F 5

```
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO     TTTTTTTTTTTTTTT     SSSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO     TTTTTTTTTTTTTTT     SSSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO     TTTTTTTTTTTTTTT     SSSSSSSSSSSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT        SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT        SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT        SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT        SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT        SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT        SSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO        TTT           SSSSSSSSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO        TTT           SSSSSSSSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO        TTT           SSSSSSSSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT                    SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT                    SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT                    SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT                    SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT                    SSS
BBB         BBB  OOO         OOO  OOO         OOO        TTT                    SSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO           TTT        SSSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO           TTT        SSSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO           TTT        SSSSSSSSSSSS
```

```
VV       VV  MM       MM  BBBBBBBB
VV       VV  MM       MM  BBBBBBBB
VV       VV  MMMM   MMMM  BB      BB
VV       VV  MMMM   MMMM  BB      BB
VV       VV  MM  MM  MM  BB      BB
VV       VV  MM  MM  MM  BB      BB
VV       VV  MM      MM  BBBBBBBB
VV       VV  MM      MM  BBBBBBBB
VV       VV  MM      MM  BB      BB
VV       VV  MM      MM  BB      BB
  VV   VV    MM      MM  BB      BB      ....
  VV   VV    MM      MM  BB      BB      ....
    VV       MM      MM  BBBBBBBB        ....
    VV       MM      MM  BBBBBBBB        ....

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

Table of contents

```
0000     1          .TITLE  VMB - VMS Primary Bootstrap Routine
0000     2          .IDENT  'V04-002'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;++
0000    29 ;
0000    30 ; FACILITY:
0000    31 ;
0000    32 ;       Bootstrap module for VAX 11/780, 11/750, and 11/730 hardware
0000    33 ;
0000    34 ; ENVIRONMENT:
0000    35 ;
0000    36 ;       Runs at IPL 31, kernel mode, memory management is OFF, IS=1
0000    37 ;       (running on interrupt stack), and code must be PIC.
0000    38 ;
0000    39 ; ABSTRACT:
0000    40 ;
0000    41 ;       This module contains the primary bootstrap code. The main
0000    42 ;       routine -- START_BOOT -- gains control from CONSOLE, boot block
0000    43 ;       0 code, or from BOOT58. The code creates a System Control
0000    44 ;       Block (SCB), initializes XDELTA if requested, initializes the
0000    45 ;       Restart Parameter Block (RPB), creates a PFN bit map describing
0000    46 ;       all of physical memory, reads in a secondary bootstrap program,
0000    47 ;       and transfers control to that bootstrap.
0000    48 ;
0000    49 ; AUTHOR:
0000    50 ;
0000    51 ;       RICHARD I. HUSTVEDT, Creation date:     18-OCT-1977
0000    52 ;
0000    53 ; MODIFIED BY:
0000    54 ;
0000    55 ;       V04-002 TCM0019         Trudy C. Matthews       07-Sep-1984
0000    56 ;               Change the venus delay constants in TENUSECTBL and UBDELAY.
0000    57 ;               Cache is not enabled in this ear y stage of bootstrap so
```

```
0000     58 ;                  the delay values must be decreased from 10 to 2.
0000     59 ;
0000     60 ;        V04-001 WMC0033         Wayne Cardoza           05-Sep-1984
0000     61 ;                .I page table mut account for bad pages.
0000     62 ;
0000     63 ;        V03-033 TCM0018         Trudy C. Matthews       01-Aug-1984
0000     64 ;                Fix more bugs in FIND_CI_790.
0000     65 ;
0000     66 ;        V03-032 CWH3032         CW Hobbs                25-Jul-1984
0000     67 ;                Remove test for 750 cpu in console boot, those will now start
0000     68 ;                with BOOT58 and ask for first floppy so that CI780 and PCS
0000     69 ;                are available to 750.  Also, lengthen buffer for confirmation
0000     70 ;                prompt so that overruns won't smash RPBBASE as soon.
0000     71 ;
0000     72 ;        V03-031 TCM0017         Trudy C. Matthews       23-Jul-1984
0000     73 ;                Fix boot adapter register space calculation in SAVE_CSR_790.
0000     74 ;                Also add support for a new RPB field: RPB$B_CTRLLTR.  This
0000     75 ;                field allows a controller letter for the boot device's
0000     76 ;                controller to be an explicit input to VMB (in R2).  We made
0000     77 ;                this option available because INIT's algorithm for deriving
0000     78 ;                the boot device's controller letter is less than perfect.
0000     79 ;
0000     80 ;        V03-030 TCM0016         Trudy C. Matthews       04-Apr-1984
0000     81 ;                Fix some bugs in routine FIND_CI_790.
0000     82 ;
0000     83 ;        V03-029 CWH3029         CW Hobbs                15-Mar-1984
0000     84 ;                Require a "Y" response when switching console volumes,
0000     85 ;                rather than accepting a carriage return.
0000     86 ;
0000     87 ;        V03-028 WHM0002         Bill Matthews           13-Mar-1984
0000     88 ;                Modify support for common system files so the interface
0000     89 ;                between VMB and SYSBOOT doesn't have to change.
0000     90 ;
0000     91 ;        V03-027 KTA3109         Kerbey T. Altmann       08-Mar-1984
0000     92 ;                Add support for loading PCS750 ucode.
0000     93 ;
0000     94 ;        V03-026 KPL0001         Peter Lieberwirth        7-Mar-1984
0000     95 ;                Use MOVZBW to fill in RPB BOOTNDT field.  We'll have no
0000     96 ;                16-bit device types until the BI, but those system routines
0000     97 ;                starting to use the word field need to be protected
0000     98 ;                against mutant memory (non-zeroed memory).
0000     99 ;
0000    100 ;        V03-025 WHM0001         Bill Matthews           24-Feb-1984
0000    101 ;                Add support for booting off a set of common system files.
0000    102 ;
0000    103 ;        V03-024 TCM0015         Trudy C. Matthews       16-Jan-1984
0000    104 ;                Add $PAMMDEF and $PR790DEF missing from TCM0014.
0000    105 ;
0000    106 ;        V03-023 WMC0023         Wayne Cardoza           12-Jan-1984
0000    107 ;                Add a missing .ENABL LSB
0000    108 ;
0000    109 ;        V03-022 TCM0014         Trudy C. Matthews       12-Dec-1983
0000    110 ;                Use PAMM to determine if an adapter is present on the
0000    111 ;                ABUS in FIND_CI_790.
0000    112 ;
0000    113 ;        V03-021 RLRSCORP        Robert L. Rappaport     8-Nov-1983
0000    114 ;                Begin to add Scorpio support.
```

```
0000   115 ;
0000   116 ;       V03-020 TCM0012         Trudy C. Matthews       08-Nov-1983
0000   117 ;               Correct several address calculation bugs in routines
0000   118 ;               SAVE_CSR_790 and FIND_CI_790.  Move VMB's "permanent" stack
0000   119 ;               allocation logic to before the first XDELTA breakpoint.  This
0000   120 ;               allows VMB to be linked with XDELTA's instruction decode
0000   121 ;               logic for easier debugging.
0000   122 ;
0000   123 ;       V03-019 TCM0011         Trudy C. Matthews       28-Oct-1983
0000   124 ;               Initialize RPB$B_FLAGS field to zero.
0000   125 ;
0000   126 ;       V03-018 CWH3018         CW Hobbs                9-Sep-1983
0000   127 ;               Change the wording of the prompt string for switching console
0000   128 ;               volumes, add a message that we are resuming load operation.
0000   129 ;
0000   130 ;       V03-017 KDM0073         Kathleen D. Morse       22-Aug-1983
0000   131 ;               Make references to TENUSECTBL and UBDELAY PIC.
0000   132 ;               Move the following cells to BOOTDRIVR.MAR:  EXE$GL_UBDELAY
0000   133 ;               and EXE$GL_TENUSEC.
0000   134 ;
0000   135 ;       V03-016 TCM0010         Trudy C. Matthews       02-Aug-1983
0000   136 ;               Store contents of PR$_SID in EXE$GB_CPUDATA cell before
0000   137 ;               any CPUDISP macros are executed.  Also update TENUSECTBL
0000   138 ;               and UBDELAY tables with appropriate values for the 11/785.
0000   139 ;
0000   140 ;       V03-015 KDM0062         Kathleen D. Morse       18-Jul-1983
0000   141 ;               Add data cells used by TIMEDWAIT macro, so that boot
0000   142 ;               drivers can use a standard time-wait macro.
0000   143 ;
0000   144 ;       V03-014 KTA3074         Kerbey T. Altmann       13-Jul-1983
0000   145 ;               If booting DIAG SUPRV, put CI pagetables in hi mem.
0000   146 ;
0000   147 ;       V03-013 RLRCPUDISP      Robert L. Rappaport     15-Jun-1983
0000   148 ;               Use new CPUDISP macro to check for supported CPU's.
0000   149 ;
0000   150 ;       V03-012 TCM0009         Trudy C. Matthews       27-Apr-1983
0000   151 ;               Change sense of BOOTR5 CRDTEST bit from an enable to an
0000   152 ;               inhibit (i.e. by default, pages with CRD errors are removed
0000   153 ;               during the memory test).
0000   154 ;
0000   155 ;       V03-011 KTA3039         Kerbey T. Altmann       08-Mar-1983
0000   156 ;               More elegant fix for TCM0008/KTA3037.
0000   157 ;
0000   158 ;       V03-010 TCM0008         Trudy C. Matthews       21-Feb-1983
0000   159 ;               Change invalid comparison made in KT13037.
0000   160 ;
0000   161 ;       V03-009 KTA3037         Kerbey T. Altmann       10-Feb-1983
0000   162 ;               Enhance SAVE_CSR_750 to handle floating adapters.
0000   163 ;
0000   164 ;       V03-008 TCM0007         Trudy C. Matthews       25-Jan-1983
0000   165 ;               Add routines to detect a CI adapter on any supported CPU.
0000   166 ;               Add a new R5 flag bit that requests that pages with
0000   167 ;               correctable memory errors be discarded at bootstrap time.
0000   168 ;
0000   169 ;       V03-007 TCM0006         Trudy C. Matthews       11-Jan-1983
0000   170 ;               Change 11/790 machine check handler to not clear out SBIA
0000   171 ;               error bits (will be done later before enabling interrupts).
```

```
0000   172 ;               Change 11/780 machine check handler to write PR$_SBIFS back
0000   173 ;               to itself to clear 11/780 error.  Define routine
0000   174 ;               SYSL$CLRSBIA for linkage to XDELTA.
0000   175 ;
0000   176 ;       V03-006 TCM0005         Trudy C. Matthews       8-Nov-1982
0000   177 ;               Add 11/790 machine check handler to Unibus memory scan
0000   178 ;               code.  Add RPB$L_BADPGS field to RPB; contains number of
0000   179 ;               bad pages found during VMB's memory scan.
0000   180 ;
0000   181 ;       V03-005 KTA3008         Kerbey T. Altmann       11-Oct-1982
0000   182 ;               Protect R7 across a call to file system.
0000   183 ;               Make UNIBUS memory scan more sophisticated.
0000   184 ;
0000   185 ;       V03-004 TCM0004         Trudy C. Matthews       28-Jul-1982
0000   186 ;               Change "7VV" symbols to "790" symbols.
0000   187 ;
0000   188 ;
0000   189 ;--
```

```
                        0000   191          .SBTTL  Declarations
                        0000   192
                        0000   193          .DEFAULT DISPLACEMENT, WORD
                        0000   194
                        0000   195 ;
                        0000   196 ; Macros to describe VMS data structures
                        0000   197 ;
                        0000   198
                        0000   199          $BQODEF                         ; Boot QIO offsets
                        0000   200          $BTDDEF                         ; Boot device definitions
                        0000   201          $DMPDEF                         ; System dump file header definitions
                        0000   202          $EHSRDEF                        ; 11/790 Error handling status reg.
                        0000   203          $IHDDEF                         ; Image header definitions
                        0000   204          $IODEF                          ; I/O function codes
                        0000   205          $IO750DEF                       ; 11/750 definitions
                        0000   206          $IO780DEF                       ; 11/780 definitions
                        0000   207          $IO730DEF                       ; 11/730 definitions
                        0000   208          $IO790DEF                       ; 11/790 definitions
                        0000   209          $IO8SSDEF                       ; 11/8SS definitions
                        0000   210          $MBADEF                         ; MASSBUS adapter registers
                        0000   211          $MCF790DEF                      ; 11/790 machine check stack frame
                        0000   212          $MSTAT2DEF                      ; 11/790 memory status register
                        0000   213          $NDTDEF                         ; Nexus device types
                        0000   214          $PAMMDEF                        ; 11/790 physical address memory map
                        0000   215          $PRDEF                          ; Processor registers
                        0000   216          $PR790DEF                       ; 11/790 Processor registers
                        0000   217          $PR8SSDEF                       ; 11/8SS Processor registers
                        0000   218          $RPBDEF                         ; Restart parameter block
                        0000   219          $SBIADEF                        ; 11/790 SBI adapter definitions
                        0000   220          $SSDEF                          ; System status codes
                        0000   221          $UBADEF                         ; UNIBUS adapter registers
                        0000   222          $UBIDEF                         ; 11/750 UNIBUS adapter
                        0000   223          $VADEF                          ; Virtual address fields
                        0000   224          $VMBARGDEF                      ; Define VMB argument list offsets
                        0000   225
                        0000   226  ;
                        0000   227  ; Field definitions of the CPU-specific data block used by VMB.EXE
                        0000   228  ;
                        0000   229
                        0000   230          $DEFINI CPU,GLOBAL
                        0000   231
                        0000   232  $DEF    CPU_W_SAVE_CSRS                 ; Routine to save CSRs.
00000002                0000   233                          .BLKW   1
                        0002   234  $DEF    CPU_W_CHECKMEM                  ; Routine to test memory.
00000004                0002   235                          .BLKW   1
                        0004   236  $DEF    CPU_W_INIT_ADAP                 ; Routine to initialize adapters.
00000006                0004   237                          .BLKW   1
                        0006   238  $DEF    CPU_W_FIND_CI                   ; Routine to find CI port.
00000008                0006   239                          .BLKW   1
                        0008   240
                        0008   241          $DEFEND CPU
                        0000   242
                        0000   243 ;
                        0000   244 ; Macros
                        0000   245 ;
                        0000   246
                        0000   247          .MACRO  ERROR,STR               ; Outputs an error string to the
```

```
0000    248          BSBW    ERROUT                          ; console terminal.
0000    249          .ASCIZ  STR
0000    250          .ENDM   ERROR
0000    251
0000    252 ;
0000    253 ; Generate a word-relative address.
0000    254 ;
0000    255
0000    256          .MACRO  RELADR  ADDRESS,BASE
0000    257          .WORD   ADDRESS-BASE
0000    258          .ENDM   RELADR
0000    259
0000    260 ;
0000    261 ; Turns a CPU identification code into the relative address of a table.
0000    262 ;
0000    263
0000    264          .MACRO  CPU_IDENT LABEL,TABLE
0000    265          RELADR  CPU_DATA_'TABLE',LABEL
0000    266          .ENDM   CPU_IDENT
0000    267
0000    268 ;
0000    269 ; Defines a table of data that is CPU-specific, and PIC.
0000    270 ;
0000    271
0000    272          .MACRO  CPU_DEF LABEL
0000    273 CPU_DATA_'LABEL':                                ; Name of table.
0000    274          RELADR  SAVE_CSR_'LABEL',CPU_DATA_'LABEL'
0000    275                                                  ; Routine to compute CSRs.
0000    276          RELADR  CHECKMEM_'LABEL',CPU_DATA_'LABEL'
0000    277                                                  ; Routine to test memory.
0000    278          RELADR  INIT_ADP_'LABEL',CPU_DATA_'LABEL'
0000    279                                                  ; Routine to init adapters.
0000    280          RELADR  FIND_CI_'LABEL',CPU_DATA_'LABEL'
0000    281                                                  ; Routine to find CI port.
0000    282          .ENDM   CPU_DEF
0000    283
0000    284
0000    285 ;
0000    286 ; Branch to a new PSECT
0000    287 ;
0000    288
0000    289          .MACRO  BRW_PSECT LABEL,PSECT=<$$$$00BOOT,LONG>
0000    290          .SHOW   EXPANSIONS
0000    291
0000    292          BRW     LABEL
0000    293 ;
0000    294 ; ***** Change Program Section
0000    295 ;
0000    296          .PSECT  PSECT
0000    297
0000    298 LABEL:
0000    299          .NOSHOW EXPANSIONS
0000    300          .ENDM   BRW_PSECT
0000    301
0000    302 ;
0000    303 ; Set new PSECT
0000    304 ;
```

```
                     0000   305
                     0000   306          .MACRO  SET_PSECT PSECT=<$$$$00BOOT,LONG>
                     0000   307          .SHOW   EXPANSIONS
                     0000   308 ;
                     0000   309 ; ***** Change Program Section
                     0000   310 ;
                     0000   311          .PSECT  PSECT
                     0000   312
                     0000   313          .NOSHOW EXPANSIONS
                     0000   314          .ENDM   SET_PSECT
                     0000   315
                     0000   316 ;
                     0000   317 ; Build a table of memory size and ranges
                     0000   318 ;
                     0000   319          .MACRO  MEM_TABLE,MEM_SIZE,START
                     0000   320          .LONG   MEM_SIZE-<MEM_SIZE/10>
                     0000   321          .WORD   START
                     0000   322          .ENDM   MEM_TABLE
                     0000   323
                     0000   324 ;
                     0000   325 ; Equated symbols
                     0000   326 ;
                     0000   327
0000007F             0000   328          IO_SIZE         = 127           ; Maximum # blocks in one read
00000001             0000   329          DEBUG           = 1             ; Assemble DEBUG code
0000000D             0000   330          CR              = 13            ; ASCII code for carriage return
0000000A             0000   331          LF              = 10            ; ASCII code for line feed
00000004             0000   332          BITMAP_PAG_CNT  = 4             ; Number of pre-allocated
00000003             0000   333          STACK_PAG_CNT   = 3             ; Number of stack pages to allocate
                     0000   334                                          ; PFN bitmap pages
                     0000   335
                     0000   336 ;
                     0000   337 ; Static storage
                     0000   338 ;
                     0000   339
                00000000    340          .PSECT  __Z99BOOT,PAGE          ; PSECT that always links at end
                     0000   341                                          ; of bootstrap.
                     0000   342 BOOTHIGH::                               ; Symbol to mark the start of
                     0000   343                                          ; the first page after the code
                     0000   344                                          ; in this module.
                00000000    345          .PSECT  BOOTDRIV_9,PAGE         ; PSECT at end of drivers
                     0000   346 OVERLAY_START:                           ; Start overlaying VMB here
                     0000   347
                     0000   348          .IF     DF,DEBUG                ; If debugging code included,
00000004             0000   349 EXE$MCHKVEC     == BOOTHIGH+4            ; define a symbol used by XDELTA
                     0000   350          .ENDC                           ; to locate the SCB.
                     0000   351
                     0000   352 ;
                     0000   353 ; Declare a code PSECT that will always link at the start of the image.
                     0000   354 ;
                     0000   355
                     0000   356          SET_PSECT                       ; Use default PSECT
                     0000
                     0000       ;
                     0000       ; ***** Change Program Section
                     0000       ;
                00000000            .PSECT  $$$$00BOOT,LONG
```

0000

```
0000    358          .SBTTL  START_BOOT, Primary bootstrap routine
0000    359
0000    360 ;++
0000    361 ; Functional description:
0000    362 ;
0000    363 ;       VMB is loaded into physical memory and gains control via a JMP
0000    364 ;       instruction from CONSOLE, boot block 0 on the boot device, or
0000    365 ;       BOOT5B. VMB gains control in the routine START_BOOT.
0000    366 ;
0000    367 ;       VMB begins by creating and initializing an SCB. If the software
0000    368 ;       bootstrap control flags specified a bootstrap breakpoint, VMB
0000    369 ;       then executes a BPT instruction that transfers control to
0000    370 ;       XDELTA.
0000    371 ;
0000    372 ;       After the XDELTA breakpoint, VMB initializes a system data
0000    373 ;       structure, i.e., a restart parameter block (RPB) that allows
0000    374 ;       a system reboot after a power failure or crash. The RPB holds
0000    375 ;       the bootstrap input registers, the boot device's CSR and bus
0000    376 ;       configuration register (CR), the address of the RPB itself, and
0000    377 ;       pointers to a primitive device driver.
0000    378 ;
0000    379 ;       VMB, the primary bootstrap, also identifies all physical memory
0000    380 ;       in the configuration by creating a bit map in which each bit
0000    381 ;       represents one page of physical memory. In the process of
0000    382 ;       testing all memory, VMB determines which NEXUSes on the system
0000    383 ;       bus are attached to adapters. For every adapter present, VMB
0000    384 ;       records the adapter type in the RPB.
0000    385 ;
0000    386 ;       Finally, VMB chooses a secondary bootstrap image -- either by
0000    387 ;       default, by boot flag settings, or by soliciting a file
0000    388 ;       specification from the user. VMB uses a minimal driver for the
0000    389 ;       bootstrap device to load the secondary image into memory, and
0000    390 ;       transfers control to that bootstrap.
0000    391 ;
0000    392 ;       The secondary bootstrap -- usually SYSBOOT.EXE -- uses the
0000    393 ;       minimal driver from VMB for reading and writing to and from the
0000    394 ;       bootstrap device. Thus, SYSBOOT is device-independent.
0000    395 ;
0000    396 ;       VMB has CPU dependencies such as system bus addresses, memory
0000    397 ;       controller registers, and bus adapter register formats.
0000    398 ;       Therefore, VMB consists of common code that applies to all VAX
0000    399 ;       implementations, and CPU-specific code that applies to one
0000    400 ;       hardware implementation only. The current version supports the
0000    401 ;       following CPUs:
0000    402 ;
0000    403 ;               11/780  (STAR)
0000    404 ;               11/750  (COMET)
0000    405 ;               11/730  (NEBULA)
0000    406 ;               11/790  (VENUS)
0000    407 ;
0000    408 ; Inputs:
0000    409 ;
0000    410 ;       R0      - <07:00> boot device type code (RPB$B_DEVTYP)
0000    411 ;
0000    412 ;                         0       MASSBUS device (RM02/3,RP04/5/6/7,RM80)
0000    413 ;                         1       RK06/7
0000    414 ;                         2       RL01/2
```

```
0000   415 ;                              3       IDC(almost an RA80) on 11/730
0000   416 ;                              17      UDA-50
0000   417 ;                                      (note: values 1 - 31 are reserved for
0000   418 ;                                       Unibus devices)
0000   419 ;                              32      HSC on CI
0000   420 ;                              64      Console block storage device
0000   421 ;
0000   422 ;                      - <15:08> reserved for future expansion
0000   423 ;
0000   424 ;                      - <31:16> device class dependent (RPB$W_ROUBVEC)
0000   425 ;
0000   426 ;                              UNIBUS  - optional vector address; 0 implies
0000   427 ;                                        use the default vector
0000   428 ;
0000   429 ;                              MASSBUS - not used
0000   430 ;
0000   431 ;               R1     - boot device's bus address
0000   432 ;
0000   433 ;                              11/780 &
0000   434 ;                              11/730  - <31:04> MBZ
0000   435 ;                                        <03:00> TR number of adapter
0000   436 ;
0000   437 ;                              11/750  - <31:24> MBZ
0000   438 ;                                        <23:00> address of the I/O page for the
0000   439 ;                                                boot device's adapter
0000   440 ;
0000   441 ;                              11/790  - <31:06> MBZ
0000   442 ;                                        <05:04> A-bus Adapter number
0000   443 ;                                        <03:00> TR number of the adapter
0000   444 ;
0000   445 ;                              11/8SS - <31:04> MBZ
0000   446 ;                                       <03:00> BI node number of adapter
0000   447 ;               R2
0000   448 ;                      - all controllers:
0000   449 ;                              <31:24> - controller letter designator (optional)
0000   450 ;
0000   451 ;                      - UNIBUS:
0000   452 ;
0000   453 ;                              <31:18> MBZ
0000   454 ;                              <17:00> UNIBUS address of the device's CSR
0000   455 ;
0000   456 ;                              -11/8SS - <31:00> BI address of boot device's CSR
0000   457 ;
0000   458 ;                      - MASSBUS:
0000   459 ;
0000   460 ;                              <31:04> MBZ
0000   461 ;                              <03:00> adapter's controller/formatter number
0000   462 ;
0000   463 ;                      - CI:
0000   464 ;                              <31:08> MBZ
0000   465 ;                              <07:00> HSC port number (station address)
0000   466 ;
0000   467 ;               R3     - boot device unit number
0000   468 ;
0000   469 ;               R4     - logical block number to boot from if bit 3 is set in R5
0000   470 ;                        (not supported on 11/750)
0000   471 ;
```

```
0000    472 ;       R5      - software boot control flags.  The value -1 is reserved.
0000    473 ;
0000    474 ;               Bit     Meaning
0000    475 ;               ---     -------
0000    476 ;
0000    477 ;                0      RPB$V_CONV.
0000    478 ;                       Conversational boot. At various points in the
0000    479 ;                       system boot procedure, the bootstrap code
0000    480 ;                       solicits parameters and other input from the
0000    481 ;                       console terminal. If the DIAG is also on, then
0000    482 ;                       the diagnostic supervisor should enter "MENU"
0000    483 ;                       mode and prompt user for devices to test.
0000    484 ;
0000    485 ;                1      RPB$V_DEBUG.
0000    486 ;                       Debug. If this flag is set, VMS maps the code
0000    487 ;                       for the XDELTA debugger into the system page
0000    488 ;                       tables of the running system.
0000    489 ;
0000    490 ;                2      RPB$V_INIBPT.
0000    491 ;                       Initial breakpoint. If RPB$V_DEBUG is set, VMS
0000    492 ;                       executes a BPT instruction immediately after
0000    493 ;                       enabling mapping.
0000    494 ;
0000    495 ;                3      RPB$V_BBLOCK.
0000    496 ;                       Secondary boot from boot block. Secondary
0000    497 ;                       bootstrap is a single 512-byte block, whose
0000    498 ;                       LBN is specified in R4.
0000    499 ;
0000    500 ;                4      RPB$V_DIAG.
0000    501 ;                       Diagnostic boot. Secondary bootstrap is image
0000    502 ;                       called [SYSMAINT]DIAGBOOT.EXE.
0000    503 ;
0000    504 ;                5      RPB$V_BOOBPT.
0000    505 ;                       Bootstrap breakpoint. Stops the primary
0000    506 ;                       and secondary bootstraps with a breakpoint
0000    507 ;                       instruction before testing memory.
0000    508 ;
0000    509 ;                6      RPB$V_HEADER.
0000    510 ;                       Image header. Takes the transfer address of the
0000    511 ;                       secondary bootstrap image from that file's
0000    512 ;                       image header. If RPB$V_HEADER is not set,
0000    513 ;                       transfers control to the first byte of the
0000    514 ;                       secondary boot file.
0000    515 ;
0000    516 ;                7      RPB$V_NOTEST.
0000    517 ;                       Memory test inhibit. Sets a bit in the PFN bit
0000    518 ;                       map for each page of memory present. Does not
0000    519 ;                       test the memory.
0000    520 ;
0000    521 ;                8      RPB$V_SOLICT.
0000    522 ;                       File name. VMB prompts for the name of a
0000    523 ;                       secondary bootstrap file.
0000    524 ;
0000    525 ;                9      RPB$V_HALT.
0000    526 ;                       Halt before transfer. Executes a HALT
0000    527 ;                       instruction before transferring control to the
0000    528 ;                       secondary bootstrap.
```

```
0000   529 ;
0000   530 ;                10      RPB$V_NOPFND.
0000   531 ;                        No PFN deletion (not implemented; intended to
0000   532 ;                        tell VMB not to read a file from the boot device
0000   533 ;                        that identifies bad or reserved memory pages,
0000   534 ;                        so that VMB does not mark these pages as valid
0000   535 ;                        in the PFN bitmap).
0000   536 ;
0000   537 ;                11      RPB$V_MPM.
0000   538 ;                        Specifies that multi-port memory is to be used
0000   539 ;                        for the total exec memory requirement.  No local
0000   540 ;                        memory is to be used.  This is for tightly-coupled
0000   541 ;                        multi-processing.
0000   542 ;
0000   543 ;                12      RPB$V_USEMPM.
0000   544 ;                        Specifies that multi-port memory should be used in
0000   545 ;                        addition to local memory, as though both were one
0000   546 ;                        single pool of pages.
0000   547 ;
0000   548 ;                13      RPB$V_MEMTEST
0000   549 ;                        Specifies that a more extensive algorithm be used
0000   550 ;                        when testing main memory for hardware uncorrectable
0000   551 ;                        (RDS) errors.
0000   552 ;
0000   553 ;                14      RPB$V_FINDMEM
0000   554 ;                        Requests use of MA780 memory if MS780 is insufficient
0000   555 ;                        for booting.  Used for 11/782 installations.
0000   556 ;
0000   557 ;                15      RPB$V_AUTOTEST
0000   558 ;                        Used by Diagnostic Supervisor.
0000   559 ;
0000   560 ;                16      RPB$V_CRDTEST
0000   561 ;                        Specifies that memory pages with correctable (CRD)
0000   562 ;                        errors NOT be discarded at bootstrap time.  By default,
0000   563 ;                        pages with CRD errors are removed from use during the
0000   564 ;                        bootstrap memory test.
0000   565 ;
0000   566 ;                <31:28> RPB$V_TOPSYS
0000   567 ;                        Specifies the top level directory number for system
0000   568 ;                        disks with multiple systems
0000   569 ;
0000   570 ;       The hardware or the CONSOLE program sets up the next 3 registers
0000   571 ;       after a system crash or power failure:
0000   572 ;
0000   573 ;       R10     - halt PC
0000   574 ;       R11     - halt PSL
0000   575 ;       AP      - halt code
0000   576 ;
0000   577 ;       SP      - <base_address + ^X200> of 64kb of good memory
0000   578 ;
0000   579 ; Implicit inputs:
0000   580 ;
0000   581 ;       When VMB gains control, physical memory looks like the diagram
0000   582 ;       below:
0000   583 ;
0000   584 ;       SP-^X200:       +---------------------------------------+
0000   585 ;                       !     Restart Parameter Block (RPB)     !
```

```
                                                H 15
0000    586 ;       SP:             +---------------------------------------+
0000    587 ;                       |       Primary bootstrap (VMB)         |
0000    588 ;                       +---------------------------------------+
0000    589 ;
0000    590 ; Outputs:
0000    591 ;
0000    592 ;       R10     - base address of region containing secondary bootstrap
0000    593 ;       R11     - address of restart parameter block
0000    594 ;       SP      - current stack pointer
0000    595 ;       PR$_SCBB- system control block address
0000    596 ;
0000    597 ; Implicit outputs:
0000    598 ;
0000    599 ;       When VMB transfers control to the secondary bootstrap, physical
0000    600 ;       memory is laid out as in the diagram below:
0000    601 ;
0000    602 ;       RPB$L_BASE:     +---------------------------------------+
0000    603 ;                       |     Restart Parameter Block (RPB)     |
0000    604 ;       base+^X200:     +---------------------------------------+
0000    605 ;                       |       Primary bootstrap (VMB)         |
0000    606 ;       PR$_SCBB:       +---------------------------------------+
0000    607 ;                       |      System control block (SCB)       |
0000    608 ;                       +---------------------------------------+
0000    609 ;                       |              PFN Bitmap               |
0000    610 ;       PFNMAP+^X800:   +---------------------------------------+
0000    611 ;                       |                 Stack                 |
0000    612 ;       SP:             +---------------------------------------+
0000    613 ;                       |     Secondary bootstrap (SYSBOOT)     |
0000    614 ;                       +---------------------------------------+
0000    615 ;
0000    616 ; The design for the PFN bitmap has been extended to handle more than
0000    617 ; 4 pages of bitmap = 8mb of memory.  Bitmaps that do not fit in the
0000    618 ; 4 page reserved area are now allocated contiguous good pages in
0000    619 ; higher memory.  Assuming that the pages are actually good, the
0000    620 ; bitmap is placed at the RPB address + 1mb.  If a page in that area is
0000    621 ; bad, then the next contiguous run of pages that is big enough is
0000    622 ; where the bitmap will be placed.  For backward compatibility, the
0000    623 ; RPB$Q_PFNMAP descriptor in the RPB points at the pre-allocated
0000    624 ; 4 page bitmap, which is correct for the low 8mb of memory.  The
0000    625 ; real descriptor for the bitmap is passed in the argument list.
0000    626 ;
0000    627 ;--
0000    628
0000    629 START_BOOT::                          ; Start of primary bootstrap.
0000    630
0000    631 ;
0000    632 ; Reserve space for a System Control Block (SCB) immediately after the
0000    633 ; VMB code. Write the address of a machine check fault handler in all
0000    634 ; vectors in the SCB. This handler is used during bootstrapping except
0000    635 ; when the bootstrap code specifically writes a different handler
0000    636 ; address in one of the SCB vectors.
0000    637 ;
0000    638 ; The low bit set in the address of the fault handler causes the handler
0000    639 ; to execute on the interrupt stack.
0000    640 ;
0000    641
0113'CF    D4   0000    642         CLRL    CONTINUE_INDEX          ; Indicate initial execution of VMB
```

```
                FFF9'  31  0004   643          BRW     START_BOOT_1            ; Continue in other psect
                       0007   644          SET_PSECT <YBTMEM,LONG>
                       0007
                       0007          ;
                       0007          ; ***** Change Program Section
                       0007          ;
                00000000             .PSECT  YBTMEM,LONG
                       0000
                       0000   645
                       0000   646 START_BOOT_1::                            ; Entry point for re-execution of VMB
                       0000   647
          56   0125'CF   9E  0000   648          MOVAB   BOOT_FAULT+1,R6         ; Get the address of a handler.
          57   0200'CF   9E  0005   649          MOVAB   BOOTRIGH+^X200,R7       ; Get the address of the first
                       000A   650                                               ; byte past the SCB.
                       000A   651
                       000A   652 ;
                       000A   653 ; Register usage in the loop below is as follows:
                       000A   654 ;
                       000A   655 ;       R6      - address (+1) of a machine check handler
                       000A   656 ;       R7      - address of 1 byte past a longword of SCB
                       000A   657 ;
                       000A   658
                       000A   659 FILL_SCB:                                   ; Fill SCB vectors.
             77   56   D0  000A   660          MOVL    R6,-(R7)                ; Write 1 vector.
          57   01FF 8F   B3  000D   661          BITW    #^X1FF,R7               ; Check for page boundary.
                  F6   12  0012   662          BNEQ    FILL_SCB                ; Need to w  te another vector.
                       0014   663
                       0014   664 ;
                       0014   665 ; Write the address of the SCB into the SCB processor register.
                       0014   666 ;
                       0014   667
             11   57   DA  0014   668          MTPR    R7,#PR$_SCBB            ; Load SCB processor register.
                       0017   669
                       0017   670 ;
                       0017   671 ; Read the system identification processor register to discover which
                       0017   672 ; kind of VAX is to be booted.
                       0017   673 ;
                       0017   674
                58   3E   DB  0017   675          MFPR    #PR$_SID,R8             , Read the CPU identification
                       001A   676                                               ; processor register.
          0000'CF   58   D0  001A   677          MOVL    R8,EXE$GB_CPUDATA       ; Save contents of SID.
       5E    58   E8 8F   78  001F   678          ASHL    #-PR$V_SID_TYPE,R8,R8   ; Get CPU identification code.
          0000'CF   58   90  0024   679          MOVB    R8,EXE$GB_CPUTYPE       ; Save processor code globally.
                       0029   680
                       0029   681          CPUDISP <<780,VALID_PROCID>,-    ; Test for VAX-11/780
                       0029   682                   <750,VALID_PROCID>,-    ;  and VAX-11/750
                       0029   683                   <730,VALID_PROCID>,-    ;  and VAX-11/730
                       0029   684                   <790,VALID_PROCID>,-    ;  and VAX-11/790
                       0029   685                   <8SS,VALID_PROCID>>,-   ;  and VAX-11/8SS
                       0029   686                  ENVIRON=VMB;
                       0058   687
                       0058   688 ;
                       0058   689 ; Processor is of known type.
                       0058   690 ;
                       0058   691
                       0058   692 VALID_PROCID:                              ; Continue after determining valid CPU.
                       0058   693
```

```
                                0058   694 ;
                                0058   695 ; Now set up temporary values for the time-wait macros, used only by
                                0058   696 ; the boot drivers.  Later on in the boot process actual values will
                                0058   697 ; be computed.
                                0058   698 ;
0000'CF   06A3'CF48   9A        0058   699         MOVZBL  TENUSECTBL-1[R8],EXE$GL_TENUSEC ; Ten micro-sec loop counter
0000'CF   06AB'CF48   9A        0060   700         MOVZBL  UBDELAY-1[R8],EXE$GL_UBDELAY ; Unibus delay loop counter
                                0068   701
                                0068   702 ;
                                0068   703 ; If the DEBUG flag is defined (meaning that XDELTA has been linked
                                0068   704 ; with this primary bootstrap), set up 2 XDELTA handlers in the SCB --
                                0068   705 ; one for breakpoints and one for tbit traps. Then initialize the
                                0068   706 ; XDELTA breakpoint table, allocate 3 pages of stack, and, if requested,
                                0068   707 ; execute a breakpoint before proceeding with the bootstrap.
                                0068   708 ;
                                0068   709
                                0068   710         .IF     DF,DEBUG                        ; If debugging is going on,
  2C A7   0000'CF   9E          0068   711         MOVAB   XDELBPT,^X2C(R7)                ; Set up BPT handler.
  28 A7   0000'CF   9E          006E   712         MOVAB   XDELTBIT,^X28(R7)               ; Set up TBIT handler.
0000'CF   0087'CF   9E          0074   713         MOVAB   INI$BRK,XDELIBRK                ; Store the initial breakpoint.
          56    5E  D0          007B   714         MOVL    SP,R6                           ; Save current top of stack.
          1000 C7   9E          007E   715         MOVAB   <<^X200>+<BITMAP_PAG_CNT*^X200>+<STACK_PAG_CNT*^X200>>(R7),-
                5E              0082   716                 SP                              ; Create a stack above the SCB and
                                0083   717                                                 ; pre-allocated bitmap.
        01 55   05              0083   718         BBC     #RPB$V_BOOBPT,R5,-              ; If no BPT was requested in the
                                0087   719                 NOBRK                           ; boot flags, just proceed.
                                0087   720
                                0087   721 ;
                                0087   722 ; Initial breakpoint.
                                0087   723 ;
                                0087   724 ; Current register status is as follows:
                                0087   725 ;
                                0087   726 ;       R0-R5   - initial VMB input values
                                0087   727 ;       R6      - SP value at start of VMB
                                0087   728 ;       R7      - address of the SCB
                                0087   729 ;       R8      - processor identification code
                                0087   730 ;       R9-FP   - initial VMB input values
                                0087   731 ;       SP      - address of a 3-page stack
                                0087   732 ;
                                0087   733 ;
                                0087   734
                                0087   735 INI$BRK::                                     ; Debugging breakpoint.
03                              0087   736         BPT                                     ; Stop in XDELTA.
                                0088   737
                                0088   738 NOBRK:                                        ; Proceed with bootstrapping.
                                0088   739         .ENDC                                   ; End of debug conditional.
                                0088   740
```

```
                        0088   742          .SBTTL  Initialize RPB
                        0088   743
                        0088   744 ;++
                        0088   745 ;
                        0088   746 ; The next section of VMB writes the restart parameters into the RPB.
                        0088   747 ; These parameters include
                        0088   748 ;
                        0088   749 ;       the VMB input registers
                        0088   750 ;       the base address of the RPB
                        0088   751 ;       the boot device's CSR
                        0088   752 ;       the boot device adapter's configuration register (CR)
                        0088   753 ;       the address of the PFN bitmap
                        0088   754 ;       the address and length of the bootst' p device driver
                        0088   755 ;
                        0088   756 ; Current register settings are as follows:
                        0088   757 ;
                        0088   758 ;       R0-R5   - unchanged from time that VMB gained control
                        0088   759 ;       R6      - SP value from time that VMB gained control
                        0088   760 ;       R7      - address of the SCB
                        0088   761 ;       R8      - processor identification code
                        0088   762 ;       R9-FP   - unchanged from time that VMB gained control
                        0088   763 ;       SP      - top of VMB's stack
                        0088   764 ;
                        0088   765 ;--
                        0088   766
                        0088   767 ;
                        0088   768 ; Store the boot parameters in the RPB, as well as the RPB starting
                        0088   769 ; address. CONSOLE-controlled restart code looks for the starting
                        0088   770 ; address contained within itself as a clue to VMS restartability.
                        0088   771 ;
                        0088   772
        7E   5B   D0    0088   773          MOVL    R11,-(SP)                       ; Save Halt PSL temporarily
    5B  FE00 C6   9E    008B   774          MOVAB   -^X200(R6),R11                  ; Calculate base address of RPB.
      14 AB   8E   D0   0090   775          MOVL    (SP)+,RPB$L_HALTPSL(R11)        ; Save Halt PSL in RPB
      10 AB   5A   D0   0094   776          MOVL    R10,RPB$L_HALTPC(R11)           ; Save Halt PC in RPB
    01F8'CF   5B   D0   0098   777          MOVL    R11,BOO$GL_RPBBASE              ; Save address for callback.
      6B   5B   D0      009D   778          MOVL    R11,RPB$L_BASE(R11)             ; Save address in RPB.
      1C AB   50   7D   00A0   779          MOVQ    R0,RPB$L_BOOTR0(R11)            ; Save boot registers R0-R1.
      24 AB   52   7D   00A4   780          MOVQ    R2,RPB$L_BOOTR2(R11)            ; Save boot registers R2-R3.
0108  CB   27 AB   90   00A8   781          MOVB    RPB$L_BOOTR2+3(R11), -          ; Move high byte of BOOTR2 to
                        00AE   782                  RPB$B_CTRLLIR(R11)              ; another RPB location.
        27 AB   94   00AE   783          CLRB    RPB$L_BOOTR2+3(R11)             ; Clear the high byte (for compatibility).
      2C AB   54   7D   00B1   784          MOVQ    R4,RPB$L_BOOTR4(R11)            ; Save boot registers R4-R5.
        0000'CF   DE   00B5   785          MOVAL   BOO$AL_VECTOR,-                 ; Save address of boot device
          34 AB         00B9   786                  RPB$L_IOVEC(R11)                ; driver
          38 AB   D4    00BB   787          C RL    RPB$L_IOVECSZ(R11)              ; Correct size of boot-driver wil be
                        00BE   788                                                  ; set by the first call to BOO$QIO
        00A3 CB   94    00BE   789          CLRB    RPB$B_FLAGS(R11)                ; Initialize flags field
      18 AB   5C   D0   00C2   790          MOVL    AP,RPB$L_HALTCODE(R11)          ; Save the halt code.
      5C AB   01   CE   00C6   791          MNEGL   #1,RPB$L_ADPPHY(R11)            ; Initialize adapter physical address
      54 AB   01   CE   00CA   792          MNEGL   #1,RPB$L_CSRPHY(R11)            ; Initialize CSR physical address
                        00CE   793
                        00CE   794 ;
                        00CE   795 ; Move device characteristics from the input registers into the RPB.
                        00CE   796 ;
                        00CE   797
      64 AB   53   B0   00CE   798          MOVW    R3,RPB$W_UNIT(R11)              ; Save the device unit number.
```

```
                66 AB    50    90  00D2    799          MOVB    R0,RPB$B_DEVTYP(R11)    ; Save the device type.
                                 00D6    800
                                 00D6    801 ;
                                 00D6    802 ; From this point on AP contains the address of the Secondary Boot
                                 00D6    803 ; argument list.
                                 00D6    804 ;
                                 00D6    805
              5C    01AB'CF    DE  00D6    806          MOVAL   SECOND_PARAM,AP         ; Address of Secondary Boot arg list
                                 00DB    807
                                 00DB    808 ;
                                 00DB    809 ; Calculate the address of the CPU-specific table. CPU_CODES is a list
                                 00DB    810 ; of offsets from start of CPU_CODES to start of a CPU-specific data
                                 00DB    811 ; table.
                                 00DB    812 ;
                                 00DB    813
              58    0670'CF48    3C  00DB    814          MOVZWL  CPU_CODES[R8],R8        ; For a CPU, get offset to CPU
                                 00E1    815                                          ; data from CPU-list table.
              58    0670'CF48    9E  00E1    816          MOVAB   CPU_CODES[R8],R8        ; Add offset to address of
                                 00E7    817                                          ; CPU-list table.
                                 00E7    818
                                 00E7    819 ;
                                 00E7    820 ; Reserve space for an PFN bitmap that will desecribe all of physical
                                 00E7    821 ; memory. Store the address and size of the PFN bitmap in the RPB.
                                 00E7    822 ;
                                 00E7    823
                   59    48 AB    9E  00E7    824          MOVAB   RPB$Q_PFNMAP+4(R11),R9  ; Get pointer to RPB bitmap
                                 00EB    825                                          ; address field.
                69    0200 C7    9E  00EB    826          MOVAB   ^X200(R7),(R9)          ; Store PFN bitmap address
                                 00F0    827                                          ; (1 page past SCB base address).
            79    00000800 8F    D0  00F0    828          MOVL    #BITMAP_PAG_CNT*512,-(R9); Store bitmap size in RPB.
                     1000 C7    9E  00F7    829          MOVAB   <<^X200>+<BITMAP_PAG_CNT*^X200>+<STACK_PAG_CNT*^X200>>(R7),-
                          5A         00FB    830                  R10                     ; Adjust pointer to 1st unused
                                     00FC    831                                          ; byte of free memory (past the SCB
                                     00FC    832                                          ; and the PFN bitmap and the stack).
                                     00FC    833 ;
                                     00FC    834 ; Now continue execution of VMB in the appropriate place, based on
                                     00FC    835 ; whether this is the first execution of VMB (normal boot procedure)
                                     00FC    836 ; or whether this is a recursive execution of VMB (as requested via
                                     00FC    837 ; RPB$V_FINDMEM flag).
                                     00FC    838 ;
            56    010D'CF    9E  00FC    839          MOVAB   CONTINUE_TBL,R6         ; Get adr of continuation code adrs
            50    0113'CF    D0  0101    840          MOVL    CONTINUE_INDEX,R0       ; Get continuation index code
               50    6640    32  0106    841          CVTWL   (R6)[R0],R0             ; Calculate address of appropriate code
                     6640    17  010A    842          JMP     (R6)[R0]                ; Continue executing where left off
                                 010D    843
                                 010D    844 ;
                                 010D    845 ; Table of continuation addresses for continuation of
                                 010D    846 ; algorithm that uses MA780 memory instead of MS780 memory
                                 010D    847 ; for booting.  (Requested via RPB$V_FINDMEM flag.)  Each
                                 010D    848 ; entry in this table points to the code at which execution
                                 010D    849 ; continues after VMB has been moved into memory on a different
                                 010D    850 ; memory controller.  This is to allow the physical addresses
                                 010D    851 ; of memory to be altered by VMB.
                                 010D    852 ;
                                 010D    853 CONTINUE_TBL:                            ; Table of continuation addresses
                                 010D    854          RELADR  NORMAL_PATH,CONTINUE_TBL ; Path for normal VMB execution
                                 010F    855          RELADR  CONT1_PATH,CONTINUE_TBL ; Continuation of VMB in MA780 memory
```

```
                0111   856          RELADR  CONT2_PATH,CONTINUE_TBL ; Continuation of VMB in MS780 memory
                0113   857
                0113   858 CONTINUE_INDEX::                        ; Index into continuation table
00000000        0113   859          .LONG   0                      ; Normal VMB path index is 0
                0117   860 NORMAL_PATH:
```

```
                                0117   862          .SBTTL  Locate and test memory
                                0117   863
                                0117   864 ;++
                                0117   865 ;
                                0117   866 ; The next section of the primary bootstrap program determines and
                                0117   867 ; records the size and pattern of available memory in a PFN bitmap.
                                0117   868 ; It does this by calling a processor specific subroutine linked with VMB.
                                0117   869 ; Each subroutine handles the sizing of memory and the testing of the
                                0117   870 ; pages.  A common routine (BOO$TEST_MEM) is called to allocate and
                                0117   871 ; initialize bitmap segments, and loop through the page testing.
                                0117   872 ; When the sizing and testing are complete, the resulting bitmap is
                                0117   873 ; neither contiguous nor necessarily dense.  There could be ranges of
                                0117   874 ; PFN's that are simply not present and will of course be assumed
                                0117   875 ; to be bad.  At this point a contiguous chunk of memory is allocated
                                0117   876 ; and a clean, dense, contiguous bitmap is formed from the pieces.
                                0117   877 ;
                                0117   878 ; Current register settings are the following:
                                0117   879 ;
                                0117   880 ;       R0-R6   - scratch
                                0117   881 ;       R7      - address of the SCB
                                0117   882 ;       R8      - address of the CPU-specific table
                                0117   883 ;       R9      - address of RPB$Q_PFNMAP
                                0117   884 ;       R10     - address of 1st byte of unused good memory
                                0117   885 ;       R11     - address of the RPB
                                0117   886 ;       AP      - address of secondary boot argument list
                                0117   887 ;       FP      - scratch
                                0117   888 ;       SP      - address of a 3 page stack
                                0117   889 ;
                                0117   890 ; In previous versions of VMB, the RPB CONFREG field was filled in with
                                0117   891 ; a code indexed by TR number which identifies the adapter type of each
                                0117   892 ; NEXUS.  This is still done for 11/780 and 11/750 for backward compatibility
                                0117   893 ; of VMB.  For 11/730 and 11/790, the CONFREG array is filled in by INITADP
                                0117   894 ; at a later stage of bootstrapping.
                                0117   895 ;
                                0117   896 ;--
                                0117   897
                                0117   898 LOC_TEST_MEM:
                                0117   899
                                0117   900 ;
                                0117   901 ; No assembly time initialization is allowed in VMB because the
                                0117   902 ; ERROR logic allows it to restart.
                                0117   903 ;
                          10 AC   D4    0117   904          CLRL    VMB$L_HI_PFN(AP)          ; Init BITMAP data for
                    0C AC 01  15  9C    011A   905          ROTL    #21,#T,VMB$L_LO_PFN(AP)   ; highest and lowest PFN
                       01E8'CF    D4    011F   906          CLRL    BITMAP_HI_INDX            ; Highest bitmap page index
                          50 5B  17  9C    0123   907          ROTL    #<32-95>,R11,R0           ; Base PFN for large bitmaps
                01EC'CF   1000 C0  DE    0127   908          MOVAL   4096(R0),BITMAP_BAS_PFN   ; is 2mb beyond RPB
                                012E   909 ;
                                012E   910 ; Initialize the pre-allocated PFN bitmap pages to zero.  There is
                                012E   911 ; at least one such page to hold the bitmap for the pre-tested
                                012E   912 ; block of memory in which we are running.  The size of this
                                012E   913 ; pre-allocated bitmap is an integral number of pages.
                                012E   914 ;
                                012E   915
                             56 04 A9   D0    012E   916          MOVL    4(R9),R6                  ; Get base of PFN bitmap.
66 69 00 66 00 2C               0132   917          MOVC5   #0,(R6),#0,(R9),(R6)      ; Set bitmap to all zeroes.
0040 8F 00 56 00 2C             0138   918          MOVC5   #0,(R6),#0,#<512*64>,-    ; Zero fill the memory descriptors in
```

```
                        00BC CB         0'3F
                                        0142   919                  #<RPB$C_MEMDSCSIZ*RPB$C_NMEMDSC>,RPB$L_MEMDSC(R11) ; the RPB
                          4C AB    D4   0142   920          CLRL    RPB$L_PFNCNT(R11)          ; Init the count of good pages
                        0104 CB    D4   0145   921          CLRL    RPB$L_BADPGS(R11)          ; Init the count of bad pages
                  01E4'CF    5A    D0   0149   922          MOVL    R10,BITMAP_VEC_PTR         ; Address of vector of bitmap addresses
6A   0800 8F    FF 8F    6A    00    2C   014E   923          MOVC5   #0,(R10),#-1,#512*4,(R10) ; Init bitmap address vector.
                                        0157   924 ;
                                        0157   925 ; A side effect of the above MOVC5 is that R1 = R10
                                        0157   926 ;
                          5A    53    D0   0157   927          MOVL    R3,R10                     ; Next free address (page bounded)
                       53    04 A9    D0   015A   928          MOVL    4(R9),R3                   ; Address of pre-allocated bitmap
                          52    04    D0   015E   929          MOVL    #BITMAP_PAG_CNT,R2         ; Number of pages of pre-alloc bitmap
                          81    53    D0   0161   930 10$:     MOVL    R3,(R1)+                   ; Store address of bitmap page
                    53    0200 C3    DE   0164   931          MOVAL   ^X200(R3),R3               ; Next pre-allocated bitmap page
                       F5 52    F5    0169   932          SOBGTR  R2,10$                     ; Loop through pre-allocated page(s)
                                        016C   933
                                        016C   934 ;
                                        016C   935 ; Allow the bitmap page containing the RPB to be above the pre-allocated
                                        016C   936 ; (8mb max) PFN bitmap.
                                        016C   937 ;
                50    5B    EB 8F    78   016C   938          ASHL    #-<9+12>,R11,R0            ; Bitmap index for RPB
                     01E4'DF40    D5   0171   939          TSTL    @BITMAP_VEC_PTR[R0]        ; Need a bitmap page for this
                                        0176   940                                             ; portion of the PFN bitmap?
                          11    18    0176   941          BGEQ    20$                        ; Branch if not
                  01E4'DF40    5A    D0   0178   942          MOVL    R10,@BITMAP_VEC_PTR[R0]    ; Use one more pre-tested page
   6A   0200 8F    00    6A    00    2C   017E   943          MOVC5   #0,(R10),#0,#512,(R10)     ; as the bitmap page
                          5A    53    D0   0186   944          MOVL    R3,R10                     ; Keep track of next free address
                                        0189   945 20$:
                                        0189   946 ;
                                        0189   947 ; In the 11/780 and 11/750 CPU implementations, the memory test routines
                                        0189   948 ; check up to 16 slots (positions)  on the system bus for adapters or memory
                                        0189   949 ; controllers. The 16-byte field RPB$B_CONFREG in the RPB stores each
                                        0189   950 ; type of adapter/controller as the bootstrap finds it. Set up registers
                                        0189   951 ; to facilitate this loop in the CPU-specific routines.
                                        0189   952 ; In the 11/730 and 11/790, the CONFREG field is not filled in by VMB, but
                                        0189   953
                                        0189   954 ; initialized in INITADP at a later stage of bootstrapping.  The difference
                                        0189   955 ; is to maintain backward compatibility of VMB for currently supported systems.
                                        0189   956 ;
                                        0189   957
        55    0090 CB    9E            0189   958          MOVAB   RPB$B_CONFREG(R11),R5      ; Point to start of
                                        018E   959                                             ; configuration field.
                                        018E   960
                                        018E   961 ;
                                        018E   962 ; Now call the CPU-specific routine to locate and test memory.
                                        018E   963 ;
                                        018E   964
              1600 8F          BB      018E   965          PUSHR   #^M<R9,R10,AP>             ; Save some registers
           50   02 A8          32      0192   966          CVTWL   CPU_W_CHECKMEM(R8),R0      ; Calculate address of a
                 6840          16      0196   967          JSB     (R8)[R0]                   ; CPU-specific memory test
                                        0199   968                                             ; routine and call it.
              1600 8F          BA      0199   969          POPR    #^M<R9,R10,AP>             ; Restore saved registers
        03    01E8'CF          D1      019D   970          CMPL    BITMAP_HI_INDX,#BITMAP_PAG_CNT-1 ; All bitmap pre-allocated?
                    0E          14      01A2   971          BGTR    ALLOC_BITMAP               ; Branch if not
50      01E8'CF    01          C1      01A4   972          ADDL3   #1,BITMAP_HI_INDX,R0       ; Actual number of the pre-allocated
                                        01AA   973                                             ; bitmap pages that were used
  44 AB    50    09          78      01AA   974          ASHL    #9,R0,RPB$Q_PFNMAP(R11)    ; Record possibly smaller descriptor
```

```
                00AD   31  01AF   975          BRW     BITMAP_IS_OK            ; Yes, don't need to move it
                       01B2   976
                       01B2   977 ;
                       01B2   978 ; Need to allocate a contiguous PFN bitmap and move the scattered
                       01B2   979 ; bitmap pages into it.
                       01B2   980 ;
                       01B2   981 ; The starting point for trying to allocate is 4096 pages (2mb) after
                       01B2   982 ; the RPB.  The only pages that can be in use here that appear to be
                       01B2   983 ; good and usable are the bitmap pages themselves when they are
                       01B2   984 ; allocated as the first good PFN in a given bitmap page.
                       01B2   985 ;
                       01B2   986
                       01B2   987 ALLOC_BITMAP:
      52  01EC'CF  01  C3  01B2   988          SUBL3   #1,BITMAP_BAS_PFN,R2    ; Desired location for bitmap
                   55  D4  01B8   989          CLRL    R5                      ; Init "first alloc failure" flag
                   5E  11  01BA   990          BRB     45$
          29 52    10  AC  F2  01BC   991 10$:     AOBLSS  VMB$L_HI_PFN(AP),R2,20$ ; Calculate the next PFN
                       01C1   992          ERROR   </%BOOT-F-Failed to allocate PFN bitmap/>
      50  52   0C  00  EF  01EA   993 20$:     EXTZV   #0,#12,R2,R0            ; Bitmap page relative PFN
      51  52   09  0C  EF  01EF   994          EXTZV   #12,#9,R2,R1            ; Index to bitmap page
        51  01E4'DF  41  D0  01F4   995          MOVL    @BITMAP_VEC_PTR[R1],R1  ; Address of bitmap page
                   07  18  01FA   996          BGEQ    30$                     ; Branch if present
          52   0FFF  8F  A8  01FC   997          BISW    #^XFFF,R2               ; Skip this entire bitmap page
                   0D  11  0201   998          BRB     40$
            09 61      50  E1  0203   999 30$:     BBC     R0,(R1),40$             ; Branch if page is bad
          50   52      09  78  0207  1000          ASHL    #9,R2,R0                ; Address from PFN
               51      50  D1  020B  1001          CMPL    R0,R1                   ; Same as bitmap page adr?
                       14  12  020E  1002          BNEQ    50$                     ; Branch if not, its ok
                           0210  1003                                              ; Otherwise treat as a bad page
            06 55      00  E2  0210  1004 40$:     BBSS    #0,R5,45$               ; Branch if not first alloc failure
      52    54   01E8'CF  C1  0214  1005          ADDL3   BITMAP_HI_INDX,R4,R2    ; Skip the entire "pre-planned"
                           021A  1006                                              ; allocation area.  Some bitmap pages
                           021A  1007                                              ; may be in it, but in the wrong place
               54  52  01  C1  021A  1008 45$:     ADDL3   #1,R2,R4                ; Reset start point of search
      53       01E8'CF  01  C1  021E  1009          ADDL3   #1,BITMAP_HI_INDX,R3    ; Reset no. of pages to look for
                     95 53  F4  0224  1010 50$:     SOBGEQ  R3,10$                  ; Next page in contiguous cluster
                           0227  1011
                           0227  1012 ;
                           0227  1013 ; Now move all the bitmap pages into the contiguous space just found
                           0227  1014 ;
                           0227  1015
                           0227  1016 MOVE_BITMAP:
               53  54  09  78  0227  1017          ASHL    #9,R4,R3                ; Starting ad.    new bitmap
                 04 A9  53  D0  022B  1018          MOVL    R3,4(R9)                ; Record base adr of PFN bitmap
               56  01E4'CF  D0  022F  1019          MOVL    BITMAP_VEC_PTR,R6       ; Adr of array of bitmap page adrs
                   51  86  D0  0234  1020 10$:     MOVL    (R6)+,R1                ; Adr of next bitmap page
                       0A  18  0237  1021          BGEQ    20$                     ; Branch if page is allocated
  63  0200 8F  00  63  00  2C  0239  1022          MOVC5   #0,(R3),#0,#512,(R3)    ; No good pages in this 2mb
                       12  11  0241  1023          BRB     50$
                   53  51  D1  0243  1024 20$:     CMPL    R1,R3                   ; No move needed?
                       07  12  0246  1025          BNEQ    30$                     ; Branch if must move the page
               53  0200  C3  DE  0248  1026          MOVAL   512(R3),R3              ; Just adjust the bitmap adr
                       06  11  024D  1027          BRB     50$
          63   61  0200  8F  28  024F  1028 30$:     MOVC3   #512,(R1),(R3)          ; Move the bitmap page
                 DA 01E8'CF  F4  0255  1029 50$:     SOBGEQ  BITMAP_HI_INDX,10$      ; Loop through all bitmap pages
            69   53   04 A9  C3  025A  1030          SUBL3   4(R9),R3,(R9)           ; Store size of PFN bitmap
                           025F  1031 BITMAP_IS_OK:
```

```
                5A   01E4'CF   D0  025F  1032          MOVL    BITMAP_VEC_PTR,R10      ; Reset first free byte pointer
                                   C264  1033
```

```
                        0264  1035          .SBTTL  Load CI microcode
                        0264  1036 ;
                        0264  1037 ; Check for the presence of a CI port on the SBI.  If present try to locate
                        0264  1038 ; the appropriate microcode file on the console medium (CI780.BIN).  If found,
                        0264  1039 ; then load it into memory.  If not found and not booting from it, merely issue
                        0264  1040 ; a warning message and continue on - presumably some human will correct the
                        0264  1041 ; situation later.  If not found and booting, then give an error and HALT!
                        0264  1042 ;
                        0264  1043
                        0264  1044          .ENABLE LSB
          04    E0      0264  1045          BBS     #RPB$V_DIAG,-               ; If this is a DIAG SUPRV boot,
    03 30 AB              0266  1046                  RPB$L_BOOTR5(R11),2$        ;  then do NOT allocate.
        FD94'   30        0269  1047          BSBW    BOO$CACHE_ALLOC             ; Allocate the FILEREAD cache
 52   66 AB    9A        026C  1048 2$:      MOVZBL  RPB$B_DEVTYP(R11),R2        ; Hold the device type
 50   06 A8    32        0270  1049          CVTWL   CPU_W_FIND_CI(R8),R0        ; Get CPU-specific routine to find CI.
       6840    16        0274  1050          JSB     (R8)[R0]                    ; Call it.
         22    12        0277  1051          BNEQ    10$                         ; NEQ => CI port on the system.
    52   20    91        0279  1052          CMPB    #BTD$K_HSCCI,R2             ; Booting off the HSC/CI?
         03    13        027C  1053          BEQL    5$                          ; Yes, error
       018D    31        027E  1054          BRW     50$                         ; No, just leave quietly
                         0281  1055
                         0281  1056 5$:      ERROR   </%BOOT-F-No such device/>
                         029B  1057
      40 8F    90        029B  1058 10$:     MOVB    #BTD$K_CONSOLE,-            ; Temp set for reading from console
      66 AB              029E  1059                  RPB$B_DEVTYP(R11)           ;  the ucode file
53  0D0E'CF    9E        02A0  1060          MOVAB   CI_UCODEV,R3                ; Pick up the ucode vector
        0806    30        02A5  1061          BSBW    OPEN_UCODE_FILE             ; Find and open it
      15 50     E8        02A8  1062          BLBS    R0,30$                      ; Found it, continue
       015C     31        02AB  1063          BRW     40$                         ; Did not find it, but not booting
                         02AE  1064 ;
                         02AE  1065 ; Found the CI and the CI ucode.  Now determine the place in memory to read
                         02AE  1066 ; the code into as well as put the page table.  The search starts at place
                         02AE  1067 ; indicated in the following table and stops just short of the FILEREAD
                         02AE  1068 ; cache (which is why the cache-init was done earlier).  If not enough
                         02AE  1069 ; memory can be found, halt with a message.
                         02AE  1070 ;
                         02AE  1071 ; NOTE: If this table is changed, a corresponding table near the label
                         02AE  1072 ;       MEM_CACHE_TABLE needs to be checked for consistency.
                         02AE  1073 ;
                         02AE  1074 MEM_TAB:
                         02AE  1075 ;                       SIZE,START
                         02AE  1076 ;                       ----  -----
                         02AE  1077          MEM_TABLE   8192,512                ; More than 4 megabyte
                         02B4  1078          MEM_TABLE   2048,256                ; More than 1 megabyte
                         02BA  1079          MEM_TABLE      0,  0                ;
                         02C0  1080
          52    D0       02C0  1081 30$:     PUSHL   R2                          ; Save old devtype
    51    4C AB    D0     02C2  1082          MOVL    RPB$L_PFNCNT(R11),R1        ; Maximum memory
 51     0104 CB    C0     02C6  1083          ADDL    RPB$L_BADPGS(R11),R1        ; Add in the bad pages
    53    E0 AF    9E     02CB  1084          MOVAB   MEM_TAB,R3                  ; Addr of table
       54    83    D0     02CF  1085 32$:     MOVL    (R3)+,R4                    ; Get the next table entry
             39    13     02D2  1086          BEQL    34$                         ; Memory too small
       50    83    3C     02D4  1087          MOVZWL  (R3)+,R0                    ; Starting page number
       54    51    D1     02D7  1088          CMPL    R1,R4                       ; More memory than this entry?
             F3    19     02DA  1089          BLSS    32$                         ; Branch if not, get next entry
    51    7F A1    9E     02DC  1090          MOVAB   127(R1),R1                  ; Set to round up
51  51    F9 8F    78     02E0  1091          ASHL    #-7,R1,R1                   ; Number of pages of page table needed
```

```
                51    0204'CF   C0  02E5  1092          ADDL    CI_UCODE_STAT+4,R1      ; Add in the ucode length
                      54   51   D0  02EA  1093          MOVL    R1,R4                   ; Will not settle for less
             55   08 AC   09   78  02ED  1094          ASHL    #9,VMB$Q_FILECACHE+4(AP),R5; Start addr of FILEREAD cache
                          06   12  02F2  1095          BNEQ    60$                     ; Okay, if somethere there, else ..
          55   4C AB   FE  8F   78  02F4  1096          ASHL    #-2,RPB$L_PFNCNT(R11),R5 ; Use 1/4 of max as limit
                          04   E1  02FA  1097 60$:      BBC     #RPB$V_DIAG,-           ; If this is NOT a DIAG SUPRV boot,
                    09 30 AB       02FC  1098                   RPB$L_BOOTR5(R11),33$   ;  skip next
                50    0300 8F   3C  02FF  1099          MOVZWL  #768,R0                 ; Start at ^X60000
                  55    4C AB   D0  0304  1100          MOVL    RPB$L_PFNCNT(R11),R5    ; Go to top
                                    0308  1101 ;
                                    0308  1102 ; Attempt to allocate the pages necessary: A page table to encompass all of
                                    0308  1103 ; physical memory and the full microcode file.  The entire size must fit
                                    0308  1104 ; at a lower address than the FILEREAD cache, since this allocation cannot
                                    0308  1105 ; be either stepped on or deallocated until INIT time.
                                    0308  1106 ;
                        FCF5'   30  0308  1107 33$:      BSBW    BOO$ALLOC_PAGES         ; Grab the pages
                          26    18  030B  1108          BGEQ    35$                     ; Success
                                    030D  1109 34$:      ERROR   </%BOOT-F-Insufficient memory for CI/>
                                    0333  1110
          30 AC    53   52   C1  0333  1111 35$:      ADDL3   R2,R3,VMB$L_CI_HIPFN(AP); Set the highest PFN used
             56    53   09   78  0338  1112          ASHL    #9,R3,R6                ; Address to read into
               20 AC    56   D0  033C  1113          MOVL    R6,VMB$Q_UCODE+4(AP)    ; Store address away
                  7E    57   7D  0340  1114          MOVQ    R7,-(SP)                ; Save registers
                                    0343  1115 ;
                                    0343  1116 ; Now attempt to find and load the PCS file for 750's.
                                    0343  1117 ;
                    0000'CF   91  0343  1118          CMPB    EXE$GB_CPUTYPE,-        ; Check if this is 750
                          02        0347  1119                   #PR$_SID_TYP750
                          03   13   0348  1120          BEQL    64$                     ; Okay
                       00A1   31   034A  1121          BRW     80$                     ; No, so skip all this
                                    034D  1122
       5F 8F      0001'CF   91  034D  1123 64$:      CMPB    EXE$GB_CPUDATA+1,#^X5F  ; Is this a REV 95 ucode machine?
                          33   1E   0353  1124          BGEQU   65$                     ; Yes, continue
                                    0355  1125          ERROR   </%BOOT-F-Base CPU not at proper rev level for CI/>
                                    0388  1126
             53    0CFE'CF   9E  0388  1127 65$:      MOVAB   PCS_UCODEV,R3           ; Pick up ucode vector
                     071E   30   038D  1128          BSBW    OPEN_UCODE_FILE         ; Find and open it
             58    0CF6'CF   7D  0390  1129          MOVQ    PCS_UCODE_STAT,R8       ; Starting LBN and size
                      FD28'   30  0395  1130          BSBW    READFILE                ; Read it in
                                    0398  1131 ;
                                    0398  1132 ; This routine loads the Patchbits.  The Patchbits are extracted one bit at
                                    0398  1133 ; a time from longwords and written to the Patchbits.  To write to the
                                    0398  1134 ; Patchbits a '1' has to be written to the Patchbit Enable Register (CMI
                                    0398  1135 ; address F0C000). Upon completion of this routine the Patchbit Enable
                                    0398  1136 ; Register is cleared.
                                    0398  1137 ;
                   56   20 AC   D0  0398  1138          MOVL    VMB$Q_UCODE+4(AP),R6    ; Grab starting address
             54   00F00000 8F   D0  039C  1139          MOVL    #^X00F00000,R4          ; Physical address of patchbits
             00F0C000 9F   01   D0  03A3  1140          MOVL    #1,@#^X00F0C000         ; Enable writing to Patchbits
                      52   D4  03AA  1141          CLRL    R2                      ; Clear counter
           64   66   01   52   EF  03AC  1142 70$:      EXTZV   R2,#1,(R6),(R4)         ; Write Patchbits
                     54   04   C0  03B1  1143          ADDL2   #4,R4                   ; Next CMI location
        F0 52   00002000 8F   F2  03B4  1144          AOBLSS  #^X2000,R2,70$          ; Done ?
                00F0C000 9F   D4  03BC  1145          CLRL    @#^X00F0C000            ; Disable writing to Patchbits
                                    03C2  1146 ;
                                    03C2  1147 ; This routine loads PCS with the microcode.
                                    03C2  1148 ;
```

```
                                03C2  1149 ;       MOVL    #^X00F08000,R4          ; Map to PCS address on CMI
            56   0400 C6   9E  03C5  1150         MOVAB   ^X400(R6),R6            ; R6 -> start of PCS Ucode in buffer
            52   0400 8F   3C  03C7  1151         MOVZWL  #^X400,R2               ; Number of microwords
                       54   DD  03CC  1152         PUSHL   R4                      ; Save first address for later enable
         50 66   14    0C   EF  03CE  1153         EXTZV   #0,#20,(R6),R0          ; Save first 20 bits for later enable
                       51   D4  03D3  1154         CLRL    R1                      ; Clear counter
                  55   04   3C  03D5  1155 72$:    MOVZWL  #4,R5                   ; Short loop control
         84 66   14    51   EF  03D8  1156 74$:    EXTZV   R1,#20,(R6),(R4)+       ; Store one 20 bit unit
                  51   14   C0  03DD  1157         ADDL2   #20,R1                  ; Increment BIT position
                    F5 55   F5  03E0  1158         SOBGTR  R5,74$                  ; Finish one microword
                    EF 52   F5  03E3  1159         SOBGTR  R2,72$                  ; Finish all microwords
      9E 50   FFF00000 8F   C9  03E6  1160         BISL3   #^XFFF00000,R0,@(SP)+   ; Set and write bits
                                03EE  1161 ;
                                03EE  1162 ; Read in the CI ucode file into the allocated area.
                                03EE  1163 ;
            58   0200'CF   7D  03EE  1164 80$:    MOVQ    CI_UCODE_STAT,R8        ; Starting LBN and size
         1C AC   59    09   78  03F3  1165         ASHL    #9,R9,VMB$Q_UCODE(AP)   ; Store size away
               56   20 AC   D0  03F8  1166         MOVL    VMB$Q_UCODE+4(AP),R6    ; Grab starting address
                    FCC1'   30  03FC  1167         BSBW    READFILE                ; Read it in
                  57   8E   7D  03FF  1168         MOVQ    (SP)+,R7                ; Restore
                       52 8ED0  0402  1169         POPL    R2                      ; Restore old devtype
                       00   E2  0405  1170         BBSS    S^#VMB$V_LOAD_SCS,-     ; Tell SYSBOOT to load SCS code
                  00 2C AC      0407  1171                 VMB$L_FLAGS(AP),40$
               66 AB   52   90  040A  1172 40$:    MOVB    R2,RPB$B_DEVTYP(R11)    ; Restore the 'real' boot device
               50   34 AB   D0  040E  1173 50$:    MOVL    RPB$L_IOVEC(R11),R0     ; Pick up vector address
                  59   5B   D0  0412  1174         MOVL    R11,R9                  ; Transfer RPB address
                  18 B040   16  0415  1175         JSB     @BQO$L_MOVE(R0)[R0]     ; Move the boot driver
                                0419  1176         .DISABLE LSB
                                0419  1177
                                0419  1178 ;
                                0419  1179 ; Initialize the boot device's bus adapter.  If not console, call a
                                0419  1180 ; CPU-specific subroutine to  onvert this data into the physical
                                0419  1181 ; addresses of the adapter's configuration register (CR) and the
                                0419  1182 ; device's control/status register (CSR). Store the addresses in the
                                0419  1183 ; RPB.
                                0419  1184 ;
                                0419  1185
                                0419  1186 ADPINIT:                               ; Initialize adapter.
            66 AB   91   0419  1187         CMPB    RPB$B_DEVTYP(R11),-     ; Skip over if booting from
            40 8F        041C  1188                 #BTD$K_CONSOLE          ; console block storage device
               58   13   041E  1189         BEQL    15$
         51   20 AB   7D  0420  1190         MOVQ    RPB$L_BOOTR1(R11),R1    ; Pick up original R1/R2
            55   68   32  0424  1191         CVTWL   CPU_W_SAVE_CSRS(R8),R5  ; Calculate address of a
               6845   16  0427  1192         JSB     (R8)[R5]                ; CPU-specific routine to
                      042A  1193                                             ; calculate and save boot device
                      042A  1194                                             ; and adapter's CSRs in RPB.
            66 AB   91   042A  1195         CMPB    RPB$B_DEVTYP(R11),-     ; Skip over if booting from
               20        042D  1196                 #BTD$K_HSCCI            ;  HSC/CI adapter
                  0B   13   042E  1197         BEQL    10$
         50   5C AB   D0  0430  1198         MOVL    RPB$L_ADPPHY(R11),R0    ; Get ADP CSR address.
         51   04 A8   32  0434  1199         CVTWL   CPU_W_INIT_ADAP(R8),R1  ; Calculate address of a
               6841   16  0438  1200         JSB     (R8)[R1]                ; CPU-specific adapter init.
                      043B  1201                                             ; routine and call it.
         50   34 AB   D0  043B  1202 10$:    MOVL    RPB$L_IOVEC(R11),R0     ; Pick up vector address
            20 AC   D0  043F  1203         MOVL    VMB$Q_UCODE+4(AP),-
            28 A0        0442  1204                 BQO$L_UCODE(R0)         ; Transfer the ucode address
         51   1C A0   D0  0444  1205         MOVL    BQO$L_UNIT_INIT(R0),R1  ; Pick up any possible routine
```

```
                 2E   13  0448  1206          BEQL    15$                             ; None
          6041   6C   FA  044A  1207          CALLG   (AP),(R0)[R1]                   ; Do any necessary unit init
              27 50   E8  044E  1208          BLBS    R0,15$                          ; All is well
                          0451  1209          ERROP   </%BOOT-F-Failed to initialize device/>
                          0478  1210
                          0478  1211 15$:
                          0478  1212
                          0478  1213 ;
                          0478  1214 ; The adapter initialization routines exit with RSB instructions. Thus,
                          0478  1215 ; control returns here, and proceeds to obtain the secondary bootstrap
                          0478  1216 ; specification as described on the next page.
                          0478  1217 ;
```

```
                        .SBTTL  Identify and read in the secondary boot image

                ;++
                ;
                ; Current register settings are the following:
                ;
                ;       R0-R6   - scratch
                ;       R7      - address of the SCB
                ;       R8      - address of the CPU-specific table
                ;       R9      - scratch
                ;       R10     - address of the 1st byte of unused memory
                ;       R11     - address of the RPB
                ;       AP      - address of secondary boot parameter list
                ;       SP      - address of a 3 page stack
                ;
                ; Registers R0-R6 and R9 are available as scratch registers.
                ;
                ; The next 2 paragraphs refer to an 11/780-specific function. The
                ; function is historical; not implemented on later processors:
                ;
                ; The primary bootstrap now looks at a software bootstrap flag --
                ; RPB$V_BBLOCK. If the flag is set, the secondary bootstrap is a single
                ; block from the boot device. The LBN is stored in RPB$L_BOOTR4.
                ; Set up size of secondary bootstrap, transfer address, memory address,
                ; and LBN fields so that subsequent code can read the boot block into
                ; memory and transfer control to the code contained therein.
                ;
                ; If the RPB$V_BBLOCK flag is not set, the primary bootstrap must load
                ; a secondary bootstrap file specified by an ASCII file specification.
                ;
                ; If another boot flag -- RPB$V_SOLICT -- is set, the primary bootstrap
                ; asks the console user to type in a file specification.
                ;
                ; If the boot flag is not set, the bootstrap just selects a standard
                ; secondary bootstrap from the boot device. The bootstrap then calls I/O
                ; subroutines that locate the file, copy it into memory above the stack,
                ; and transfer control to the image's transfer address.
                ;
                ;--

                ;
                ; If we are booting from the console block storage device, it is
                ; necessary to switch the media to the system floppy or cartridge.
                ; Type a message on console and wait for carriage return.
                ;

  66 AB   91                    CMPB    RPB$B_DEVTYP(R11),-     ; Booting from console block
  40 8F                         #BTD$K_CONSOLE                  ; storage device?
     3C   12                    BNEQ    104$                    ; No
0C52'CF   9F                    PUSHAB  REMOVEPROMPT2           ; Pass first part of remove message
0C31'CF   9F                    PUSHAB  REMOVEPROMPT1           ; Pass second part of message
59   5E   D0                    MOVL    SP,R9                   ; Pass the address for the resume message
     32   10                    BSBB    105$                    ; Print the message
5E   08   C0                    ADDL2   #8,SP                   ; Pop the arguments
01F0'CF   9F            103$:   PUSHAB  INPBUF                  ; Push address of input buffer
     04   DD                    PUSHL   #4                      ; Push size of buffer
0BE7'CF   9F                    PUSHAB  SWITCHPROMPT            ; Push address of prompt string
```

```
                0000'CF    03   FB  0499  1276          CALLS   #3,BOO$READPROMPT        ; Prompt and wait for <ret>
                01F1'CF    20   8A  049E  1277          BICB    #32,INPBUF+1            ; Force answer to upper case (ASCIC string)
            59  8F  01F1'CF   91  04A3  1278          CMPB    INPBUF+1,#^A"Y"         ; We want to see a Yes
                       E4   12  04A9  1279          BNEQ    103$                    ; Not "Y", ask again
                  0C96'CF   9F  04AB  1280          PUSHAB  RESUMEPROMPT2           ; Pass first part of resume message
                  0C70'CF   9F  04AF  1281          PUSHAB  RESUMEPROMPT1           ; Pass second part of message
                       59   5E  D0  04B3  1282          MOVL    SP,R9                   ; Pass the address for the resume message
                            CS  10  04B6  1283          BSBB    105$                    ; Print the message
                       5E   08  C0  04B8  1284          ADDL2   #8,SP                   ; Pop the arguments
                          007A  31  04BB  1285  104$:     BRW     20$                     ; And continue
                                    04BE  1286
                                    04BE  1287 ;
                                    04BE  1288 ; Local subroutine to print a message with a volume label stuck in the
                                    04BE  1289 ; middle.
                                    04BE  1290 ; Input:       R9 -> two longwords containing the addresses of the front
                                    04BE  1291 ;                    and rear parts of the message (two ASCIZ strings)
                                    04BE  1292 ;
                                    04BE  1293  105$:                                   ; Push arguments for QIO.
                            5B  DD  04BE  1294          PUSHL   R11                     ; Push phony channel number.
                            00  DD  04C0  1295          P SHL   #^                      ; Physical read mode.
                            21  DD  04C2  1296          P               ADLBLK          ; Read logical block function.
                            01  DD  04C4  1297          P                               ; Starting LBN.
                     7E   0200  8F  3C  04C6  1298          MOVZWL          ,(SP)           ; Transfer size in bytes.
                            5A  DD  04CB  1299          PUSHL   R10                     ; Buffer address
                0000'CF     06  FB  04CD  1300          CALLS   #6,B^$QIO               ; Call a bootstrap QIO routine.
                          29 50  E8  04D2  1301          BLBS    R0,10$
                                    04D5  1302          ERROR   <%BOOT-F-Unable to read console volume/>
                            7E  7C  04FE  1303  101$:     CLRQ    -(SP)                   ; Null buffer means print only
                            89  DD  0500  1304          PUSHL   (R9)+                   ; Move address of first message
                0000'CF     03  FB  0502  1305          CALLS   #3,BOO$READPROMPT       ; Print the message
            55    01D8  CA  9E  0507  1306          MOVAB   472(R10),R5             ; Point R5 at the volume label
                  50    0C  D0  050C  1307          MOVL    #12,R0                  ; Length of volume label
                51    6540  9E  050F  1308          MOVAB   (R5)[R0],R1             ; Point R1 one past the end
                  52    71  90  0513  1309  201$:     MOVB    -(R1),R2                ; R2 contains the byte
                        05  13  0516  1310          BEQL    301$                    ; Byte is null, skip it
                  52    20  91  0518  1311          CMPB    #^A" ",R2               ; Is it a space
                        05  12  051B  1312          BNEQ    401$                    ; End of name found, print it
                     F3 50  F5  051D  1313  301$:     SOBGTR  R0,201$                 ; Do the whole string
                        0C  11  0520  1314          BRB     501$                    ; No name, don't bother not printing it
                                    0522  1315
                     01 A1  94  0522  1316  401$:     CLRB    1(R1)                   ; Put a null after the string
                        7E  7C  0525  1317          CLRQ    -(SP)                   ; Null buffer is print only
                        55  DD  0527  1318          PUSHL   R5                      ; R5 -> front of volume label
            0000'CF     03  FB  0529  1319          CALLS   #3,W^BOO$READPROMPT     ; Type the first string
                        7E  7C  052E  1320  501$:     CLRQ    -(SP)                   ; Null buffer means print only
                        69  DD  0530  1321          PUSHL   (R9)                    ; Pass address of second message
            0000'CF     03  FB  0532  1322          CALLS   #3,BOO$READPROMPT       ; Print the message
                            05  0537  1323          RSB
                                    0538  1324
                                    0538  1325 ;
                                    0538  1326 ; The next code paragraph is historical: refers only to the 11/780:
                                    0538  1327 ; If the boot block flag is set, prepare I/O input registers.
                                    0538  1328 ;
                                    0538  1329
                        03  E1  0538  1330  20$:     BBC     #RPB$V_BBLOCK,-         ; If the boot from boot block
                     30 AB          053A  1331                  RPB$L_BOOTR5(R11),-     ; flag is not set, proceed to
                        0D          053C  1332                  TEST_SOLICIT            ; test the solicit bit.
```

```
        55   02   D0  053D          MOVL    #2,R5                   ; block to go, and transfer
                      0540                                          ; address from start of block.
     58   2C AB   D0  0540          MOVL    RPB$L_BOOTR4(R11),R8    ; Get boot block LBN.
        59   01   D0  0544          MOVL    #1,R9                   ; Set block size to 1 block.
           0123   31  0547          BRW     READIN_BOOT             ; Proceed to read in block.
                      054A
                      054A  ;
                      054A  ; If the "solicit for secondary bootstrap file" flag is not set,
                      054A  ; just use a predefined file specification.
                      054A  ;
                      054A
                      054A  TEST_SOLICIT:                           ; Check for solicit.
             08   E1  054A          BBC     #RPB$V_SOLICT,-         ; If "solicit" flag is not
          30 AB       054C                  RPB$L_BOOTR5(R11),-     ; set, just use a default file
             17       054E                  DEFAULT_SECOND          ; specification.
                      054F
                      054F  ;
                      054F  ; To solicit a file name, call a device-independent subroutine that
                      054F  ; writes a prompt string to the console terminal, and then reads the
                      054F  ; user typed file name. All device specifications are ignored.
                      054F  ;
                      054F
          68 AB   9F  054F          PUSHAB  RPB$T_FILE(R11)         ; Set address of input buffer.
             27   DD  0552          PUSHL   #39                     ; Set maximum character count.
       0BC0'CF    9F  0554          PUSHAB  NAMEPROMPT              ; Set address of prompt string.
   0000'CF   03   FB  0558          CALLS   #3,BOO$READPROMPT       ; Prompt and read string.
        59   01   D0  055D          MOVL    #1,R9                   ; No retry if this file is not found
     57   68 AB   9E  0560          MOVAB   RPB$T_FILE(R11),R7      ; Get address of file name string
             3F   11  0564          BRB     FILEBOOT                ; Go try to read the file.
                      0566
                      0566  ;
                      0566  ; If the solicit boot flag was not set, use a default file name string.
                      0566  ; Usually, this file name is [SYSEXE]SYSBOOT.EXE. However, if the
                      0566  ; diagnostic boot flag is set, the file name is [SYSMAINT]DIAGBOOT.EXE.
                      0566  ;
                      0566
                      0566  DEFAULT_SECOND:                         ; Use default file name.
   57   0B8B'CF   9E  0566          MOVAB   VMSFILE,R7              ; Assume SYSBOOT.EXE.
             04   E1  056B          BBC     #RPB$V_DIAG,-           ; If the diagnostic flag is not
          30 AB       056D                  RPB$L_BOOTR5(R11),-     ; set, SYSBOOT is correct.
             05       056F                  COPY_NAME
   57   0B9F'CF   9E  0570          MOVAB   DIAGFILE,R7             ; Otherwise, use predefined
                      0575                                          ; name of diagnostic boot.
                      0575
                      0575  ;
                      0575  ; Copy the file name to the RPB.
                      0575  ;
                      0575
                      0575  COPY_NAME:                              ; Copy file name.
        04   1C   EF  0575          EXTZV   #RPB$V_TOPSYS,#RPB$S_TOPSYS,-
     59   30 AB       0578                  RPB$L_BOOTR5(R11),R9    ; Value of 0-F means top level
                      057B                                          ; system directory "SYS0" - "SYSF"
        09   59   D1  057B          CMPL    R9,#9                   ; 0 - 9 ?
             03   15  057E          BLEQ    10$                     ; Branch if yes
        59   07   LC  0580          ADDL    #<<^A/A/>-<^A/9/>-1>,R9 ; Add bias to make A - F
   0BBA'CF   59   80  0583  10$:    ADDB    R9,W^FIL$GT_TOPSYS+4    ; Form "SYSn"
        50   67   9A  0?88          MOVZBL  (R7),R0                 ; Size of name string
```

```
                68 AB    50    05    81  058B  1390          ADDB3   #5,R0,RPB$T_FILE(R11)   ; Add "SYSn" size and one for a dot
                  69 AB     5B 8F    90  0590  1391          MOVB    #^A/[/,RPB$T_FILE+1(R11); Start directory spec
                         0BB7'CF    D0  0595  1392          MOVL    W^FIL$GT_TOPSYS+1,-     ; Put top level sys name
                            6A AB        0599  1393                  RPB$T_FILE+2(R11)       ;
             6E AB    01 A7    50    28  059B  1394          MOVC3   R0,1(R7),RPB$T_FILE+6(R11); Move name into RPB
                         6E AB    2E    90  05A1  1395          MOVB    #^A/./,RPB$T_FILE+6(R11); Put in dot
                                        05A5  1396
                                        05A5  1397 ;
                                        05A5  1398 ; Call a device-independent routine, FIL$OPENFILE to locate the named
                                        05A5  1399 ; file on the disk.
                                        05A5  1400 ;
                                        05A5  1401 ; Registers set up are the following:
                                        05A5  1402 ;
                                        05A5  1403 ;       R7      - address of counted ascii file name string
                                        05A5  1404 ;       R10     - address of 1st byte of unused memory
                                        05A5  1405 ;       R11     - address of the RPB
                                        05A5  1406 ;
                                        05A5  1407
                                        05A5  1408
                                        05A5  1409 FILEBOOT:                                    ; Locate the file.
                   FA58'    30    05A5  1410          BSBW    BOO$CACHE_OPEN                  ; Open the FILEREAD cache
                  01 A7     9F    05A8  1411 10$:     PUSHAB  1(R7)                           ; Address of file name string.
                     7E     67    9A    05AB  1412          MOVZBL  (R7),-(SP)              ; Character count of file name.
                     56     7E    DE    05AE  1413          MOVAL   -(SP),R6                ; Allocate scratch for channel
                                        05B1  1414                                          ; and get adr of scratch storage
                     3C AB  DF    05B1  1415          PUSHAL  RPB$L_FILLBN(R11)               ; RPB fields that receive file
                                        05B4  1416                                          ; statistics during OPEN.
                        6A  DF    05B4  1417          PUSHAL  (R10)                           ; File header buffer at end of
                                        05B6  1418                                          ; memory.
                  0200 CA     DF    05B6  1419          PUSHAL  ^X200(R10)                      ; Index file header buffer at
                                        05BA  1420                                          ; end of memory.
                    04 A6     DF    05BA  1421          PUSHAL  4(R6)                           ; Address in file name desc.
                       66     DF    05BD  1422          PUSHAL  (R6)                            ; Address of phony channel.
            0000'CF    05     FB    05BF  1423          CALLS   #5,FIL$OPENFILE                 ; Call FILREAD to locate file.
                  5E    0C     C0    05C4  1424          ADDL2   #12,SP                          ; Clean up scratch space
                    48 50     E8    05C7  1425          BLBS    R0,FILE_CONTIG                  ; Branch on success.
                                        05CA  1426
                                        05CA  1427 ;
                                        05CA  1428 ; File was not found.  If looking up default secondary boot file in
                                        05CA  1429 ; SYS0 top level system directory, try the lookup with a null TOPSYS.
                                        05CA  1430 ;
                                        05CA  1431
            0910 8F    50    B1    05CA  1432          CMPW    R0,#SS$_NOSUCHFILE              ; If not "no such file"
                       11    12    05CF  1433          BNEQ    30$                             ; then don't consider a second try
                  0BB6'CF    95    05D1  1434          TSTB    W^FIL$GT_TOPSYS                 ; Try this second lookup once only
                       0B    13    05D5  1435          BEQL    30$                             ; Branch if no TOPSYS in use
                  0BB6'CF    94    05D7  1436          CLRB    W^FIL$GT_TOPSYS                 ; Disable TOPSYS
            0BBA'CF    30    91    05DB  1437          CMPB    #^A/0/,W^FIL$GT_TOPSYS+4        ; Trying to boot from [SYS0.SYSEXE]?
                       C6    13    05E0  1438          BEQL    10$                             ; Yes, try again from [SYSEXE]
                                        05E2  1439 30$:
                                        05E2  1440
                                        05E2  1441 ;
                                        05E2  1442 ; File was not found. Report an error.
                                        05E2  1443 ;
                                        05E2  1444
            50    0908 8F    B1    05E2  1445          CMPW    #SS$_NOSUCHDEV,R0               ; "No such device" error?
                       03    12    05E7  1446          BNEQ    FILOPN_ERR                      ; Branch if not.
```

```
                FAB7'  31  05E9  1447          BRW     NOSUCHDEV               ; Report "no such device" error.
                       05EC  1448
                       05EC  1449 FILOPN_ERR:                                 ; Report unknown error.
                       05EC  1450          ERROR   </%BOOT-F-Unable to locate BOOT file/>
                       0612  1451                                              ; Output error message.
                       0612  1452
                       0612  1453 ;
                       0612  1454 ; File was located successfully. Make sure that the file is contiguous.
                       0612  1455 ; The file statistics block is the following:
                       0612  1456 ;
                       0612  1457 ;          +----------------------+
                       0612  1458 ;          |     starting LBN     | (0 if file not contiguous)
                       0612  1459 ;          +----------------------+
                       0612  1460 ;          |    size in blocks    |
                       0612  1461 ;          +----------------------+
                       0612  1462 ;
                       0612  1463
                       0612  1464          ASSUME  RPB$L_FILSIZ EQ RPB$L_FILLBN+4
                       0612  1465
                       0612  1466 FILE_CONTIG:                                ; Test for contiguity.
             58  3C AB   7D  0612  1467          MOVQ    RPB$L_FILLBN(R11),R8    ; Get file statistics.
                58   D5  0616  1468          TSTL    R8                      ; Contiguous file?
                23   12  0618  1469          BNEQ    READ_HEADER             ; Yes, continue.
                       061A  1470          ERROR   </%BOOT-F-Bootfile not contiguous/>
                       063D  1471                                              ; Output error message.
                       063D  1472
                       063D  1473 ;
                       063D  1474 ; If the software boot control flags indicate that that transfer
                       063D  1475 ; address of the secondary bootstrap is stored in the image file's
                       063D  1476 ; header block, read that header block. Otherwise, assume that the
                       063D  1477 ; transfer address is simply the 1st byte in the image file.
                       063D  1478 ;
                       063D  1479
                       063D  1480 READ_HEADER:                                 ; Read header if necessary.
                   55  D4  063D  1481          CLRL    R5                      ; Assume no transfer address.
                   06  E1  063F  1482          BBC     #RPB$V_HEADER,-         ; If no header requested,
                30 AB       0641  1483                  RPB$L_BOOTR5(R11),-     ; then just branch past header
                   29       0643  1484                  READIN_BOOT             ; reading code.
                56  5A  D0  0644  1485          MOVL    R10,R6                  ; Start of free memory
                59  01  D0  0647  1486          MOVL    #1,R9                   ; Header is always only 1 block.
                   FA73'    064A  1487          BSBW    READFILE                ; Read header block.
             58  3C AB  7D  064D  1488          MOVQ    RPB$L_FILLBN(R11),R8    ; R8 = 1st LBN, R9 = block count
                52  59  7D  0651  1489          MOVQ    R9,R2                   ; R2 = block count, R3 = hdr adr
                   F9A9'    0654  1490          BSBW    BOO$IMAGE_ATT           ; Get image attributes
                       0657  1491 ;
                       0657  1492 ; R1 = image header block count
                       0657  1493 ; R2 = size of file in blocks excluding symbol table and patch text
                       0657  1494 ;
             00A0  CB  51  D0  0657  1495          MOVL    R1,RPB$B_HDRPGCNT(R11)  ; Store image header block count
             59  52  51  C3  065C  1496          SUBL3   R1,R2,R9                ; Blocks in image after header block(s)
                58  51  C0  0660  1497          ADDL    R1,R8                   ; LBN of first block beyond headr block(s)
             51  02 AA  3C  0663  1498          MOVZWL  IHD$W_ACTIVOFF(R10),R1  ; Get offset to image
                       0667  1499                                              ; activation data in header.
                51  5A  C0  0667  1500          ADDL    R10,R1                  ; Form transfer vector address.
                55  61  D0  066A  1501          MOVL    (R1),R5                 ; Get transfer address.
                       066D  1502
                       066D  1503 ;
```

```
                     066D  1504 ; Now read in the file. If the file is too large for the remaining
                     066D  1505 ; memory space, see if the required additional pages are usable.
                     066D  1506 ; If they are, use them.  If not issue a fatal diagnostic and HALT
                     066D  1507 ;
                     066D  1508 ; Registers set up now are the following:
                     066D  1509 ;
                     066D  1510 ;       R5      - transfer address
                     066D  1511 ;       R8      - starting LBN of file (after header)
                     066D  1512 ;       R9      - size of file in blocks
                     066D  1513 ;       R10     - address of 1st byte in free memory
                     066D  1514 ;       R11     - address of the RPB
                     066D  1515 ;       AP      - secondary boot argument list
                     066D  1516 ;
                     066D  1517
                     066D  1518 READIN_BOOT:
                     066D  1519
                     066D  1520         BRW_PSECT READIN_BOOT_1
                     066D
                     066D
                     066D                       BRW     READIN_BOOT_1
                     0670         ;
                     0670         ; ***** Change Program Section
                     0670         ;
                    00000007              .PSECT  $$$$COBOOT,LONG
                     0007
                     0007         READIN_BOOT_1:
                     0007  1521
                     0007  1522 ;
                     0007  1523 ; The following code disables XDELTA and sets up to read the secondary
                     0007  1524 ; bootstrap in over most of VMB.  The picture of memory is still
                     0007  1525 ; preserved for backwards compatibility.  Only the size of the primary
                     0007  1526 ; boot is smaller.
                     0007  1527 ;
                     0007  1528
     57  0000'CF    DE  0007  1529          MOVAL   OVERLAY_START,R7        ; End of all drivers, page aligned
       50    44 AB   7D  000C  1530          MOVQ    RPB$Q_PFNMAP(R11),R0    ; Descriptor for PFN bitmap
          52   50    7D  0010  1531          MOVQ    R0,R2                   ; Save a copy
     51   04   09    9C  0013  1532          ROTL    #9,#BITMAP_PAG_CNT,R1   ; Max pre-allocated byte count
          51   50    D1  0017  1533          CMPL    R0,R1                   ; Use the smaller for the
               03    15  001A  1534          BLEQ    20$                     ; backward compatible
          50   51    D0  001C  1535          MOVL    R1,R0                   ; maximum 8mb bitmap
     51   0200 C7    DE  001F  1536 20$:     MOVAL   ^X200(R7),R1            ; backward comp  ible bitmap
       44 AB   50    7D  0024  1537          MOVQ    R0,RPB$Q_PFNMAP(R11)    ; Store new descriptor for small bitmap
          5E   53    D1  0028  1538          CMPL    R3,SP                   ; If this is a large bitmap,
               03    14  002B  1539          BGTR    30$                     ; then it is above VMB
          52   50    7D  002D  1540          MOVQ    R0,R2                   ; Otherwise use the new
       14 AC   52    7D  0030  1541 30$:     MOVQ    R2,VMB$Q_PFNMAP(AP)     ; descriptor of the small bitmap
     50   0200 C0    DE  0034  1542          MOVAL   ^X200(R0),R0            ; Additional page of SCB to move
               55    DD  0039  1543          PUSHL   R5                      ; Preserve this from MOVC
          01FC'CF    D6  003B  1544          INCL    MUST_HALT               ; Disable restart from ERROUT
67   0000'CF    50    28  003F  1545          MOVC3   R0,BOOTHIGH,(R7)        ; Move the SCB and pre-allocated bitmap
               20    BA  0045  1546          POPR    #^M<R5>                 ; Restore image start offset
     5E   0600 C3    9E  0047  1547          MOVAB   <STACK_PAG_CNT*^X200>(R3),SP ; Move stack adjacent to bitmap
          5A   5E    D0  004C  1548          MOVL    SP,R10                  ; First free address above VMB
          56   5A    D0  004F  1549          MOVL    R10,R6                  ; Buffer for read
                         0052  1550
                         0052  1551 ;
```

```
                                0052  1552 ; Will the desired number of blocks fit in the space remaining in the
                                0052  1553 ; pre-tested 64kb of memory?  If not, check that the additional pages
                                0052  1554 ; required are usable.  If they are, then read it all, otherwise quit.
                                0052  1555 ;
                                0052  1556
                                0052  1557 CHECK_BOOT_FIT:
      50    5B    17    9C      0052  1558         ROTL    #<32-9>,R11,R0          ; PFN for RPB
        50    7F  A0    DE      0056  1559         MOVAL   127(R0),R0              ; Last PFN guaranteed to be good
      51    5A    17    9C      005A  1560         ROTL    #<32-9>,R10,R1          ; Starting PFN for read
            51    59    C0      005E  1561         ADDL    R9,R1                   ; Last+1 PFN needed to be good
                  05    11      0061  1562         BRB     30$                     ; Zero or more iterations
   1E 18 BC     50    E1      0063  1563 10$:     BBC     R0,@VMB$Q_PFNMAP+4(AP),SECOND_TOO_BIG ; Branch if cannot
                                0068  1564                                         ; read the entire secondary boot
      F7 50     51    F2      0068  1565 30$:     AOBLSS  R1,R0,10$               ; Check the next page
                                006C  1566
                                006C  1567 ;
                                006C  1568 ; Disable XDELTA, no debugging from here on.  The following read will
                                006C  1569 ; overwrite the XDELTA code.
                                006C  1570 ;
                                006C  1571
28 A7     0125'CF             9E  006C  1572         MOVAB   BOOT_FAULT+1,^X28(R7)   ; Shut off XDELTA TBIT handler and
2C A7     0125'CF             9E  0072  1573         MOVAB   BOOT_FAULT+1,^X2C(R7)   ; BPT handler in new copy of SCB
            11    57    DA      0078  1574         MTPR    R7,#PR$_SCBB            ; Set new SCB address
                                007B  1575
                                007B  1576 ;
                                007B  1577 ; Now read the secondary boot code into memory
                                007B  1578 ;
                                007B  1579
                  43    10      007B  1580         BSBB    READFILE                ; Read.
                                007D  1581
                                007D  1582 ;
                                007D  1583 ; The secondary bootstrap is now in memory. If the software boot control
                                007D  1584 ; flags asked for a HALT before the secondary bootstrap gains control,
                                007D  1585 ; execute a HALT instruction. Otherwise, transfer control to the new
                                007D  1586 ; bootstrap image.
                                007D  1587 ;
                                007D  1588
         09   E1                007D  1589         BBC     #RPB$V_HALT,-           ; If boot flags don't call for
   30 AB                        007F  1590                 RPB$L_BOOTR5(R11),-     ; halt, just transfer to new
         01                     0081  1591                 START_SECOND            ; bootstrap image.
                  00            0082  1592         HALT                            ; Otherwise, HALT.
                                0083  1593
                                0083  1594 START_SECOND:                           ; Transfer to secondary boot.
    654A        17              0083  1595         JMP     (R5)[R10]               ; Execute JUMP.
                                0086  1596
                                0086  1597 ;
                                0086  1598 ; Secondary bootstrap does not fit in the pre-tested 64kb and
                                0086  1599 ; one or more of the required adjacent pages is not usable.
                                0086  1600 ;
                                0086  1601
                                0086  1602 SECOND_TOO_BIG:
                                0086  1603         ERROR   </%BOOT-F-Boot file too big/>
```

```
00A3  1605
00A3  1606 ;
00A3  1607 ; No such device error reporting.
00A3  1608 ;
00A3  1609
00A3  1610 NOSUCHDEV:                                   ; "No such device" I/O error.
00A3  1611          ERROR   </%BOOT-F-Nonexistent drive/>
00C0  1612                                              ; Output error message.
```

```
                        00C0  1614          .SBTTL  READFILE, Reads bootstrap file in large chunks
                        00C0  1615
                        00C0  1616 ;++
                        00C0  1617 ;
                        00C0  1618 ; Functional description:
                        00C0  1619 ;
                        00C0  1620 ;       Calls the device-independent bootstrap QIO routine to read
                        00C0  1621 ;       a file. Divides the file into pieces as large as possible, so
                        00C0  1622 ;       that the read is a small number (like 1) of DMA transfers.
                        00C0  1623 ;
                        00C0  1624 ; Inputs:
                        00C0  1625 ;
                        00C0  1626 ;       R6      - buffer address
                        00C0  1627 ;       R8      - logical block number (LBN)
                        00C0  1628 ;       R9      - number of blocks in file
                        00C0  1629 ;
                        00C0  1630 ; Implicit inputs:
                        00C0  1631 ;
                        00C0  1632 ;       IO_SIZE - largest number of blocks possible in single transfer
                        00C0  1633 ;
                        00C0  1634 ; Outputs:
                        00C0  1635 ;
                        00C0  1636 ;       Registers R1-R4, and R5 must be preserved.
                        00C0  1637 ;
                        00C0  1638 ;       R6      - buffer address updated past last byte read
                        00C0  1639 ;       R8      - LBN updated to block after last block read
                        00C0  1640 ;       R9      - blocks in file (reduced to number not read)
                        00C0  1641 ;
                        00C0  1642 ; Implicit outputs:
                        00C0  1643 ;
                        00C0  1644 ;--
                        00C0  1645
                        00C0  1646 READFILE:                                    ; Read file into memory.
         32    10       00C0  1647          BSBB    READX                      ; Do the actual read
       2E 50   E8       00C2  1648          BLBS    R0,10$                     ; Success
50   0908 8F   B1       00C5  1649          CMPW    #SS$_NOSUCHDEV,R0          ; "No such device" error?
         D7    13       00CA  1650          BEQL    NOSUCHDEV                  ; Branch if yes.
                        00CC  1651          ERROR   <\%BOOT-F-I/O error reading boot file\>
                        00F3  1652                                             ; Output error message.
         05             00F3  1653 10$:     RSB
                        00F4  1654
                        00F4  1655 READX:
   57    7F 8F    9A     00F4  1656          MOVZBL  #IO_SIZE,R7                ; Assume maximum transfer size.
      59 57    D1       00F8  1657          CMPL    R7,R9                      ; Minimize with file size.
         03    15       00FB  1658          BLEQ    10$                        ; Branch if file larger than
                        00FD  1659                                             ; maximum transfer size.
      57 59    D0       00FD  1660          MOVL    R9,R7                      ; Set to remaining file size.
                        0100  1661
                        0100  1662 10$:                                        ; Push arguments for QIO.
         5B    DD       0100  1663          PUSHL   R11                        ; Push phony channel number.
         00    DD       0102  1664          PUSHL   #0                         ; Physical read mode.
         21    DD       0104  1665          PUSHL   #IO$_READLBLK              ; Read logical block function.
         58    DD       0106  1666          PUSHL   R8                         ; Starting LBN.
7E       57 09 9C       0108  1667          ROTL    #9,R7,-(SP)                ; Transfer size in bytes.
         56    DD       010C  1668          PUSHL   R6                         ; Buffer address
   56 04 AE    C0       010E  1669          ADDL    4(SP),R6                   ; Update buffer address.
      58 57    C0       0112  1670          ADDL    R7,R8                      ; Update LBN.
```

```
0000'CF    06   FB  0115  1671          CALLS   #6,B00$QIO                      ; Call a bootstrap QIO routine.
       01  50   E8  011A  1672          BLBS    R0,20$                          ; Continue on success.
                05  011D  1673          RSB
                    011E  1674
                    011E  1675 20$:                                             ; Read more file if any left.
           59   57   C2  011E  1676          SUBL    R7,R9                       ; Decrement blocks remaining.
                9D   14  0121  1677          BGTR    READFILE                    ; Continue if not done.
                05  0123  1678          RSB                                     ; Return to caller when done.
                    0124  1679
                    0124  1680 ;
                    0124  1681 ; Entry point for reading a single logical block from a device.  Needed
                    0124  1682 ; by the RT file open routine.
                    0124  1683 ;
                    0124  1684
                    0124  1685          SET_PSECT <YFILEREAD,BYTE>
                    0124
                    0124        ;
                    0124        ; ***** Change Program Section
                    0124        ;
                00000000        .PSECT  YFILEREAD,BYTE
                    0000
                    0000  1686
                  03C0  0000  1687          .ENTRY  FIL$READ_LBN,^M<R6,R7,R8,R9>
                    0002  1688
                    0002  1689
           56   08  AC   D0  0002  1690          MOVL    8(AP),R6                ; The buffer address
           58   04  AC   D0  0006  1691          MOVL    4(AP),R8                ; The LBN to grab
              59   01      D0  000A  1692          MOVL    #1,R9                   ; One block only
                 00E4'     30  000D  1693          BSBW    READX                   ; Do it
                    04  0010  1694          RET                                 ; Leave
```

```
                     1696          .SBTTL  CPU-specific Tables.
                     1697
                     1698          SET_PSECT <YBTMEM,LONG>
                     1699
                     1700  ;
                     1701  ; ***** Change Program Section
                     1702  ;
00000670             1703          .PSECT  YBTMEM,LONG
                     1704
                     1705  ;
                     1706  ; Map CPU identification codes to CPU-specific data tables.
                     1707  ;
                     1708
                     1709  CPU_CODES:                              ; CPU identification table.
0000                 1710          .WORD   0                       ; Code 0 is not valid.
                     1711          CPU_IDENT-                      ; Identify 11/780 (code 1).
                     1712                  LABEL=CPU_CODES,-       ; Name of table.
                     1713                  TABLE=780               ; Table address.
                     1714
                     1715          CPU_IDENT,-                     ; Identify 11/750 (code 2).
                     1716                  LABEL=CPU_CODES,-       ; Name of table.
                     1717                  TABLE=750               ; Table address.
                     1718
                     1719          CPU_IDENT,-                     ; Identify 11/730 (code 3)
                     1720                  LABEL=CPU_CODES,-       ; Name of table.
                     1721                  TABLE=730               ; Table address
                     1722
                     1723          CPU_IDENT,-                     ; Identify 11/790 (code 4)
                     1724                  LABEL=CPU_CODES,-       ; Name of table
                     1725                  TABLE=790               ; Table address
                     1726
                     1727          CPU_IDENT,-                     ; Identify 11/8SS (code 5)
                     1728                  LABEL=CPU_CODES,-       ; Name of table
                     1729                  TABLE=8SS               ; Table address
                     1730
                     1731  ;
                     1732  ; Tables of CPU-specific information.
                     1733  ;
                     1734
                     1735          CPU_DEF LABEL=780               ; 11/780 CPU-specific table.
                     1736
                     1737          CPU_DEF LABEL=750               ; 11/750 CPU-specific table.
                     1738
                     1739          CPU_DEF LABEL=730               ; 11/730 CPU-specific table.
                     1740
                     1741          CPU_DEF LABEL=790               ; 11/790 CPU-specific table
                     1742
                     1743          CPU_DEF LABEL=8SS               ; 11/8SS CPU-specific table
                     1744
                     1745  ;
                     1746  ; Tables of cpu-specific time-wait data cells values.
                     1747  ;
                     1748
                     1749  TENUSECTBL:                             ; Ten micro-second loop counter
04                   1750          .BYTE   4                       ;   11/780 and 11/785 value
01                   1751          .BYTE   1                       ;   11/750 value
```

```
01  06A6  1747          .BYTE   1                       ;   11/730 value
02  06A7  1748          .BYTE   2                       ;   11/790 value
05  06A8  1749          .BYTE   5                       ;   11/8SS value pulled from a hat
      06A9  1750
      06A9  1751 UBDELAY:                               ; Unibus delay loop counter
04  06A9  1752          .BYTE   4                       ;   11/780 and 11/785 value
01  06AA  1753          .BYTE   1                       ;   11/750 value
01  06AB  1754          .BYTE   1                       ;   11/730 value
02  06AC  1755          .BYTE   2                       ;   11/790 value
05  06AD  1756          .BYTE   5                       ;   11/8SS value pulled from a hat
```

```
                        .SBTTL  Test memory

                ;++
                ;
                ; Functional description:
                ;
                ;       This routine tests a given range of PFN's and builds one or
                ;       more bitmap pages.  Each bitmap page address is recorded in
                ;       the corresponding long word in the bitmap page address array.
                ;
                ; Calling Sequence:
                ;
                ;       JSB     BOO$TEST_MEM
                ;
                ; Inputs:
                ;
                ;       R2 = address of page test routine
                ;       R3 = number of pages to test
                ;       R9 = starting page number
                ;       R11 = address of Restart Parameter Block
                ;       BITMAP_VEC_PTR - contains the base address of the bitmap
                ;               page address array.
                ;       SECOND_PARAM - is the address of the Secondary Boot parameter
                ;               list.  The lowest and highest PFN's seen are to be
                ;               recorded here.
                ;
                ; Outputs:
                ;
                ;       R0,R1,R2,R3,R6,R9 altered
                ;       others preserved
                ;
                ;       Store address of newly allocated bitmap page(s) in the vector
                ;               pointed to by BITMAP_VEC_PTR and indexed by the
                ;               high 9 bits of the PFN's covered by the bitmap page.
                ;       Record the lowest (inclusive) and highest (exclusive) PFN's
                ;               seen in the offsets VMB$L_LO_PFN and VMB$L_HI_PFN
                ;               of the Secondary Boot Parameter list (SECOND_PARAM).
                ;       Record the highest bitmap page index used in the vector of
                ;               bitmap page addresses (BITMAP_HI_INDX).
                ;
                ;--
                ;
                ; Scratch storage offset definitions
                ;
                        $OFFSET 0,NEGATIVE,<-
                        NXT_PFN,-               ; Starting PFN for next segment of bitmap
                        NXT_PAGCNT,-            ; Page count remaining
                        PRETST_PFN,-            ; Pre-tested starting PFN
                        PRETST_CNT,-            ; Pre-tested page count
                        PAGTST,-                ; Address of page test routine
                        <SCRATCH_SIZE,0>-       ; Size of scratch area
                        >
FFFC            NXT_PFN:
FFF8            NXT_PAGCNT:
FFF4            PRETST_PFN:
FFF0            PRETST_CNT:
```

```
                        FFEC        PAGTST:
                        FFEC        SCRATCH_SIZE:
                        06AE  1811
                        06AE  1812 BOO$TEST_MEM::
                35B0 8F    BB   06AE  1813          PUSHR   #^M<R4,R5,R7,R8,R10,FP,AP>
           5C   01A8'CF    DE   06B2  1814          MOVAL   SECOND_PARAM,AP         ; Point at Secondary Boot Parameter list
                5D    5E   D0   06B7  1815          MOVL    SP,FP                   ; Use FP as a local pointer
                5E    14   C2   06BA  1816          SUBL    #-SCRATCH_SIZE,SP       ; Reserve local storage
        F4 AD    5B    17   9C   06BD  1817          ROTL    #<32-9>,R11,PRETST_PFN(FP) ; Set start of pre-tested pages
        F0 AD    01    07   9C   06C2  1818          ROTL    #7,#1,PRETST_CNT(FP)    ; 128 pre-tested pages
           EC AD    52    D0   06C7  1819          MOVL    R2,PAGTST(FP)           ; Save address of page test routine
           50    59    53   C1   06CB  1820          ADDL3   R3,R9,R0                ; Last + 1 PFN
             10 AC    50   D1   06CF  1821          CMPL    R0,VMB$L_HI_PFN(AP)     ; Higher than last highest?
                      04   15   06D3  1822          BLEQ    10$                     ; Branch if not
              10 AC   50   D0   06D5  1823          MOVL    R0,VMB$L_HI_PFN(AP)     ; Yes, record the highest value
              0C AC   59   D1   06D9  1824 10$:     CMPL    R9,VMB$L_LO_PFN(AP)     ; Is the starting PFN smaller than
                              06DD  1825                                          ; the currently recorded smallest?
                      04   18   06DD  1826          BGEQ    20$                     ; Branch if not
              0C AC   59   D0   06DF  1827          MOVL    R9,VMB$L_LO_PFN(AP)     ; Yes, record the smallest one seen
                              06E3  1828 20$:
                              06E3  1829 NEXT_BITMAP:
     52   59   0C   00   EF   06E3  1830          EXTZV   #0,#12,R9,R2            ; No. of pages from beginning
                              06E8  1831                                          ; of bitmap page (4096 bits/page)
                53   52   C0   06E8  1832          ADDL    R2,R3                   ; Page count is larger by that amount
                51   53   D0   06EB  1833          MOVL    R3,R1                   ; Save this page count
           50   01   0C   9C   06EE  1834          ROTL    #12,#1,R0               ; Max of 4096 pages per bitmap page
                50   53   D1   06F2  1835          CMPL    R3,R0                   ; More than a bitmap's worth?
                      03   15   06F5  1836          BLEQ    10$                     ; Branch if not
                53   50   D0   06F7  1837          MOVL    R0,R3                   ; Yes, use the max
        F8 AD   51   53   C3   06FA  1838 10$:     SUBL3   R3,R1,NXT_PAGCNT(FP)    ; No. of pages left for next iteration
                53   52   C2   06FF  1839          SUBL    R2,R3                   ; Actual no. of pages to test
                53   59   C0   0702  1840          ADDL    R9,R3                   ; Last + 1 PFN to be tested
             FC AD   53   D0   0705  1841          MOVL    R3,NXT_PFN(FP)          ; Save for next iteration if any
     5A   59    F4   BF   78   0709  1842          ASHL    #-12,R9,R10             ; Index to bitmap page address array
        01E8'CF    5A   D1   070E  1843          CMPL    R10,BITMAP_HI_INDX      ; Is this the highest index seen?
                      05   15   0713  1844          BLEQ    20$                     ; Branch if not
        01E8'CF    5A   D0   0715  1845          MOVL    R10,BITMAP_HI_INDX      ; Keep track of the highest index
                56   00   D2   071A  1846 20$:     MCOML   #0,R6                   ; Bitmap page address not set yet
              30 AB   07   E0   071D  1847          BBS     #RPB$V_NOTEST,RPB$L_BOOTR5(R11),- ; If not testing memory
                      49         0721  1848                  INIT_BITMAP             ; then fill in bit map as all good
                              0722  1849
                              0722  1850 ;
                              0722  1851 ; Test each page for gross errors unless RPB$V_NOTEST is set. For each
                              0722  1852 ; page available and good (if tested), set the corresponding bit in
                              0722  1853 ; the PFN bitmap.
                              0722  1854 ;
                              0722  1855
                              0722  1856 MEM_LOOP:                                 ; Test one controller's worth.
              F4 AD   59   D1   0722  1857          CMPL    R9,PRETST_PFN(FP)       ; Is this a the next pre-tested page?
                      0D   12   0726  1858          BNEQ    TEST_PAGE               ; Branch if not, go test it
                              0728  1859
                              0728  1860 ;
                              0728  1861 ; Just handle these pre-tested pages one at a time.  There are only
                              0728  1862 ; 128 of them, and handling them as a unit means they are restricted
                              0728  1863 ; to being in the same bitmap segment.  It is required that they be
                              0728  1864 ; in one controller, but this further breakdown into 4096 page units
                              0728  1865 ; which fit into a one page bitmap is artificial.
```

```
                     0728  1866 ;
                     0728  1867
        F4 AD    D6  0728  1868          INCL    PRETST_PFN(FP)              ; Yes, step to next PFN
     11 F0 AD    F5  072B  1869          SOBGTR  PRETST_CNT(FP),GOOD_PAGE ; Count these pages
                     072F  1870                                              ; Treat it as good, don't retest
  F4 AD    00    D2  072F  1871          MCOML   #0,PRETST_PFN(FP)           ; No, more pre-tested pages left
           0B    11  0733  1872          BRB     GOOD_PAGE                   ; Last pre-tested page is good too
                     0735  1873 TEST_PAGE:
50 59      09    78  0735  1874          ASHL    #9,R9,R0                    ; Convert page # to physical address
51 01      06    9C  0739  1875          ROTL    #6,#1,R1                    ; Loop counter for 64 quadwords.
     EC BD    16    073D  1876          JSB     @PAGTST(FP)                 ; Test this page
                     0740  1877 GOOD_PAGE:                                   ; Good page here.
           56    D5  0740  1878          TSTL    R6                          ; Is the bitmap set up yet?
           27    19  0742  1879          BLSS    INIT_BITMAP                 ; Branch if no, must init one
                     0744  1880 COUNT_PAGE:
03 66      59    E2  0744  1881          BBSS    R9,(R6),NEXT_PAGE           ; Set bit in bitmap; do next page
     4C AB    D6  0748  1882          INCL    RPB$L_PFNCNT(R11)           ; Increment good page count.
                     074B  1883
                     074B  1884 ;
                     074B  1885 ; If more pages remain in this bitmap segment, continue looping.
                     074B  1886 ;
                     074B  1887
                     074B  1888 NEXT_PAGE:                                   ; Do next page, if any.
D3 59      53    F2  074B  1889          AOBLSS  R3,R9,MEM_LOOP  ; Process next page if any
                     074F  1890 CHK_NEXT_BITMAP:
59   FC AD    D0  074F  1891          MOVL    NXT_PFN(FP),R9              ; Starting PFN for next bitmap segment
53   F8 AD    D0  0753  1892          MOVL    NXT_PAGCNT(FP),R3           ; Another bitmap segment to do?
           8A    12  0757  1893          BNEQ    NEXT_BITMAP                 ; Branch if yes
     5E    5D    D0  0759  1894          MOVL    FP,SP                       ; Clean off local storage
     35B0  8F    BA  075C  1895          POPR    #^M<R4,R5,R7,R8,R10,FP,AP> ; Recover saved registers
           05    05  0760  1896          RSB                                 ; and return
                     0761  1897 ;
                     0761  1898 ; Handler that gains control when a page has gross memory errors.
                     0761  1899 ; Entered via a BRW from the actual machine specific handler after it
                     0761  1900 ; has done the machine specific thing necessary to clear the condition
                     0761  1901 ;
                     0761  1902
                     0761  1903 BOO$PAGE_MCHECK::                            ; Skip current page.
     5E    5D    14  C3  0761  1904          SUBL3   #-SCRATCH_SIZE,FP,SP    ; Restore stack pointer.
           0104  CB  D6  0765  1905          INCL    RPB$L_BADPGS(R11)           ; Count number of bad pages found.
                 E0  11  0769  1906          BRB     NEXT_PAGE                   ; And get the next page
                     076B  1907 ;
                     076B  1908 ; Allocate and initialize a page of bitmap for this bitmap segment
                     076B  1909 ;
                     076B  1910 INIT_BITMAP:
   56   01E4'DF4A    D0  076B  1911          MOVL    @BITMAP_VEC_PTR[R10],R6 ; Is bitmap page allocated?
                 38  18  0771  1912          BGEQ    GOT_BITMAP                  ; Branch if yes
52   01EC'CF    5A  C1  0773  1913          ADDL3   R10,BITMAP_BAS_PFN,R2   ; Desired PFN for this bitmap page
50   52    0C    00  EF  0779  1914          EXTZV   #0,#12,R2,R0                ; Bitmap page relative PFN
51   52    09    0C  EF  077E  1915          EXTZV   #12,#9,R2,R1                ; Index to bitmap page
   51   01E4'DF41    D0  0783  1916          MOVL    @BITMAP_VEC_PTR[R1],R1  ; Address of bitmap page
                 0A  19  0789  1917          BLSS    20$                         ; Branch if entire bitmap page not there
        06 61    50  E1  078B  1918          BBC     R0,(R1),20$                 ; Branch if desired page was bad
                     078F  1919                                              ; or not yet tested
     56    52    09  78  078F  1920          ASHL    #9,R2,R6                    ; Form bitmap adr from PFN
                 04  11  0793  1921          BRB     30$
     56    59    09  78  0795  1922 20$:     ASHL    #9,R9,R6                    ; Use first good page mapped by this
```

```
                                0799  1923
                                0799  1924 30$:     PUSHL   R3                              ; Save last PFN to be tested
   66   0200 8F    00    66    00    2C  079B  1925          MOVC5   #0,(R6),#0,#512,(R6)     ; Init all PFN's bad
                                08    BA  07A3  1926          POPR    #^M<R3>                 ; Restore last PFN to be tested
                01E4'DF4A    56    D0  07A5  1927          MOVL    R6,@BITMAP_VEC_PTR[R10] ; Record adr of bitmap page
                                07AB  1928 GOT_BITMAP:
              50    59    FD  8F    78  07AB  1929          ASHL    #-3,R9,R0               ; Form byte offset from the beginning
                 50    01FF  8F    AA  07B0  1930          BICW    #^X1FF,R0               ; of the bitmap to PFN 0
                    56    50    C2  07B5  1931          SUBL    R0,R6                   ; so that BBSS PFN,(R6) will work
                  30 AB    07    E1  07B8  1932          BBC     #RPB$V_NOTEST,RPB$L_BOOTR5(R11),- ; Branch if testing memory
                                87        07BC  1933          COUNT_PAGE
                                07BD  1934 ;
                                07BD  1935 ; Set bits for pages which are not to be tested, but assumed good.
                                07BD  1936 ;
              52    53    59    C3  07BD  1937          SUBL3   R9,R3,R2                ; Count of pages to mark good
                 4C AB    52    C0  07C1  1938          ADDL    R2,RPB$L_PFNCNT(R11)    ; Add them in as good pages
                    50    20    D0  07C5  1939          MOVL    #32,R0                  ; Set 32 bits per iteration
              51    52    1F    CB  07C8  1940          BICL3   #31,R2,R1               ; Even no. of longwords worth
                 51    7149  9E  07CC  1941          MOVAB   -(R1)[R9],R1            ; R1=R1-1+R9 = last PFN inclusive
                                07D0  1942                                              ; for 32 at a time loop
   66   50   59    FFFFFFFF  8F    F0  07D0  1943 20$:     INSV    #-1,R9,R0,(R6)          ; Mark these pages good
           FFF1 59    50    51    F1  07D9  1944          ACBL    R1,R0,R9,20$            ; R9=R9+R0, If R9 LEQ R1 then GOTO 20$
           50   52    0?    00    EF  07DF  1945          EXTZV   #0,#5,R2,R0             ; No. of bits remaining to be set
                                04    13  07E4  1946          BEQL    30$                     ; Branch if all bits are set
                                52    D4  07E6  1947          CLRL    R2                      ; Stop the next time around
                                E6    11  07E8  1948          BRB     20$                     ; Set the remaining bits
                             FF62    31  07EA  1949 30$:     BRW     CHK_NEXT_BITMAP
```

```
                                M 1
07ED  1951          .SBTTL  SAVE_CSRS, CPU-specific routines
07ED  1952
07ED  1953 ;++
07ED  1954 ;
07ED  1955 ; Functional description:
07ED  1956 ;
07ED  1957 ;       One routine per CPU implementation follows. Each routine
07ED  1958 ;       determines from input registers the addresses of the boot
07ED  1959 ;       device's CSR and the attached adapter's CSR, and stores these
07ED  1960 ;       addresses in the RPB.
07ED  1961 ;
07ED  1962 ; Inputs:
07ED  1963 ;
07ED  1964 ;       R1      - address of the boot device's adapter
07ED  1965 ;
07ED  1966 ;                       11/780  - <31:4> MBZ
07ED  1967 ;                                 <3:0> TR number
07ED  1968 ;                       11/750  - <31:24> MBZ
07ED  1969 ;                                 <23:0> address of the I/O page for the
07ED  1970 ;                                        boot device's adapter
07ED  1971 ;
07ED  1972 ;       R2      - UNIBUS:
07ED  1973 ;
07ED  1974 ;                       <31:18> MBZ
07ED  1975 ;                       <17:3> UNIBUS address of the device's CSR
07ED  1976 ;                       <2:0> MBZ
07ED  1977 ;
07ED  1978 ;       R2      - MASSBUS or CI:
07ED  1979 ;
07ED  1980 ;                       not used
07ED  1981 ;
07ED  1982 ;       R11     - address of the Restart Parameter Block
07ED  1983 ;
07ED  1984 ; Implicit inputs:
07ED  1985 ;
07ED  1986 ;       The boot device may be located on any UNIBUS, MASSBUS, or CI  adapter.
07ED  1987 ;
07ED  1988 ; Outputs:
07ED  1989 ;
07ED  1990 ;       R8, R11, and SP must be preserved.
07ED  1991 ;
07ED  1992 ; Implicit outputs:
07ED  1993 ;
07ED  1994 ;       The RPB fields RPB$L_ADPPHY and RPB$L_CSRPHY contain the
07ED  1995 ;       addresses of the adapter's register space and the boot device's
07ED  1996 ;       register space respectively. For CI or MASSBUS devices, these values
07ED  1997 ;       are identical.
07ED  1998 ;
07ED  1999 ;--
```

```
                        07ED  2001          .SBTTL  SAVE_CSR_780, Save CSRs for 11/780
                        07ED  2002          .SBTTL  SAVE_CSR_730, Save CSRs for 11/730
                        07ED  2003          .SBTTL  SAVE_CSR_790, Save CSRs for 11/790
                        07ED  2004
                        07ED  2005 ;++
                        07ED  2006 ;
                        07ED  2007 ; SAVE_CSR_780, Save CSRs for 11/780
                        07ED  2008 ; SAVE_CSR_730, Save CSRs for 11/730
                        07ED  2009 ; SAVE_CSR_790, Save CSRs for 11/790
                        07ED  2010 ;
                        07ED  2011 ; Implicit inputs:
                        07ED  2012 ;
                        07ED  2013 ;       R1 contains the system bus address in the form of a TR number
                        07ED  2014 ;       because TR numbers map to fixed physical addresses.
                        07ED  2015 ;
                        07ED  2016 ;       Bus adapters on the 780 start at address IO780$AL_IOBASE for TR 0,
                        07ED  2017 ;       and increment by ^X2000 each subsequent adapter. UNIBUS address
                        07ED  2018 ;       space for the first UNIBUS in the configuration starts at
                        07ED  2019 ;       IO780$AL_UBOSP. The second through fourth UNIBUS start 64kb after
                        07ED  2020 ;       the previous UNIBUS. UNIBUS device CSRs are in the last 64kb of
                        07ED  2021 ;       the UNIBUS address space.
                        07ED  2022 ;
                        07ED  2023 ;       The I/O space layout for the 11/730 is the same as that
                        07ED  2024 ;       described above for the 11/780 except that the base of
                        07ED  2025 ;       TR 0 register space is IO730$AL_IOBASE and UNIBUS 0 space
                        07ED  2026 ;       starts at IO730$AL_UBOSP.  Also, only one UNIBUS will ever
                        07ED  2027 ;       be configured on an 11/730.
                        07ED  2028 ;
                        07ED  2029 ;       The I/O space on the 11/790 allows for multiple (2) SBIA's.
                        07ED  2030 ;       The offsets for NEXUS within an SBI are the same as on the 11/780.
                        07ED  2031 ;       For the 11/790, R1 can have a value from 1 to 16, with bits 4 and 5
                        07ED  2032 ;       indicating which A-bus adapter hosts the system bus.
                        07ED  2033 ;
                        07ED  2034 ; Outputs:
                        07ED  2035 ;
                        07ED  2036 ;       R5-R11 and AP-SP are preserved.
                        07ED  2037 ;
                        07ED  2038 ;--
                        07ED  2039
                        07ED  2040 SAVE_CSR_780:                            ; Save CSRs for the 11/780
 53   20000000 9F   DE  07ED  2041          MOVAL   a#IO780$AL_IOBASE,R3     ; Get adapter 0 space.
 54   20100000 9F   DE  07F4  2042          MOVAL   a#IO780$AL_UBOSP,R4      ; Get UB 0 space
              2D   11  07FB  2043          BRB     SAVE_CSR_COMMON          ; Join common code
                        07FD  2044
                        07FD  2045 SAVE_CSR_730:                            ; Save CSRs for the 11/730.
 53   00F20000 9F   DE  07FD  2046          MOVAL   a#IO730$AL_IOBASE,R3     ; Get adapter 0 space
 54   00F00000 9F   DE  0804  2047          MOVAL   a#IO730$AL_UBOSP,R4      ; Get UB0 space
              1D   11  080B  2048          BRB     SAVE_CSR_COMMON          ; Join common code
                        080D  2049
                        080D  2050 SAVE_CSR_790:
 53   20000000 9F   DE  080D  2051          MOVAL   a#IO790$AL_IOA0,R3       ; Assume start of 1st SBIA
54   51   02   04   EF  0814  2052          EXTZV   #RPB$V_ABUS,#RPB$S_ABUS,R1,R4    ; Get A-bus adapter number
53   02   19   54   F0  0819  2053          INSV    R4,#25,#RPB$S_ABUS,R3    ; Insert to make physical address ********
 54   00100000 E3   DE  081E  2054          MOVAL   IO790$AL_UBOSP(R3),R4    ; Get UB 0 space for this SBIA
51   51   04   00   EF  0825  2055          EXTZV   #RPB$V_NEXUS,#RPB$S_NEXUS,R1,R1 ; Extract nexus number
                        082A  2056
                        082A  2057
```

```
                                082A  2058 SAVE_CSR_COMMON:                        ; Common code from here on
                                082A  2059
                                082A  2060 ;
                                082A  2061 ; Compute and save adapter configuration register address:
                                082A  2062 ;
                                082A  2063
          50    51    0D    9C  082A  2064         ROTL    #13,R1,R0               ; Convert TR # to adapter
                50    53    C0  082E  2065         ADDL    R3,R0                   ; configuration address
            5C  AB    50    D0  0831  2066         MOVL    R0,RPB$L_ADPPHY(R11)    ; Store in RPB
            54  AB    50    D0  0835  2067         MOVL    R0,RPB$L_CSRPHY(R11)    ; Store in device CSR too in
                                0839  2068                                         ; case this is a MASSBUS or CI device.
                                0839  2069
                                0839  2070 ;
                                0839  2071 ; Read the configuration register to determine whether the adapter is
                                0839  2072 ; for a UNIBUS, MASSBUS, or CI. Bits <31:3> of the configuration register
                                0839  2073 ; identify the adapter type.
                                0839  2074 ;
                                0839  2075
                53    60    D0  0839  2076         MOVL    (R0),R3                 ; Read the adapter's CR.
          00A1  CB    53    9B  083C  2077         MOVZBW  R3,RPB$W_BOOTNDT(R11)   ; Save boot adapter's nexus device type.
          50    53    03    CB  0841  2078         BICL3   #3,R3,R0                ; Clear the bits representing
                                0845  2079                                         ; a UBA number to obtain the
                                0845  2080                                         ; canonical adapter type.
                50    28    91  0845  2081         CMPB    #NDT$_UBO,R0            ; Is this a UNIBUS adapter?
                      01    13  0848  2082         BEQL    UBA_ADAPTER             ; Yes. Go compute device's CSR.
                            05  084A  2083         RSB                             ; RPB$L_CSRPHY already set for
                                084B  2084                                         ; MASSBUS or CI, so return.
                                084B  2085
                                084B  2086 UBA_ADAPTER:                             ; Compute UNIBUS device's CSR.
            54  AB    52    54  C9  084B  2087         BISL3   R4,R2,-             ; Add UNIBUS CSR value to
                                0850  2088                 RPB$L_CSRPHY(R11)       ; base of all UNIBUS address
                                0850  2089                                         ; space.
                      50    D4  0850  2090         CLRL    R0                      ; Get start of this ub space
          50    02    12    53  F0  0852  2091         INSV    R3,#18,#2,R0        ; relative to UBO space.
                54  AB    50  C0  0857  2092         ADDL    R0,RPB$L_CSRPHY(R11)    ; Adjust CSR address to this UB space.
                            05  085B  2093         RSB                             ; Return.
```

```
085C  2095          .SBTTL  SAVE_CSR_750, Save CSRs for 11/750
085C  2096
085C  2097 ;
085C  2098 ;*****
085C  2099 ; Add code to accept a slot number between 0 and 15 in R1.
085C  2100 ;*****
085C  2101 ;
085C  2102
085C  2103 ;++
085C  2104 ;
085C  2105 ; SAVE_CSR_750, Save CSRs for 11/750
085C  2106 ;
085C  2107 ; Implicit inputs:
085C  2108 ;
085C  2109 ;       For a MASSBUS boot device, R1 contains a 24-bit address of the
085C  2110 ;       MASSBUS adapter's address space. The MBAs start at fixed
085C  2111 ;       addresses that correspond to NEXUS slots 20-23:
085C  2112 ;
085C  2113 ;               MBA0    ^XF28000, slot 20       I0750$AL_MBBASE
085C  2114 ;               MBA1    ^XF2A000, slot 21       I0750$AL_MBBASE+^X2000
085C  2115 ;               MBA2    ^XF2C000, slot 22       I0750$AL_MBBASE+^X4000
085C  2116 ;               MBA3    ^XF2E000, slot 23       I0750$AL_MBBASE+^X6000
085C  2117 ;
085C  2118 ;       For a UNIBUS boot device, R1 contains a 24-bit address of the
085C  2119 ;       boot device UNIBUS' I/O page. The I/O pages start at fixed
085C  2120 ;       addresses that correspond to UNIBUS adapters also at fixed
085C  2121 ;       addresses that correspond to NEXUS slots 24-25:
085C  2122 ;
085C  2123 ;               UNIBUS 0 I/O page       ^XFFE000,
085C  2124 ;                 I0750$AL_UBBASE, UBI0 ^XF30000, slot 24
085C  2125 ;
085C  2126 ;               UNIBUS 1 I/O page       ^XFBE000,
085C  2127 ;                 I0750$AL_UBBASE+^X2000,
085C  2128 ;                                 UBI1 ^XF32000, slot 25
085C  2129 ;
085C  2130 ;       UNIBUS address space:
085C  2131 ;
085C  2132 ;               UBI0    ^XFC0000:^XFFFFFF       I0750$AL_UB0SP
085C  2133 ;               UBI1    ^XF80000:^XFBFFFF       I0750$AL_UB0SP+^X40000
085C  2134 ;
085C  2135 ;       For a boot device in the floating space, R1 contains a 24-bit
085C  2136 ;       address of the adapter's address space. The floating adapters
085C  2137 ;       start at the CMI address that correspond to NEXUS slots 26-31:
085C  2138 ;
085C  2139 ;               float_A ^XF34000, slot 26       I0750$AL_FLOAT
085C  2140 ;                 .
085C  2141 ;               float_F ^XF3E000, slot 31       I0750$AL_FLOAT+^XA000
085C  2142 ;
085C  2143 ;       The routine reduces the adapter's physical address to a slot
085C  2144 ;       number from 0-15, and stores the number in the RPB for later
085C  2145 ;       use by INIT.
085C  2146 ;
085C  2147 ; Outputs:
085C  2148 ;
085C  2149 ;       R1-R11 and AP-SP are preserved.
085C  2150 ;
085C  2151 ; Implicit outputs:
```

```
                                085C  2152 ;
                                085C  2153 ;       This routine derives the slot number (less 16) for the boot
                                085C  2154 ;       device's adapter and loads the number into RPB$L_BOOTR1. Later
                                085C  2155 ;       INIT uses the value in RPB$L_BOOTR1 as an index into the RPB
                                085C  2156 ;       CONFREG field to find the adapter type of the boot device.
                                085C  2157 ;
                                085C  2158 ;--
                                085C  2159
                                085C  2160 SAVE_CSR_750:                            ; Save CSRs for the 11/750.
00F40000 8F    51    D1         085C  2161          CMPL    R1,#IO750$AL_FLOAT+ -
                                0863  2162                   <6*IO750$AL_PERNEX>    ; Is this adapter in the float space?
               13    1E         0863  2163          BGEQU   10$                    ; No, must be MASSBUS or UNIBUS
00F34000 8F    51    D1         0865  2164          CMPL    R1,#IO750$AL_FLOAT     ; Is this adapter in the float space?
               0A    1F         086C  2165          BLSSU   10$                    ; No, must be MASSBUS or UNIBUS
                                086E  2166 ;
                                086E  2167 ; Read the configuration register to determine the adapter. Bits <31:3> of
                                086E  2168 ; the configuration register identify the adapter type.
                                086E  2169 ;
         53    61    D0         086E  2170          MOVL    (R1),R3                ; Read the adapter's CR.
    00A1 CB    53    9B         0871  2171          MOVZBW  R3,RPB$W_BOOTNDT(R11)  ; Save boot adapter's nexus device type.
               0E    11         0876  2172          BRB     20$                    ; Continue in common code
                                0878  2173
00F30000 8F    51    D1         0878  2174 10$:     CMPL    R1,#IO750$AL_UBBASE    ; Is this a MASSBUS adapter?
               0F    1E         087F  2175          BGEQU   DERIVE_UBIADDR         ; No. Go derive UBI address.
    00A1 CB    20    9B         0881  2176          MOVZBW  #NDT$_MB, -            ; Save boot adapter's nexus device
                                0886  2177                  RPB$W_BOOTNDT(R11)     ; type.
      5C AB    51    D0         0886  2178 20$:     MOVL    R1,RPB$L_ADPPHY(R11)   ; Yes, store MBA's CR in RPB.
      54 AB    51    D0         088A  2179          MOVL    R1,RPB$L_CSRPHY(R11)   ; And store in RPB$L_CSRPHY too.
               40    11         088E  2180          BRB     DERIVE_SLOTNUM         ; Branch to derive slot number.
                                0890  2181
                                0890  2182 DERIVE_UBIADDR:                         ; Reduce R1 to UBI address.
   00FC0000 8F    51    D1         0890  2183          CMPL    R1,#IO750$AL_UB0SP     ; Is this UNIBUS 1 or UNIBUS 0?
               1D    1A         0897  2184          BGTRU   10$                    ; UNIBUS 0. Go load addresses.
    00A1 CB    29    9B         0899  2185          MOVZBW  #NDT$_UB1, -           ; Save boot adapter's nexus device
                                089E  2186                  RPB$W_BOOTNDT(R11)     ; type.
     00F30000 8F    C1         089E  2187          ADDL3   #IO750$AL_UBBASE,-     ; UNIBUS 1. Load the ADPPHY
     00002000 8F               08A4  2188                  #IO750$AL_PERNEX,-     ; field with UBI1 address.
           5C AB               08A9  2189                  RPB$L_ADPPHY(R11)
54 AB  52  00FBE000 8F    C9   08AB  2190          BISL3   #<IO750$AL_UB0SP-      ; Create and store away the
                                08B4  2191                  -^X40000+^O760000>,R2,- ; boot device's CSR address.
                                08B4  2192                  RPB$L_CSRPHY(R11)
               16    11         08B4  2193          BRB     GET_UBI_ADDR           ; Branch to common UBI code.
                                08B6  2194
                                08B6  2195 10$:                                   ; UNIBUS 0. Load addresses.
    00A1 CB    28    9B         08B6  2196          MOVZBW  #NDT$_UB0, -           ; Save boot adapter's nexus device
                                08BB  2197                  RPB$W_BOOTNDT(R11)     ; type.
     00F30000 8F    D0         08BB  2198          MOVL    #IO750$AL_UBBASE,-     ; Store address of 1st UNIBUS
           5C AB               08C1  2199                  RPB$L_ADPPHY(R11)      ; adapter.
54 AB  52  00FFE000 8F    C9   08C3  2200          BISL3   #<IO750$AL_UB0SP-      ; Calculate and load the full
                                08CC  2201                  +^O760000>,R2,-        ; CSR address into RPB.
                                08CC  2202                  RPB$L_CSRPHY(R11)
                                08CC  2203
                                08CC  2204 GET_UBI_ADDR:                          ; Obtain UBI address.
         51    5C AB    D0      08CC  2205          MOVL    RPB$L_ADPPHY(R11),R1   ; Get address of associated UBI.
                                08D0  2206
                                08D0  2207 DERIVE_SLOTNUM:                        ; Derive and save slot number.
   50    51    00F20000 8F    C3   08D0  2208          SUBL3   #IO750$AL_IOBASE,R1,R0 ; Get adapter's offset from base
```

```
                                08D8  2209
                                                                        ; of I/O space.
            50  00002000 8F    C6  08D8  2210      DIVL    #I0750$AL_PERNEX,R0     ; Divide by size of one adapter.
                 20 AB    50   D0  08DF  2211      MOVL    R0,RPB$L_BOOTR1(R11)    ; Save the slot number.
                                05  08E3  2212      RSB                             ; Return to caller.
```

```
                        08E4  2214          .SBTTL  SAVE_CSR_8SS, Save CSRs for 11/8SS
                        08E4  2215
                        08E4  2216 ;+
                        08E4  2217 ;
                        08E4  2218 ; SAVE_CSR_8SS, Save CSRs for 11/8SS
                        08E4  2219 ;
                        08E4  2220 ; Inputs:
                        08E4  2221 ;
                        08E4  2222 ;       R1  =  BI node number for Boot device's adapter
                        08E4  2223 ;       R2  =  BI address of Boot device's CSR if booting on BUA
                        08E4  2224 ;       R11 => RPB
                        08E4  2225 ;
                        08E4  2226 ; Implicit inputs:
                        08E4  2227 ;
                        08E4  2228 ;       The BI physical address space is as follows:
                        08E4  2229 ;
                        08E4  2230 ;               I/O space begins at ^x20000000 - symbolically IO8SS$AL_IOBASE
                        08E4  2231 ;
                        08E4  2232 ;               Each node has an 8KB register space.  The first node
                        08E4  2233 ;               register space begins at IO8SS$AL_IOBASE, and the next
                        08E4  2234 ;               one begins at an 8KB offset from it, and the next 8KB, etc.
                        08E4  2235 ;
                        08E4  2236 ;               Each node has a 256KB node space reserved for it.  These
                        08E4  2237 ;               node spaces begin at physical address ^x20400000 -
                        08E4  2238 ;               symbolically this is IO8SS$AL_NODESP.  The node space
                        08E4  2239 ;               for node 0 begins at this physical address and the
                        08E4  2240 ;               subsequent ones begin at 256KB intervals.
                        08E4  2241 ;
                        08E4  2242 ; Outputs:
                        08E4  2243 ;
                        08E4  2244 ;       Registers R0 - R3 modified.  All others preserved.
                        08E4  2245 ;
                        08E4  2246 ;-
                        08E4  2247
                        08E4  2248
                        08E4  2249 SAVE_CSR_8SS:
                        08E4  2250
    50    51    0D    78  08E4  2251          ASHL    #13,R1,R0               ; Multiply node # by 8K so that
                        08E8  2252                                            ;  R0 contains the offset, into I/O
                        08E8  2253                                            ;  space for this node's register space.
                        08E8  2254
50    20000000 8F    C0  08E8  2255          ADDL    #IO8SS$AL_IOBASE,R0     ; R0 => Node's register space.
      5C AB    50    D0  08EF  2256          MOVL    R0,RPB$L_ADPPHY(R11)    ; Store in RPB.
                        08F3  2257
         53    60    D0  08F3  2258          MOVL    (R0),R3                 ; Read node Device type register.
      53 0102  8F    B1  08F6  2259          CMPW    #<NDT$_BUA&^xFFFF>,R3   ; See if a UNIBUS adapter. NOTE
                        08FB  2260                                            ;  anding with ^xFFFF is to avoid
                        08FB  2261                                            ;  truncation warning from assembler.
               03     12  08FB  2262          BNEQ    10$                     ; If NOT, branch around.
         50    52    D0  08FD  2263          MOVL    R2,R0                   ; If UNIBUS device, R0 => CSR.
                        0900  2264 10$:
      54 AB    50    D0  0900  2265          MOVL    R0,RPB$L_CSRPHY(R11)    ; Save address of CSR in RPB.
               05     05  0904  2266          RSB
```

```
                        .SBTTL  INIT_ADAP, CPU-specific adapter initialization routine

;++
;
; Functional description:
;
;       One routine per CPU implementation follows. Each routine
;       initializes the adapter for the boot device, if the adapter
;       requires such an initialization.
;
; Inputs:
;
;       R0      - physical address of the adapter
;       R1-R6   - scratch
;       R7      - address of the SCB
;       R8      - address of the CPU-specific table
;       R9      - scratch
;       R10     - address of the 1st unused byte of good memory
;       R11     - address of the RPB
;
; Implicit inputs:
;
;       If the value in R0 is identical to the value in RPB$L_CSRPHY,
;       the adapter is for a UNIBUS. Otherwise, the adapter is for a
;       MASSBUS.
;
; Outputs:
;
;       R7-R8, R10-R11, and SP must be preserved.
;
; Implicit outputs:
;
;       The boot device's adapter is initialized.
;
;--
```

```
                        0905  2304          .SBTTL  INIT_ADP_780, Initialize 11/780 boot device adapter
                        0905  2305          .SBTTL  INIT_ADP_790, Initialize 11/790 boot device adapter
                        0905  2306
                        0905  2307 ;++
                        0905  2308 ;
                        0905  2309 ; INIT_ADP_780, Initialize boot device adapter on the 11/780
                        0905  2310 ; INIT_ADP_790, Initialize boot device adapter on the 11/790
                        0905  2311 ;
                        0905  2312 ; Implicit inputs:
                        0905  2313 ;
                        0905  2314 ;       Both the UNIBUS and MASSBUS adapters must be initialized for
                        0905  2315 ;       the 11/780 and the 11/790.  The initialization bit is in the same position
                        0905  2316 ;       in both the UNIBUS and MASSBUS adapter configuration register.
                        0905  2317 ;
                        0905  2318 ;       This routine sets the initialization bit in the CR; then loops
                        0905  2319 ;       to wait for initialization completion for the UNIBUS; assumes
                        0905  2320 ;       immediate completion for the MASSBUS.
                        0905  2321 ;
                        0905  2322 ;--
                        0905  2323
                        0905  2324 INIT_ADP_780:                          ; Initialize 11/780 adapters.
55      097C'CF   DE    0905  2325          MOVAL   W^MCHK_780,R5           ; Get address of 780 mcheck handler.
              05  11    090A  2326          BRB     INIT_ADP_COMMON         ; Go to common code.
                        090C  2327
                        090C  2328 INIT_ADP_790:
55      0994'CF   DE    090C  2329          MOVAL   W^MCHK_790,R5           ; Get address of 790 mcheck handler.
                        0911  2330
                        0911  2331 INIT_ADP_COMMON:
              01  D0    0911  2332          MOVL    #MBA$M_CR_INIT,-        ; Set the initialize bit. Bit
            04  A0      0913  2333                  MBA$L_CR(R0)            ; # is same for MBA and UBA.
      54    AB  50  D1  0915  2334          CMPL    R0,RPB$L_CSRPHY(R11)    ; Is ADP CSR address = device
                        0919  2335                                          ; CSR address?
              01  12    0919  2336          BNEQU   WAIT_UBA                ; No. This is a UBA.
                  05    091B  2337          RSB                             ; Yes. MBA. Return, no wait.
                        091C  2338
                        091C  2339 ;
                        091C  2340 ; For a UNIBUS adapter, must wait for a CSR bit to be written by the
                        091C  2341 ; adapter before knowing that initialization is complete.
                        091C  2342 ;
                        091C  2343
                        091C  2344 WAIT_UBA:                            ; Wait for the UBA.
                        091C  2345          .ENABLE LSB
00010000      8F    D3  091C  2346          BITL    #UBA$M_CSR_UBIC,-       ; Did the adapter set the init
              60        0922  2347                  UBA$L_CSR(R0)           ; complete bit?
              F7  13    0923  2348          BEQL    WAIT_UBA                ; No, wait longer.
                        0925  2349
                        0925  2350 ;
                        0925  2351 ; Now check for any UNIBUS memory that may be on the adapter. If found
                        0925  2352 ; disable the corresponding map registers.
                        0925  2353 ;
                        0925  2354
            04  A7  DD  0925  2355          PUSHL   4(R7)                   ; Save machine check vector
      04    A7  55  D0  0928  2356          MOVL    R5,4(R7)                ; Replace with a temporary
        56      5E  D0  092C  2357          MOVL    SP,R6                   ; Save stack pointer
0003FFFF      8F  CB  092F  2358          BICL3   #^X0003FFFF,-
      51    54  AB      0935  2359                  RPB$L_CSRPHY(R11),R1    ; Get the start of UNIBUS space
          52      51  D0  0938  2360          MOVL    R1,R2                   ; Copy
```

```
           53   0003C000 E2   9E  093B          MOVAB   ^X3C000(R2),R3            ; Last 8KB of memory
        04 A0   7C000000 8F   D0  0942          MOVL    #^X7C000000,UBA$L_CR(R0); Disable all UNIBUS map regs
                          62  B5  094A 10$:     TSTW    (R2)                      ; See if anything responds
             08 A0     08 A0  D0  094C          MOVL    UBA$L_SR(R0),UBA$L_SR(R0) ; Clear and read status
                          10  12  0951          BNEQ    20$                       ; Nothing there, try higher
                   52     51  D1  0953 15$:     CMPL    R1,R2                     ; Found memory, first time in?
                          06  13  0956          BEQL    17$                       ; Yes, skip next test
                     0000'CF  D5  0958          TSTL    BOO$GL_UMR_DIS            ; Any registers already disabled?
                          55  13  095C          BEQL    UNI_MEM_ERR               ; No, memory not start at 0
            0000'CF       10  C0  095E 17$:     ADDL2   #16,BOO$GL_UMR_DIS        ; Up the count of registers to disable
    FFDD 52   00002000 8F 53  F1  0963 20$:     ACBL    R3,#^X2000,R2,10$         ; Continue thru the entire space
            0000'CF       16  78  096D          ASHL    #22,BOO$GL_UMR_DIS,-      ; Div by 16 and shift left 26 bits
                  04 A0           0972                  UBA$L_CR(R0)              ; Disable the UMR's
                  04 A7     8E D0 0974          POPL    4(R7)                     ; Restore machine check vector
                              05  0978          RSB                               ; Initialization complete.
                                  0979
                                  0979          .ALIGN  LONG
                                  097C MCHK_780:
                54        30  DB  097C          MFPR    #PR$_SBIFS,R4             ; Get SBI fault status register
             00 54        19  E5  097F          BBCC    #25,R4,25$                ; Clear "error 1st pass" bit
                30        54  DA  0983 25$:     MTPR    R4,#PR$_SBIFS             ; Write back to clear SBI fault
             54     04    AE  D0  0986          MOVL    4(SP),R4                  ; Pick up summary parameter
                5E        56  D0  098A          MOVL    R6,SP                     ; Clear off frame
                05        54  91  098D          CMPB    R4,#5                     ; Is it Read Data Substitute?
                          C1  13  0990          BEQL    15$                       ; Yes, then it is a read w/bad parity
                          CF  11  0992          BRB     20$                       ; No, its nonexitent
                                  0994
                                  0994          .ALIGN  LONG
                                  0994 MCHK_790:
       54     0000004A 8F     DB  0994          MFPR    #PR$_EHSR,R4              ; Get error handling status register.
            00 54         05  E5  099B          BBCC    #EHSR$V_VMS,R4,30$        ; Clear "VMS entered" bit.
      0000004A 8F         54  DA  099F 30$:     MTPR    R4,#PR$_EHSR              ; Write back to clear machine check.
             54     3C    AE  D0  09A6          MOVL    MCF790$L_MSTAT2(SP),R4    ; Pick up memory status register.
                5E        56  D0  09AA          MOVL    R6,SP                     ; Clear mcheck frame off stack.
             B2 54        02  E0  09AD          BBS     #MSTAT2$V_IOBUFF,R4,20$   ; Branch if NXM => no Unibus memory.
                          A0  11  09B1          BRB     15$                       ; Ignore any other error.
                                  09B3          .DISABLE LSB
                                  09B3
                                  09B3 UNI_MEM_ERR:
                                  09B3          ERROR   </%BOOT-F-UNIBUS memory does not start at 0/>
```

```
                        09E0  2402          .SBTTL  INIT_ADP_750, Initialize boot device 11/750 adapter
                        09E0  2403          .SBTTL  INIT_ADP_730, Initialize boot device 11/730 adapter
                        09E0  2404
                        09E0  2405 ;++
                        09E0  2406 ;
                        09E0  2407 ; INIT_ADP_750, Initialize boot device adapter on the 11/750
                        09E0  2408 ; INIT_ADP_730, Initialize boot device adapter on the 11/730
                        09E0  2409 ;
                        09E0  2410 ; Implicit inputs:
                        09E0  2411 ;
                        09E0  2412 ;       The massbus is initialized by setting the init bit in the
                        09E0  2413 ;       MBA adapter control register.  The unibus is initialized by
                        09E0  2414 ;       setting the UB reset IPR.  (On the 11/730 the adapter of
                        09E0  2415 ;       the boot device will always be a unibus.)
                        09E0  2416 ;
                        09E0  2417 ;--
            00000024    09E0  2418          MCK_BER = ^X24                  ; Offset into machine check frame
                        09E0  2419                                          ;  for Bus Error Register
            00000003    09E0  2420          NEX     = 3                     ; Bit position for non-existent mem
            00000800    09E0  2421          STEP    = ^X800                 ; Granularity of the step in memory
                        09E0  2422                                          ;  probe algorithm
                        09E0  2423
                        09E0  2424 INIT_ADP_750:                            ; Initialize 11/750 adapters.
                        09E0  2425 INIT_ADP_730:                            ; Initialize the 11/730 adapters.
                        09E0  2426          .ENABLE LSB
    54 AB    50    D1   09E0  2427          CMPL    R0,RPB$L_CSRPHY(R11)    ; Is ADP CSR addr = device CSR addr?
          05    12      09E4  2428          BNEQ    10$                     ; No, must be UNIBUS
          01    D0      09E6  2429          MOVL    #MBA$M_CR_INIT,-        ; Set the initialize bit
       04 A0            09E8  2430                  MBA$L_CR(R0)            ;
                05      09EA  2431          RSB                             ; Done
                        09EB  2432
       37    00    DA   09EB  2433 10$:     MTPR    #0,#PR$_UBRESET         ; Reset the UNIBUS adapter.
                        09EE  2434
                        09EE  2435 ;
                        09EE  2436 ; Now check for any UNIBUS memory that may be on the adapter. If found
                        09EE  2437 ; disable the corresponding map registers.
                        09EE  2438 ;
                        09EE  2439
         51    01F0 8F    3C   09EE  2440          MOVZWL  #496,R1                 ; Pick up number to disable
         52    0800 C0    DE   09F3  2441          MOVAL   UBA$L_MAP(R0),R2        ; Address of first
                  82    D4   09F8  2442 20$:     CLRL    (R2)+                   ; Invalidate it
                  FB 51 F5   09FA  2443          SOBGTR  R1,20$                  ; Loop until done
                  04 A7 DD   09FD  2444          PUSHL   4(R7)                   ; Save machine check vector
       04 A7    5C'AF DE   0A00  2445          MOVAL   B^MCHK_750,4(R7)        ; Replace with a temporary
             56    5E    D0   0A05  2446          MOVL    SP,R6                   ; Save stack pointer
          0003FFFF 8F    CB   0A08  2447          BICL3   #^X0003FFFF,-
            51    54 AB        0A0E  2448                  RPB$L_CSRPHY(R11),R1    ; Get the start of UNIBUS space
             52    51    D0   0A11  2449          MOVL    R1,R2                   ; Copy
    0208'CF    F800 C1    9E   0A14  2450          MOVAB   -STEP(R1),LAST_MEM      ; Initialize cell
     53    0003C000 E2    9E   0A1B  2451          MOVAB   ^X3C000(R2),R3          ; Last 8KB of memory
                  62    B5   0A22  2452 30$:     TSTW    (R2)                    ; See if anything responds
             52    51    D1   0A24  2453 40$:     CMPL    R1,R2                   ; First time in?
                  10    13   0A27  2454          BEQL    45$                     ; Yes, skip next test
             0000'CF    D5   0A29  2455          TSTL    B00$GL_UMR_DIS          ; Any registers already disabled?
                  19    13   0A2D  2456          BEQL    55$                     ; No, memory not start at 0
          14 020C'CF    E8   0A2F  2457          BLBS    MEM_FLAG,55$            ; Yes, now the far side of a hole
          0208'CF    52    D0   0A34  2458          MOVL    R2,LAST_MEM             ; Update counter
```

```
                0000'CF    04    C0  0A39  2459 45$:    ADDL2   #STEP/512,B00$GL_UMR_DIS; Up the count of registers to disable
    FFDA 52  00000800 8F    53    F1  0A3E  2460 50$:    ACBL    R3,#STEP,R2,30$         ; Continue thru the entire space
                  52    0208'CF    C2  0A48  2461 55$:    SUBL2   LAST_MEM,R2             ; Find size of hole
                                       0A4D  2462 ;
                                       0A4D  2463 ; NOTE: This test is based on the fact that the bootdriver will only
                                       0A4D  2464 ;       allow a maximum of 127 pages in a single IO.  If that assumption
                                       0A4D  2465 ;       changes, this test will have to change.
                                       0A4D  2466 ;
            00010800 8F    52    D1  0A4D  2467          CMPL    R2,#<128*512>+STEP      ; Is it big enough?
                           23    1F  0A54  2468          BLSSU   TOO_MEM_ERR             ; No, give error
                       04 A7 8ED0  0A56  2469          POPL    4(R7)                   ; Restore old machine check handler
                             05    0A5A  2470          RSB                             ; Done
                                       0A5B  2471
                                       0A5B  2472          .ALIGN  LONG
                                       0A5C  2473 MCHK_750:                            ; Temporary machine check handler
                          26    0F    DA  0A5C  2474          MTPR    #^XF,#PR$_MCESR         ; Reset machine check
                          54    08    D0  0A5F  2475          MOVL    #<1@NEX>,R4             ; Set up
                          0C    6E    D1  0A62  2476          CMPL    (SP),#^X0C              ; Is this a 730 frame?
                                04    13  0A65  2477          BEQL    60$                     ; Yes, no further check
                        54    24 AE    D0  0A67  2478          MOVL    MCK_BER(SP),R4          ; Save Bus error register
                           5E    56    D0  0A6B  2479 60$:     MOVL    R6,SP                   ; Clear machine check stack frame
                        B2 54    03    E1  0A6E  2480          BBC     #NEX,R4,40$             ; Branch if parity error on Umem
                     020C'CF    01    D0  0A72  2481          MOVL    #1,MEM_FLAG             ; Else non-existent memory, set flag
                                C5    11  0A77  2482          BRB     50$                     ; Continue in line
                                       0A79  2483          .DISABLE LSB                    ;
                                       0A79  2484
                                       0A79  2485 TOO_MEM_ERR:
                                       0A79  2486          ERROR   </%BOOT-F-Too much UNIBUS memory/>
```

```
                     0A9B  2488          .SBTTL  INIT_ADP_8SS, Initialize boot device 11/8SS adapter
                     0A9B  2489
                     0A9B  2490 ;++
                     0A9B  2491 ;
                     0A9B  2492 ; INIT_ADP_8SS, Initialize boot device adapter on the 11/8SS
                     0A9B  2493 ;
                     0A9B  2494 ; Implicit inputs:
                     0A9B  2495 ;
                     0A9B  2496 ;
                     0A9B  2497 ;--
                     0A9B  2498
                     0A9B  2499 INIT_ADP_8SS:                                ; Initialize 11/8SS adapters.
                     0A9B  2500
     51    20 AB    01   78  0A9B  2501          ASHL    #1,RPB$L_BOOTR1(R11),R1 ; R1 = 1 in bit position corresponding
                     0AA0  2502                                               ;  to Adapter BI Node.
0000005F 8F    51    DA   0AA0  2503          MTPR    R1,#PR8SS$_BIINIT       ; Send BI INIT command to this node.
       54 AB    50    D1   0AA7  2504          CMPL    R0,RPB$L_CSRPHY(R11)    ; See if a BUA.
                00    13   0AAB  2505          BEQL    10$                     ; EQL implies NO.
                     0AAD  2506 10$:
                     05   0AAD  2507          RSB                             ; Return to caller.
```

```
                                0AAE  2509          .SBTTL  OPEN_UCODE_FILE, Find and open a ucode file on console
                                0AAE  2510
                                0AAE  2511 ;++
                                0AAE  2512 ;
                                0AAE  2513 ; Functional description:
                                0AAE  2514 ;
                                0AAE  2515 ;       This routine contains code common to all file opens for any file
                                0AAE  2516 ;       on the console.  It also handles errors.
                                0AAE  2517 ;
                                0AAE  2518 ; Inputs:
                                0AAE  2519 ;
                                0AAE  2520 ;       R2      Boot device type
                                0AAE  2521 ;       R3      Address of vector of address
                                0AAE  2522 ;
                                0AAE  2523 ; Outputs:
                                0AAE  2524 ;
                                0AAE  2525 ;       R0      Status
                                0AAE  2526 ;
                                0AAE  2527 ;--
                                0AAE  2528
                                0AAE  2529 OPEN_UCODE_FILE:
            0CEF'CF    83   D0  0AAE  2530          MOVL    (R3)+,UCODE_NAME         ; Set the error name
            51    83   53   C1  0AB3  2531          ADDL3   R3,(R3)+,R1              ; Pick up name for ucode file
                  50   81   9A  0AB7  2532          MOVZBL  (R1)+,R0                 ; Size to R0
                  7E   50   7D  0ABA  2533          MOVQ    R0,-(SP)                 ; Form descriptor
            7E    83   53   C1  0ABD  2534          ADDL3   R3,(R3)+,-(SP)           ; Temp area for the stat block
                       5A   DD  0AC1  2535          PUSHL   R10                      ; Start of memory buffer area
                     08 AE   7F  0AC3  2536          PUSHAQ  8(SP)                    ; File descriptor
            0000'CF    05   FB  0AC6  2537          CALLS   #5,RTF$OPENFILE          ; Look it up on the console medium
                    24 50   E8  0ACB  2538          BLBS    R0,40$                   ; Success, try to read it in
                    05 63   E8  0ACE  2539          BLBS    (R3),10$                 ; If set, all errors fatal
                  52   20   91  0AD1  2540          CMPB    #BTD$K_HSCCI,R2          ; Booting off the CI?
                       06   12  0AD4  2541          BNEQ    20$                      ; No, leave severity
    0CCF'CF    46   8F   90  0AD6  2542 10$:       MOVB    #^A/F/,UCODE_SEVER       ; Yes, change to FATAL
                  7E   7C  0ADC  2543 20$:       CLRQ    -(SP)                    ; Null input buffer descriptor
                0CC7'CF    9F  0ADE  2544          PUSHAB  UCODE_FAIL               ; Error text
              0000'CF    03   FB  0AE2  2545          CALLS   #3,BOO$READPROMPT        ; Report the problem
                    07 63   E8  0AE7  2546          BLBS    (R3),30$                 ; If set, all errors fatal
                       50   D4  0AEA  2547          CLRL    R0                       ; Set possible return indicator
                  52   20   91  0AEC  2548          CMPB    #BTD$K_HSCCI,R2          ; Booting off the CI?
                       01   12  0AEF  2549          BNEQ    40$                      ; No, finish the boot anyway
                            00  0AF1  2550 30$:       HALT                             ; Yes, **** FATAL ERROR ****
                                0AF2  2551
                            05  0AF2  2552 40$:       RSB
```

```
                        0AF3  2554          .SBTTL  FIND_CI, CPU-specific routine to locate CI port
                        0AF3  2555
                        0AF3  2556 ;++
                        0AF3  2557 ;
                        0AF3  2558 ; Functional description:
                        0AF3  2559 ;
                        0AF3  2560 ;       One routine per CPU implementation follows. Each routine
                        0AF3  2561 ;       attempts to locate a CI port on the CPU.
                        0AF3  2562 ;
                        0AF3  2563 ; Inputs:
                        0AF3  2564 ;
                        0AF3  2565 ;       R7      Address of SCB
                        0AF3  2566 ;
                        0AF3  2567 ; Implicit inputs:
                        0AF3  2568 ;
                        0AF3  2569 ;       11/780 and 11/750:      RPB$B_CONFREG array
                        0AF3  2570 ;       11/730:                 No CI implementation exists for the 11/730
                        0AF3  2571 ;       11/790:                 Knowledge of I/O space layout
                        0AF3  2572 ;
                        0AF3  2573 ; Outputs:
                        0AF3  2574 ;
                        0AF3  2575 ;       CC = EQL means no CI on the system; CC = NEQ means CI present.
                        0AF3  2576 ;       R0,R1 destroyed,
                        0AF3  2577 ;       All other registers preserved.
                        0AF3  2578 ;
                        0AF3  2579 ;--
                        0AF3  2580
                        0AF3  2581 ;
                        0AF3  2582 ; For the 11/780 and 11/750, the size and test memory routines have set
                        0AF3  2583 ; up an array in the RPB that contains the type code of each adapter on
                        0AF3  2584 ; the system.  Search that array for a CI adapter type code.
                        0AF3  2585 ;
                        0AF3  2586 FIND_CI_780:
                        0AF3  2587 FIND_CI_750:
          10    38  3A  0AF3  2588          LOCC    S^#NDT$_CI,#16,-         ; Look for the CI780
    0090  CB          0AF6  2589                  RPB$B_CONFREG(R11)       ; in the CONFREG array.
                  05  0AF9  2590 10$:    RSB                              ; Return w/ condition code set
                      0AFA  2591                                           ; appropriately.
                      0AFA  2592
                      0AFA  2593 FIND_CI_730:
              50  D4  0AFA  2594          CLRL    R0                       ; Signal no CI730.
                  05  0AFC  2595          RSB
                      0AFD  2596
                      0AFD  2597 FIND_CI_8SS:
              50  D4  0AFD  2598          CLRL    R0                       ; Signal no CI8SS for now.
                  05  0AFF  2599          RSB
                      0B00  2600
                      0B00  2601 ;
                      0B00  2602 ; For the 11/790, the array of adapter type codes isn't set up until later
                      0B00  2603 ; in the boot process.  Search the SBI nexuses for a CI adapter.
                      0B00  2604 ;
                      0B00  2605          .ENABLE LSB
                      0B00  2606 FIND_CI_790:
              1C  BB  0B00  2607          PUSHR   #^M<R2,R3,R4>            ; Save some registers.
        04    A7  DD  0B02  2608          PUSHL   4(R7)                    ; Save machine check vector.
04    A7  0B75'CF   DE  0B05  2609          MOVAL   NXMCHK_790+1,4(R7)       ; Replace with a temporary.
              54  5E  D0  0B08  2610          MOVL    SP,R4                    ; Save stack pointer.
```

```
                51   20000000 8F    D0   0B0E  2611          MOVL    #IO790$AL_IOA0,R1       ; Get address space for 1st ABUS slot.
            00000041 8F        51    DA   0B15  2612 10$:     MTPR    R1,#PR790$_PAMLOC       ; Request PAMM code for this phys addr.
                50   00000040 8F    DB   0B1C  2613          MFPR    #PR790$_PAMACC,R0       ; Read the PAMM code.
                     50     04 00    ED   0B23  2614          CMPZV   #PAMM$V_CODE,#PAMM$S_CODE, -    ; Is there an adapter present?
                                OF         0B27  2615                  R0,#PAMM$C_NEXM
                                27   13    0B28  2616          BEQL    30$                     ; No, go to next slot.
                50   00080000 E1    D0   0B2A  2617          MOVL    IO790$AL_IOACR(R1),R0   ; Read ABUS configuration register.
            01  50     04 04    ED   0B31  2618          CMPZV   #4,#4,R0,#IO790$C_SBIA  ; Is there an SBI here?
                                19   12    0B36  2619          BNEQ    30$                     ; No, go to next slot.
                                           0B38  2620 ;
                                           0B38  2621 ; Found an SBI.  Now look for a CI adapter.
                                           0B38  2622 ;
                     52   2000 C1    DE   0B38  2623          MOVAL   IO790$AL_PERNEX(R1),R2  ; Get addr of SBI TR #1 (#0 is unused).
                          50   62    D0   0B3D  2624 20$:     MOVL    (R2),R0                 ; Read config. register on SBI.
                          38   50    91   0B40  2625          CMPB    R0,S^#NDT$_CI           ; Is this a CI adapter?
                               24    13   0B43  2626          BEQL    40$                     ; Yes; we found one.
                     52   2000 C2    DE   0B45  2627 25$:     MOVAL   IO790$AL_PERNEX(R2),R2  ; Step to next TR.
            20       52     06 0C    ED   0B4A  2628          CMPZV   #12,#6,R2,#^X20         ; Have we looked at all slots on this SBI?
                                EC   19    0B4F  2629          BLSS    20$                     ; Not yet, go on to the next.
                                           0B51  2630 ;
                                           0B51  2631 ; Looked unsuccessfully at all slots on this SBI.  See if there's another SBI.
                                           0B51  2632 ;
                51   02000000 E1    DE   0B51  2633 30$:     MOVAL   IO790$AL_PERABS(R1),R1  ; Step to next ABUS slot.
                51   28000000 8F    D1   0B58  2634          CMPL    #IO790$AL_IOA0+<IO790$AL_PERABS*4>, -    ; Have we looked at
                                           0B5F  2635                  R1                      ; all possible ABUS slots?
                                B4   14    0B5F  2636          BGTR    10$                     ; Not yet, go on to next.
                     04 A7      8E   D0    0B61  2637          MOVL    (SP)+,4(R7)             ; Restore machine check handler to SCB.
                                50   D4    0B65  2638          CLRL    R0                      ; Signal failure.
                                07   11    0B67  2639          BRB     50$                     ; Return.
                                           0B69  2640 ;
                                           0B69  2641 ; Come here on finding a CI adapter.
                                           0B69  2642 ;
                     04 A7      8E   D0    0B69  2643 40$:     MOVL    (SP)+,4(R7)             ; Restore machine check handler to SCB.
                          50    01   D0    0B6D  2644          MOVL    #1,R0                   ; Signal success.
                                1C   BA    0B70  2645 50$:     POPR    #^M<R2,R3,R4>           ; Restore registers.
                                     05    0B72  2646          RSB
                                           0B73  2647 ;
                                           0B73  2648 ; Non-existent memory machine check handler.
                                           0B73  2649 ;
                                           0B73  2650          .ALIGN  LONG
                                           0B74  2651 NXMCHK_790:
            50   0000004A 8F    DB   0B74  2652          MFPR    #PR$_EHSR,R0            ; Get error handling status register.
                      00 50    05   E5    0B7B  2653          BBCC    #EHSR$V_VMS,R0,100$     ; Clear "VMS entered" bit.
            0000004A  8F       50   DA    0B7F  2654 100$:    MTPR    R0,#PR$_EHSR            ; Write back to clear machine check.
                           5E   54   D0    0B86  2655          MOVL    R4,SP                   ; Clear mcheck frame off stack.
                                BA   11    0B89  2656          BRB     25$                     ; Scan next SBI slot.
                                           0B8B  2657
                                           0B8B  2658          .DISABLE        LSB
```

```
                                                0B8B  2660          .SBTTL  Strings used in File I/O
                                                0B8B  2661
                                                0B8B  2662 VMSFILE:                             ; Name of standard secondary
53 59 53 5D 45 58 45 53 59 53 5B 00'            0B8B  2663          .ASCIC  /[SYSEXE]SYSBOOT.EXE/ ; bootstrap image file.
               45 58 45 2E 54 4F 4F 42          0B97
                                     13         0B8B
                                                0B9F  2664
                                                0B9F  2665 DIAGFILE:                            ; Name of standard diagnostic
44 5D 54 4C 49 41 4D 53 59 53 5B 00'            0B9F  2666          .ASCIC  /[SYSMAINT]DIAGBOOT.EXE/ ; secondary bootstrap image.
   45 58 45 2E 54 4F 4F 42 47 41 49             0BAB
                                     16         0B9F
                                                0BB6  2667
                                                0BB6  2668 FIL$GT_TOPSYS::
                        30 53 59 53 00'         0BB6  2669          .ASCIC  /SYS0/                  ; Default top level system dir name
                                     04         0BB6
                              00000BC0          0BBB  2670          .BLKB   10-<.-FIL$GT_TOPSYS>    ; Fill to 10 bytes
                                                0BC0  2671
                                                0BC0  2672 NAMEPROMPT:                          ; Prompt string for secondary
00 3A 65 6C 69 66 74 6F 6F 42 0A 0D             0BC0  2673          .ASCIZ  <CR><LF>/Bootfile:/     ; boot file name.
                                                0BCC  2674
                                                0BCC  2675 DEVPROMPT:                           ; Prompt string for secondary
63 69 76 65 64 20 74 6F 6F 42 0A 0D             0BCC  2676          .ASCIZ  <CR><LF>/Boot device name (ddcu):/
75 63 64 64 28 20 65 6D 61 6E 20 65             0BD8
                        00 3A 29                0BE4
                                                0BE7  2677                                      ; device mnemonic.
                                                0BE7  2678
                                                0BE7  2679 SWITCHPROMPT:                        ; Prompt string for switching console media
68 74 20 74 72 65 73 6E 49 07 0A 0D             0BE7  2680          .ASCII  <CR><LF><7>/Insert the first standalone system volume /
6E 61 74 73 20 74 73 72 69 66 20 65             0BF3
65 74 73 79 73 20 65 6E 6F 6C 61 64             0BFF
            20 65 6D 75 6C 6F 76 20 6D          0C0B
59 22 20 72 65 74 6E 65 20 64 6E 61             0C14  2681          .ASCIZ  /and enter "YES" when ready: /
61 65 72 20 6E 65 68 77 20 22 53 45             0C20
                        00 20 3A 79 64          0C2C
                                                0C31  2682
                                                0C31  2683 ; Message to remove current volume.  Note that besides giving the user some useful
                                                0C31  2684 ; feedback, this message causes the volume to be rewound so that the volume label
                                                0C31  2685 ; can be read.  This will prevent device timeouts (processor timeouts) on TU58 drive
                                                0C31  2686 ; since the processor has been seen to timeout waiting for a TU58 to rewind.
                                                0C31  2687
20 65 73 61 65 6C 50 07 0A 0A 0A 0D             0C31  2688 REMOVEPROMPT1:  .ASCIZ  <CR><LF><LF><LF><7>/Please remove the volume "/<7>
76 20 65 68 74 20 65 76 6F 6D 65 72             0C3D
            00 07 22 20 65 6D 75 6C 6F          0C49
63 20 65 68 74 20 6D 6F 72 66 20 22             0C52  2689 REMOVEPROMPT2:  .ASCIZ  /" from the console device./<7><CR><LF>
63 69 76 65 64 20 65 6C 6F 73 6E 6F             0C5E
                  00 0A 0D 07 2E 65             0C6A
                                                0C70  2690
                                                0C70  2691 ; Message to resume loading
                                                0C70  2692
6C 20 67 6E 69 6D 75 73 65 52 0A 0D             0C70  2693 RESUMEPROMPT1:  .ASCIZ  <CR><LF>/Resuming load operation on volume "/
6F 69 74 61 72 65 70 6F 20 64 61 6F             0C7C
20 65 6D 75 6C 6F 76 20 6E 6F 20 6E             0C88
                              00 22             0C94
74 73 20 65 73 61 65 6C 70 20 2C 22             0C96  2694 RESUMEPROMPT2:  .ASCIZ  /", please stand by . . ./<CR><LF><LF>
2E 20 2E 20 2E 20 79 62 20 64 6E 61             0CA2
                        00 0A 0A 0D             0CAE
                                                0CB2  2695
```

```
                                OCB2  2696 CI_UCODE_FILE:
      4E 49 42 2E 30 38 37 49 43 00'  OCB2  2697         .ASCIC  /CI780.BIN/             ; Name of the binary file
                                09    OCB2
                                OCBC  2698
                                OCBC  2699 PCS_UCODE_FILE:
   4E 49 42 2E 30 35 37 53 43 50 00'  OCBC  2700         .ASCIC  /PCS750.BIN/            ; Name of the binary file
                                0A    OCBC
                                OCC7  2701
                                OCC7  2702 UCODE_FAIL:
6E 55 2D 57 2D 54 4F 4F 42 25 0A 0D  OCC7  2703         .ASCIZ  <CR><LF>/%BOOT-W-Unable to locate ucode file - xxxx/<CR><LF>
61 63 6F 6C 20 6F 74 20 65 6C 62 61  OCD3
6C 69 66 20 65 64 6F 63 75 20 65 74  OCDF
   00 0A 0D 78 78 78 78 20 2D 20 65  OCEB
                    00000CCF  OCF6  2704 UCODE_SEVER = UCODE_FAIL+8              ; Severity code
                    00000CEF  OCF6  2705 UCODE_NAME = UCODE_FAIL+40              ; File name
                              OCF6  2706
                              OCF6  2707 PCS_UCODE_STAT:
                    00000000  OCF6  2708         .LONG   0
                    00000000  OCFA  2709         .LONG   0
                              OCFE  2710
                              OCFE  2711 PCS_UCODEV:
                 20 53 43 50  OCFE  2712         .ASCII  /PCS /
                    FFFFFFBA  0D02  2713         .LONG   PCS_UCODE_FILE-.
                    FFFFFFF0  0D06  2714         .LONG   PCS_UCODE_STAT-.
                    00000001  0D0A  2715         .LONG   1                       ; Fatal
                              0D0E  2716
                              0D0E  2717 CI_UCODEV:
                 20 20 49 43  0D0E  2718         .ASCII  /CI  /
                    FFFFFFA0  0D12  2719         .LONG   CI_UCODE_FILE-.
                    FFFFF4EA' 0D16  2720         .LONG   CI_UCODE_STAT-.
                    00000000  0D1A  2721         .LONG   0                       ; Non-fatal
                              0D1E  2722
                              0D1E  2723
```

```
                        2725          .SBTTL  Unexpected machine check handler, DEBUG labels
                        2726
                        2727          SET_PSECT                             ; Back to default PSECT
                        2728
                        2729 ;
                        2730 ; Define handlers needed by XDELTA.
                        2731 ;
                        2732
                        2733          .ALIGN  LONG                          ; All handlers longword-aligned.
                        2734
                        2735          .IF     DF,DEBUG                      ; If debugging,
                        2736
                        2737 EXE$ACVIOLAT::                                 ; Access violation vector.
                        2738 EXE$BREAK::                                    ; BPT vector.
                        2739 EXE$ROPRAND::                                  ; Reserved operand vector.
                        2740 EXE$TBIT::                                     ; Trace trap vector.
                        2741 MMG$PAGEFAULT::                               ; Pagefault exception vector.
                        2742
                        2743          .ENDC                                 ; End of debugging conditional.
                        2744
                        2745 ;
                        2746 ; Fault handler for unexpected exception conditions during bootstrap.
                        2747 ;
                        2748
                        2749 BOOT_FAULT:                                    ; Handler for most of SCB.
                        2750          ERROR   </%BOOT-F-Unexpected Exception/>
                        2751                                                ; Output error message.
                        2752
                        2753          .ALIGN  LONG                          ; All handlers longword-aligned.
                        2754
                        2755 ;
                        2756 ; Machine check handler.
                        2757 ;
                        2758
                        2759 UNEXP_MCHK::
                        2760          ERROR   </%BOOT-F-Unexpected Machine Check/>
                        2761                                                ; Output error message.
                        2762
                        2763 ;
                        2764 ; Labels required by XDELTA.
                        2765 ;
                        2766
                        2767          .IF     DF,DEBUG                      ; If debugging, define labels.
                        2768
                        2769 INI$RDONLY::                                   ; Dummy change protection
                        2770 INI$WRITABLE::                                 ; routines.
                        2771 SYSL$CLRSBIA::                                ; Dummy routine to clear SBIA errors.
                        2772          RSB                                   ; Just return.
                        2773
                        2774 EXE$GL_FLAGS::                                ; Dummy flags longword.
                        2775 EXE$GL_SCB::                                  ; Dummy SCB address pointer.
```

```
00000000  0169  2776 EXE$V_SIMULATOR == 0                    ; This is not a simulator.
          0169  2777
          0169  2778 PFN$AB_STATE::
          0169  2779 PFN$AB_TYPE::
          0169  2780 PFN$AL_BAK::
          0169  2781 PFN$AL_PTE::
          0169  2782 PFN$Ax_BLINK::
          0169  2783 PFN$Ax_FLINK::
          0169  2784 PFN$AW_REFCNT::
          0169  2785 PFN$AW_SWPVBN::
          0169  2786
          0169  2787 SYS$IOBASE::
          0169  2788 SCH$GL_CURPCB::
          0169  2789 SCH$GL_PCBVEC::
          0169  2790
          0169  2791 XDS$GT_WORD_PFN::
00000000  0169  2792         .LONG   0
          016D  2793
          016D  2794         .ENDC
```

```
                        016D  2796          .SBTTL  Error message subroutine
                        016D  2797
                        016D  2798 ;++
                        016D  2799 ;
                        016D  2800 ; Functional description:
                        016D  2801 ;
                        016D  2802 ;       This routine outputs a descriptive error message to the
                        016D  2803 ;       console terminal. Then the routine restores the original
                        016D  2804 ;       register settings from the time that the primary bootstrap
                        016D  2805 ;       gained control, and executes a HALT instruction.
                        016D  2806 ;
                        016D  2807 ; Inputs:
                        016D  2808 ;
                        016D  2809 ;       SP      - points to address of message text
                        016D  2810 ;
                        016D  2811 ; Implicit inputs:
                        016D  2812 ;
                        016D  2813 ;       The RPB contains the contents of the original boot registers,
                        016D  2814 ;       and the original stack pointer.
                        016D  2815 ;
                        016D  2816 ; Outputs:
                        016D  2817 ;
                        016D  2818 ;       Registers R0-R6, R10-R11, and AP are restored from the RPB.
                        016D  2819 ;
                        016D  2820 ; Implicit outputs:
                        016D  2821 ;
                        016D  2822 ;       When the HALT instruction executes, the console terminal reports
                        016D  2823 ;       a halt and prompts the user for input.
                        016D  2824 ;
                        016D  2825 ;--
                        016D  2826
                        016D  2827 ERROUT::                                     ; Report an error and HALT.
     58    8E    D0     016D  2828          MOVL    (SP)+,R8                   ; Get pointer to message text.
           7E    7C     0170  2829          CLRQ    -(SP)                      ; Null input buffer and size
                        0172  2830                                             ; arguments.
           58    DD     0172  2831          PUSHL   R8                         ; Address of prompt string.
0000'CF    03    FB     0174  2832          CALLS   #3,B00$READPROMPT          ; Report the error.
5B   01F8'CF    D0     0179  2833          MOVL    B00$GL_RPBBASE,R11         ; Get the RPB address.
     59    5E    D0     017E  2834          MOVL    SP,R9                      ; Save old SP
     5E    6B    D0     0181  2835          MOVL    RPB$L_BASE(R11),SP         ; Set SP to base of memory.
5E   0200  CE    9E     0184  2836          MOVAB   ^X200(SP),SP               ; Set SP to 1st page past RPB.
  50    1C AB    7D     0189  2837          MOVQ    RPB$L_BOOTR0(R11),R0       ; Restore R0-R1.
  52    24 AB    7D     018D  2838          MOVQ    RPB$L_BOOTR2(R11),R2       ; Restore R2-R3.
  54    2C AB    7D     0191  2839          MOVQ    RPB$L_BOOTR4(R11),R4       ; Restore R4-R5.
  5C    18 AB    D0     0195  2840          MOVL    RPB$L_HALTCODE(R11),AP     ; Restore halt code.
  5A    10 AB    7D     0199  2841          MOVQ    RPB$L_HALTPC(R11),R10      ; Restore halt PC and PSL.
                        019D  2842 ERRHLT:
                     00 019D  2843          HALT                               ; Halt the processor.
                        019E  2844
                        019E  2845 ;
                        019E  2846 ; If the user types CONTINUE to the console program, try to restart
                        019E  2847 ; the bootstrap process.
                        019E  2848 ;
                        019E  2849
 FA 01FC'CF    E8     019E  2850          BLBS    MUST_HALT,ERRHLT           ; Branch if cannot retart
       FE5A    31     01A3  2851          BRW     START_BOOT                 ; Restart the boot.
```

```
                     01A6  2853          .SBTTL  Declarations located at end of bootstrap
                     01A6  2854
                     01A6  2855          .ALIGN  LONG
                     01A8  2856 ;
                     01A8  2857 ; Parameter list handed from primary boot to secondary boot
                     01A8  2858 ; The first location contains the argument count.  It is intended
                     01A8  2859 ; that the secondary boot will know what is in the list based on
                     01A8  2860 ; the argument count and the VMB version number.  This means that
                     01A8  2861 ; new information should be placed at new offsets even if older
                     01A8  2862 ; stuff becomes obsolete.  The VMB version number can be used to
                     01A8  2863 ; totally change the argument meanings if necessary.
                     01A8  2864 ;
                     01A8  2865 SECOND_PARAM:
            000001AC 01A8  2866          FIL$GQ_CACHE    == .+VMB$Q_FILECACHE ; FILEREAD cache descriptor
            000001CC 01A8  2867          BOO$GB_SYSTEMID == .+VMB$B_SYSTEMID  ; SCS system id
            0000000E 01A8  2868          .LONG   <VMB$C_ARGBYTCNT-45/4   ; Size of argument list
            000001E4 01AC  2869          .BLKB   VMB$C_ARGBYTCNT-4       ; Reserve space for the arguments
                     01E4  2870
                     01E4  2871 BITMAP_VEC_PTR:
            000001E8 01E4  2872          .BLKL   1                       ; Address of bitmap page array
                     01E8  2873 BITMAP_HI_INDX:                          ; Highest index to
            000001EC 01E8  2874          .BLKL   1                       ; bitmap page address array
                     01EC  2875 BITMAP_BAS_PFN:                         ; Starting PFN for PFN bitmap
            000001F0 01EC  2876          .BLKL   1                       ; when it is bigger than the
                     01F0  2877                                          ; pre-allocated bitmap
00000000 00000000    01F0  2878 INPBUF: .QUAD   0                       ; Dummy input buffer, with a cushion
                     01F8  2879 BOO$GL_RPBBASE::
            000001FC 01F8  2880          .BLKL   1                       ; RPB base address
                     01FC  2881 MUST_HALT:
            00000000 01FC  2882          .LONG   0                       ; If LBS, ERROUT should not restart
                     0200  2883                                          ; rather it must halt.
                     0200  2884 CI_UCODE_STAT:
00000000 00000000    0200  2885          .LONG   0,0                     ; Statistics block for ucode file
                     0208  2886 LAST_MEM:
            00000000 0208  2887          .LONG   0                       ; Last memory location probed
                     020C  2888 MEM_FLAG:
            00000000 020C  2889          .LONG   0                       ; Flag for memory found
                     0210  2890
                     0210  2891 ;
                     0210  2892 ; If debugging flag is set, align end of bootstrap on a longword
                     0210  2893 ; boundary.
                     0210  2894 ;
                     0210  2895
                     0210  2896          .IF     DF,DEBUG
                     0210  2897          .ALIGN  LONG
                     0210  2898          .ENDC
                     0210  2899
                     0210  2900          .END    START_BOOT
```

```
$$BASE                  =  00000001
$$DISPL                 =  00000006
$$GENSW                 =  00000001
$$HIGH                  =  00000005
$$LIMIT                 =  00000004
$$LOW                   =  00000001
$$MNSW                  =  00000001
$$MXSW                  =  00000001
ADPINIT                    00000419 R      05
ALLOC_BITMAP               000001B2 R      05
BITMAP_BAS_PFN             000001EC R      04
BITMAP_HI_INDX             000001E8 R      04
BITMAP_IS_OK               0000025F R      05
BITMAP_PAG_CNT          =  00000004
BITMAP_VEC_PTR             000001E4 R      04
BOO$ALLOC_PAGES            ********   X    05
BOO$AL_VECTOR              ********   X    05
BOO$CACHE_ALLOC            ********   X    05
BOO$CACHE_OPEN             ********   X    05
BOO$GB_SYSTEMID         =  000001CC RG     04
BOO$GL_RPBBASE             000001F8 RG     04
BOO$GL_UMR_DIS             ********   X    05
BOO$IMAGE_ATT              ********   X    05
BOO$PAGE_MCHECK            00000761 RG     05
BOO$QIO                    ********   X    05
BOO$READPROMPT             ********   X    05
BOO$TEST_MEM               000006AE RG     05
BOOTHIGH                   00000000 RG     02
BOOT_FAULT                 00000124 R      04
BQO$C_MOVE              =  00000018
BQO$L_UCODE             =  00000028
BQO$L_UNIT_INIT         =  0000001C
BTD$K_CONSOLE           =  00000040
BTD$K_HSCCI             =  00000020
CHECKMEM_730               ********   X    05
CHECKMEM_750               ********   X    05
CHECKMEM_780               ********   X    05
CHECKMEM_790               ********   X    05
CHECKMEM_8SS               ********   X    05
CHECK_BOOT_FIT             00000052 R      04
CHK_NEXT_BITMAP            0000074F R      05
CI_UCODEV                  00000D0E R      05
CI_UCODE_FILE              00000CB2 R      05
CI_UCODE_STAT              00000200 R      04
CONT1_PATH                 ********   X    05
CONT2_PATH                 ********   X    05
CONTINUE_INDEX             00000113 RG     05
CONTINUE_TBL               0000010D R      05
COPY_NAME                  00000575 R      05
COUNT_PAGE                 00000744 R      05
CPU_CODES                  00000670 R      05
CPU_DATA_730               0000068C R      05
CPU_DATA_750               00000684 R      05
CPU_DATA_780               0000067C R      05
CPU_DATA_790               00000694 R      05
CPU_DATA_8SS               0000069C R      05
CPU_W_CHECKMEM             00000002  G

CPU_W_FIND_CI              00000006  G
CPU_W_INIT_ADAP            00000004  G
CPU_W_SAVE_CSRS            00000000  G
CR                      =  0000000D
DEBUG                   =  00000001
DEFAULT_SECOND             00000566 R      05
DERIVE_SLOTNUM             000008D0 R      05
DERIVE_UBIADDR             00000890 R      05
DEVPROMPT                  00000BCC R      05
DIAGFILE                   00000B9F R      05
DIR...                  =  FFFFFFFF
EHSR$V_VMS              =  00000005
ERRHLT                     0000019D R      04
ERROUT                     0000016D RG     04
EXE$ACVIOLAT               00000124 RG     04
EXE$BREAK                  00000124 RG     04
EXE$GB_CPUDATA             ********   X    05
EXE$GB_CPUTYPE             ********   X    05
EXE$GL_FLAGS               00000169 RG     04
EXE$GL_SCB                 00000169 RG     04
EXE$GL_TENUSEC             ********   X    05
EXE$GL_UBDELAY             ********   X    05
EXE$MCHKVEC             =  00000004 RG     02
EXE$ROPRAND                00000124 RG     04
EXE$TBIT                   00000124 RG     04
EXE$V_SIMULATOR         =  00000000  G
FIL$GQ_CACHE            =  000001AC RG     04
FIL$GT_TOPSYS              00000BB6 RG     05
FIL$OPENFILE               ********   X    05
FIL$READ_LBN               00000000 RG     06
FILEBOOT                   000005A5 R      05
FILE_CONTIG                00000612 R      05
FILL_SCB                   0000000A R      05
FILOPN_ERR                 000005EC R      05
FIND_CI_730                00000AFA R      05
FIND_CI_750                00000AF3 R      05
FIND_CI_780                00000AF3 R      05
FIND_CI_790                00000B00 R      05
FIND_CI_8SS                00000AFD R      05
GET_UBI_ADDR               000008CC R      05
GOOD_PAGE                  00000740 R      05
GOT_BITMAP                 000007AB R      05
IHD$W_ACTIVOFF          =  00000002
INI$BRK                    00000087 RG     05
INI$RDONLY                 00000168 RG     04
INI$WRITABLE               00000168 RG     04
INIT_ADP_730               000009E0 R      05
INIT_ADP_750               000009E0 R      05
INIT_ADP_780               00000905 R      05
INIT_ADP_790               0000090C R      05
INIT_ADP_8SS               00000A9B R      05
INIT_ADP_COMMON            00000911 R      05
INIT_BITMAP                0000076B R      05
INPBUF                     000001F0 R      04
IO$_READLBLK            =  00000021
I0730$AL_IOBASE         =  00F20000
I0730$AL_UBOSP          =  00FC0000
```

```
I0750$AL_FLOAT                  = 00F34000
I0750$AL_IOBASE                 = 00F20000
I0750$AL_PERNEX                 = 00002000
I0750$AL_UBOSP                  = 00FC0000
I0750$AL_UBBASE                 = 00F30000
I0780$AL_IOBASE                 = 20000000
I0780$AL_UBOSP                  = 20100000
I0790$AL_IOA0                   = 20000000
I0790$AL_IOACR                  = 00080000
I0790$AL_PERABS                 = 02000000
I0790$AL_PERNEX                 = 00002000
I0790$AL_UBOSP                  = 00100000
I0790$C_SBIA                    = 00000001
I08SS$AL_IOBASE                 = 20000000
IO_SIZE                         = 0000007F
LAST_MEM                          00000208 R      04
LF                              = 0000000A
LOC_TEST_MEM                      00000117 R      05
MBA$L_CR                        = 00000004
MBA$M_CR_INIT                   = 00000001
MCF790$L_MSTAT2                 = 0000003C
MCHK_750                          00000A5C R      05
MCHK_780                          0000097C R      05
MCHK_790                          00000994 R      05
MCK_BER                         = 00000024
MEM_FLAG                          0000020C R      04
MEM_LOOP                          00000722 R      05
MEM_TAB                           000002AE R      05
MMG$PAGEFAULT                     00000124 RG     04
MOVE_BITMAP                       00000227 R      05
MSTAT2$V_IOBUFF                 = 00000002
MUST_HALT                         000001FC R      04
NAMEPROMPT                        00000BC0 R      05
NDT$_BUA                        = 80000102
NDT$_CI                         = 00000038
NDT$_MB                         = 00000020
NDT$_UB0                        = 00000028
NDT$_UB1                        = 00000029
NEX                             = 00000003
NEXT_BITMAP                       000006E3 R      05
NEXT_PAGE                         0000074B R      05
NOBRK                             00000088 R      05
NORMAL_PATH                       00000117 R      05
NOSUCHDEV                         000000A3 R      04
NXMCHK_790                        00000B74 R      05
NXT_PAGCNT                        FFFFFFF8
NXT_PFN                           FFFFFFFC
OPEN_UCODE_FILE                   00000AAE R      05
OVERLAY_START                     00000000 R      03
PAGTST                            FFFFFFEC
PAMM$C_NEXM                     = 0000000F
PAMM$S_CODE                     = 00000004
PAMM$V_CODE                     = 00000000
PCS_UCODEV                        00000CFE R      05
PCS_UCODE_FILE                    00000CBC R      05
PCS_UCODE_STAT                    00000CF6 R      05
PFN$AB_STATE                      00000169 RG     04
PFN$AB_TYPE                       00000169 RG     04
PFN$AL_BAK                        00000169 RG     04
PFN$AL_PTE                        00000169 RG     04
PFN$AW_REFCNT                     00000169 RG     04
PFN$AW_SWPVBN                     00000169 RG     04
PFN$AX_BLINK                      00000169 RG     04
PFN$AX_FLINK                      00000169 RG     04
PR$V_SID_TYPE                   = 00000018
PR$_EHSR                        = 0000004A
PR$_MCESR                       = 00000026
PR$_SBIFS                       = 00000030
PR$_SCBB                        = 00000011
PR$_SID                         = 0000003E
PR$_SID_TYP730                  = 00000003
PR$_SID_TYP750                  = 00000002
PR$_SID_TYP780                  = 00000001
PR$_SID_TYP790                  = 00000004
PR$_SID_TYP8SS                  = 00000005
PR$_UBRESET                     = 00000037
PR790$_PAMACC                   = 00000040
PR790$_PAMLOC                   = 00000041
PR8SS$_BIINIT                   = 0000005F
PRETST_CNT                        FFFFFFF0
PRETST_PFN                        FFFFFFF4
READFICE                          000000C0 R      04
READIN_BOOT                       0000066D R      05
READIN_BOOT_1                     00000007 R      04
READX                             000000F4 R      04
READ_HEADER                       0000063D R      05
REMOVEPROMPT1                     00000C31 R      05
REMOVEPROMPT2                     00000C52 R      05
RESUMEPROMPT1                     00000C70 R      05
RESUMEPROMPT2                     00000C96 R      05
RPB$B_CONFREG                   = 00000090
RPB$B_CTRLLTR                   = 00000108
RPB$B_DEVTYP                    = 00000066
RPB$B_FLAGS                     = 000000A3
RPB$B_HDRPGCNT                  = 000000A0
RPB$C_MEMDSCSIZ                 = 00000008
RPB$C_NMEMDSC                   = 00000008
RPB$L_ADPPHY                    = 0000005C
RPB$L_BADPGS                    = 00000104
RPB$L_BASE                      = 00000000
RPB$L_BOOTR0                    = 0000001C
RPB$L_BOOTR1                    = 00000020
RPB$L_BOOTR2                    = 00000024
RPB$L_BOOTR4                    = 0000002C
RPB$L_BOOTR5                    = 00000030
RPB$L_CSRPHY                    = 00000054
RPB$L_FILLBN                    = 0000003C
RPB$L_FILSIZ                    = 00000040
RPB$L_HALTCODE                  = 00000018
RPB$L_HALTPC                    = 00000010
RPB$L_HALTPSL                   = 00000014
RPB$L_IOVEC                     = 00000034
RPB$L_IOVECSZ                   = 00000038
RPB$L_MEMDSC                    = 000000BC
```

```
RPB$L_PFNCNT            =  0000004C
RPB$Q_PFNMAP            =  00000044
RPB$S_ABUS              =  00000002
RPB$S_NEXUS             =  00000004
RPB$S_TOPSYS            =  00000004
RPB$T_FILE              =  00000068
RPB$V_ABUS              =  00000004
RPB$V_BBLOCK            =  00000003
RPB$V_BOOBPT            =  00000005
RPB$V_DIAG              =  00000004
RPB$V_HALT              =  00000009
RPB$V_HEADER            =  00000006
RPB$V_NEXUS             =  00000000
RPB$V_NOTEST            =  00000007
RPB$V_SOLICT            =  00000008
RPB$V_TOPSYS            =  0000001C
RPB$W_BOOTNDT           =  000000A1
RPB$W_UNIT              =  00000064
RTF$OPENFILE              ********    X   05
SAVABS...               =  FFFFFFEC
SAVE_CSR_730              000007FD R      05
SAVE_CSR_750              0000085C R      05
SAVE_CSR_780              000007ED R      05
SAVE_CSR_790              0000080D R      05
SAVE_CSR_8SS              000008E4 R      05
SAVE_CSR_COMMON           0000082A R      05
SBIA$L_CR                 00000000
SBIA$L_CSR                00000004
SBIA$L_DIAGNOS            0000000C
SBIA$L_DMAACA             00000018
SBIA$L_DMAAID             0000001C
SBIA$L_DMABCA             00000020
SBIA$L_DMABID             00000024
SBIA$L_DMACCA             00000028
SBIA$L_DMACID             0000002C
SBIA$L_DMAICA             00000010
SBIA$L_DMAIID             00000014
SBIA$L_MAINT              00000044
SBIA$L_SBIERR             00000034
SBIA$L_SBIQC              0000004C
SBIA$L_SBISILO            00000030
SBIA$L_SBISTS             0000003C
SBIA$L_SILOCMP            00000040
SBIA$L_SUMRY              00000008
SBIA$L_TMOADDRS           00000038
SBIA$L_UNJAM              00000048
SCH$GL_CURPCB             00000169 RG     04
SCH$GL_PCBVEC             00000169 RG     04
SCRATCH_SIZE              FFFFFFEC
SECOND_PARAM              000001A8 R      04
SECOND_TOO_BIG            00000086 R      04
SS$_NOSUCHDEV           =  00000908
SS$_NOSUCHFILE          =  00000910
STACK_PAG_CNT           =  00000003
START_BOOT                00000000 RG     04
START_BOOT_1              00000000 RG     05
START_SECOND              00000083 R      04
STEP                    =  00000800
SWITCHPROMPT              00000BE7 R      05
SYS$IOBASE                00000169 RG     04
SYSL$CLRSBIA              00000168 RG     04
TENUSECTBL                000006A4 R      05
TEST_PAGE                 00000735 R      05
TEST_SOLICIT              0000054A R      05
TOO_MEM_ERR               00000A79 R      05
UBA$L_CR                =  00000004
UBA$L_CSR               =  00000000
UBA$L_MAP               =  00000800
UBA$L_SR                =  00000008
UBA$M_CSR_UBIC          =  00010000
UBA_ADAPTER               0000084B R      05
UBDELAY                   000006A9 R      05
UCODE_FAIL                00000CC7 R      05
UCODE_NAME              =  00000CEF R      05
UCODE_SEVER             =  00000CCF R      05
UNEXP_MCHK                00000144 RG     04
UNI_MEM_ERR               000009B3 R      05
VALID_PROCID              00000058 R      05
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
VMB$V_LOAD_SCS          =  00000000
VMSFICE                   00000B8B R      05
WAIT_UBA                  0000091C R      05
XDELBPT                   ********    X   05
XDELIBRK                  ********    X   05
XDELTBIT                  ********    X   05
XDS$GT_WORD_PFN           00000169 RG     04
```

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	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Z99BOOT	00000000 (0.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
BOOTDRIV_9	00000000 (0.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
$$$$00BOOT	00000210 (528.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
YBTMEM	00000D1E (3358.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
YFILEREAD	00000011 (17.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.09	00:00:00.75
Command processing	126	00:00:00.81	00:00:04.10
Pass 1	649	00:00:27.16	00:00:51.63
Symbol table sort	0	00:00:03.75	00:00:05.05
Pass 2	656	00:00:08.14	00:00:15.54
Symbol table output	1	00:00:00.27	00:00:00.68
Psect synopsis output	1	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1464	00:00:40.26	00:01:17.78

The working set limit was 1950 pages.
158217 bytes (310 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2311 non-local and 96 local symbols.
2900 source lines were read in Pass 1, producing 36 object records in Pass 2.
54 pages of virtual memory were used to define 50 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]790DEF.MLB;1	5
_255DUA28:[BOOTS.OBJ]BOOTS.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	16
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	34

2440 GETS were required to define 34 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:VMB/OBJ=OBJ$:VMB MSRC$:VMB/UPDATE=(ENH$:VMB)+EXECML$/L!B+LIB$:BOOTS.MLB/LIB+SHRLIB$:790DEF.MLB/LIB

SYSGENMN
LIS

VMB
LIS

SYSGEN
LIS

SYSGTTSTR
LIS

SYSGENCMD
LIS

SYSGTERM
LIS

TS8BOOTIO
LIS

CDDENTRY
LIS

CDD

CDDSHR
MAP

CDDEXC2
LIS

CDDLIB
LIS

CDDEXC
LIS

WRITEBOOT
LIS

CDDLIB
B32