list based on
                                00D0   267 ; the argument count and the VMB version number.  This means that
                                00D0   268 ; new information should be placed at new offsets even if older
                                00D0   269 ; stuff becomes obsolete.  The VMB version number can be used to
                                00D0   270 ; totally change the argument meanings if necessary.
                                00D0   271 ;
                                00D0   272
                                00D0   273 second_param:
                    000000D4    00D0   274         fil$gq_cache    == .+vmb$q_filecache ;FILEREAD cache descriptor
                    000000F4    00D0   275         boo$gb_systemid == .+vmb$b_systemid  ;SCS system id
                    0000000E    00D0   276         .long   <vmb$c_argbytcnt-4>/4   ;Size of argument list
                                00D4   277         .rept   vmb$c_argbytcnt-4       ;Reserve space for the arguments
                                00D4   278         .byte   0
                          00    00D4   279         .endr
                                010C   280
                                010C   281 file_cache_desc:                         ;saved cache desc
                    00000000    010C   282         .long   0                       ;to re-init the cache after error
                    00000000    0110   283         .long   0
                                0114   284
                                0114   285 ;
                                0114   286 ; address of the RPB as a global
                                0114   287 ;
                                0114   288
                                0114   289 boo$gl_rpbbase::
                    00000000    0114   290         .long   0
                                0118   291
                                0118   292 ;
                                0118   293 ; machine check support
                                0118   294 ;
```

```
                     read write data
                     0118   295
                     0118   296 machine_check_continue:                ;contains 0 or that address to
0000011C             0118   297         .blkl   1                      ;transfer to after a machine check
                     011C   298
                     011C   299 ;
                     011C   300 ; error device name
                     011C   301 ;
                     011C   302
                     011C   303 boot_device_name:
00000000             011C   304         .long   0
       00            0120   305         .byte   0
```

```
0121    307          .sbttl  boot code
0121    308 ;++
0121    309 ; ROM_START
0121    310 ;
0121    311 ; functional description:
0121    312 ;
0121    313 ; This code is entered after the MicroVAX I microcode has completed its
0121    314 ; restart/boot/halt sequence. It runs at IPL 31, in Kernel mode on the
0121    315 ; interrupt stack. The action is to initialize an RPB, setup a bitmap of
0121    316 ; useable memory pages and load the next part of the system boot based on
0121    317 ; the input flag settings.
0121    318 ;
0121    319 ; If the inputs include a specific boot device name that device and only
0121    320 ; that device is booted. On the otherhand, if no specific boot device
0121    321 ; is specified then a priority ordered sequence of boot devices is tried.
0121    322 ; (In this case R0 will be 0 or contain all blanks.
0121    323 ;
0121    324 ; As follows:
0121    325 ;
0121    326 ;       DU units 0,1,2,3
0121    327 ;       Other disks
0121    328 ;       ROM (See below for an explanation of how the ROM is found.)
0121    329 ;       QNA
0121    330 ;
0121    331 ;       If none of these devices provide a bootstrap then a message is
0121    332 ;       displayed followed by a HALT.
0121    333 ;
0121    334 ; ROM systems are recognized by the boot memory search. A ROM system must
0121    335 ; be aligned on a 4KB boundary and contain a foot print which is the same
0121    336 ; as the second part of the boot block described below.
0121    337 ;
0121    338 ; If the boot is from a mass storage device then for each valid volume
0121    339 ; that is found, the volume is searched as a Files-11 volume and then
0121    340 ; the secondary boot image is found. If the volume is not a Files-11 volume
0121    341 ; then block 0 of the volume is read and checked to see if it meets the
0121    342 ; standard for the boot block format. If not, the volume is not used and the
0121    343 ; next volume is tried unless a specific device was specified by the user.
0121    344 ;
0121    345 ; The boot block format is:
0121    346 ;
0121    347 ;                       +-------+-------+-------+-------+
0121    348 ;       BB+0:           |     1 |     n |     any value |
0121    349 ;                       +-------+-------+-------+-------+
0121    350 ;                       | low LBN       | High LBN      |
0121    351 ;                       +-------+-------+-------+-------+
0121    352 ;
0121    353 ; This second part is used for both the boot block and the ROM system.
0121    354 ;
0121    355 ;                       +-------+-------+-------+-------+
0121    356 ;       BB+(2*n)+0:     |  Chk  |     k |       18(Hex) |
0121    357 ;                       +-------+-------+-------+-------+
0121    358 ;                       | any value, most likely 0      |
0121    359 ;                       +-------+-------+-------+-------+
0121    360 ;       BB+(2*n)+8:     | size in blocks of the image   |
0121    361 ;                       +-------+-------+-------+-------+
0121    362 ;       BB+(2*n)+12:    | load offset                   |
0121    363 ;                       +-------+-------+-------+-------+
```

```
0121    364 ;       BB+(2*n)+16:    ! offset into image to start    !
0121    365 ;                       +-------+-------+-------+-------+
0121    366 ;       BB+(2*n)+20:    ! sum of the previous three LW's'
0121    367 ;                       +-------+-------+-------+-------+
0121    368 ;
0121    369 ; The input bits in R5 can contain a bit that disables the Files-11 search.
0121    370 ;
0121    371 ; If a Files-11 boot is done then the file booted is either:
0121    372 ;
0121    373 ;       SYSBOOT.EXE - default
0121    374 ;       DIAGBOOT.EXE - R5 bit setting
0121    375 ;       solicited from the console
0121    376 ;
0121    377 ; Details of how the memory look and the register settings when the
0121    378 ; secondary bootstraps are entered are documented where the exits occur.
0121    379 ;
0121    380 ; inputs:
0121    381 ;
0121    382 ;       r0 = boot device name in ASCII or 0 if none specified
0121    383 ;       r1 = switch pack settings 1 is "ON", 0 is "OFF"
0121    384 ;
0121    385 ;               Bit     Meaning
0121    386 ;               ---     -------
0121    387 ;
0121    388 ;               7       Enable disk search during bootstrap
0121    389 ;
0121    390 ;               6       1 if VT100/VT200 console, 0 if QVSS video option
0121    391 ;
0121    392 ;               4-5     Halt action
0121    393 ;               3       Console Break enabled
0121    394 ;
0121    395 ;               2       Reserved
0121    396 ;
0121    397 ;               0-1     Console baud rate
0121    398 ;
0121    399 ;       R5 = software boot control flags from the /N boot command qualifier.
0121    400 ;
0121    401 ;       The following bits are used by this boot ROM code:
0121    402 ;
0121    403 ;               Bit     Meaning
0121    404 ;               ---     -------
0121    405 ;
0121    406 ;                3      RPB$V_BBLOCK.
0121    407 ;                       If set, the attempt to Files-11 boot is skipped
0121    408 ;                       and only the boot block type boot is done.
0121    409 ;
0121    410 ;                4      RPB$V_DIAG.
0121    411 ;                       Diagnostic boot. Secondary bootstrap is image
0121    412 ;                       called [SYSMAINT]DIAGBOOT.EXE.
0121    413 ;
0121    414 ;                6      RPB$V_HEADER.
0121    415 ;                       Image header. Takes the transfer address of the
0121    416 ;                       secondary bootstrap image from that file's
0121    417 ;                       image header. If RPB$V_HEADER is not set,
0121    418 ;                       transfers control to the first byte of the
0121    419 ;                       secondary boot file.
0121    420 ;
```

```
0121   421 ;                8       RPB$V_SOLICT.
0121   422 ;                        File name. Prompt for the name of a
0121   423 ;                        secondary bootstrap file.
0121   424 ;
0121   425 ;                9       RPB$V_HALT.
0121   426 ;                        Halt before transfer. Executes a HALT
0121   427 ;                        instruction before transferring control to the
0121   428 ;                        secondary bootstrap.
0121   429 ;
0121   430 ;               <31:28> RPB$V_TOPSYS
0121   431 ;                        Specifies the top level directory number for system
0121   432 ;                        disks with multiple systems
0121   433 ;
0121   434 ;        The following bits are NOT used by this boot ROM code:
0121   435 ;
0121   436 ;                Bit     Meaning
0121   437 ;                ---     -------
0121   438 ;
0121   439 ;                 0      RPB$V_CONV.
0121   440 ;                        Conversational boot. At various points in the
0121   441 ;                        system boot procedure, the bootstrap code
0121   442 ;                        solicits parameters and other input from the
0121   443 ;                        console terminal. If the DIAG is also on, then
0121   444 ;                        the diagnostic supervisor should enter "MENU"
0121   445 ;                        mode and prompt user for dev ces to test.
0121   446 ;
0121   447 ;                 1      RPB$V_DEBUG.
0121   448 ;                        Debug. If this flag is set, VMS maps the code
0121   449 ;                        for the XDELTA debugger into the system page
0121   450 ;                        tables of the running system.
0121   451 ;
0121   452 ;                 2      RPB$V_INIBPT.
0121   453 ;                        Initial breakpoint. If RPB$V_DEBUG is set, VMS
0121   454 ;                        executes a BPT instruction immediately after
0121   455 ;                        enabling mapping.
0121   456 ;
0121   457 ;                 5      RPB$V_BOOBPT.
0121   458 ;                        Bootstrap breakpoint. Stops the primary
0121   459 ;                        and secondary bootstraps with a breakpoint
0121   460 ;                        instruction before testing memory.
0121   461 ;
0121   462 ;                 7      RPB$V_NOTEST.
0121   463 ;                        Memory test inhibit. Sets a bit in the PFN bit
0121   464 ;                        map for each page of memory present. Does not
0121   465 ;                        test the memory.
0121   466 ;
0121   467 ;                10      RPB$V_NOPFND.
0121   468 ;                        No PFN deletion (not implemented; intended to
0121   469 ;                        tell VMB not to read a file from the boot device
0121   470 ;                        that identifies bad or reserved memory pages,
0121   471 ;                        so that VMB does not mark these pages as valid
0121   472 ;                        in the PFN bitmap).
0121   473 ;
0121   474 ;                11      RPB$V_MPM.
0121   475 ;                        Specifies that multi-port memory is to be used
0121   476 ;                        for the total exec memory requirement.  No local
0121   477 ;                        memory is to be used.  This is for tightly-coupled
```

```
                                multi-processing.  If the DIAG is also on, then
                                the diagnostic supervisor enters "AUTOTEST" mode.

                        12      RPB$V_USEMPM.
                                Specifies that multi-port memory should be used in
                                addition to local memory, as though both were one
                                single pool of pages.

                        13      RPB$V_MEMTEST
                                Specifies that a more extensive algorithm be used
                                when testing main memory for hardware uncorrectable
                                (RDS) errors.

                        14      RPB$V_FINDMEM
                                Requests use of MA780 memory if MS780 is insufficient
                                for booting.  Used for 11/782 installations.

                ;       r10 = original PC
                ;       r11 = original PSL
                ;       AP = halt code
                ;       SP = address of 64K memory block + 200 hex
                ;
                ; implicit inputs:
                ;
                ;       IPL is 31, interrupt stack.
                ;       The first instruction of this code is at SP.
                ;
                ;       All of the system's memory controllers have been
                ;       initialized to have parity error detect ON. This
                ;       means that the 64K memory block that contains this
                ;       code has correct parity. The cache is enabled and
                ;       will continue to be enabled throughout.
                ;
                ; When the secondary bootstrap code gains control memory will look like:
                ;
                ; 0     +---------------------------------------+
                ;       + RPB                                   +
                ; 200   +---------------------------------------+
                ;       + 8K of Boot Code                       +
                ;       + boot driver preamble starts at 200    +
                ; 4200  +---------------------------------------+ (PR$_SCBB value)
                ;       + 2 Pages of SCB                        +
                ; 4600  +---------------------------------------+
                ;       + 2 Pages of PFN Bit Map described by   +
                ;       + RPB fields                            +
                ; 4A00  +---------------------------------------+
                ;       + available for stack (3 Pages)         +
                ; 5000  +---------------------------------------+
                ;       + Secondary boot code image             +
                ;       .
                ;       .
                ;       .
                ;
                ; The register contents when control is passed to the secondary
                ; bootstrap are:
                ;
                ;       R11 = base address of RPB
```

```
                        0121    535 ;       AP = address of the secondary boot parameter block alla VMB
                        0121    536 ;       SP = current stack pointer
                        0121    537 ;       PR$_SCBB = SCB address
                        0121    538 ;--
                        0121    539
                        0121    540 ;
                        0121    541 ; this will be the first location in the boot ROM
                        0121    542 ;
                        0121    543
                    00000000    544         .psect  $$$$00boot,long
                        0000    545 ROM_BASE:
                 011E'  31  0000    546         brw     rom_start               ;transfer control to actual code
      83 82 81 80 06    0003    547         .byte   6,^x80,^x81,^x82,^x83   ;footprint
                        0008    548
                    00000121    549         .psect  $$$$04boot,long         ;
                        0121    550
                        0121    551 ROM_START:
                        0121    552         .default        displacement,word
```

```
                                0121   554          .sbttl  SCB initialization and XDELTA breakpoint
                                0121   555
                     26   0F    DA   0121   556          mtpr    #^xf,#pr$_mcesr         ;reset any machine checks
                                0124   557
                                0124   558 ;
                                0124   559 ; setup SCB for the duration of this execution
                                0124   560 ;
                                0124   561
                57   4400 CE    9E   0124   562          movab   k_scb_addr+1024-k_rom_code_addr(sp),r7
                                0129   563                                          ;address scb plus two pages
                  59   FF 8F    9A   0129   564          movzbl  #255,r9                         ;setloop count DIV 2
                77   06D9'CF    DE   012D   565 10$:      moval   unfielded_scb_int+1,-(r7) ;address general error routine
                       F8 59    F4   0132   566          sobgeq  r9,10$                  ;continue in loop
                       E0 AF    D4   0135   567          clrl    machine_check_continue ;init machine check continue address
                     0615'CF    9E   0138   568          movab   machine_check_detect+1,- ;init machine check vector
                       04 A7          013C   569                  scb_a_mcheck(r7)        ;
                     0735'CF    9E   013E   570          movab   write_timeout_int+1,- ;init write timeout vector
                       60 A7          0142   571                  scb_a_write_timeout(r7);
                     11   57    DA   0144   572          mtpr    r7,#pr$_scbb            ;insert scb address in PR
                                0147   573
                                0147   574 ;
                                0147   575 ; Read the system identification processor register to discover which
                                0147   576 ; kind of VAX is to be booted.
                                0147   577 ;
                                0147   578
                     58   3E    DB   0147   579          mfpr    #pr$_sid,r8             ;Read the CPU identification
                                014A   580                                          ;processor register.
             58   58   E8 8F    78   014A   581          ashl    #-pr$v_sid_type,r8,r8   ;Get CPU identification code.
                0000'CF   58    90   014F   582          movb    r8,exe$gb_cputype       ;Save processor code globally
                                0154   583                                          ;in boot driver desc table
                                0154   584
                                0154   585 ;
                                0154   586 ; If the DEBUG flag is defined (meaning that XDELTA has been linked
                                0154   587 ; with this primary bootstrap), set up 2 XDELTA handlers in the SCB --
                                0154   588 ; one for breakpoints and one for tbit traps. Then initialize the
                                0154   589 ; XDELTA breakpoint table, allocate 3 pages of stack, and, if requested,
                                0154   590 ; execute a breakpoint before proceeding with the bootstrap.
                                0154   591 ;
                                0154   592
                                0154   593          .if     df,debug
                                0154   594          movab   xdt$breakpoint+1,scb_a_breakpoint(r7) ;Set up BPT handler.
                                0154   595          movab   xdt$trace_trap+1,scb_a_trace_trap(r7) ;Set up TBIT handler.
                                0154   596          movab   ini$brk,xdt$initial_break ;Store the initial breakpoint.
                                0154   597          movl    sp,r6                   ;Save current top of stack.
                                0154   598          movab   k_secondary_boot_addr-k_rom_code_addr(sp),sp ;Address a stack,
                                0154   599                                          ;add 1 page for the SCB
                                0154   600          bbc     #rpb$v_boobpt,r5,nobrk  ;If no BPT was requested in the
                                0154   601                                          ;boot flags, just proceed.
                                0154   602
                                0154   603 ;
                                0154   604 ; Initial breakpoint.
                                0154   605 ;
                                0154   606 ; Current register status is as follows:
                                0154   607 ;
                                0154   608 ;       R0-R5   - initial input values
                                0154   609 ;       R6      - SP value at start of ROM code
                                0154   610 ;       R7      - address of the SCB
```

```
0154   611 ;       R8      - processor identification code
0154   612 ;       R9      - destroyed
0154   613 ;       R10-FP  - initial input values
0154   614 ;       SP      - address of a 3-page stack
0154   615 ;
0154   616 ; Code following the breakpoint is going to restore SP to its original
0154   617 ; value. If you want to modify SP in XDELTA, modify R6 instead.
0154   618 ;
0154   619
0154   620 ini$brk::                                ;Debugging breakpoint.
0154   621
0154   622         bpt                              ;Stop in XDELTA.
0154   623
0154   624 NOBRK:                                   ;Proceed with bootstrapping.
0154   625         movl    r6,sp                    ;restore stack pointer
0154   626         .ENDC
```

```
                                0154   628          .sbttl  rpb initialization
                                0154   629 ;
                                0154   630 ; initialize and address the RPB
                                0154   631 ;
                                0154   632
          56   FE00 CE    9E   0154   633          movab   k_rpb_addr-k_rom_code_addr(sp),r6 ;address rpb with temp reg
            1C A6    50    7D   0159   634          movq    r0,rpb$l_bootr0(r6)     ;save registers
            24 A6    52    7D   015D   635          movq    r2,rpb$l_bootr2(r6)     ;
            2C A6    54    7D   0161   636          movq    r4,rpb$l_bootr4(r6)     ;
            10 A6    5A    7D   0165   637          movq    r10,rpb$l_haltpc(r6)    ;save halt PC and PSL
                 66  56    D0   0169   638          movl    r6,rpb$l_base(r6)       ;address of RPB
            A4 AF    56    D0   016C   639          movl    r6,boo$gl_rpbbase       ;also globally
            18 A6    5C    D0   0170   640          movl    ap,rpb$l_haltcode(r6)   ;save halt code
                 04 A6    D4   0174   641          clrl    rpb$l_restart(r6)       ;init header fields
                 0C A6    D4   0177   642          clrl    rpb$l_rstrtflg(r6)      ;
            08 A6    01    CE   017A   643          mnegl   #1,rpb$l_chksum(r6)     ;
       34 A6    0000'CF    9E   017E   644          movab   boo$al_vector,rpb$l_iovec(r6) ;insert address of driver
            00   6E    00    2C   0184   645          movc5   #0,(sp),#0,-            ;init remainder of RPB
       38 A6    00D0 8F         0188   646                  #rpb$c_length-rpb$l_iovecsz,rpb$l_iovecsz(r6)
                 5B   56    D0   018D   647          movl    r6,r11                  ;set future RPB address
               00B0 CB    57    D0   0190   648          movl    r7,rpb$l_scbb(r11)      ;save scbb address in RPB
               0090 CB    28    90   0195   649          movb    #ndt$_ub0,rpb$b_confreg(r11) ;one Qbus on Micro-VAX I.
               00A1 CB    28    90   019A   650          movb    #ndt$_ub0,rpb$b_bootndt(r11) ;Pretend this is UNIBUS.
                                019F   651
                                019F   652 ;
                                019F   653 ; init the secondary bootstrap parameter block
                                019F   654 ;
                                019F   655
       5C   FF2D CF    9E   019F   656          movab   second_param,ap         ;load its base address
         0C AC    01    CE   01A4   657          mnegl   #1,vmb$l_lo_pfn(ap)     ;set pfn data
         10 AC    01    CE   01A8   658          mnegl   #1,vmb$l_hi_pfn(ap)     ;set pfn data
                                01AC   659
                                01AC   660 ;
                                01AC   661 ; address larger stack and setup free memory pointer
                                01AC   662 ;
                                01AC   663
       5E   5000 CB    9E   01AC   664 15$:     movab   k_secondary_boot_addr(r11),sp; address target for I/O
                                01B1   665                                      ;and create a three page stack
                 5A   5E    D0   01B1   666          movl    sp,r10                  ;copy to address of free memory
```

```
                                01B4   668          .sbttl  memory initialization
                                01B4   669 ;
                                01B4   670 ; initialize parity memory
                                01B4   671 ;
                                01B4   672 ; allocate and init RPB PFN bit map
                                01B4   673 ;
                                01B4   674
          44 AB    01    0A    78  01B4   675          ashl    #10,#1,rpb$q_pfnmap(r11) ;set size to 1024 bytes
          48 AB    4600 CB     9E  01B9   676          movab   k_pfn_map_addr(r11),rpb$q_pfnmap+4(r11) ;set up pfn desc addr
4600 CB   44 AB    00    6E    00  2C  01BF   677          movc5   #0,(sp),#0,rpb$q_pfnmap(r11),k_pfn_map_addr(r11) ;init to zeros
            FF4A CF    E4'AF     9E  01C8   678          movab   b^5$,machine_check_continue ;setup continue address
          59    00001FFF 8F     D0  01CE   679          movl    #k_max_memory_pages-1,r9 ;set page number of last memory page
             07 20 AB    06     E0  01D5   680          bbs     #switch_v_qvss,rpb$l_bootr1(r11),3$;if set console not QVSS
          59    00000200 8F     C2  01DA   681          subl    #512,r9                 ;else - last 256K used for QVSS video
                                01E1   682 3$:       setipl  #^x1d-1                 ;lower IPL to allow write timeout
                                01E4   683
                                01E4   684 ;
                                01E4   685 ; sweep all of memory to set parity and establish bit map
                                01E4   686 ;
                                01E4   687
        FF2D CF    029F'CF     9E  01E4   688 5$:       movab   nxm_memory,machine_check_continue ;enable machine check
                           56   7C  01EB   689          clrq    r6                      ;set first physical address
                                01ED   690                                          ;zero page count
             58    00BC CB     9E  01ED   691          movab   rpb$l_memdsc(r11),r8    ;address first memory descriptor
                           68   D4  01F2   692          clrl    (r8)                    ;init page count field
               04 A8    01    CE  01F4   693          mnegl   #1,4(r8)                ;set very low PFN
                                01F8   694
                                01F8   695 ;
                                01F8   696 ; write 0's to a page
                                01F8   697 ;
                                01F8   698
                                01F8   699 page_boundary:
                                01F8   700
          56    57    09    78  01F8   701          ashl    #9,r7,r6                ;compute page address
                                01FC   702
                                01FC   703 ;
                                01FC   704 ; don't write 0's in the 64k where this code is
                                01FC   705 ;
                                01FC   706
             56    5B    D1    01FC   707          cmpl    r11,r6                  ;compare addresses to find out
                         42    12  01FF   708          bneq    20$                     ;br if not in the 64KB of good memory
               0C AC     D5    0201   709          tstl    vmb$l_lo_pfn(ap)        ;low pfn inited?
                         04    18  0204   710          bgeq    10$                     ;br if yes
             0C AC     57    D0  0206   711          movl    r7,vmb$l_lo_pfn(ap)     ;insert base of area as lowest PFN
                04 A8    D5    020A   712 10$:       tstl    4(r8)                   ;memory desc PFN set yet?
                         04    18  020D   713          bgeq    15$                     ;br if yes
             04 A8     57    D0  020F   714          movl    r7,4(r8)                ;insert this low PFN
                50    01    CE  0213   715 15$:       mnegl   #1,r0                   ;set all bits in a register
4600 CB    20    57    50    F0  0216   716          insv    r0,r7,#32,k_pfn_map_addr(r11) ;enable 128 pages
4604 CB    20    57    50    F0  021D   717          insv    r0,r7,#32,k_pfn_map_addr+4(r11) ;of good memory
4608 CB    20    57    50    F0  0224   718          insv    r0,r7,#32,k_pfn_map_addr+8(r11) ;
460C CB    20    57    50    F0  022B   719          insv    r0,r7,#32,k_pfn_map_addr+12(r11) ;
          50    7F 8F     9A  0232   720          movzbl  #127,r0                 ;load page count in r0
            57    6740     9E  0236   721          movab   (r7)[r0],r7             ;adjust PFN to last page tested
             4C AB    50    C0  023A   722          addl    r0,rpb$l_pfncnt(r11)    ;adjust good page count < 8000
                68    50    C0  023E   723          addl    r0,(r8)                 ;adjust desc size
                         3D    11  0241   724          brb     40$                     ;continue in common
```

```
                                0243    725
                                0243    726 ;
                                0243    727 ; write and then read a single memory page
                                0243    728 ;
                                0243    729
                                0243    730 ;
                                0243    731 ; loop to check for correct parity and verify contents
                                0243    732 ;
                                0243    733
                          66    7C    0243    734 20$:    clrq    (r6)                    ;write memory to zero
                    50    02    D0    0245    735         movl    #k_bus.timeout,r0       ;initialize for error exit
                    51    86    7D    0248    736         movq    (r6)+,r1                ;read memory back for parity detect
                                      024B    737                                         ;first read checks for page present
                                      024B    738                                         ;skip out if not present to nxm_memory
                                      024B    739                                         ;skip out on parity error
                          52    12    024B    740         bneq    nxm_memory              ;if eql then correct read back
                                      024D    741                                         ;if neq then odd case of PROM
                                      024D    742
                                      024D    743 ;
                                      024D    744 ; check for page cross
                                      024D    745 ;
                                      024D    746
                 56   01FF 8F    B3    024D    747         bitw    #^x1ff,r6               ;page cross?
                           EF    12    0252    748         bneq    20$                     ;br if no, keep going
                                0254    749
                                0254    750 ;
                                0254    751 ; try to write a non-zero value and verify that it can be done
                                0254    752 ;
                                0254    753 ; This is to detect pages in ROM's that are all zeros
                                0254    754 ;
                                0254    755
          0FFFFFFC E6    3F    B0    0254    756         movw    #63,<1a28>-4(r6)        ;write memory, page is present
                                      025B    757                                         ;with don't cache bit
          0FFFFFFC E6    3F    B1    025B    758         cmpw    #63,<1a28>-4(r6)        ;read back correct?
                         3B    12    0262    759         bneq    nxm_memory              ;if neq then some non RAM memory
                      FC A6    B4    0264    760         clrw    -4(r6)                  ;reset to zero
                                0267    761
                                0267    762 ;
                                0267    763 ; page is written - parity appears correct
                                0267    764 ;
                                0267    765
    4600 CB    01    57    01    F0    0267    766 30$:    insv    #1,r7,#1,k_pfn_map_addr(r11) ;insert bit in PFN map
                        0C AC    D5    026E    767         tstl    vmb$l_lo_pfn(ap)        ;low pfn inited?
                                04    18    0271    768         bgeq    35$             ;br if yes
                     0C AC      57    D0    0273    769         movl    r7,vmb$l_lo_pfn(ap) ;insert lowest PFN
                        04 A8    D5    0277    770 35$:    tstl    4(r8)                   ;memory desc PFN set yet?
                                04    18    027A    771         bgeq    40$             ;br if yes
                     04 A8      57    D0    027C    772         movl    r7,4(r8)            ;insert this low PFN
                     10 AC      57    D0    0280    773 40$:    movl    r7,vmb$l_hi_pfn(ap) ;insert highest
                                68    D6    0284    774         incl    (r8)                ;count in current memory desc
                           4C AB    D6    0286    775         incl    rpb$l_pfncnt(r11)   ;count as good page
                                0289    776
                                0289    777 ;
                                0289    778 ; come here to move to next page
                                0289    779 ;
                                0289    780
                                0289    781 next_page:
```

```
                     0289   782
          FF69 57 01 59 F1 0289   783          acbl    r9,#1,r7,page_boundary  ; continue until end of memory
                     028F   784
                     028F   785 ;
                     028F   786 ; restore IPL and setup SCB for booting
                     028F   787 ;
                     028F   788
                     028F   789          setipl  #ipl$_power             ;reset IPL
                FE82 CF D4  0292   790          clrl    machine_check_continue  ;reset machine check continue addr
       23 00000F0D 8F DA    0296   791          mtpr    #Led_memory_ok,#pr$_txdb; set lights
                   39 11    0299   792          brb     begin_boot
                     029B   793
                     029B   794 ;
                     029B   795 ; come here when a page does not exist
                     029B   796 ;
                     029B   797
                     029B   798 nxm_memory:
                     029B   799
                   50 01 D1 029B   800          cmpl    #k_parity.error,r0      ;expected error?
                      12 12 029E   801          bneq    20$                     ;parity is ok
                     02A0   802
                     02A0   803 ;
                     02A0   804 ; reset the memory controllers to clear parity error
                     02A0   805 ;
                     02A0   806
                   51 0F D0 02A0   807          movl    #15,r1                  ;set loop count of controllers
       52 20001440 8F D0    02A3   808          movl    #msv11_csr_base,r2      ;address base of controlller CSR's
                82 01 B0    02AA   809 10$:     movw    #msv11_csr_parity_enable,(r2)+;blast all possible CSR's
                  FA 51 F4  02AD   810          sobgeq  r1,10$                  ;continue until done
                     D3 11  02B0   811          brb     next_page               ;no check for PROM needed on parity
                     02B2   812                                                 ;problem
                     02B2   813 ;
                     02B2   814 ; bus timeout means non existant memory on a read, the page is not present
                     02B2   815 ;
                     02B2   816 ; But, page could be PROM memory
                     02B2   817 ;
                     02B2   818
             50 02 D1    02B2   819 20$:     cmpl    #k_bus.timeout,r0       ;expected error?
                15 12    02B5   820          bneq    fatal_memory_error      ;br if unexpected
                     02B7   821
                     02B7   822 ;
                     02B7   823 ; record holes in memory via descriptors
                     02B7   824 ;
                     02B7   825
                68 D5    02B7   826          tstl    (r8)                    ;this desc in use?
                CA 13    02B9   827          beql    next_page               ;br if no, don't move to next
       50 00FC CB 9E     02BB   828          movab   <rpb$c_nmemdsc*rpb$c_memdscsiz>+rpb$l_memdsc(r11),r0
             58 50 D1    02C0   829          cmpl    r0,r8                   ;overrun area?
                C0 12    02C3   830          bneq    next_page               ;br if yes
             58 08 C0    02C5   831          addl    #rpb$c_memdscsiz,r8     ;address next memory desc
                68 D4    02C8   832          clrl    (r8)                    ;set count to zero
                B9 11    02CA   833          brb     next_page               ;
                     02CC   834
                     02CC   835 ;
                     02CC   836 ; memory initialization error
                     02CC   837 ;
                     02CC   838
```

```
02D0    839 fatal_memory_error:
02D0    840
02D0    841         fatal_message   memerr
```

```
                     02D8   843 ;++
                     02D8   844 ; begin_boot - start the booting process
                     02D8   845 ;
                     02D8   846 ; functional description:
                     02D8   847 ;
                     02D8   848 ; This sequence is entered after the RPB and PFN bitmap are set up.
                     02D8   849 ;
                     02D8   850 ; The process of selecting a boot device and type of boot operation starts
                     02D8   851 ; here.
                     02D8   852 ;
                     02D8   853 ; inputs:
                     02D8   854 ;
                     02D8   855 ;       r11 = address of the RPB
                     02D8   856 ;       ap = address of the secondary parameter block
                     02D8   857 ;
                     02D8   858 ;--
                     02D8   859
                     02D8   860 begin_boot:
                     02D8   861
                     02D8   862 ;
                     02D8   863 ; If the "solicit for secondary bootstrap file" flag is not set,
                     02D8   864 ; just use a predefined file specification.
                     02D8   865 ;
                     02D8   866
  03   E0             02D8   867         bbs     #rpb$v_bblock,-         ;br if not files-11 boot
30 AB                02DA   868                 rpb$l_bootr5(r11),-     ;
  43                 02DC   869                 25$                     ;
  08   E1            02DD   870         bbc     #rpb$v_solict,-         ;If "solicit" flag is not
30 AB                02DF   871                 rpb$l_bootr5(r11),-     ;set, just use a default file
  13                 02E1   872                 10$                     ;specification.
                     02E2   873
                     02E2   874 ;
                     02E2   875 ; To solicit a file name, call a device-independent subroutine that
                     02E2   876 ; writes a prompt string to the console terminal, and then reads the
                     02E2   877 ; user typed file name. All device specifications are ignored.
                     02E2   878 ;
                     02E2   879
    20 AB    DD      02E2   880         pushl   rpb$l_bootr1(r11)       ;Pass options switch settings
    68 AB    9F      02E5   881         pushab  rpb$t_file(r11)         ;Set address of input buffer.
    27       DD      02E8   882         pushl   #39                     ;Set maximum character count.
  FD47 CF    9F      02EA   883         pushab  nameprompt              ;Set address of prompt string.
0000'CF      04      02EE   884         calls   #4,boo$readprompt       ;Prompt and read string.
    2B       11      02F3   885         brb     25$                     ;Go try to read the file.
                     02F5   886
                     02F5   887 ;
                     02F5   888 ; If the solicit boot flag was not set, use a default file name string.
                     02F5   889 ; Usually, this file name is [SYSEXE]SYSBOOT.EXE. However, if the
                     02F5   890 ; diagnostic boot flag is set, the file name is [SYSMAINT]DIAGBOOT.EXE.
                     02F5   891 ;
                     02F5   892
57   FD07 CF    9E      02F5   893 10$:     movab   vmsfile,r7              ;Assume SYSBOOT.EXE.
    04       E1      02FA   894         bbc     #rpb$v_diag,-           ;If the diagnostic flag is not
    30 AB            02FC   895                 rpb$l_bootr5(r11),-     ;set, SYSBOOT is correct.
    05               02FE   896                 15$
57   FD16 CF    9E      02FF   897         movab   diagfile,r7             ;Otherwise, use predefined
                     0304   898                                         ;name of diagnostic boot.
                     0304   899
```

```
                                0304   900 ;
                                0304   901 ; Copy the file name to the RPB.
                                0304   902 ;
                                0304   903
                    50   67  9A 0304   904 15$:    movzbl  (r7),r0                 ;Size of name string
                         50  D6 0307   905        incl    r0                      ;Include the byte count character
                68 AB    67  50  28 0309   906        movc3   r0,(r7),rpb$t_file(r11) ;Move name into RPB
                         04  1C  EF 030E   907        extzv   #rpb$v_topsys,#rpb$s_topsys,-
                      50   30 AB      0311   908                rpb$l_bootr5(r11),r0    ;Value of 0-F means top level
                                0314   909                                        ;system directory "SYS0" - "SYSF"
                         09  50  D1 0314   910        cmpl    r0,#9                   ;0 - 9 ?
                             03  15 0317   911        bleq    20$                     ;Branch if yes
                         50  07  C0 0319   912        addl    #<<^A/A/>-<^A/9/>-1>,r0 ;Add bias to make A - F
                      6D AB  50  80 031C   913 20$:    addb    r0,rpb$t_file+5(r11)    ;Form "SYSn"
                                0320   914
                                0320   915 ;
                                0320   916 ; extract and stabilize device name info
                                0320   917 ;
                                0320   918
                    57  FD1D  CF  9E 0320   919 25$:    movab   boot_device_list,r7     ;address descriptor list
                7E  1C AB  80A0A0A0  8F  CB 0325   920        bicl3   #^x80A0A0A0,rpb$l_bootr0(r11),-(sp) ;make name uppercase
                                032E   921                                        ;remove possible parity bit
                         20  6E  91 032E   922        cmpb    (sp),#^a/ /             ;special non-name?
                             0E  14 0331   923        bgtr    35$                     ;br if gtr then specific device
                                0333   924
                                0333   925 ;
                                0333   926 ; non-specific device name
                                0333   927 ;
                                0333   928
                             6E  D4 0333   929        clrl    (sp)                    ;specify non name
                             07  E0 0335   930        bbs     #switch_v_disk_boot,-   ;br if entire list is to be searched
                          10 20 AB      0337   931                rpb$l_bootr1(r11),40$   ;
                    57   FD11  CF  9E 033A   932        movab   no_disk_boot_device_list,r7;address alternate descriptor list
                             09  11 033F   933        brb     40$                     ;continue
                                0341   934
                                0341   935 ;
                                0341   936 ; specific device name
                                0341   937 ;
                                0341   938
                    FDD6 CF     6E  D0 0341   939 35$:    movl    (sp),boot_device_name   ;save specified name
                      03 AE     30  82 0346   940        subb    #^a/0/,3(sp)            ;reduce unit number
                                034A   941
                                034A   942 ;
                                034A   943 ; start with first entry in boot device list and try each one until a
                                034A   944 ; boot occurs or the list is empty
                                034A   945 ;
                                034A   946
                         67 AB  94  034A   947 40$:    clrb    rpb$b_slave(r11)        ;no slave or
                         64 AB  B4  034D   948        clrw    rpb$w_unit(r11)         ;unit info
                    5A   5000   CB  9E 0350   949        movab   k_secondary_boot_addr(r11),r10 ;set nominal load address
                66 AB    05  A7  90 0355   950        movb    bd_b_type(r7),rpb$b_devtyp(r11) ;load device type
                             6E  95 035A   951        tstb    (sp)                    ;special non-name?
                             18  13 035C   952        beql    45$                     ;br if yes, no specific device
            67   6E   18   00  ED 035E   953        cmpzv   #0,#24,(sp),bd_l_name(r7); compare three characters for equal
                             2C  12 0363   954        bneq    50$                     ;br if no match
                04 A7    03 AE  91 0365   955        cmpb    3(sp),bd_b_high_unit(r7) ;unit in range?
                             25  1A 036A   956        bgtru   50$                     ;br if no
```

```
                04 A7    03 AE    90  036C    957          movb    3(sp),bd_b_high_unit(r7);boot specific unit only
                64 AB    03 AE    9B  0371    958          movzbw  3(sp),rpb$w_unit(r11)   ;
                 FDA1 CF    67    D0  0376    959 45$:     movl    bd_l_name(r7),boot_device_name;build error device name
          FD9D CF    30    64 AB    81  037B    960          addb3   rpb$w_unit(r11),#^a/0/,boot_device_name+3;
                      0A B747      16  0382    961          jsb     @bd_a_routine(r7)[r7]   ;use device specific routine
                        2B 50      E8  0386    962          blbs    r0,100$                 ;br if success, transfer control
                     24 50    01   E1  0389    963          bbc     #1,r0,55$               ;severe or fatal error?
                                       038D    964                                      ;br if fatal error
                           6E    95  038D    965          tstb    (sp)                    ;special non-name?
                           20    12  038F    966          bneq    55$                     ;br if specific device
                        57    0E   C0  0391    967 50$:     addl2   #bd_s_bd,r7             ;try next device
              23   00000F0D 8F    DA  0394    968          mtpr    #led_memory_ok,#pr$_txdb; tell operator
                           67    B5  039B    969          tstw    bd_l_name(r7)           ;is there another?
                           AB    12  039D    970          bneq    40$                     ;continue if not end of list
                                       039F    971
                                       039F    972 ;
                                       039F    973 ; device data base search done without a match or valid boot device
                                       039F    974 ;
                                       039F    975
                50    0848 8F    3C  039F    976          movzwl  #ss$_devassign,r0       ;list end, specific name error
                           6E    95  03A4    977          tstb    (sp)                    ;special non-name?
                           09    12  03A6    978          bneq    55$                     ;br if specific device
                50    0908 8F    3C  03A8    979          movzwl  #ss$_nosuchdev,r0       ;list end, generic error
                     FD6B CF    D4  03AD    980          clrl    boot_device_name        ;no specific name
                                       03B1    981 55$:     fatal_message                   ;issue error in r0
                                       03B4    982
                                       03B4    983 ;
                                       03B4    984 ; secondary image in place, transfer control to it
                                       03B4    985 ;
                                       03B4    986
                5E    5000 CB    9E  03B4    987 100$:    movab   k_secondary_boot_addr(r11),sp ;load fresh sp
                                       03B9    988
                                       03B9    989 ;
                                       03B9    990 ; restore SCBB values
                                       03B9    991 ;
                                       03B9    992
              23   00000F0F 8F    DA  03B9    993          mtpr    #led_transfer_control,#pr$_txdb; tell user
                57    4600 CB    9E  03C0    994          movab   k_scb_addr+1024(r11),r7 ;address scb plus two pages
                  59    FF 8F    9A  03C5    995          movzbl  #255,r9                 ;setloop count DIV 2
                77    06E1'CF    DE  03C9    996 110$:    moval   secondary_scb_int+1,-(r7) ;address general error routine
                        F8 59    F4  03CE    997          sobgeq  r9,110$                 ;continue in loop
                                       03D1    998
                                       03D1    999 ;
                                       03D1   1000 ; recompute size of bitmap
                                       03D1   1001 ;
                                       03D1   1002
          50    10 AC    FD 8F    78  03D1   1003          ashl    #-3,vmb$l_hi_pfn(ap),r0 ;get last valid PFN
             44 AB    50    01    C1  03D7   1004          addl3   #1,r0,rpb$q_pfnmap(r11) ;set size of map in bytes
              14 AC    44 AB    7D  03DC   1005          movq    rpb$q_pfnmap(r11),vmb$q_pfnmap(ap);copy pfnmap desc
                                       03E1   1006
                                       03E1   1007 ;
                                       03E1   1008 ; halt system prior to entering secondary boot if requested
                                       03E1   1009 ;
                                       03E1   1010
                           09    E1  03E1   1011          bbc     #rpb$v_halt,-           ;If boot flags don't call for
                        30 AB          03E3   1012                   rpb$l_bootr5(r11),-     ;halt, just transfer to new
                           07          03E5   1013                   120$                    ;bootstrap image.
```

```
                23   00000F05 8F    DA  03E6  1014          mtpr    #console_halt,#pr$_txdb ;Otherwise, HALT.
                                65    17  03ED  1015 120$:   jmp     (r5)                    ;Execute JUMP.
```

```
                              03EF        .sbttl  specific device boot subroutine
                              03EF   ;++
                              03EF   ; prom_boot
                              03EF   ;
                              03EF   ; functional description:
                              03EF   ;
                              03EF   ; This routine tries to boot from a PROM system i age that may be in
                              03EF   ; memory. Each bad page 16KB boundary is tested t  see if it is readable.
                              03EF   ;
                              03EF   ; inputs:
                              03EF   ;
                              03EF   ;       r7 = address of the internal boot device description
                              03EF   ;       r10 = address of the secondary boot's memory
                              03EF   ;       r11 = RPB address
                              03EF   ;       ap = address of the secondary parameter block
                              03EF   ;
                              03EF   ; outputs:
                              03EF   ;
                              03EF   ;       r0 = ss$_norom - no rom present, severe error
                              03EF   ;
                              03EF   ;       or
                              03EF   ;
                              03EF   ;       r0 = 1 if success
                              03EF   ;       r5 = transfer address
                              03EF   ;
                              03EF   ;       r7,r11 are preserved
                              03EF   ;--
                              03EF
                              03EF   prom_boot:
                              03EF
                              03EF   ;
                              03EF   ; cycle up through memory
                              03EF   ;
                              03EF
FD23 CF    15'AF      9E      03EF        movab   b^20$,machine_check_continue;implant for read timeout
              53      D4      03F5        clrl    r3                      ;initial page address
18 4600 CB    53      E0      03F7   10$: bbs     r3,k_pfn_map_addr(r11),20$ ;br if that boundary is not bad
   51    53   09      78      03FD        ashl    #9,r3,r1                ;compute address
         18   61      B1      0401        cmpw    (r1),#^x18              ;try to read that memory
                              0404                                        ;may machine check
              0F      12      0404        bneq    20$                     ;br if not key
            02DF      30      0406        bsbw    verify_boot_block       ;verify the boot block
           09 50      E9      0409        blbc    r0,20$                  ;br if not correct
                              040C
                              040C   ;
                              040C   ; PROM found, boot from it
                              040C   ;
                              040C
55    10 A1   51      C1      040C        addl3   r1,16(r1),r5            ;compute starting address
                              0411
                              0411   ;
                              0411   ; reset machine state
                              0411   ;
                              0411
           34 AB      7C      0411        clrq    rpb$l_iovec(r11)        ;no driver
                      05      0414        rsb                             ;done
                              0415
```

```
                                0415  1074 ;
                                0415  1075 ; move onto next 16KB boundary
                                0415  1076 ;
                                0415  1077
    FFD8 53    20    00001FFF 8F    F1  0415  1078 20$:     acbl    #k_max_memory_pages-1,#32,r3,10$; continue until done
                        50    801A 8F    3C  041F  1079          movzwl  #ss$_norom!2,r0         ;set severe error code
                              FCF0 CF    D4  0424  1080          clrl    machine_check_continue  ;
                                         05  0428  1081          rsb
```

```
                     0429  1083 ;++
                     0429  1084 ; network_boot
                     0429  1085 ;
                     0429  1086 ; functional description:
                     0429  1087 ;
                     0429  1088 ; This routine tries to boot from a network device
                     0429  1089 ;
                     0429  1090 ; inputs:
                     0429  1091 ;
                     0429  1092 ;       r7 = address of the internal boot device description
                     0429  1093 ;       r10 = address of the secondary boot's memory
                     0429  1094 ;       r11 = RPB address
                     0429  1095 ;       ap = address of the secondary parameter block
                     0429  1096 ;
                     0429  1097 ; outputs:
                     0429  1098 ;
                     0429  1099 ;       r0 = 1 if success
                     0429  1100 ;       r5 = transfer address
                     0429  1101 ;
                     0429  1102 ;       or
                     0429  1103 ;
                     0429  1104 ;       r0 = ss$_nosuchdev      - CSR does not exist - severe
                     0429  1105 ;          = ss$_bufferovf      - secondary bootstrap does not fit - fatal
                     0429  1106 ;          = ss$_devinact       - device could not be inited - fatal
                     0429  1107 ;          = ss$_ctrlerr        - I/O error during operation - fatal
                     0429  1108 ;          = ss$_devoffline     - device is offline - severe
                     0429  1109 ;
                     0429  1110 ;       r7,r11 are preserved
                     0429  1111 ;--
                     0429  1112
                     0429  1113 network_boot:
                     0429  1114
0281    30           0429  1115         bsbw    validate_csr            ;test CSR of device
                     042C  1116                                         ;return implies success
                     042C  1117 ;
                     042C  1118 ; boot via the Ethernet
                     042C  1119 ;
                     042C  1120
     52    7E    DE  042C  1121         moval   -(sp),r2                ;address target for transfer address
       68  AB    9F  042F  1122         pushab  rpb$t_file(r11)         ;address to store node name
       28  AB    9F  0432  1123         pushab  rpb$l_bootr3(r11)       ;address to store node address
           62    9F  0435  1124         pushab  (r2)                    ;address to store transfer address
     4600  CB    9F  0437  1125         pushab  k_pfn_map_addr(r11)     ;address of bit map
     5000  CB    9F  043B  1126         pushab  k_secondary_boot_addr(r11) ;buffer space
53   7000  CB    9E  043F  1127         movab   k_secondary_boot_addr+<16*512>(r11),r3 ;image load address
           63    9F  0444  1128         pushab  (r3)                    ;image load address
0000'CF    06    FB  0446  1129         calls   #6,boo$downline_load    ;try QNA boot
55   53    8E    C1  044B  1130         addl3   (sp)+,r3,r5             ;compute transfer address
       03  50    E9  044F  1131         blbc    r0,10$                  ;br if not success
       34  AB    7C  0452  1132         clrq    rpb$l_iovec(r11)        ;no driver
                 05  0455  1133 10$:    rsb                             ;done
```

```
                     0456  1135 ;++
                     0456  1136 ; disk_boot
                     0456  1137 ;
                     0456  1138 ; functional description:
                     0456  1139 ;
                     0456  1140 ; This routine tries to boot from a disk device
                     0456  1141 ;
                     0456  1142 ; inputs:
                     0456  1143 ;
                     0456  1144 ;       r7 = address of the internal boot device description
                     0456  1145 ;       r10 = address of the secondary boot's memory
                     0456  1146 ;       r11 = RPB address
                     0456  1147 ;       ap = address of the secondary parameter block
                     0456  1148 ;
                     0456  1149 ; outputs:
                     0456  1150 ;
                     0456  1151 ;       r0 = 1 if success
                     0456  1152 ;       r5 = transfer address
                     0456  1153 ;
                     0456  1154 ;       or
                     0456  1155 ;
                     0456  1156 ;       r0 = ss$_nosuchdev      - CSR does not exist - severe
                     0456  1157 ;          = ss$_nosuchfile     - file is not on the volume - fatal
                     0456  1158 ;          = ss$_filnotcntg     - boot file is not contiguous - fatal
                     0456  1159 ;          = ss$_bufferovf      - secondary bootstrap does not fit - fatal
                     0456  1160 ;          = ss$_devinact       - device could not be inited - fatal
                     0456  1161 ;          = ss$_ctrlerr        - I/O error during operation - fatal
                     0456  1162 ;          = ss$_devoffline     - device is offline - severe
                     0456  1163 ;
                     0456  1164 ;       r7,r11 are preserved
                     0456  1165 ;--
                     0456  1166
                     0456  1167 disk_boot:
                     0456  1168
          0254   30  0456  1169         bsbw    validate_csr            ;check CSR and return if success
                     0459  1170
                     0459  1171 ;
                     0459  1172 ; move and initialize the disk driver
                     0459  1173 ;
                     0459  1174
    52   34 AB   D0  0459  1175         movl    rpb$l_iovec(r11),r2     ;address boot driver
    59   6B   9E  045D  1176         movab   k_rpb_addr(r11),r9      ;load move parameter
    18 B242   16  0460  1177         jsb     @bqo$l_move(r2)[r2]     ;call move code
                     0464  1178
                     0464  1179 ;
                     0464  1180 ; try low to high units, removable first, non-removable second
                     0464  1181 ;
                     0464  1182 ; Build a mask with two sets of 8 bits. The first 8 bits are the available
                     0464  1183 ; "soft" disk units and the second 8 are the available "hard". The mask
                     0464  1184 ; starts with rpb$w_unit to bd_b_high_unit set in each set.
                     0464  1185 ;
                     0464  1186
          58   D4  0464  1187         clrl    r8                      ;no units to search
    56   64 AB   3C  0466  1188         movzwl  rpb$w_unit(r11),r6      ;build the basic mask
    51   04 A7   9A  046A  1189         movzbl  bd_b_high_unit(r7),r1   ;get high
58   51   56   C3  046E  1190         subl3   r6,r1,r8                ;number of units
          58   D6  0472  1191         incl    r8                      ;plus 1
```

```
                58   01    58   78  0474  1192          ashl    r8,#1,r8                ;form mask
                           58   D7  0478  1193          decl    r8                      ;
                58   58    56   78  047A  1194          ashl    r6,r8,r8                ;move to correct bit pos
             58 08   08    58   F0  047E  1195          insv    r8,#8,#8,r8             ;duplicate mask
                     51    56   91  0483  1196          cmpb    r6,r1                   ;high = low - one unit?
                           02   12  0486  1197          bneq    10$                     ;br if yes, enter search
                           58   94  0488  1198          clrb    r8                      ;no soft disk search
                                    048A  1199
                                    048A  1200 ;
                                    048A  1201 ; select a unit from the mask
                                    048A  1202 ;
                                    048A  1203
          56   58    10   00   EA  048A  1204 10$:      ffs     #0,#16,r8,r6            ;get the unit number
            64 AB    56   08   AB  048F  1205          bicw3   #^x8,r6,rpb$w_unit(r11) ;set unit number, less mask flag
       FC84 CF  64 AB    30   81  0494  1206          addb3   #^a/0/,rpb$w_unit(r11),boot_device_name+3; new unit in name
                                    049B  1207
                                    049B  1208 ;
                                    049B  1209 ; now, init that unit on the controller
                                    049B  1210 ;
                                    049B  1211
                52   34 AB    D0  049B  1212          movl    rpb$l_iovec(r11),r2     ;address boot driver
                51   1C A2    D0  049F  1213          movl    bgo$l_unit_init(r2),r1  ;Pick up device init routine
                           20   13  04A5  1214          beql    30$                     ;None
                                    04A5  1215
                                    04A5  1216 ;
                                    04A5  1217 ; init the controller and a specific unit
                                    04A5  1218 ;
                                    04A5  1219 ; it is OK for the unit to be offline but not for the controller to fail
                                    04A5  1220 ;
                                    04A5  1221
             6241   6C    FA  04A5  1222          callg   (ap),(r2)[r1]           ;do any necessary unit init
                 09 50    E8  04A9  1223          blbs    r0,20$                  ;br if unit is online
                                    04AC  1224
                                    04AC  1225 ;
                                    04AC  1226 ; If the unit is not online, it is a fatal error if the controller failed.
                                    04AC  1227 ;
                                    04AC  1228
       0084 8F    50   B1  04AC  1229          cmpw    r0,#ss$_devoffline      ;offline?
                     36   12  04B1  1230          bneq    50$                     ;br if no, more fatal error
                     21   11  04B3  1231          brb     35$                     ;continue with next unit
                                    04B5  1232
                                    04B5  1233 ;
                                    04B5  1234 ; controller is up, unit is online, make removable, non-removable tests
                                    04B5  1235 ;
                                    04B5  1236 ; success from the online is:
                                    04B5  1237 ;
                                    04B5  1238 ;       #1 unit is online, can't detect hard or soft
                                    04B5  1239 ;       #9 unit is online, hard disk
                                    04B5  1240 ;       #25 unit is online, soft disk
                                    04B5  1241 ;
                                    04B5  1242
          0C 56    03   E0  04B5  1243 20$:      bbs     #3,r6,30$               ;br if hard disk mask, try unit
                                    04B9  1244
                                    04B9  1245 ;
                                    04B9  1246 ; looking for a soft disk - can the controller can tell?
                                    04B9  1247 ;
                                    04B9  1248
```

```
                06 50    03    E1   04B9  1249          bbc     #3,r0,25$               ;   if not detectable soft or hard
                                    04BD  1250
                                    04BD  1251 ;
                                    04BD  1252 ; looking for a soft disk and the controller can tell
                                    04BD  1253 ;
                                    04BD  1254
                04 50    04    E0   04BD  1255          bbs     #4,r0,30$               ;br if soft disk flag set, try unit
                         1B    11   04C1  1256          brb     40$                     ;continue in common
                                    04C3  1257
                                    04C3  1258 ;
                                    04C3  1259 ; since the controller can't tell, shut off tests in soft mask
                                    04C3  1260 ; but do this unit anyway
                                    04C3  1261 ;
                                    04C3  1262
                         58    94   04C3  1263 25$:    clrb    r8                      ;no more soft disk tests
                                    04C5  1264
                                    04C5  1265 ;
                                    04C5  1266 ; try a boot of this unit
                                    04C5  1267 ;
                                    04C5  1268
                   03C0 8F    BB   04C5  1269 30$:    pushr   #^m<r6,r7,r8,r9>        ;save context values
                        1F    10   04C9  1270          bsbb    boot_disk_unit          ;try this unit
                   03C0 8F    BA   04CB  1271          popr    #^m<r6,r7,r8,r9>        ;restore context values
                     17 50    E8   04CF  1272          blbs    r0,50$                  ;br if success
                13 50    01    E1   04D2  1273          bbc     #1,r0,50$               ;br if fatal error
                                    04D6  1274                                          ;continue if just severe error
              51    56    08    C1   04D6  1275 35$:    addl3   #8,r6,r1                ;clear bit in both masks
                00 58    51    E5   04DA  1276          bbcc    r1,r8,40$               ;hard mask or greater
                00 58    56    E5   04DE  1277 40$:    bbcc    r6,r8,45$               ;and soft or hard
                         58    B5   04E2  1278 45$:    tstw    r8                      ;more units?
                         A4    12   04E4  1279          bneq    10$                     ;br if yes
                                    04E6  1280                                          ;exit with error status of last unit
             50    02    88   04E6  1281          bisb    #1@1,r0                 ;make error semi-success
                                    04E9  1282
                                    04E9  1283 ;
                                    04E9  1284 ; fixup name with real booted device and transfer control
                                    04E9  1285 ;
                                    04E9  1286
                         05   04E9  1287 50$:    rsb
```

```
                        04EA  1289          .sbttl  boot a specific disk unit routine
                        04EA  1290 ;++
                        04EA  1291 ; boot_disk_unit
                        04EA  1292 ;
                        04EA  1293 ; functional description:
                        04EA  1294 ;
                        04EA  1295 ; This routine tries a boot of a particular disk unit. The device and
                        04EA  1296 ; driver are present and verified. This routine is used for each unit on
                        04EA  1297 ; which a boot is to be tried. RPB$B_UNIT contains the unit information.
                        04EA  1298 ;
                        04EA  1299 ; inputs:
                        04EA  1300 ;
                        04EA  1301 ;       r9 = rpb address
                        04EA  1302 ;       r10 = address  of the secondary boot's memory
                        04EA  1303 ;       r11 = RPB address
                        04EA  1304 ;       ap = address of the secondary parameter block
                        04EA  1305 ;
                        04EA  1306 ; outputs:
                        04EA  1307 ;
                        04EA  1308 ;       r0 = ss$_success
                        04EA  1309 ;       r5 = transfer address
                        04EA  1310 ;
                        04EA  1311 ;       or
                        04EA  1312 ;
                        04EA  1313 ;       r0 = ss$_nosuchdev      - CSR does not exits
                        04EA  1314 ;          = ss$_nosuchfile     - file is not on the volume
                        04EA  1315 ;          = ss$_filnotcntg     - boot file is not contiguous
                        04EA  1316 ;          = ss$_bufferovf      - secondary bootstrap does not fit
                        04EA  1317 ;          = ss$_devinact       - device could not be inited
                        04EA  1318 ;          = ss$_ctrlerr        - I/O error during operation
                        04EA  1319 ;          = ss$_devoffline     - device is offline
                        04EA  1320 ;
                        04EA  1321 ;       r10 and r11 are preserved.
                        04EA  1322 ;--
                        04EA  1323
                        04EA  1324 boot_disk_unit:
                        04EA  1325
                        04EA  1326 ;
                        04EA  1327 ; do forced boot block boot
                        04EA  1328 ;
                        04EA  1329 ; If RPB$V_BBLOCK is set then read LBN 0 and transfer control to the
                        04EA  1330 ; block.
                        04EA  1331 ;
                        04EA  1332
            03   E0     04EA  1333          bbs     #rpb$v_bblock,-         ;br if direct boot block boot
      63 30 AB           04EC  1334                  rpb$l_bootr5(r11),80$   ;
                        04EF  1335
                        04EF  1336 ;
                        04EF  1337 ; init the file read cache if this is a FILES-11 boot
                        04EF  1338 ;
                        04EF  1339
FBDE CF    FC19 CF   7D  04EF  1340          movq    file_cache_desc,fil$gq_cache ;reload the descriptor
              11   12  04F6  1341          bneq    10$                     ;br if done
           FB05'   30  04F8  1342          bsbw    boo$cache_alloc         ;allocate the cache
      50   8000 8F   3C  04FB  1343          movzwl  #ss$_memerr,r0          ;assume no memory
FC05 CF    FBD0 CF   7D  0500  1344          movq    fil$gq_cache,file_cache_desc ;save the descriptor
              48   13  0507  1345          beql    75$                     ;br if cache not allocated
```

```
                        0509  1346
                        0509  1347 ;
                        0509  1348 ; Call a device-independent routine, FIL$OPENFILE to locate the named
                        0509  1349 ; file on the disk.
                        0509  1350 ;
                        0509  1351 ; the cache open is where the drive is mounted so it can fail if there is
                        0509  1352 ; no physical volume
                        0509  1353 ;
                        0509  1354
          FAF4'  30     0509  1355 10$:    bsbw    boo$cache_open          ;Open the FILEREAD cache
        23 50    E9     050C  1356        blbc    r0,55$                  ;br if error
        69 AB    9F     050F  1357 15$:    pushab  rpb$t_file+1(r11)       ;Address of file name string.
    7E  68 AB    9A     0512  1358        movzbl  rpb$t_file(r11),-(sp)   ;Character count of file name.
           7E    D4     0516  1359        clrl    -(sp)                   ;Allocate scratch for channel
                        0518  1360                                        ;and get adr of scratch storage
        3C AB    DF     0518  1361        pushal  rpb$l_fillbn(r11)       ;RPB fields that receive file
                        051B  1362                                        ;statistics during OPEN.
           6A    DF     051B  1363        pushal  (R10)                   ;File header buffer at end of
                        051D  1364                                        ;memory.
      0200 CA    DF     051D  1365        pushal  512(R10)                ;Index file header buffer at
                        0521  1366                                        ;end of memory.
        10 AE    DF     0521  1367        pushal  16(sp)                  ;Address in file name desc.
        10 AE    DF     0524  1368        pushal  16(sp)                  ;Address of phony channel.
0000'CF    05    FB     0527  1369        calls   #5,fil$openfile         ;Call FILREAD to locate file.
     5E    0C    C0     052C  1370        addl2   #12,sp                  ;Clean up scratch space
        5B 50    E8     052F  1371        blbs    r0,boot_file            ;Branch on success.
                        0532  1372
                        0532  1373 ;
                        0532  1374 ; the volume is not a files-11 volume, try boot block booting, if the error
                        0532  1375 ; related to a file structure problem
                        0532  1376 ;
                        0532  1377
  08C0 8F    50    B1     0532  1378 55$:    cmpw    r0,#ss$_filestruct      ;test for file structure error code
           19    13     0537  1379        beql    80$                     ;br if that's what it is
  0810 8F    50    B1     0539  1380        cmpw    r0,#ss$_badfilehdr      ;test for file structure error code
           12    13     053E  1381        beql    80$                     ;br if that's what it is
  0828 8F    50    B1     0540  1382        cmpw    r0,#ss$_badirectory     ;test for file structure error code
           0B    13     0545  1383        beql    80$                     ;br if that's what it is
  0808 8F    50    B1     0547  1384        cmpw    r0,#ss$_badchksum       ;test for file structure error code
           04    13     054C  1385        beql    80$                     ;br if that's not what it is
        50  02    C8     054E  1386        bisl    #1@1,r0                 ;make non-fatal
                  05     0551  1387 75$:    rsb                             ;and go back to caller
                        0552  1388
                        0552  1389 ;
                        0552  1390 ; read LBN 0 as boot block
                        0552  1391 ;
                        0552  1392
           58   D4      0552  1393 80$:    clrl    r8                      ;block to read
        59 01   D0      0554  1394        movl    #1,r9                   ;size to read
        56 5A   D0      0557  1395        movl    r10,r6                  ;Start of free memory
             0080  30     055A  1396        bsbw    readfile                ;read the block to R10
          F1 50  E9     055D  1397        blbc    r0,75$                  ;br if error
                        0560  1398
                        0560  1399 ;
                        0560  1400 ; validate the boot block
                        0560  1401 ;
                        0560  1402
```

```
                50    08C2 8F    3C  0560  1403          movzwl  #ss$_filestruct!2,r0    ;set error code, semi-success
                  52    02 AA    9A  0565  1404          movzbl  2(r10),r2               ;get offset to secondary id field
                  01    03 AA    91  0569  1405          cmpb    3(r10),#1               ;next field a BR instruction
                              E2   12  056D  1406          bneq    75$                     ;br if no
                    51    6A42   3E  056F  1407          movaw   (r10)[r2],r1            ;address next field
                                   0573  1408                                          ;this must be in the same page!
                          0172     30  0573  1409          bsbw    verify_boot_block       ;check boot block
                        D8 50      E9  0576  1410          blbc    r0,75$                  ;br if not a valid block
             58   04 AA    10      9C  0579  1411          rotl    #16,4(r10),r8           ;get secondary image LBN
                  59    08 A1      D0  057E  1412          movl    8(r1),r9                ;get image size
                  5A    0C A1      C0  0582  1413          addl    12(r1),r10              ;compute load address
             55   10 A1    5A      C1  0586  1414          addl3   r10,16(r1),r5           ;compute transfer address
                              42   11  058B  1415          brb     readin_boot             ;boot block is valid, read file
                                   058D  1416
                                   058D  1417 ;
                                   058D  1418 ; File was located successfully. Make sure that the file is contiguous.
                                   058D  1419 ; The file statistics block is the following:
                                   058D  1420 ;
                                   058D  1421 ;       +----------------------+
                                   058D  1422 ;       |     starting LBN     | (0 if file not contiguous)
                                   058D  1423 ;       +----------------------+
                                   058D  1424 ;       |    size in blocks    |
                                   058D  1425 ;       +----------------------+
                                   058D  1426 ;
                                   058D  1427
                                   058D  1428 boot_file:                              ;Test for contiguity.
                  58    3C AB      7D  058D  1429          movq    rpb$l_fillbn(r11),r8    ;Get file statistics.
                              58   D5  0591  1430          tstl    r8                      ;Contiguous file?
                              06   12  0593  1431          bneq    60$                     ;Yes, continue.
                50    02AC 8F      3C  0595  1432          movzwl  #ss$_filnotcntg,r0      ;search fatal error
                                   05  059A  1433          rsb
                                   059B  1434
                                   059B  1435 ;
                                   059B  1436 ; If the software boot control flags indicate that that transfer
                                   059B  1437 ; address of the secondary bootstrap is stored in the image file's
                                   059B  1438 ; header block, read that header block. Otherwise, assume that the
                                   059B  1439 ; transfer address is simply the 1st byte in the image file.
                                   059B  1440 ;
                                   059B  1441
                55    5A     D0    C59B  1442 60$:    movl    r10,r5                  ;Assume no special transfer address.
                        06     E1  059E  1443          bbc     #rpb$v_header,-         ;If no header requested,
                  30 AB            05A0  1444                  rpb$l_bootr5(r11),-     ;then just branch past header
                        2C         05A2  1445                  readin_boot             ;reading code.
                56    5A     D0    05A3  1446          movl    r10,r6                  ;Start of free memory
                59    01     D0    05A6  1447          movl    #1,r9                   ;Header is always only 1 block.
                        32     10  05A9  1448          bsbb    readfile                ;Read header block.
                  5E 50      E9    05AB  1449          blbc    r0,no_fit               ;br if error
             58   3C AB      7D    05AE  1450          movq    rpb$l_fillbn(r11),r8    ;R8 = 1st LBN, R9 = block count
                52    59     7D    05B2  1451          movq    r9,r2                   ;R2 = block count, R3 = hdr adr
                    FA48'    30    05B5  1452          bsbw    boo$image_att           ;Get image attributes
                                   05B8  1453
                                   05B8  1454 ;
                                   05B8  1455 ; R1 = image header block count
                                   05B8  1456 ; R2 = size of file in blocks excluding symbol table and patch text
                                   05B8  1457 ;
                                   05B8  1458
             00A0 CB    51     D0    05B8  1459          movl    r1,rpb$b_hdrpgcnt(r11)  ;Store image header block count
```

```
                59   52   51   C3  05BD  1460          subl3   r1,r2,r9                ;Blocks in image after header block(s)
                     58   51   C0  05C1  1461          addl    r1,r8                   ;LBN of first block beyond headr block
                  51   02 AA   3C  05C4  1462          movzwl  ihd$w_activoff(r10),r1  ;Get offset to image
                                   05C8  1463                                          ;activation data in header.
                       51   5A   C0  05C8  1464          addl    r10,r1                  ;Form transfer vector address.
                    55   614A   9E  05CB  1465          movab   (r1)[r10],r5            ;Get transfer address.
                                   05CF  1466
                                   05CF  1467 ;
                                   05CF  1468 ; Now read in the file. If the file is too large for the remaining
                                   05CF  1469 ; memory space, see if the required additional pages are usable.
                                   05CF  1470 ; If they are, use them.  If not issue a fatal diagnostic and HALT.
                                   05CF  1471 ;
                                   05CF  1472 ; Registers set up now are the following:
                                   05CF  1473 ;
                                   05CF  1474 ;       R5      - transfer address
                                   05CF  1475 ;       R8      - starting LBN of file (after header)
                                   05CF  1476 ;       R9      - size of file in blocks
                                   05CF  1477 ;       R10     - address of 1st byte in free memory
                                   05CF  1478 ;       R11     - address of the RPB
                                   05CF  1479 ;       AP      - secondary boot argument list
                                   05CF  1480 ;
                                   05CF  1481
                                   05CF  1482 readin_boot:
                                   05CF  1483
          14 AC    44 AB   7D  05CF  1484          movq    rpb$q_pfnmap(r11),vmb$q_pfnmap(ap);setup bitmap desc
                56   5A   D0  05D4  1485          movl    r10,r6                  ;buffer for read
                                   05D7  1486
                                   05D7  1487 ;
                                   05D7  1488 ; Will the desired number of blocks fit in the space remaining in the
                                   05D7  1489 ; pre-tested 64kb of memory?  If not, check that the additional pages
                                   05D7  1490 ; required are usable.  If they are, then read it all, otherwise quit.
                                   05D7  1491 ;
                                   05D7  1492
                 0136   30  05D7  1493          bsbw    verify_image_memory     ;verify pages for image
               2F 50    E9  05DA  1494          blbc    r0,no_fit               ;br if error
                                   05DD  1495
                                   05DD  1496 ;
                                   05DD  1497 ; Now read the secondary boot code into memory
                                   05DD  1498 ;
                                   05DD  1499 ;       Calls the device-independent bootstrap QIO routine to read
                                   05DD  1500 ;       a file. Divides the file into pieces as large as possible, so
                                   05DD  1501 ;       that the read is a small number (like 1) of DMA transfers.
                                   05DD  1502 ;
                                   05DD  1503 ; Registers:
                                   05DD  1504 ;
                                   05DD  1505 ;       R5      - secondary boot transfer address
                                   05DD  1506 ;       R6      - buffer address
                                   05DD  1507 ;       R8      - logical block number (LBN)
                                   05DD  1508 ;       R9      - number of blocks in file
                                   05DD  1509 ;
                                   05DD  1510
                                   05DD  1511 readfile:                               ;Read file into memory.
       57   7F   8F   9A  05DD  1512          movzbl  #k_max_io_pages,r7      ;Assume maximum transfer size.
            59   57   D1  05E1  1513          cmpl    r7,r9                   ;Minimize with file size.
                 03   15  05E4  1514          bleq    10$                     ;Branch if file larger than
                                   05E6  1515                                  ;maximum transfer size.
            57   59   D0  05E6  1516          movl    r9,r7                   ;Set to remaining file size.
```

```
                        05E9  1517 10$:
                              1518          pushl   r11                     ;Push arguments for QIO.
                     5B DD    05E9  1518          pushl   r11                     ;Push arguments for QIO.
                     00 DD    05EB  1519          pushl   #0                      ;Push phony channel number.
                  7E 21 3C    05ED  1520          movzwl  #io$_readlblk,-(sp)     ;Read logical block function.
                     58 DD    05F0  1521          pushl   r8                      ;Starting LBN.
               7E    57 09 9C 05F2  1522          rotl    #9,r7,-(sp)             ;Transfer size in bytes.
                     56 DD    05F6  1523          pushl   r6                      ;Buffer address
                 56    04 AE  05F8  1524          addl    4(sp),r6                ;Update buffer address.
                  58    57 C0 05FC  1525          addl    r7,r8                   ;Update LBN.
            0000'CF    06 FB 05FF  1526          calls   #6,boo$qio              ;Call a bootstrap QIO routine.
                  05 50 E9  0604  1527          blbc    r0,30$                  ;Continue on success.
                  59    57 C2 0607  1528          subl    r7,r9                   ;Decrement blocks remaining.
                        D1  14 060A  1529          bgtr    readfile                ;Continue if not done.
                            060C  1530
                            060C  1531 ;
                            060C  1532 ;       R0      - status
                            060C  1533 ;       R5      - secondary boot transfer address
                            060C  1534 ;       R6      - buffer address updated past last byte read
                            060C  1535 ;       R8      - LBN updated to block after last block read
                            060C  1536 ;       R9      - blocks in file (reduced to number not read)
                            060C  1537 ;
                            060C  1538
                            060C  1539 30$:
                        05  060C  1540 no_fit: rsb                             ;Return to caller when done.
```

```
                        060D  1542          .sbttl  scb interrupt routines
                        060D  1543 ;++
                        060D  1544 ; ignore_scb_int
                        060D  1545 ;
                        060D  1546 ; functional description:
                        060D  1547 ;
                        060D  1548 ; This sequence runs via an SCB vectored interrupt.
                        060D  1549 ;
                        060D  1550 ; inputs:
                        060D  1551 ;
                        060D  1552 ;       none
                        060D  1553 ;
                        060D  1554 ; outputs:
                        060D  1555 ;
                        060D  1556 ;       none
                        060D  1557 ;--
                        060D  1558
                        060D  1559          .align  long
                        0610  1560
                        0610  1561 ignore_scb_int:
                     02 0610  1562          rei                             ;
```

```
                                0611  1564 ;++
                                0611  1565 ; machine_check_detect
                                0611  1566 ;
                                0611  1567 ; functional description:
                                0611  1568 ;
                                0611  1569 ; This sequence runs when it is enabled in the machine check vector.
                                0611  1570 ; The action is to alter the return address to a value in r1 and continue.
                                0611  1571 ;
                                0611  1572 ; inputs:
                                0611  1573 ;
                                0611  1574 ;       machine check stack
                                0611  1575 ;       machine_check_continue  = address of the continuation code or 0
                                0611  1576 ;
                                0611  1577 ; outputs:
                                0611  1578 ;
                                0611  1579 ;       r0 = machine check code
                                0611  1580 ;--
                                0611  1581
                                0611  1582         .align  long
                                0614  1583
                                0614  1584 machine_check_detect:
                                0614  1585
     26  000000FF  8F    DA   0614  1586         mtpr    #^xff,#pr$_mcesr        ;clear machine check error
         FAF9  CF    D5   061B  1587         tstl    machine_check_continue  ;change return PC?
               0D    13   061F  1588         beql    10$                     ;if eql then no, unexpected
           50  04    AE   D0   0621  1589         movl    4(sp),r0                ;load reason
               5E    8E   C0   0625  1590         addl    (sp)+,sp                ;pop stack
           6E  FAEC  CF   D0   0628  1591         movl    machine_check_continue,(sp) ;actually change return PC
               02    02   062D  1592         rei                             ;continue
                          062E  1593 10$:     fatal_message   scbint
```

```
                        0636  1595 ;++
                        0636  1596 ; fatal_error
                        0636  1597 ;
                        0636  1598 ; functional description:
                        0636  1599 ;
                        0636  1600 ; This routine is entered when a fatal error is to be displayed.
                        0636  1601 ; The input code is a standard ss$_ value and it is matched to a text
                        0636  1602 ; string by scanning a table of longword entries. The first work of the
                        0636  1603 ; longword is the low word of the ss$ code and the next word is the
                        0636  1604 ; displacement to the message text.
                        0636  1605 ;
                        0636  1606 ; inputs:
                        0636  1607 ;
                        0636  1608 ;       r0 = internal error code
                        0636  1609 ;
                        0636  1610 ; outputs:
                        0636  1611 ;
                        0636  1612 ;       The boot is abandoned, the registers are restored to
                        0636  1613 ;       reflect the initial contents and the system is halted.
                        0636  1614 ;--
                        0636  1615
                        0636  1616 fatal_error:
                        0636  1617
5B   0000'CF   9E       0636  1618         movab   rom_base,r11            ;recalculate RPB address
5B     01FF 8F   AA     063B  1619         bicw    #^x1ff,r11              ;
5B     FE00 CB   9E     0640  1620         movab   -k_rom_code_addr(r11),r11;
       FACF CF   D4     0645  1621         clrl    machine_check_continue  ;disable error continue
51     FA32 CF   9E     0649  1622         movab   message_base,r1          ;address message desc
       50   03   CA     064E  1623         bicl    #3,r0                   ;remove severity bits
       50   81   B1     0651  1624 10$:     cmpw    (r1)+,r0                ;compare code
            06   13     0654  1625         beql    15$                     ;br if found
            81   B5     0656  1626         tstw    (r1)+                   ;advance and test for zero offset?
            F7   12     0658  1627         bneq    10$                     ;continue in not found
            2C   11     065A  1628         brb     20$                     ;if list end then no message
     50     61   32     065C  1629 15$:     cvtwl   (r1),r0                 ;fetch displacement from cell
       20   AB   DD     065F  1630         pushl   rpb$l_bootr1(r11)       ;Pass options switch settings
            7E   7C     0662  1631         clrq    -(sp)                   ;no read data
         6140   9F     0664  1632         pushab  (r1)[r0]                ;address of message text
                       0667  1633
                       0667  1634 ;
                       0667  1635 ; output the header part followed by the input code's message
                       0667  1636 ;
                       0667  1637
       20   AB   DD     0667  1638         pushl   rpb$l_bootr1(r11)       ;Pass options switch settings
            7E   7C     066A  1639         clrq    -(sp)                   ;setup header
     F9FD   CF   9F     066C  1640         pushab  message_header          ;
0000'CF     04   FB     0670  1641         calls   #4,boo$readprompt       ;output header
0000'CF     04   FB     0675  1642         calls   #4,boo$readprompt       ;output message
       20   AB   DD     067A  1643         pushl   rpb$l_bootr1(r11)       ;Pass options switch settings
            7E   7C     067D  1644         clrq    -(sp)                   ;output device name
     FA99   CF   9F     067F  1645         pushab  boot_device_name
0000'CF     04   FB     0683  1646         calls   #4,boo$readprompt       ;
                       0688  1647
                       0688  1648 ;
                       0688  1649 ; reload the input registers
                       0688  1650 ;
                       0688  1651
```

```
                5E    6B   D0  0688  1652 20$:    movl    rpb$l_base(r11),sp      ;load sp
             5E  0200  CE   9E  068B  1653         movab   ^x200(sp),sp            ;
             50    1C  AB   7D  0690  1654         movq    rpb$l_bootr0(r11),r0    ;load r0,r1
             52    24  AB   7D  0694  1655         movq    rpb$l_bootr2(r11),r2    ;load r2,r3
             54    2C  AB   7D  0698  1656         movq    rpb$l_bootr4(r11),r4    ;load r4,r5
             5C    18  AB   D0  069C  1657         movl    rpb$l_haltcode(r11),ap  ;load halt code
             5A    10  AB   7D  06A0  1658         movq    rpb$l_haltpc(r11),r10   ;restore PC,PSL
                               06A4  1659
                               06A4  1660 ;
                               06A4  1661 ; halt system, continue will restart the boot
                               06A4  1662 ;
                               06A4  1663
          23  00000F05  8F   DA  06A4  1664 25$:    mtpr    #console_halt,#pr$_txdb ;halt processor
                        F7   11  06AB  1665         brb     25$                     ;
```

```
                     06AD  1667 ;++
                     06AD  1668 ; validate_csr - test for present CSR
                     06AD  1669 ;
                     06AD  1670 ; functional description:
                     06AD  1671 ;
                     06AD  1672 ; This routine tests for a device CSR and returns to the caller's caller
                     06AD  1673 ; if the CSR is not present. The CSR address is calculated from the base
                     06AD  1674 ; CSR address and the controller number.
                     06AD  1675 ;
                     06AD  1676 ; inputs:
                     06AD  1677 ;
                     06AD  1678 ;       r7 = boot device descriptor address
                     06AD  1679 ;       r11 = rpb address
                     06AD  1680 ;
                     06AD  1681 ; outputs:
                     06AD  1682 ;
                     06AD  1683 ;       return to caller implies that the device is present
                     06AD  1684 ;       return to caller's caller with r0 = ss$_devassign+2
                     06AD  1685 ;
                     06AD  1686 ;       The RPB$L_PHYCSR value is filled in.
                     06AD  1687 ;
                     06AD  1688 ;       r0,r1 are destroyed
                     06AD  1689 ;--
                     06AD  1690
                     06AD  1691 validate_csr:
                     06AD  1692
          FA65 CF    BB'AF    9E  06AD  1693         movab   b^10$,machine_check_continue ;chance machine check addr
               51      06 A7    D0  06B3  1694         movl    bd_a_csr(r7),r1         ;get csr address
                          50    D4  06B7  1695         clrl    r0                      ;set present flag
                          61    B5  06B9  1696         tstw    (r1)                    ;test
                    FA59 CF    D4  06BB  1697 10$:     clrl    machine_check_continue  ;zap machine check address
                          50    D5  06BF  1698         tstl    r0                      ;CSR present?
                          09    13  06C1  1699         beql    20$                     ;continue
               50         084A 8F    3C  06C3  1700         movzwl  #ss$_devassign!2,r0     ;set error but semi-success
                          51 8ED0    06C8  1701         popl    r1                      ;pop return to caller
                             05    06CB  1702         rsb                             ;done
                54 AB        51    D0  06CC  1703 20$:     movl    r1,rpb$l_csrphy(r11)    ;save csr address
            23   00000F0E 8F        DA  06D0  1704         mtpr    #led_boot_inprogress,#pr$_txdb; tell operator
                             05    06D7  1705         rsb                             ; return to caller
```

```
                        1707 ;++
                        1708 ; unfielded_scb_int
                        1709 ; secondary_scb_int
                        1710 ;
                        1711 ; functional description:
                        1712 ;
                        1713 ; This routine is executed if an unwanted SCB interrupt occurs during
                        1714 ; booting. An error message is displayed and the system is halted.
                        1715 ;
                        1716 ; inputs:
                        1717 ;
                        1718 ;       scb interrupt stack
                        1719 ;
                        1720 ; outputs:
                        1721 ;
                        1722 ;       none
                        1723 ;--
                        1724
                        1725         .align  long
                        1726
06D8                    1727 unfielded_scb_int:
06D8                    1728
06D8                    1729         fatal_message   scbint
06E0                    1730
06E0                    1731 secondary_scb_int:
06E0                    1732
06E0                    1733         fatal_message   2ndint
```

```
                        ;++
                        ; verify_boot_block
                        ;
                        ; functional description:
                        ;
                        ; This routine verifyes a small memory section as a boot block descriptor.
                        ; It is used to verify a disk boot block or a ROM id block.
                        ;
                        ;
                        ;                +-------+-------+-------+-------+
                        ;       BB+0:    |     1 |     n |     any value |
                        ;                +-------+-------+-------+-------+
                        ;                | Low LBN       | High LBN      |
                        ;                +-------+-------+-------+-------+
                        ;
                        ;                +-------+-------+-------+-------+
                        ;       BB+(2*n)+0:  | Chk   |     k |       18(Hex) |
                        ;                +-------+-------+-------+-------+
                        ;                | any value, most likely 0      |
                        ;                +-------+-------+-------+-------+
                        ;       BB+(2*n)+8:  | size in blocks of the image   |
                        ;                +-------+-------+-------+-------+
                        ;       BB+(2*n)+12: | offset to load                |
                        ;                +-------+-------+-------+-------+
                        ;       BB+(2*n)+16: | offset into image to start    |
                        ;                +-------+-------+-------+-------+
                        ;       BB+(2*n)+20: | sum of the previous three LW's|
                        ;                +-------+-------+-------+-------+
                        ;
                        ; inputs:
                        ;
                        ;       r1 = address of the block
                        ;
                        ; outputs:
                        ;
                        ;       r0 = true or false
                        ;       r1 = original address
                        ;
                        ;       r2 is destroyed
                        ;--
                        verify_boot_block:
                        
50   D4  06E8           clrl    r0                      ;assume not a valid block
18   61  B1  06EA       cmpw    (r1),#^x18              ;VAX instruction set id?
20   12  06ED           bneq    10$                     ;br if no
52 02 A1 18  81  06EF   addb3   #^x18,2(r1),r2          ;get optional value
52   52  92  06F4       mcomb   r2,r2                   ;ones's complement it
03 A1 52  91  06F7      cmpb    r2,3(r1)                ;check check sum byte
12   12  06FB           bneq    10$                     ;continue if no match
52 0C A1 08 A1 C1 06FD  addl3   8(r1),12(r1),r2         ;check other words
52 10 A1 C0  0703       addl    16(r1),r2               ;get augment to load address
14 A1 52  D1  0707      cmpl    r2,20(r1)               ;match?
02  12  070B           bneq    10$                     ;br if no
50  D6  070D           incl    r0                      ;success
    05  070F  10$:     rsb
```

```
                     0710  1792 ;++
                     0710  1793 ; verify_image_memory
                     0710  1794 ;
                     0710  1795 ; functional description:
                     0710  1796 ;
                     0710  1797 ; This routine checks for n contiguous pages from the established load
                     0710  1798 ; address.
                     0710  1799 ;
                     0710  1800 ; inputs:
                     0710  1801 ;
                     0710  1802 ;       r9 = desired page count
                     0710  1803 ;       r10 = target load address
                     0710  1804 ;       r11 = address of the RPB
                     0710  1805 ;       ap = boot argument list
                     0710  1806 ;
                     0710  1807
                     0710  1808 verify_image_memory:
                     0710  1809
 50   0601 8F    3C  0710  1810          movzwl  #ss$_bufferovf,r0       ;set error code
 52   5B   17    9C  0715  1811          rotl    #<32-9>,r11,r2          ;PFN for RPB
   52   7F A2    DE  0719  1812          moval   127(r2),r2              ;Last PFN guaranteed to be good
 51   5A   17    9C  071D  1813          rotl    #<32-9>,r10,r1          ;Starting PFN for read
      51   59    C0  0721  1814          addl    r9,r1                   ;Last+1 PFN needed to be good
           05    11  0724  1815          brb     30$                     ;Zero or more iterations
07 18 BC   52    E1  0726  1816 10$:     bbc     r2,@vmb$q_pfnmap+4(ap),40$ ;Branch if cannot
                     072B  1817                                          ;read the entire secondary boot
   F7 52   51    F2  072B  1818 30$:     aoblss  r1,r2,10$               ;Check the next page
      50   01    D0  072F  1819          movl    #1,r0                   ;correct
                     0732  1820 40$:     rsb
```

```
                     0733  1822 ;++
                     0733  1823 ; write_timeout
                     0733  1824 ;
                     0733  1825 ; functional description:
                     0733  1826 ;
                     0733  1827 ; This sequence runs when a write timeout interrupt occurs.
                     0733  1828 ;
                     0733  1829 ; inputs:
                     0733  1830 ;
                     0733  1831 ;       PC/PSL are on the stack
                     0733  1832 ;       machine_check_continue = address to continue at or 0
                     0733  1833 ;
                     0733  1834 ; outputs:
                     0733  1835 ;
                     0733  1836 ;       r0 = error code
                     0733  1837 ;--
                     0733  1838
                     0733  1839
                     0733  1840         .align  long
                     0734  1841
                     0734  1842 write_timeout_int:
                     0734  1843
      6E   F9E0 CF   D0  0734  1844         movl    machine_check_continue,(sp) ;reset PC
           9D   13   D0  0739  1845         beql    unfielded_scb_int       ;unexpected error if no continue addr
           50   02   D0  073B  1846         movl    #k_bus.timeout,r0       ;set code
                     02  073E  1847         rei                             ;done
                         073F  1848
                         073F  1849         .end    ROM_START
```

```
A_2NDINT                000000E7E R  03
A_BADFILENAME           000000A8 R   03
A_BUFFEROVF             000000BA R   03
A_CTRLERR               000000EA R   03
A_DEVASSIGN             00000036 R   03
A_DEVINACT              00000101 R   03
A_DEVOFFLINE            00000123 R   03
A_FILESTRUCT            00000066 R   03
A_FILNOTCNTG            0000008A R   03
A_MEMERR                00000136 R   03
A_NOROM                 000001B1 R   03
A_NOSUCHDEV             00000060 R   03
A_NOSUCHFILE            0000004D R   03
A_NOSUCHNODE            000001CB R   03
A_SCBINT                00000153 R   03
BDT$L_ACTION            00000004
BDT$L_ADDR              0000000C
BDT$L_AUXDRNAME         00000018
BDT$L_CPUTYPE           00000000
BDT$L_DEVNAME           00000024
BDT$L_DEVTYPE           00000002
BDT$L_DRIVRNAME         00000014
BDT$L_ENTRY             00000010
BDT$L_SIZE              00000008
BDT$L_UNIT_DISC         00000020
BDT$L_UNIT_INIT         0000001C
BD_A_CSR                00000006
BD_A_ROUTINE            0000000A
BD_B_HIGH_UNIT          00000004
BD_B_TYPE               00000005
BD_L_NAME               00000000
BD_S_BD                 0000000E
BEGIN_BOOT              000002D8 R       02
BOO$AC_VECTOR           ********   X     02
BOO$CACHE_ALLOC         ********   X     02
BOO$CACHE_OPEN          ********   X     02
BOO$DOWNLINE_LOAD       ********   X     02
BOO$GB_SYSTEMID       = 000000F4 RG      02
BOO$GL_RPBBASE          00000114 RG      02
BOO$IMAGE_ATT           ********   X     02
BOO$QIO                 ********   X     02
BOO$READPROMPT          ********   X     02
BOOT_DEVICE_LIST        00000041 R       02
BOOT_DEVICE_NAME        0000011C R       02
BOOT_DISK_UNIT          000004EA R       02
BOOT_FILE               0000058D R       02
BQO$C_MOVE            = 00000018
BQO$L_UNIT_INIT       = 0000001C
BTD$K_PROM            = 00000008
BTD$K_QNA             = 00000060
BTD$K_UDA             = 00000011
CONSOLE_HALT          = 00000F05
DIAGFILE                00000019 R       02
DISK_BOOT               00000456 R       02
EXE$GB_CPUTYPE          ********   X     02
FATAL_ERROR             00000636 R       02
FATAL_MEMORY_ERROR      000002D0 R       02
FIL$GQ_CACHE          = 000000D4 RG      02
FIL$OPENFILE            ********   X     02
FILE_CACHE_DESC           0000010C R       02
IGNORE_SCB_INT            00000610 R       02
IHD$W_ACTIVOFF        = 00000002
IO$_READLBLK          = 00000021
IPL$_POWER            = 0000001F
K_BUS.T.MEOUT         = 00000002
K_MAX_IO_PAGES        = 0000007F
K_MAX_MEMORY_PAGES    = 00002000
K_PARITY.ERROR        = 00000001
K_PFN_MAP_ADDR        = 00004600
K_ROM_CODE_ADDR       = 00000200
K_RPB_ADDR            = 00000000
K_SCB_ADDR            = 00004200
K_SECONDARY_BOOT_ADDR = 00005000
LAST_MSG              = 000001CB R       03
LED_BOOT_INPROGRESS   = 00000F0E
LED_MEMORY_OK         = 00000F0D
LED_TRANSFER_CONTROL  = 00000F0F
MACHINE_CHECK_CONTINUE    00000118 R       02
MACHINE_CHECK_DETECT      00000614 R       02
MESSAGE_BASE              0000007F R       02
MESSAGE_HEADER            0000006D R       02
MSV11_CSR_BASE        = 20001440
MSV11_CSR_PARITY_ENABLE = 00000001
NAMEPROMPT                00000035 R       02
NDT$_UBO              = 00000028
NETWORK_BOOT              00000429 R       02
NEXT_PAGE                 00000289 R       02
NO_DISK_BOOT_DEVICE_LIST  0000004F R       02
NO_FIT                    0000060C R       02
NXM_MEMORY                0000029F R       02
PAGE_BOUNDARY             000001F8 R       02
PHY_A_IO_SPACE        = 20000000
PR$D_SID_TYPE         = 00000018
PR$_IPL               = 00000012
PR$_MCESR             = 00000026
PR$_SCBB              = 00000011
PR$_SID               = 0000003E
PR$_TXDB              = 00000023
PROM_BOOT                 000003EF R       02
READFILE                  000005DD R       02
READIN_BOOT               000005CF R       02
ROM_BASE                  00000000 R       04
ROM_START                 00000121 R       02
RPB$B_BOOTNDT         = 000000A1
RPB$B_CONFREG         = 00000090
RPB$B_DEVTYP          = 00000066
RPB$B_HDRPGCNT        = 000000A0
RPB$B_SLAVE           = 00000067
RPB$C_LENGTH          = 00000108
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$V_DIAG                      = 00000004
RPB$V_HALT                      = 00000009
RPB$V_HEADER                    = 00000006
RPB$V_SOLICT                    = 00000008
RPB$V_TOPSYS                    = 0000001C
RPB$W_UNIT                      = 00000064
SCB_A_BREAKPOINT                = 0000002C
SCB_A_MCHECK                    = 00000004
SCB_A_TRACE_TRAP                = 00000028
SCB_A_WRITE_TIMEOUT             = 00000060
SECONDARY_SCB_INT                 000006E0 R      02
SECOND_PARAM                      000000D0 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$_SCBINT                      = 00008008
SWITCH_V_DISK_BOOT              = 00000007
SWITCH_V_QVSS                   = 00000006
UNFIELDED_SCB_INT                 000006D8 R      02
VALIDATE_CSR                      000006AD R      02
VERIFY_BOOT_BLOCK                 000006E8 R      02
VERIFY_IMAGE_MEMORY               00000710 R      02
VMB$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_FICECACHE                   00000004
VMB$Q_NODENAME                    00000034
VMB$Q_PFNMAP                      00000014
VMB$Q_UCODE                       0000001C
VMSFICE                           00000000 R      02
WRITE_TIMEOUT_INT                 00000734 R      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
$$$$04BOOT	0000073F (1855.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$$10BOOT	000001E9 (489.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$$00BOOT	00000008 (8.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	16	00:00:00.08	00:00:00.24
Command processing	75	00:00:00.55	00:00:01.06
Pass 1	356	00:00:14.01	00:00:20.29
Symbol table sort	0	00:00:01.90	00:00:02.02
Pass 2	291	00:00:04.39	00:00:06.76
Symbol table output	21	00:00:00.11	00:00:00.25
Psect synopsis output	4	00:00:00.03	00:00:00.23
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	763	00:00:21.08	00:00:30.86

The working set limit was 1800 pages.
85253 bytes (167 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1184 non-local and 57 local symbols.
1849 source lines were read in Pass 1, producing 23 object records in Pass 2.
24 pages of virtual memory were used to define 23 macros.

Macro library statistics

Macro library name	Macros defined
DISK$STARWORK03:[GAMACHE.UV1ROM.VMS]LIBUV1.ML	6
DISK$STARWORK03:[GAMACHE.UV1ROM.OBJ]VMB.MLB;3	4
SYS$SYSROOT:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	16

1209 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MAC/LIS=LIS$:VMBUVAX1/OBJ=OBJ$:VMBUVAX1 VMB$:VMBUVAX1+OBJ$:VMB/LIB+VMS$:LIBUV1/LIB
```

LUV1ROM

PQBTDRIVR
LIS

QVSS
LIS

SEARCHMSG
LIS

VMBUVAX1
MAP

SETUSER
LIS

VMBUVAX1
LIS

BOOTDRUV1
LIS

FILERDUV1
LIS

BOOTIOUV1
LIS

CONIO
LIS

NETBOOT
LIS

0431 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

WELCOME
TXT

LOGIN
COM

SYSTARTUP
COM

XQBOOT
LIS

EDTINI
EDT

REMOVE
COM

VERIFY
LIS

VERIFY

UVMS

UVSTARTUP
COM

VERIFY
MAP

ADDUSER
COM

MODPARAMS
DAT

RESTUSER
COM

BACKUSER
COM

SUCCESS
TXT

SYLOGIN
COM