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
```

FILEID**SYSBOOT

```
  SSSSSSSS  YY      YY    SSSSSSSS  BBBBBBBB      OOOOOO      OOOOOO    TTTTTTTTTT
  SSSSSSSS  YY      YY    SSSSSSSS  BBBBBBBB      OOOOOO      OOOOOO    TTTTTTTTTT
SS          YY      YY  SS          BB      BB  OO      OO  OO      OO      TT
SS          YY      YY  SS          BB      BB  OO      OO  OO      OO      TT
SS            YY  YY    SS          BB      BB  OO      OO  OO      OO      TT
SS            YY  YY    SS          BB      BB  OO      OO  OO      OO      TT
  SSSSSS        YY        SSSSSS    BBBBBBBB    OO      OO  OO      OO      TT
  SSSSSS        YY        SSSSSS    BBBBBBBB    OO      OO  OO      OO      TT
        SS      YY              SS  BB      BB  OO      OO  OO      OO      TT
        SS      YY              SS  BB      BB  OO      OO  OO      OO      TT
        SS      YY              SS  BB      BB  OO      OO  OO      OO      TT          ....
        SS      YY              SS  BB      BB  OO      OO  OO      OO      TT          ....
SSSSSSSS        YY      SSSSSSSS    BBBBBBBB      OOOOOO      OOOOOO        TT          ....
SSSSSSSS        YY      SSSSSSSS    BBBBBBBB      OOOOOO      OOOOOO        TT          ....

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        .TITLE  SYSBOOT - VMS Secondary Bootstrap Routine
0000        .IDENT  'V04-000'
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000   ;++
0000   ;
0000   ; Facility:  System bootstrapping and initialization
0000   ;
0000   ; Abstract:
0000   ;
0000   ;
0000   ; Environment:
0000   ;       Mode=Kernel ,  Memory Management OFF
0000   ;       IS=1 ,  IPL=31
0000   ;
0000   ; Author:  RICHARD I. HUSTVEDT, Creation date:  28-APR-1978
0000   ;
0000   ; MODIFIED BY:
0000   ;
0000   ;       V03-062 KPL0102         P Lieberwirth           24-Aug-1984
0000   ;               Fix bug in GETCONLOC_790 that enabled the logical console
0000   ;               imcorrectly.
0000   ;
0000   ;       V03-061 WHM0008         Bill Matthews           25-Jul-1984
0000   ;               Initialize the keyboard translation table address for QVSS
0000   ;               console terminal support.
0000   ;
0000   ;       V03-060 TCM0019         Trudy C. Matthews       24-Jul-1984
0000   ;               Check to see if the version of VMB has support for the
0000   ;               RPB$B_CTRLLTR field.
0000   ;
0000   ;       V03-059 WHM0007         Bill Matthews           16-Jul-1984
0000   ;               Add support for QVSS as a console terminal.
0000   ;
0000   ;       V03-058 WMC0058         Wayne Cardoza           05-Jul-1984
0000   ;               Don't second guess LRPMIN.
```

```
0000 ;
0000 ;      V03-057 WMC0057         Wayne Cardoza           28-Jun-1984
0000 ;              Survive very fragmented dump file.
0000 ;
0000 ;      V03-056 WHM0006         Bill Matthews           14-May-1984
0000 ;              Fix to not copy SYSBOOT's copy of EXE$GL_TODR over the copy
0000 ;              from SYS.EXE
0000 ;
0000 ;      V03-055 KDM0101         Kathleen D. Morse       02-May-1984
0000 ;              Change minimum cpu microcode rev level check from 5 to 4
0000 ;              for MicroVAX I.
0000 ;
0000 ;      V03-054 WHM0005         Bill Matthews           02-May-1984
0000 ;              Always use VAXVMSSYS.PAR. Don't read the SYSGEN parameter area
0000 ;              twice.
0000 ;
0000 ;      V03-053 KDM0098         Kathleen D. Morse       24-Apr-1984
0000 ;              Add minimum cpu microcode rev level check for MicroVAX I.
0000 ;
0000 ;      V03-052 WHM0004         Bill Matthews           16-Apr-1984
0000 ;              Don't use a separate parameter file if USESYSPARAMS is set to
0000 ;              0 in SYS.EXE. (Needed for standalone backup environment)
0000 ;
0000 ;      V03-051 KPL0101         Peter Lieberwirth       11-Apr-1984
0000 ;              Zero the high byte of RPB$W_BOOTNDT if VMB version is
0000 ;              high enough to have set up RPB$B_BOOTNDT, but not high
0000 ;              enough so as to have properly initialized the high byte
0000 ;              (version 12).
0000 ;
0000 ;      V03-050 WHM0003         Bill Matthews           04-Apr-1984
0000 ;              Added support to read in the SYSGEN parameters from the
0000 ;              the default system parameter file.
0000 ;              Added support to use file to handle ascii sysgen parameters
0000 ;              that are longer than 4 bytes.
0000 ;
0000 ;      V03-049 WMC0004         Wayne Cardoza           28-Mar-1984
0000 ;              Save the highest useable PFN.
0000 ;
0000 ;      V03-048 CWH3048         CW Hobbs                15-Mar-1984
0000 ;              Require a "Y" response rather than a carriage return when
0000 ;              switching console volumes.  Also rewind the old volume before
0000 ;              asking for the switch so that VAX-11/750 processors do not
0000 ;              timeout waiting for the TU58 to rewind.
0000 ;
0000 ;      V03-047 WHM0002         Bill Matthews   08-Mar-1984
0000 ;              Modification made to VMB so interface change made in WHM0001
0000 ;              can be backed out.
0000 ;
0000 ;      V03-046 KPL0001         Peter Lieberwirth       8-Mar-1984
0000 ;              Zero fill RPB$W_BOOTNDT from RPB$B_CONFREG for old versions
0000 ;              of VMB.
0000 ;
0000 ;      V03-045 WHM0001         Bill Matthews   05-Mar-1984
0000 ;              Add support for booting from a set of common system files.
0000 ;
0000 ;      V03-044 MMD0248         Meg Dumont,     27-Feb-1984  10:48
0000 ;              Add support for $MTACCESS installation specific accessibility
```

```
0000 ;                  routine
0000 ;
0000 ;          V03-043 LY00B6          Larry Yetto             10-FEB-1984 15:30
0000 ;                  Fix truncation errors.
0000 ;
0000 ;          V03-042 TMK0001         Todd M. Katz            31-Jan-1984
0000 ;                  Fix a whole host of truncation errors by changing word
0000 ;                  to long offsets
0000 ;
0000 ;          V03-041 WMC0003         Wayne Cardoza           21-Dec-1983
0000 ;                  Prevent PHYSICAL_PAGES parameter from confusing PFN bit map.
0000 ;                  Make PFN link word size converge.
0000 ;
0000 ;          V03-040 TCM0018         Trudy C. Matthews       08-Dec-1983
0000 ;                  Initialize EXE$GB_CPUDATA twice: once before the first
0000 ;                  CPUDISP and again after SYSBOOT reads the system parameters
0000 ;                  off the disk (which wipes out the first initialization).
0000 ;                  Also, set "enable mask write" bit when enabling venus'
0000 ;                  logical console line in GETCONLOC_790.
0000 ;                  Only make interrupts through SCB vector 0 harmless on a 780
0000 ;                  (and not on venus).
0000 ;
0000 ;          V03-039 DWT0149         David W. Thiel          14-Nov-1983
0000 ;                  Condition loading of SCSLOA and CLUSTRLOA on the
0000 ;                  now 3-valued SYSGEN parameter VAXCLUSTER.
0000 ;
0000 ;          V03-038 ACG0372         Andrew C. Goldstein,    11-Nov-1983  10:00
0000 ;                  Tighten page protection on various system structures
0000 ;
0000 ;          V03-037 TCM0017         Trudy C. Matthews       27-Oct-1983
0000 ;                  Initialize EXE$GB_CPUTYPE cell before initial breakpoint;
0000 ;                  XDELTA expects that field to be set up.
0000 ;
0000 ;          V03-036 KDM0087         Kathleen D. Morse       20-Oct-1983
0000 ;                  Load SCB emulation vectors from the boot-strap emulator
0000 ;                  linked into SYSBOOT, not from what VMB left in them.
0000 ;
0000 ;          V03-035 KDM0084         Kathleen D. Morse       23-Sep-1983
0000 ;                  Add MicroVAX I to CPUDISP.  Add a patch area for SYSBOOT.
0000 ;
0000 ;          V03-034 CWH3034         CW Hobbs                28-Aug-1983
0000 ;                  Make it possible for SYS.EXE to cross volume boundaries
0000 ;                  when booting from the console.  Means that no files can
0000 ;                  be kept open across calls to read SYS.EXE.
0000 ;
0000 ;          V03-033 TCM0016         Trudy C. Matthews       2-Aug-1983
0000 ;                  Store SID in CPUDATA field before executing any CPUDISP
0000 ;                  macros.  Add 11/785-specific path to CPUDISP macro that
0000 ;                  checks hardware/microcode revs.
0000 ;
0000 ;          V03-032 KDM0061         Kathleen D. Morse       15-Jul-1983
0000 ;                  Change code to use cpu-specific definition for IPR ACCS,
0000 ;                  PR780$_ACCS, instead of cpu-independent PR$_ACCS.
0000 ;
0000 ;          V03-031 TCM0015         Trudy C. Matthews       27-Jul-1983
0000 ;                  Add support for the 11/785's different PCS/WCS version numbers.
0000 ;
```

```
0000 ;        V03-030 MSH0001         Maryann Hinden          13-Jul-1983
0000 ;                BOO$GETPARAM no longer takes an argument.
0000 ;
0000 ;        V03-029 KTA3061         Kerbey T. Altmann       26-Jun-1983
0000 ;                Fixed truncation error.
0000 ;
0000 ;        V03-028 ADE0001         Alan D. Eldridge        23-Jun-1983
0000 ;                Change "64" to CXB$C_OVERHEAD when calculating LRP size.
0000 ;
0000 ;        V03-027 KTA3059         Kerbey T. Altmann       21-Jun-1983
0000 ;                Add support for boot device name from VMB.
0000 ;
0000 ;        V03-026 TCM0014         Trudy C. Matthews       15-Jun-1983
0000 ;                Fix console protocol bug in GETCONLOC_790.
0000 ;
0000 ;        V03-025 KDM0045         Kathleen D. Morse       31-May-1983
0000 ;                Change word displacements to longword.
0000 ;
0000 ;        V03-024 KDM0044         Kathleen D. Morse       02-May-1983
0000 ;                Load floating point emulator, if necessary, and set
0000 ;                appropriate indicators in EXE$GL_ARCHFLAG.
0000 ;
0000 ;        V03-023 KTA3048         Kerbey T. Altmann       14-Apr-1983
0000 ;                Grab G&H status from console for 11/780.  Ascertain
0000 ;                FPA status.
0000 ;
0000 ;        V03-022 TCM0013         Trudy C. Matthews       23-Feb-1983
0000 ;                Check VMB version for version which passes number of bad
0000 ;                pages found during bootstrap memory scan in RPB$L_BADPGS.
0000 ;
0000 ;        V03-021 KTA3038         Kerbey T. Altmann       11-Feb-1983
0000 ;                Add copying of possible boot node name.
0000 ;
0000 ;        V03-020 DWT0073         David W. Thiel          28-Jan-1983
0000 ;                Load cluster code based on SGN$V_LOADCLUSTR instead
0000 ;                of VMS5.
0000 ;
0000 ;        V03-019 TCM0012         Trudy C. Matthews       21-Jan-1983
0000 ;                Add routine label SYSL$CLRSBIA for linking with XDELTA.
0000 ;
0000 ;        V03-018 STJ3056         Steven T. Jeffreys      21-Jan-1983
0000 ;                Added code to laod $ERAPAT and $CHKPRT loadable code
0000 ;                into pool, if necessary.
0000 ;
0000 ;        V03-017 SRB0059         Steve Beckhardt         6-Jan-1983
0000 ;                Added code to load cluster loadable code into pool,
0000 ;                if necessary.
0000 ;
0000 ;        V03-016 WMC0002         Wayne Cardoza           30-Dec-1982
0000 ;                Exclude system service vectors from PFN database.
0000 ;
0000 ;        V03-015 TCM0011         Trudy C. Matthews       15-Dec-1982
0000 ;                Correct protocol error in GETCONLOC_790.
0000 ;
0000 ;        V03-014 TCM0010         Trudy C. Matthews       10-Dec-1982
0000 ;                Use CPUDATA cell to store 11/790 microcode rev level.
0000 ;
```

```
          0000  ;       V03-013 JWH0126         Jeffrey W. Horn         11-Nov-1982
          0000  ;               Fix bug in computation of SWP$GB_SHLP1PT which occurs
          0000  ;               when SWP$C_SHLFPTE is greater than 127.
          0000  ;
          0000  ;       V03-012 KTA3018         Kerbey T. Altmann       08-Nov-1982
          0000  ;               Remove loading of INILOA.
          0000  ;
          0000  ;       V03-011 TCM0009         Trudy C. Matthews       26-Oct-1982
          0000  ;               Add 11/790-specific paths through CPU-dependent code.
          0000  ;
          0000  ;       V03-010 LJK0168         Lawrence J. Kenah       2-Jun-1982
          0000  ;               Move code that maps pageable exec so that pages containing
          0000  ;               it are not included in PFN data base when system paging
          0000  ;               is turned off (SYSPAGING = 0).
          0000  ;
          0000  ;--
          0000
          0000  ;
          0000  ; Include files:
          0000  ;
          0000          $ARCDEF                         ; Define architectural flag bits
          0000          $BOODEF                         ; Define Boot Control Block Offsets
          0000          $BQODEF                         ; Define Boot qio offsets
          0000          $BTDDEF                         ; Define Boot devices
          0000          $CXBDEF                         ; Define Complex Chained Buffer
          0000          $DPTDEF                         ; Define Driver Prologue Definitions
          0000          $DYNDEF                         ; Define dynamic pool codes
          0000          $FH2DEF                         ; Define level 2 file header offsets
          0000          $IHDDEF                         ; Define Image Header offsets
          0000          $IO750DEF                       ; Define 11/750 I/O space
          0000          $IODEF                          ; Define I/O function code values
          0000          $IRPDEF                         ; Define I/O request packet offsets
          0000          $NDTDEF                         ; Define nexus device type codes
          0000          $PHDDEF                         ; Define process header offsets
          0000          $PRDEF                          ; Define process register numbers
          0000          $PRMDEF                         ; Define SYSGEN parameter offsets
          0000          $PR780DEF                       ; Define 11/780 processor registers
          0000          $PSLDEF                         ; Define program status longword fields
          0000          $PTEDEF                         ; Define page table entry fields
          0000          $RPBDEF                         ; Define restart parameter block offsets
          0000          $SECDEF                         ; Define process section block
          0000          $TPADEF                         ; Define TPARSE offsets
          0000          $VADEF                          ; Define fields in virtual address
          0000          $VMBARGDEF                      ; Define VMB argument list offsets
          0000          $WCBDEF                         ; Define Window Control Block offsets
          0000          $WSLDEF                         ; Define Working Set lis entry
          0000                                          ;***
00000001  0000  DEBUG=1                                 ;***
          0000                                          ;***
          0000
          0000  ;
          0000  ; Macros:
          0000  ;
          0000          .MACRO  ERROR,STR               ;
          0000          BSBW    ERROUT                  ; Output error string
          0000          .ASCIZ  \'STR'\                 ;
          0000          .ENDM   ERROR                   ;
```

```
          0000
          0000          .MACRO  MSG,STR                       ;
          0000          BSBW    BOO$FACMSG                    ; Output message
          0000          .ASCIZ  \'STR'\                       ;
          0000          .ENDM   MSG                           ;
          0000
          0000     ;
          0000     ; Equated Symbols:
          0000     ;
          0000     ;
00000FFC  0000          R2_R11 = ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>    ; Mask for R2 to R11
0000007F  0000          IOSIZE = 127                          ; Do I/O in very large chunks
00004000  0000          MAXPGS = 16384                        ; Allow for 8Mb of memory
00000001  0000          DEBUG  = 1                            ; Assemble DEBUG code
0000000D  0000          CR     = 13                           ; Character code for carriage return
0000000A  0000          LF     = 10                           ; Character code for line feed
00000007  0000          BELL   = 7                            ; ASCII bell
          0000
00000018  0000          RP_DT  = 24                           ; Offset to drive type register for massbus
00000003  0000          RTRV_PAG_CNT = 3                      ; Retrieval pointer page count
          0000                                                ; 1 for SYS.EXE
          0000                                                ; 1 for SYSDUMP.DMP
          0000                                                ; 1 for the other small files
          0000
          0000     ;
          0000     ; 11/780 microcode revision levels.
          0000     ;
0000006D  0000          FPLA_VLOC_780 = ^O155                 ; Offset to FPLA version number location
0000000C  0000          FPLA_780 = ^X0C                       ; FPLA version number
0000006A  0000          PCS_VLOC_780  = ^O152                 ; Offset to PCS version number location
00000001  0000          PCS_780 = ^X01                        ; PCS version number
0000006B  0000          WCSP_VLOC_780 = ^O153                 ; Offset to WCS primary version location
0000000C  0000          WCSP_780 = ^X0C                       ; WCS primary version number
0000006C  0000          WCSS_VLOC_780 = ^O154                 ; Offset to WCS secondary version location
00000012  0000          WCSS_780 = ^X12                       ; WCS secondary version
00000064  0000          GHOPT_VLOC     = ^O144                ; GH opt flag location
          0000
          0000     ;
          0000     ; 11/785 microcode revision levels.
          0000     ;
00000017  0000          SID_785 = 23                          ; SID bit that differentiates an 11/785
          0000                                                ; from an 11/780.
          0000
0000006A  0000          PCS_VLOC_785  = ^O152                 ; Offset to PCS version number
000000[illegible]  0000          PCS_785 = ^X04                        ; PCS version number
0000006B  0000          MTCH_VLOC_785 = ^O153                 ; Offset to PCS/WCS match version number.
00000004  0000          MTCH_785 = ^X04                       ; PCS/WCS match version number.
0000006C  0000          WCSP_VLOC_785 = ^O154                 ; Offset to WCS primary version number.
00000001  0000          WCSP_785 = ^X01                       ; WCS primary version number.
0000006D  0000          WCSS_VLOC_785 = ^O155                 ; Offset to WCS secondary version number.
00000000  0000          WCSS_785 = ^X00                       ; WCS secondary version number.
          0000
          0000     ;
          0000     ; 11/750 revision levels.
          0000     ;
          0000
00000000  0000          HWREV_750=0                           ; Minimum req'd hardware ECO level
```

```
0000003E  0000        MICREV_750=62                   ; Minimum req'd microcode rev level
          0000
          0000  ;
          0000  ; 11/730 revision levels.
          0000  ;
00000000  0000        HWREV_730=0                     ; Minimum req'd hardware ECO level
00000033  0000        MICREV_730=51                   ; Minimum req'd microcode rev level
          0000
          0000  ;
          0000  ; 11/790 revision levels.
          0000  ;
00000012  0000        MICREV_REQ_790 = ^X12           ; Code to request ucode rev level
00000000  0000        MICREV_790=0                    ; Minimum req'd microcode rev level
          0000
          0000  ;
          0000  ; MicroVAX I revision levels.
          0000  ;
00000004  0000        MICREV_UV1 = ^X04               ; Minimum req'd microcode rev level
          0000                                        ; Note:  REV 5 is required to use
          0000                                        ; the 19'' QVSS as console terminal.
          0000
          0000  ;
          0000  ; Offsets into Statistics Blocks kept for each file that is looked up
          0000  ;
          0000        $OFFSET 0,POSITIVE,<-
          0000        STAT_L_VBN,-                    ; Starting VBN (after header)
          0000        STAT_L_BYTECNT,-                ; Size in bytes of image
          0000        STAT_L_SYSVA,-                  ; Base relative adr of pool adr
          0000        STAT_L_NAME,-                   ; Base relative adr of ASCIC file name
          0000        STAT_L_MAP,-                    ; Address of virtual to logical map
          0000        STAT_L_VBN2ADR,-                ; Buf Adr containing VBN 2
          0000        <STAT_C_SIZE,0> -               ; Size of STAT block
          0000        >
          0000  STAT_L_VBN:
          0004  STAT_L_BYTECNT:
          0008  STAT_L_SYSVA:
          000C  STAT_L_NAME:
          0010  STAT_L_MAP:
          0014  STAT_L_VBN2ADR:
          0018  STAT_C_SIZE:
          0000  ;
          0000  ; Own Storage:
          0000  ;
      00000000        .PSECT  __Z99BOOT,PAGE
          0000  BOOTHIGH:
      00000000        .PSECT  $$$$00BOOT,LONG         ;
```

```
0000            .SBTTL  Secondary Bootstrap Main Routine
0000    ;++
0000    ;
0000    ; Functional Description:
0000    ;
0000    ; Calling Sequence:
0000    ;       JMP     EXE$SYSBOOT
0000    ;
0000    ; Input Parameters:
0000    ;       R10 - Base address of secondary bootstrap
0000    ;       R11 - Pointer to Restart Parameter Block
0000    ;       AP  - Argument list from VMB (Version 5 or later)
0000    ;       SP  - Stack pointer
0000    ;       PR$_SCBB - System Control Block Base (SCB)
0000    ;
0000    ;       Memory layout at start of secondary bootstrap:
0000    ;
0000    ;       +------------------------------------------+ :BASE
0000    ;       !                                          !
0000    ;       !       Restart Parameter Block (RPB)      !
0000    ;       !                                          !
0000    ;       +------------------------------------------+ :BASE+^X200
0000    ;       !                                          !
0000    ;       !         Primary Bootstrap Code           !
0000    ;       !   (up to the end of drivers only)        !
0000    ;       !                                          !
0000    ;       +------------------------------------------+ :PR$_SCBB
0000    ;       !                                          !
0000    ;       !          System Control Block            !
0000    ;       !                                          !
0000    ;       +------------------------------------------+ :PFNMAP
0000    ;       !                                          !
0000    ;       !          Small PFN Bitmap                !
0000    ;       !                                          !
0000    ;       +------------------------------------------+ :PFNMAP+X (X=^X200/400/600
0000    ;       !                                          !     or 800 as determ by VMB)
0000    ;       !          Bootstrap Stack                 !
0000    ;       !            (3 pages)                     !
0000    ;       +------------------------------------------+ :(SP)
0000    ;       !                                          !
0000    ;       !        Secondary Bootstrap Code          !
0000    ;       !                                          !
0000    ;       +------------------------------------------+
0000    ;
0000    ;       The PFN bitmap passed in by VMB describes all the memory that
0000    ; has been tested and proven usable.  Some of that memory is actively in
0000    ; use and must not be allocated and overwritten by this SYSBOOT code.
0000    ; For example the portion of memory in which this code is currently
0000    ; running has not been marked as "in use."  Allocation of memory
0000    ; is done from high PFN to low.  It is expected that we will not get
0000    ; to the pages we are running in, and thus having them appear usable
0000    ; means that they don't have to be set usable before transferring
0000    ; control to INIT.
0000    ;       There are 3 other regions of memory that have the same attribute.
0000    ; The first is the FIL$OPENFILE cache.  If memory allocation overlaps
0000    ; this cache, it will automatically be disabled.  The second region
0000    ; is the combination CI microcode and page table.  If memory allocation
```

```
                                0000   ; overlaps this area, a fatal error halt is taken, as the infomation
                                0000   ; contained therein is essential to system operation. The third region
                                0000   ; is the large PFN bitmap.  This region is only created for memories
                                0000   ; in excess of 8mb, and it is positioned at 2mb.  It is assumed that
                                0000   ; none of the allocation logic will get down to this point for such
                                0000   ; a large memory, and thus this is not a problem.
                                0000   ;       There are 3 points at which SYSBOOT is interested in regions
                                0000   ; which are in use, but marked allocatable.  The first point
                                0000   ; is when SYSBOOT artificially limits the size of memory to the
                                0000   ; physical page count SYSGEN parameter (near label HAVEPRM).
                                0000   ;       The second place is when SYSBOOT is trying to allocate a
                                0000   ; contiguous run of pages for the SPT (near label ALLOCSPT).
                                0000   ;       The final place is in the routine ALLOCPFN which allocates
                                0000   ; a PFN.
                                0000   ;
                                0000   ; Output Parameters:
                                0000   ;
                                0000   ;--
                                0000
                                0000   EXE$SYSBOOT::                                   ; Secondary bootstrap start address
        23A4'CF     5B    D0    0000           MOVL    R11,W^BOO$GL_RPBBASE    ; Save base of RPB
                                0005   ;
                                0005   ; Get pointer to SCB used by primary bootstrap and fill with pointers to
                                0005   ; exception handlers for secondary bootstrap.
                                0005   ;
             58     11    DB    0005           MFPR    #PR$_SCBB,R8            ; Address of VMB's SCB
        58   0200 C8      DE    0008           MOVAL   ^X200(R8),R8            ; Fill SCB back to front
        56  1DF1'CF       DE    000D           MOVAL   W^BOOT_FAULT+1,R6       ; Unexpected fault handler vector
             78     56    D0    0012   10$:    MOVL    R6,-(R8)                ; Store a vector
        58   01FF   8F    B3    0015           BITW    #^X1FF,R8               ; Check for page boundary
                    F6    12    001A           BNEQ    10$                     ; Need another vector
  00C8 C8  00000000'EF    9E    001C           MOVAB   VAX$EMULATE,^XC8(R8)    ; Set emulation vector contents
  00CC C8  00000000'EF    9E    0025           MOVAB   VAX$EMULATE_FPD,^XCC(R8) ; Set FPD emulation vector contents
     04 A8   1E0C'CF      9E    002E           MOVAB   W^UNEXP_MCHK,4(R8)      ; Machine checks are not expected
                                0034   ;
                                0034   ; Get CPU type and check if valid.
                                0034   ;
                                0034
                                0034   CHK_CPU_TYPE:                                   ; Get and check CPU type
             50     3E    DB    0034           MFPR    #PR$_SID,R0             ; Read SID
        0000'CF     50    D0    0037           MOVL    R0,W^EXE$GB_CPUDATA     ; Store SID in SYSPARAM data field
   50    50    E8   8F    78    003C           ASHL    #-PR$V_SID_TYPE,R0,R0   ; Right justify type
                    10    13    0041           BEQL    10$                     ; Branch if type=0 (illegal)
             08     50    91    0043           CMPB    R0,#PR$_SID_TYPMAX      ; Type exceeds maximum?
                    24    1B    0046           BLEQU   20$                     ; Branch if not (legal)
                                0048   ;***
                                0048   ;***The following 4 instructions are present to allow
                                0048   ;***11/780's with uninitialized sid's to run.
                                0048   ;***
          FF 8F     50    91    0048           CMPB    R0,#-1                  ; CPU type=-1?
                    05    12    004C           BNEQ    10$                     ; branch if not
             50     50    8E    004E           MNEGB   R0,R0                   ; if so, set CPU type to 11/780
                    19    11    0051           BRB     20$                     ;  and continue
                                0053   ;***
                                0053   10$:    MSG     <-F-Unknown processor>
                          00    006B           HALT                            ; *****Fatal Error*****
        0000'CF     50    90    006C   20$:    MOVB    R0,W^EXE$GB_CPUTYPE     ; Save CPU TYPE in SYSPARAM
```

```
       23A8'CF    50   90  0071        MOVB    R0,W^CPUTYPE           ; Save it in a safe place too
                           0076
                           0076  ;
                           0076  ; Check for alternate console terminal and if necessary fix up displacements
                           0076  ; for CON$GETCHAR and CON$PUTCHAR in CONIO.
                           0076  ;
                           0076        CPUDISP <<780,40$>,-           ; *Dispatch on CPU type*
                           0076                <750,40$>,-            ;
                           0076                <730,40$>,-
                           0076                <790,40$>,-
                           0076                <UV1,30$>,-
                           0076                <785,40$>>,-
                           0076                ENVIRON=VMB
                           00C1
       24 20 AB   06   E0  00C1  30$:  BBS     #6,RPB$L_BOOTR1(R11),40$;branch if QVSS is NOT the console terminal
            FFFD'8F    B0  00C6        MOVW    #<QVSS$INPUT-CON$GETCHAR-3>,-;calculate displacment
        00000001'EF        00CA                CON$GETCHAR+1           ;Fix up the BRW instruction in CON$GETCHAR
            FFFD'8F    B0  00CF        MOVW    #<QVSS$OUTPUT-CON$PUTCHAR-3>,-;calculate displacment
        00000001'EF        00D3                CON$PUTCHAR+1           ;Fix up the BRW instruction in CON$PUTCHAR
     2320'CF   5357 8F  B0  00D8        MOVW    #^A/WS/,W^MODEL_UV1     ; Use SYSLOAWS1.EXE
00000000'EF  FFFFFF90'EF DE  00DF        MOVAL   QVSS$KEY-112,QVSS$KEYTABLE; INITIALIZE THE KEYBOARD TRANSLATION TABL
                           00EA
                           00EA  40$:
                           00EA  ;
                           00EA  ; If debug code is included, connect debugger to BPT and TBIT exception
                           00EA  ; vectors, then execute a BPT instruction to give control to the debugger.
                           00EA  ;
                           00EA        .IF     DF,DEBUG               ;
   2C A8   00000000'EF  9E  00EA        MOVAB   XDELBPT,^X2C(R8)       ; Set BPT exception vector to debugger
   28 A8   00000000'EF  9E  00F2        MOVAB   XDELTBIT,^X28(R8)      ;  and TBIT vector also
 00000000'EF     07'AF  9E  00FA        MOVAB   B^INI$BRK,XDELIBRK     ; Make correct break address in brk table
      01 30 AB    05   E1  0102        BBC     #RPB$V_BOOBPT,RPB$L_BOOTR5(R11),NOBRK   ; Check for bootstrap BPT
                           0107  INI$BRK::                            ;
                      03   0107        BPT                            ; Breakpoint initially, changed to NOP
                           0108  NOBRK:                               ; No bootstrap break point
                           0108        .ENDC                          ;
                           0108
    2198'CF  00000000'EF DE  0108        MOVAL   L^BOOTHIGH,W^BOO$GL_FREEMEM ; Save pointer to free memory
         23A0'CF      5A  D0  0111        MOVL    R10,W^SYSBOOT_BASE      ; Save base address of this code
    23CC'CF   44 AB   03  78  0116        ASHL    #3,RPB$Q_PFNMAP(R11),W^MEM_HI_PFN ; Record highest PFN
                           011D                                        ; if VMB didn't pass it in.
         23D8'CF   44 AB  7D  011D        MOVQ    RPB$Q_PFNMAP(R11),W^SMALL_PFNMAP ; Save descriptor for backward
                           0123                                        ; compatible (max 8mb) PFN bitmap
                           0123  ;
                           0123  ; Save VMB version number so that various code paths can check it
                           0123  ; A zero in VMB_VERSION means that we were started by the release 1 VMB
                           0123  ;
            50    34 AB   D0  0123        MOVL    RPB$L_IOVEC(R11),R0     ; Address of I/O vectors
                  10 A0   AD  0127        XORW3   BQO$W_VERSION(R0),-
            51    12 A0       012A                BQO$W_VERCHECK(R0),R1   ; If we have a version #, the
                     51   B6  012D        INCW    R1                      ; version and its complement are there
                     06   12  012F        BNEQ    BOOTNDT_TST             ; Branch if release 1 VMB
                  10 A0   3C  0131        MOVZWL  BQO$W_VERSION(R0),-
              239C'CF         0134                W^VMB_VERSION           ; Save the VMB version number
                           0137  ;
                           0137  ; Determine if VMB has already set up RPB$B_BOOTNDT field.
                           0137  ;
```

```
                                     0137  BOOTNDT_TST:
                 06  239C'CF   D1    0137          CMPL    W^VMB_VERSION,#6          ; Did VMB set up the BOOTNDT field?
                          0E   18    013C          BGEQ    10$                       ; Version 6 or later -- yes.
                 51    20 AB   D0    013E          MOVL    RPB$L_BOOTR1(R11),R1      ; Get index to boot adapter.
       00A1 CB    0090 CB41    9B    0142          MOVZBW  RPB$B_CONFREG(R11)[R1],-; Get boot adapter's nexus device
                                     014A                  RPB$W_BOOTNDT(R11)        ; type from RPB$B_CONFREG array.
                          0B   11    014A          BRB     END_BOOTNDT_TST           ;
                 0C  239C'CF   D1    014C  10$:    CMPL    W^VMB_VERSION,#12         ; Did VMB init the high byte of
                          04   18    0151          BGEQ    END_BOOTNDT_TST           ;  W_BOOTNDT properly? br if yes.
                     00A2 CB   94    0153          CLRB    RPB$W_BOOTNDT+1(R11)      ;  Zero the high byte.
                                     0157  END_BOOTNDT_TST:
                                     0157  ;
                                     0157  ; Determine if VMB has passed in an argument list in AP
                                     0157  ;
                 05  239C'CF   D1    0157          CMPL    W^VMB_VERSION,#5          ; Is there an argument list in AP?
                          03   18    015C          BGEQ    1$                        ; Yes, process it
                        0089   31    015E          BRW     ADD_CACHE                 ; Branch if not
                                     0161  ;
                                     0161  ; The argument count field of the argument list describes what
                                     0161  ; is present in the argument list.
                                     0161  ;
                       04 AC   7D    0161  1$:     MOVQ    VMB$Q_FILECACHE(AP),-     ; No additional test
                     2165'CF         0164                  W^FIL$GQ_CACHE            ;  is needed for file cache
                    6C    05   B1    0167          CMPW    #VMB$Q_PFNMAP/4,(AP)      ; Are the PFNMAP, LO and HI
                                     016A                                            ; PFN arguments present?
                          49   14    016A          BGTR    5$                        ; Branch if not
                       14 AC   7D    016C          MOVQ    VMB$Q_PFNMAP(AP),-        ; Yes, update RPB
                       44 AB         016F                  RPB$Q_PFNMAP(R11)         ;  descriptor for PFN bitmap
                                     0171          ASSUME  VMB$L_HI_PFN EQ VMB$L_LO_PFN+4
                       0C AC   7D    0171          MOVQ    VMB$L_LO_PFN(AP),-
                     23C8'CF         0174                  W^MEM_LO_PFN              ; Record the low and hi PFN's
                    6C    0B   B1    0177          CMPW    #VMB$L_FLAGS/4,(AP)       ; Are the UCODE thru FLAGS args present?
                          39   14    017A          BGTR    5$                        ; Branch if not
                       1C AC   7D    017C          MOVQ    VMB$Q_UCODE(AP),-         ; Yes, move the descriptor
                     23E0'CF         017F                  W^UCODE_LEN
                       24 AC   D0    0182          MOVL    VMB$B_SYSTEMID(AP),-      ;  and the system id
                 00000000'EF         0185                  BOO$GB_SYSTEMID
                       28 AC   B0    018A          MOVW    VMB$B_SYSTEMID+4(AP),-
                 00000004'EF         018D                  BOO$GB_SYSTEMID+4
                       2C AC   D0    0192          MOVL    VMB$L_FLAGS(AP),-
                     23E8'CF         0195                  W^VMB_FLAGS               ;  and the flags
                    6C    0C   91    0198          CMPB    #VMB$L_CI_HIPFN/4,(AP)    ; Is the CI_HIPFN arg present?
                          18   14    019B          BGTR    5$                        ; Branch if not
                       30 AC   D0    019D          MOVL    VMB$L_CI_HIPFN(AP),-
                     23EC'CF         01A0                  W^CI_HI_PFN               ; Yes, copy it
                    6C    0D   91    01A3          CMPB    #VMB$Q_NODENAME/4,(AP)    ; Is the NODENAME arg present?
                          0D   14    01A6          BGTR    5$                        ; Branch if not
                       34 AC   D5    01A8          TSTL    VMB$Q_NODENAME(AP)        ; But is it null?
                          08   13    01AB          BEQL    5$                        ; Yes, don't copy
                       34 AC   7D    01AD          MOVQ    VMB$Q_NODENAME(AP),-
                 00000000'EF         01B0                  BOO$GB_NODENAME ; No, copy it
                                     01B5  5$:
                                     01B5  ;
                                     01B5  ; Get the top level system directory name (if present) out of the name
                                     01B5  ; of the secondary bootstrap stored by VMB in the RPB.
                                     01B5  ;
                 57    68 AB   9E    01B5          MOVAB   RPB$T_FILE(R11),R7        ; Address of bootstrap name
```

```
              56     87    9A  01B9          MOVZBL  (R7)+,R6                ; R6 = size, R7 = adr of string
      67      56  5D 8F    3A  01BC          LOCC    #^A/]/,R6,(R7)          ; Find the directory portion
                     06    12  01C1          BNEQ    10$                     ; of the name string
         67   56     3E    3A  01C3          LOCC    #^A/>/,R6,(R7)          ; Alternate syntax?
                     24    13  01C7          BEQL    END_LOOKUP              ; Branch if no directory present
                     57    D6  01C9  10$:    INCL    R7                      ; Step over the open bracket
              51     57    C2  01CB          SUBL    R7,R1                   ; Count in directory string
         67   51     2E    3A  01CE          LOCC    #^A/./,R1,(R7)          ; Any top level directory?
                     19    13  01D2          BEQL    END_LOOKUP              ; Branch if not
              51     57    C2  01D4          SUBL    R7,R1                   ; Size of top level dir name
              09     51    D1  01D7          CMPL    R1,#9                   ; Size of name ok?
                     11    14  01DA          BGTR    END_LOOKUP              ; Branch if not
         53   216D'CF      9E  01DC          MOVAB   W^FIL$GT_TOPSYS,R3      ; Location to store ASCIC string
              83     51    90  01E1          MOVB    R1,(R3)+                ; Store string size
         63   67     51    28  01E4          MOVC3   R1,(R7),(R3)            ; And store the directory name
                     03    11  01E8          BRB     END_LOOKUP
                               01EA  ;
                               01EA  ; VMB version does not have the FIL$OPENFILE cache
                               01EA  ;
                               01EA  ADD_CACHE:
                  FE13'    30  01EA          BSBW    BOO$CACHE_INIT          ; Otherwise init our own cache
                               01ED  END_LOOKUP:
         50       44 AB    7D  01ED          MOVQ    RPB$Q_PFNMAP(R11),R0    ; Get descriptor for PFN bitmap
              5E     51    D1  01F1          CMPL    R1,SP                   ; Is this the small bitmap?
                     0B    19  01F4          BLSS    10$                     ; Branch if yes
         50        7140    9E  01F6          MOVAB   -(R1)[R0],R0            ; Address of last byte inclusive
23D4'CF  50       F7 8F    78  01FA          ASHL    #-9,R0,W^PFNMAP_HI_PFN  ; Save highest PFN
                               0201  10$:
                               0201  ;
                               0201  ; With the latest version of VMB, essentially all the code is already
                               0201  ; gone.  This secondary bootstrap was read in over nearly all the VMB
                               0201  ; code.  So the following code, which used to lay some data over VMB's
                               0201  ; code, now simply places it at the end of SYSBOOT.  Even with old
                               0201  ; versions of VMB this still works since the sum of the 2 bootstraps
                               0201  ; and their data still does not exceed 64kb.
                               0201  ;
                               0201  OVERLAY_VMB:
                               0201          ASSUME  BOO$C_LENGTH&3 EQ 0     ; Integral number of long words
      52  03     09        9C  0201          ROTL    #9,#RTRV_PAG_CNT+<<BOO$C_LENGTH-64+511>@-9>,R2
                               0205                                          ; If front of Boot Control Block is
                               0205                                          ; larger than 64 bytes add a page
      53  2198'CF          D0  0205          MOVL    W^BOO$GL_FREEMEM,R3     ; Place BOOTCB and retrieval pointers
                               020A                                          ; at the end of SYSBOOT
      56  53     52        C1  020A          ADDL3   R2,R3,R6                ; Note where next free page is
     23C4'CF     53        D0  020E          MOVL    R3,W^BOOTCB             ; Address of boot control block
       53     28 A3        DE  0213          MOVAL   BOO$C_LENGTH(R3),R3     ; First address in rtrv buffer
       52     D8 A2        9E  0217          MOVAB   -BOO$C_LENGTH(R2),R2    ; Size of rtrv buffer
     23B4'CF     52        7D  021B          MOVQ    R2,W^RTRV_BUF_DSC       ; Set retrieval buffer descriptor
                               0220  ;
                               0220  ; Allocate space to keep the first block beyond the image header for
                               0220  ; each of the drivers and loadable code.  They must be read in to find
                               0220  ; the size of the driver.  This avoids reading them a second time.
                               0220  ;
      55  0C'    09        9C  0220          ROTL    #9,S^#<LOAD_IMAGE_CNT+1>,R5 ; No. of bytes needed
     23BC'CF     55        7D  0224          MOVQ    R5,W^VBN2_BUF_DSC       ; Save descriptor of buffer space
2198'CF   56     55        C1  0229          ADDL3   R5,R6,W^BOO$GL_FREEMEM  ; Note where next free page is
                               022F
```

```
                              022F  ;*****************************************************************************
                              022F  ;*      DEALLOCATE ANY PAGES SPECIFIED BY BAD PAGE FILE
                              022F  ;*
                              022F  ;*****************************************************************************
                              022F
                              022F  CHKVERSION:                               ; Check for minimum FPLA/WCS/PCS version
                              022F          .ENABL  LSB
                              022F                                            ;
                              022F          CPUDISP <<780,CHKVERS_780>,-      ; *Dispatch on CPU type*
                              022F                   <750,CHKVERS_750>,-      ;
                              022F                   <730,CHKVERS_730>,-
                              022F                   <790,CHKVERS_790>,-
                              022F                   <UV1,CHKVERS_UV1>,-
                              022F                   <785,CHKVERS_785>>,-
                              022F                  ENVIRON=VMB
                              027A
                              027A  CHKVERS_750:                              ; For 11/750, check SID
              52     00   9A  027A          MOVZBL  #HWREV_750, R2            ; R2 <- hardware ECO level
              53     3E   9A  027D          MOVZBL  #MICREV_750, R3           ; R3 <- microcode version
                     06   11  0280          BRB     CHECK_SID
                              0282
                              0282  CHKVERS_730:                              ; For 11/730, check SID
              52     00   9A  0282          MOVZBL  #HWREV_730,R2             ; R2 <- hardware ECO level
              53     33   9A  0285          MOVZBL  #MICREV_730,R3            ; R3 <- microcode version
                              0288  CHECK_SID:
              51     3E   DB  0288          MFPR    #PR$_SID,R1               ; Get SID
              52     51   91  028B          CMPB    R1, R2                    ; Current ECO level high enough?
                     0D   1F  028E          BLSSU   10$                       ; Branch if not
                              0290  CHKVERS_UCODE:
   51    51   F8 8F       78  0290          ASHL    #-8,R1,R1                 ; Position microcode version
              53     51   91  0295          CMPB    R1, R3                    ; Current microcode version high enough?
                     03   1F  0298          BLSSU   10$                       ; Branch if not
                   0147   31  029A          BRW     CHKVERS_END               ; Else versions ok
                              029D  10$:    MSG     <-W-ECO or microcode version less than minimum required for VMS.>
                   0101   31  02E0          BRW     BOO_HALT1                 ; Allow continue command to override
                              02E3          .DSABL  LSB
                              02E3                                            ;
                              02E3  CHKVERS_790:                              ; For 11/790:
              51     12   D0  02E3          MOVL    #MICREV_REQ_790,R1        ; Request microcode revision level
              52     02   D0  02E6          MOVL    #2,R2                     ; Number of bytes of data = 2
        53      23AC'CF   9E  02E9          MOVAB   W^CPUDATA,R3              ; Address of buffer to receive data.
                   1A04   30  02EE          BSBW    GETCONLOC_790             ; Read version from console memory.
        00      23AC'CF   B1  02F1          CMPW    W^CPUDATA,#MICREV_790     ; Current microcode version ok?
                     03   1F  02F6          BLSSU   10$                       ; Branch if not
                   00E9   31  02F8          BRW     CHKVERS_END               ; Else version ok.
                              02FB  10$:    MSG     <-W-Microcode version less than minimum required for VMS>
                   00AB   31  0336          BRW     BOO_HALT1                 ; Branch to halt.
                              0339
                              0339  CHKVERS_UV1:
              51     3E   DB  0339          MFPR    #PR$_SID,R1               ; Get SID
              53     04   9A  033C          MOVZBL  #MICREV_UV1,R3            ; R3 <- microcode version
                   FF4E   31  033F          BRW     CHKVERS_UCODE             ; Join common code.
                              0342
                              0342  CHKVERS_785:
        55      1CEC'CF   9E  0342          MOVAB   W^VERSVECT_785,R5         ; Get address of 785 version vector.
                1CF5'CF   DF  0347          PUSHAL  W^VERSNUM_785+4           ; Save addr of required 785 values
                     09   11  034B          BRB     CHK_780_785               ; Join common code.
```

```
                            034D
                            034D  CHKVERS_780:                                        ; For 11/780, continue with check
         55  1CE3'CF   9E   034D          MOVAB   W^VERSVECT_780,R5                   ; Get address of 780 version vector
             1CEC'CF   DF   0352          PUSHAL  W^VERSNUM_780+4                     ; Save addr of required 780 values
                            0356  CHK_780_785:
         56  23AC'CF   9E   0356          MOVAB   W^CPUDATA,R6                        ; Pointer to extra data
             51   85   9A   035B  10$:    MOVZBL  (R5)+,R1                            ; Get offset to version code
                  08   13   035E          BEQL    30$                                 ; Zero if end of list
                1955   30   0360  20$:    BSBW    GETCONLOC_780                       ; Ask console for value
             86   50   90   0363          MOVB    R0,(R6)+                            ; Store it away
                  F3   11   0366          BRB     10$                                 ; Loop
                            0368
             55   8E   D0   0368  30$:    MOVL    (SP)+,R5                            ; Retrieve addr of end of required
                            036B                                                      ; microcode rev levels
             75   76   B1   036B          CMPW    -(R6),-(R5)                         ; Check all microcode versions against
                  32   1F   036E          BLSSU   40$                                 ; minimum required
             75   76   91   0370          CMPB    -(R6),-(R5)
                  2D   1F   0373          BLSSU   40$
             75   76   91   0375          CMPB    -(R6),-(R5)
                  28   1F   0378          BLSSU   40$                                 ; Failure
                            037A  ;
                            037A  ; Now pick up FPA status, WCS size, and G&H option flags
                            037A  ;
             50   28   DB   037A          MFPR    #PR780$_ACCS,R0                     ; Pick up FPA status
     50  50   F2  8F   9C   037D          ROTL    #-14,R0,R0                          ; Get bit in right position
23B0'CF  50   02       89   0382          BISB3   #2,R0,W^CPUDATA+4                   ; Set the flag
         51   64  8F   9A   0388          MOVZBL  #GHOPT_VLOC,R1                      ; Get offset to GH options flags
                1929   30   038C          BSBW    GETCONLOC_780                       ; Ask console for value
     50  50   01       89   038F          BISB3   #1,R0,R0                            ; Get the flag
        23B0'CF   50   88   0393          BISB    R0,W^CPUDATA+4                      ; Set the flag
             50   2D   DB   0398          MFPR    #PR$_WCSD,R0                        ; Get WCS size
        23B1'CF   50   90   039B          MOVB    R0,W^CPUDATA+5                      ; Transfer to storage
                  42   11   03A0          BRB     CHKVERS_END                         ; All okay
                            03A2
                            03A2  40$:    MSG     <-W-FPLA,PCS or WCS version less than minimum required for VMS.>
                            03E4  BOO_HALT1:                                          ;
                            03E4  ;
                            03E4  ;                       Halt removed to leave warning message only.
                            03E4  ;
                            03E4  CHKVERS_END:                                        ; *End of CPU-dependent code*
                            03E4                                                      ;
                            03E4  GETCURRENT:                                         ; Read current parameter values
         52  23F0'CF   9E   03E4          MOVAB   W^SYS_STAT,R2                       ; Address of SYSTEM statistics block
                1198   30   03E9  5$:     BSBW    IMAGE_OPEN                          ; Locate SYS.EXE
               31 50   E8   03EC          BLBS    R0,15$                              ; Branch if successful
                            03EF  ;
                            03EF  ; If we were looking for SYS.EXE in SYS0.DIR, then try to boot from
                            03EF  ; SYSEXE.DIR in the MFD for compatibility with Release 2 format system disks
                            03EF  ;
             216D'CF   95   03EF          TSTB    W^FIL$GT_TOPSYS                     ; Try this second lookup once only
                  0B   13   03F3          BEQL    10$                                 ; Branch if no TOPSYS in use
             216D'CF   94   03F5          CLRB    W^FIL$GT_TOPSYS                     ; Disable TOPSYS
     2171'CF      30   91   03F9          CMPB    #^A/0/,W^FIL$GT_TOPSYS+4            ; Trying to boot from [SYS0.SYSEXE]?
                  E9   13   03FE          BEQL    5$                                  ; Yes, try again from [SYSEXE]
                            0400  10$:    MSG     <-F-Unable to locate SYS.EXE>
                       00   041F          HALT                                        ; ***** Fatal Error *****
             53   01   D0   0420  15$:    MOVL    #1,R3                               ; Virtual block 1 is image header
```

```
          54  2400'CF   D0 0423          MOVL    W^SYS_STAT+STAT_L_MAP,R4 ; Virtual to logical block map
          56  2198'CF   D0 0428          MOVL    W^BOO$GL_FREEMEM,R6      ; Set buffer address
          59  01   09   9C 042D          ROTL    #9,#1,R9                 ; Read only header
              1459      30 0431          BSBW    READ_VIRTUAL             ; Read image header
              03 50     E8 0434          BLBS    R0,50$                   ; Branch if successful
              0FCA      31 0437          BRW     READ_SYS_ERR             ; Error reading SYS.EXE - fatal
              53   56   D0 043A  50$:    MOVL    R6,R3                    ; Set base address
    52  23F4'CF    17   9C 043D          ROTL    #<32-9>,W^SYS_STAT+STAT_L_BYTECNT,R2 ; No. of pages in file
              FBBA'     30 0443          BSBW    BOO$IMAGE_ATT            ; Get image attributes
                           0446  ;
                           0446  ; R1 = count of image header blocks
                           0446  ; R2 = last VBN in image - excluding symbol table and patch text
                           0446  ; R4 = preserved = virtual to logical map
                           0446  ;
          53  51   01   C1 0446          ADDL3   #1,R1,R3                 ; Starting VBN of image beyond hdr
          23F0'CF  53   D0 044A          MOVL    R3,W^SYS_STAT+STAT_L_VBN ; Save in stat block
              52   51   C2 044F          SUBL    R1,R2                    ; Blocks in image excluding image header
                           0452                                           ; symbol table, and patch text
    23F4'CF   52   09   78 0452          ASHL    #9,R2,W^SYS_STAT+STAT_L_BYTECNT ; Save byte count of image file
          5A  23C4'CF   D0 0458          MOVL    W^BOOTCB,R10             ; Address of boot control block
       14 AA  54   5A   C3 045D          SUBL3   R10,R4,BOO$L_SYS_MAP(R10) ; Save virtual to logical map
          0C AA    53   D0 0462          MOVL    R3,BOO$L_SYS_VBN(R10)    ; Save starting VBN of SYS
          10 AA    52   D0 0466          MOVL    R2,BOO$L_SYS_SIZE(R10)   ; and its size from that VBN
                           046A                                           ; to the end of executable image
                           046A  ;
                           046A  ; Add enough space to map for SYS.EXE to convert it into a Window Control Block in
                           046A  ; INIT.  If booting from the console, allow for a couple of extra pointers in case
                           046A  ; we read from another volume(s).
                           046A  ;
          56  23B4'CF   7D 046A          MOVQ    W^RTRV_BUF_DSC,R6        ; R6 = size, R7 = address of
                           046F                                           ; retrieval buffer
      50  64   FD 8F    78 046F          ASHL    #-3,(R4),R0              ; Number of retrieval pointers
     40 8F     66 AB    91 0474          CMPB    RPB$B_DEVTYP(R11),#BTD$K_CONSOLE ; Booting from the console?
                   09   12 0479          BNEQ    55$                      ; No
              50   03   C0 047B          ADDL2   #3,R0                    ; Allow a couple of extra pointers for a
                           047E                                           ;  segmented read during standalone operatio
    2398'CF   50   01   C3 047E          SUBL3   #1,R0,W^CONEOF_MAX_PTR   ; Store this count, adjusted for the dummy
                           0484                                           ;  rtrv pointer needed for each segment
              50   06   C4 0484  55$:    MULL    #6,R0                    ; 6 bytes each for WCB
                           0487  ;
                           0487  ; Note that SYS.EXE is smaller than 65K blocks and thus each retrieval
                           0487  ; pointer must be less than 65K blocks.  So 1 retrieval pointer cannot
                           0487  ; result in more than one window control block pointer.
                           0487  ;
              50   30   C0 0487          ADDL    #WCB$W_P1_COUNT,R0       ; Size required for WCB
          51  64   04   C1 048A          ADDL3   #4,(R4),R1               ; Size taken up by map right now
              50   51   C2 048E          SUBL    R1,R0                    ; Additional bytes needed to
                           0491                                           ; convert into a Window Control Block
                   0C   15 0491          BLEQ    60$                      ; Branch if already big enough
              50   03   C0 0493          ADDL    #3,R0                    ; Round up to long word boundary
              50   03   CA 0496          BICL    #3,R0                    ;
              56   50   C2 0499          SUBL    R0,R6                    ; Reserve the additional space
              57   50   C0 049C          ADDL    R0,R7                    ;
                           049F  60$:
                           049F  ;
                           049F  ; Make separate virtual to logical maps for SYSPARAM and for the
                           049F  ; non-resident portion of BUGCHECK.  This reduces the amount of
```

```
                                049F  ; code needed in the system to deal with these portions of SYS.EXE
                                049F  ; that are no longer required to be contiguous.
                                049F  ;
      53  FFC00000'8F       C0  049F          ADDL    #<MMG$A_SYSPARAM-^X80000000>@-9,R3 ; VBN of SYSPARAM
              55   00'      D0  04A6          MOVL    S^#<<BOO$C_SYSPARSZ+511>@-9>,R5 ; Number of blocks in SYSPARAM
   04 AA      57   5A       C3  04A9          SUBL3   R10,R7,BOO$L_PARAM_MAP(R10) ; Offset to map for SYSPARAM
                 136E       30  04AE          BSBW    BOO$SETMAP              ; Set virtual to logical map for
                                04B1                                          ; SYSPARAM portion of SYS.EXE
         54   2400'CF       D0  04B1          MOVL    W^SYS_STAT+STAT_L_MAP,R4 ; Get map for SYS again
         53   0000'8F       3C  04B6          MOVZWL  #<BUG$A_PAGED-^X80000000>@-9,R3 ; Starting VBN of paged
         53   23F0'CF       C0  04BB          ADDL    W^SYS_STAT+STAT_L_VBN,R3 ; BUGCHECK code
              55   00'      D0  04C0          MOVL    S^#<BUG$A_PAGEDEND-BUG$A_PAGED+511>@-9,R5
                                04C3                                          ; Blocks of paged BUGCHECK code
   24 AA      57   5A       C3  04C3          SUBL3   R10,R7,BOO$L_BUG_MAP(R10) ; Offset to map for BUGCHECK code
                 1354       30  04C8          BSBW    BOO$SETMAP              ; Set virtual to logical map for
                                04CB                                          ; BUGCHECK portion of SYS.EXE
       23B4'CF     56       7D  04CB          MOVQ    R6,W^RTRV_BUF_DSC       ; Record space used in rtrv ptr buf
                                04D0  ;
                                04D0  ; Read the sysgen parameters from VAXVMSSYS.PAR
                                04D0  ;
   00000000'EF     00       FB  04D0          CALLS   #0,BOO$USECUR           ; Read the .PAR file
                                04D7  ;
                                04D7  ; If this is a conversational bootstrap, then prompt terminal for command
                                04D7  ; input to specify a starting set of parameters and any modifications
                                04D7  ; desired.  In the event that a conversational boot was not selected and
                                04D7  ; there is no current parameter set, the defaults will be used anyway.
                                04D7  ;
        11 30 AB     00     E1  04D7          BBC     #RPB$V_CONV,RPB$L_BOOTR5(R11),NOCONVER
                                04DC                                          ; Branch if not conversational
       0000'CF     00       FB  04DC          CALLS   #0,W^BOO$GETPARAM       ; Prompt for parameters
                                04E1  ;
                                04E1  ; Restore fields that could have been destroyed by a USE DEFAULT command.
                                04E1  ;
  0000'CF   23A8'CF         90  04E1          MOVB    W^CPUTYPE,W^EXE$GB_CPUTYPE ; Restore CPU type to sys params
       0000'CF     3E       DB  04E8          MFPR    #PR$_SID,W^EXE$GB_CPUDATA  ; Restore SID to sys params
                                04ED  NOCONVER:
                                04ED  ;
                                04ED  ; Set paged code boundary in SYSPARAM area with value assembled into SYSBOOT
0000'CF   0000254A'EF       D0  04ED          MOVL    BOO$GL_PGDCOD,W^MMG$GL_PGDCOD;
                                04F6
                                04F6  ;
                                04F6  ; Now open SYSDUMP.DMP and record its retrieval pointers in the
                                04F6  ; boot control block.
                                04F6  ;
          18 AA    01       D0  04F6          MOVL    #1,BOO$L_DMP_VBN(R10)   ; Set dump file starting VBN
                                04FA
                                04FA          ASSUME  BOO$L_DMP_MAP EQ BOO$L_DMP_SIZE+4
               1C AA        7C  04FA          CLRQ    BOO$L_DMP_SIZE(R10)     ; Zero size and map in case no dump file
         52   2408'CF       9E  04FD          MOVAB   W^SYSDUMP_STAT,R2       ; Adr of SYSDUMP stat block
              0000'CF       D4  0502          CLRL    W^EXE$GW_PGFL_FID       ; No file id for page file
              0004'CF       B4  0506          CLRW    W^EXE$GW_PGFL_FID+4     ;
      00 0000'CF   00'      E5  050A          BBCC    S^#EXE$V_PAGFILDMP,W^EXE$GL_DEFFLAGS,10$ ; Assume dump is
                                0510                                          ; not going to be in the page file
         54   23B4'CF       7D  0510  10$:    MOVQ    W^RTRV_BUF_DSC,R4       ; Save old descriptor for size check
  0C A2   00002177'9F       DE  0515          MOVAL   @#BOO$GT_SYSDUMP,STAT_L_NAME(R2) ; Look up SYSDUMP.DMP
                 1064       30  051D          BSBW    IMAGE_OPEN              ; Locate SYSDUMP.DMP
               4C 50        E9  0520          BLBC    R0,15$                  ; Branch if not found
```

```
   56   54   23B4'CF   C3  0523          SUBL3   W^RTRV_BUF_DSC,R4,R6          ; Bytes used for dump file
  000002EC 8F     56   D1  0529          CMPL    R6,-                          ; See if we used less than half
                           0530                  #<<RTRV_PAG_CNT*512>-BOO$C_LENGTH>/2
                  5D   19  0530          BLSS    20$                           ; No problem
        23B4'CF   54   7D  0532          MOVQ    R4,W^RTRV_BUF_DSC             ; Forget about dump
                           0537          MSG     <-W-DMPFRG SYSDUMP.DMP is too fragmented to be used>
                  2C   11  056D          BRB     60$
  0C A2    00002183'9F 9E  056F  15$:    MOVAB   @#BOO$GT_PAGEFILE,STAT_L_NAME(R2) ; Didn't find SYSDUMP.DMP
                100A   30  0577          BSBW    IMAGE_OPEN                    ; Lookup PAGEFILE.SYS instead
               1E 50   E9  057A          BLBC    R0,60$                        ; Branch if this failed too
     0000'CF   08 A1   D0  057D          MOVL    FH2$W_FID_NUM(R1),W^EXE$GW_PGFL_FID ; Save the file id
     0004'CF   0C A1   B0  0583          MOVW    FH2$W_FID_RVN(R1),W^EXE$GW_PGFL_FID+4 ; for the page file
  00 0000'CF      00'  E2  0589          BBSS    S^#EXE$V_PAGFILDMP,W^EXE$GL_DEFFLAGS,20$ ; Note dump file
                           058F                                                ; is at the front of the page file
          04 A2   17   9C  058F  20$:    ROTL    #<32-9>,STAT_L_BYTECNT(R2),- ; Record size of file
                1C AA      0593                  BOO$L_DMP_SIZE(R10)           ; in blocks
   20 AA    10 A2  5A  C3  0595          SUBL3   R10,STAT_L_MAP(R2),BOO$L_DMP_MAP(R10) ; Offset to SYSDUMP map
                           059B
                           059B  ;
                           059B  ; Now calculate the size of nonpaged pool needed for the bootstrap
                           059B  ; system disk driver.  This includes the BOOTCB (boot control block)
                           059B  ; and any possible ucode file needed.
                           059B  ;
        56   23B4'CF   7D  059B  60$:    MOVQ    W^RTRV_BUF_DSC,R6             ; Get size and adr of what remains
                           05A0                                                ; of the retrieval pointer buffer
                           05A0                                                ; R7 = end of BOOTCB
         50   57   5A  C3  05A0          SUBL3   R10,R7,R0                     ; BOOTCB size
          08 AA    50  B0  05A4          MOVW    R0,BOO$W_SIZE(R10)            ; Save size in BOOTCB
               6A  50  3C  05A8          MOVZWL  R0,BOO$L_CHECKSUM(R10)        ; Save size to checksum for INIT
           0663 8F     B0  05AB          MOVW    #<DYN$C_BOOTCB@8!DYN$C_INIT>,-
                0A AA      05AF                  BOO$B_TYPE(R10)               ; Set type and subtype
        50   23E0'CF   C0  05B1          ADDL2   W^UCODE_LEN,R0                ; Add in any possible ucode file size
  2424'CF   50   38 AB C1  05B6          ADDL3   RPB$L_IOVECSZ(R11),R0,-       ; Save size with Boot Driver added
                           05BD                  W^BOODRV_STAT+STAT_L_BYTECNT
                           05BD                                                ; in Boot Driver statistics block
                  87   7C  05BD          CLRQ    (R7)+                         ; Since allocation will be rounded
                  87   7C  05BF          CLRQ    (R7)+                         ; up to 16 byte boundary, clean out
                           05C1                                                ; any left over garbage
              56   10  C2  05C1          SUBL    #16,R6                        ; and allocate an additional 16 bytes
        23B4'CF   56   7D  05C4          MOVQ    R6,W^RTRV_BUF_DSC             ; of retrieval buffer space
                           05C9  ;
                           05C9  ; The boot device driver file will be opened and read to obtain
                           05C9  ; the driver size.  This will be passed to the EXEC initialization.
                           05C9  ; Determining the name of the boot driver file is done differently,
                           05C9  ; depending on the version of VMB being used.
                           05C9  ;
                           05C9  LOOKUPDRIVER:
            239C'CF    D5  05C9          TSTL    W^VMB_VERSION                 ; Release 2 or later VMB?
                  1F   13  05CD          BEQL    2$                            ; Branch if not
          50    34 AB  D0  05CF          MOVL    RPB$L_IOVEC(R11),R0           ; Get address of I/O vectors
       54    0C B040   9E  05D3          MOVAB   @BQO$L_DRIVRNAME(R0)[R0],R4 ; Yes, get addr. of driver name list
       0A    239C'CF   D1  05D8          CMPL    W^VMB_VERSION,#10             ; Is it version 10?
                  0C   19  05DD          BLSS    19$                           ; No, skip this
        51    30 B040  9E  05DF          MOVAB   @BQO$L_DEVNAME(R0)[R0],R1 ; Yes, transfer boot device name
    00000000'EF   61   3C  05E4          MOVZWL  (R1),L^BOO$GL_DEVNAME         ;  to BOOPARAM block
                009E   31  05EB  19$:    BRW     20$
                           05EE
```

```
                              05EE  ;
                              05EE  ; The type of boot device will be determined and the proper driver file
                              05EE  ; will be looked up and read to obtain the proper driver size. This will
                              05EE  ; be passed to the exec initialization.  If the type of boot
                              05EE  ; device does not correspond to one of the standard disk devices, then
                              05EE  ; the driver name will be set to UNKDRIVER.EXE and a fatal error message
                              05EE  ; given if the driver file is not found.
                              05EE  ;
                              05EE  2$:
         55   5C AB    D0     05EE              MOVL    RPB$L_ADPPHY(R11),R5     ; Get physical address of bus adapter
              66 AB    91     05F2              CMPB    RPB$B_DEVTYP(R11),-      ; Booting from console block
              40 8F           05F5                      #BTD$K_CONSOLE           ; storage device?
                 3F    12     05F7              BNEQ    5$                       ; No
         54  2213'CF   9E     05F9              MOVAB   W^DDNAME,R4              ; Yes, assume TU58
                              05FE              CPUDISP <<780,4$>,-              ; 11/780 floppy
                              05FE                       <750,20$>,-             ; 11/750 TU58
                              05FE                       <730,20$>,-             ; 11/730 TU58
                              05FE                       <UV1,20$>,-             ; MicroVAX I should never get here
                              05FE                      >,-                      ;
                              05FE                      ENVIRON=VMB
                              0631                                               ; This code path can't be executed on 11/790
                              0631
         54  2206'CF   9E     0631  4$:         MOVAB   W^DXNAME,R4              ; Console RX01 driver name
                 54    11     0636              BRB     20$
                              0638
                              0638  5$:         CASE    RPB$B_DEVTYP(R11),<-     ; Case on boot device type
                              0638                      10$,-                    ; MASSBUS device
                              0638                      9$,-                     ; RK06/7
                              0638                      8$,-                     ; RL02
                              0638                      >,LIMIT=#BTD$K_MB,TYPE=B
                 3B    11     0643              BRB     12$                      ; Unknown device
                              0645
         54  21F9'CF   9E     0645  8$:         MOVAB   W^DLNAME,R4              ; RL02 driver name
                 40    11     064A              BRB     20$
         54  21DF'CF   9E     064C  9$:         MOVAB   W^DMNAME,R4              ; RK06/7 driver name
                 39    11     0651              BRB     20$                      ;
         55  0400 C5   9E     0653  10$:        MOVAB   ^X400(R5),R5             ; Compute address of device registers
 55   03  07   64 AB   F0     0658              INSV    RPB$W_UNIT(R11),#7,#3,R5; For boot device
         50    18 A5   D0     065E              MOVL    RP_DT(R5),R0             ; Read device type number
         54  21D2'CF   9E     0662              MOVAB   W^DBNAME,R4              ; Assume RP04/RP05/RP06 (DBDRIVER)
         12      50    91     0667              CMPB    R0,#^X12                 ; Is drive type = RP04/5/6
                 20    15     066A              BLEQ    20$                      ; Branch if yes
         50      14    91     066C              CMPB    #^X14,R0                 ; Is device RM03?
                 16    13     066F              BEQL    15$                      ; Yes
         50      16    91     0671              CMPB    #^X16,R0                 ; IS DEVICE = RM80?
                 11    13     0674              BEQL    15$                      ; IF EQL, YES
         50      17    91     0676              CMPB    #^X17,R0                 ; IS DEVICE = RM05?
                 0C    13     0679              BEQL    15$                      ; IF EQL, YES
         50      22    91     067B              CMPB    #^X22,R0                 ; IS DEVICE = RP07?
                 07    13     067E              BEQL    15$                      ; Yes
         54  2220'CF   9E     0680  12$:        MOVAB   W^UNKNAME,R4             ; Use driver for unknown devices
                 05    11     0685              BRB     20$                      ;
         54  21EC'CF   9E     0687  15$:        MOVAB   W^DRNAME,R4              ; Use DRDRIVER for RM03/RP07
         03  239C'CF   D1     068C  20$:        CMPL    W^VMB_VERSION,#3         ; Did this VMB set up the MEMDSC in RPB?
                 03    18     0691              BGEQ    25$                      ; Branch if it did
               0ECD    30     0693              BSBW    ADD_RPB_MEMDSC           ; PUT MEMDSC IN RPB
 2444'CF 54  23A0'CF   C3     0696  25$:        SUBL3   W^SYSBOOT_BASE,R4,-      ; Save base relative address
```

```
                              069E                  W^DSKDRV_STAT+STAT_L_NAME ;  of driver name
         07  239C'CF    D1    069E          CMPL    W^VMB_VERSION,#7           ; Version 7 or greater?
                  20    19    06A3          BLSS    30$                        ; No, leave now
      50     34 AB      D0    06A5          MOVL    RPB$L_IOVEC(R11),R0        ; Get address of I/O vectors
      54     20 A0      D0    06A9          MOVL    BQO$L_AUXDRNAME(R0),R4     ; Get addr. of aux driver name list
                  0B    13    06AD          BEQL    27$                        ; None, leave
             54   50    C0    06AF          ADDL2   R0,R4                      ; Get absolute address
 245C'CF     54  23A0'CF C3   06B2          SUBL3   W^SYSBOOT_BASE,R4,-        ; Save base relative address
                              06BA                  W^PRTDRV_STAT+STAT_L_NAME ;  of driver name
                              06BA  27$:
         09  239C'CF    D1    06BA          CMPL    W^VMB_VERSION,#9           ; Did VMB set up RPB BADPGS field?
                  04    18    06BF          BGEQ    30$                        ; Yes, leave now
           0104 CB      D4    06C1          CLRL    RPB$L_BADPGS(R11)          ; Zero number of bad pages found in
                              06C5                                             ; bootstrap memory scan.
                              06C5  30$:
         0D  239C'CF    D1    06C5          CMPL    W^VMB_VERSION,#13          ; Did VMB set up RPB CTRLLTR field?
                  04    18    06CA          BGEQ    40$                        ; Yes, leave now.
           0108 CB      94    06CC          CLRB    RPB$B_CTRLLTR(R11)         ; Zero controller letter field.
                              06D0  40$:
                              06D0  ;
                              06D0  ; Derive the name of the image containing  loadabe CPU-dependent
                              06D0  ; code.  Image names supported thus far:
                              06D0  ;
                              06D0  ;       SYSLOA780.EXE    if CPU_TYPE has PR$_SID_TYP780.
                              06D0  ;       SYSLOA750.EXE    if CPU_TYPE has PR$_SID_TYP750.
                              06D0  ;       SYSLOA730.EXE    if CPU_TYPE has PR$_SID_TYP730.
                              06D0  ;       SYSLOA790.EXE    if CPU_TYPE has PR$_SID_TYP790.
                              06D0  ;       SYSLOAUV1.EXE    if CPU_TYPE has PR$_SID_TYPUV1.
                              06D0  ;       SYSLOAWS1.EXE    if CPU_TYPE has PR$_SID_TYPUV1 and bit 6 in boot r1 is set.
                              06D0  ;
                              06D0  ;
         50  0000'CF    9A    06D0          MOVZBL  W^EXE$GB_CPUTYPE,R0        ; Get CPU type code
      56   2304'CF40    DE    06D5          MOVAL   W^MODEL_TABLE-4[R0],R6     ; Get addr of model list
 2301'CF     66   03    28    06DB          MOVC3   #3,(R6),W^NAME_XXX         ; Copy model name into XXX
                              06E1                                             ;   field of SYSLOAXXX.EXE
                              06E1
                              06E1  ;
                              06E1  ; Conditionally look up loadable $ERAPAT code (ERAPATLOA.EXE).
                              06E1  ;
                              06E1  LOOKUP_LOADABLE_CODE:
                 00'    E0    06E1          BBS     S^#SGN$V_LOADERAPAT,-      ; Is $ERAPAT load flag set?
         04 0000'CF           06E3                  W^SGN$GL_LOADFLAGS,20$     ;   if so, branch around the CLRL
            24D4'CF     D4    06E7          CLRL    W^ERAPATLOA_STAT+STAT_L_NAME ; Set for no lookup
                              06EB
                              06EB  ;
                              06EB  ; Conditionally look up loadable $CHKPRT code (CHKPRTLOA.EXE).
                              06EB  ;
                 00'    E0    06EB  20$:    BBS     S^#SGN$V_LOADCHKPRT,-      ; Is $CHKPRT load flag set?
         04 0000'CF           06ED                  W^SGN$GL_LOADFLAGS,25$     ;   if so, branch around the CLRL
            24EC'CF     D4    06F1          CLRL    W^CHKPRTLOA_STAT+STAT_L_NAME ; Set for no lookup
                              06F5
                              06F5  ; Conditionally look up loadable $MTACCESS code (MTACCESS.EXE).
                              06F5
   04 0000'CF    00'    E0    06F5  25$:    BBS     S^#SGN$V_LOADMTACCESS,W^SGN$GL_LOADFLAGS,30$ ; If SET branch around
            2534'CF     D4    06FB          CLRL    W^MTACCESSLOA_STAT+STAT_L_NAME ; Set for no look up
                              06FF  ;
                              06FF  ; Conditionally look up loadable SCS code (SCSLOA.EXE).
```

```
                              06FF  ; Conditionally lookup loadable cluster code (CLUSTRLOA.EXE).
                              06FF  ;
      01  0000'CF     91      06FF  30$:     CMPB    W^CLU$GB_VAXCLUSTER,#1   ; Test VAXCLUSTER parameter
                13    1A      0704           BGTRU   40$                      ; Load SCSLOA and CLUSTRLOA
                04    13      0706           BEQLU   32$                      ; Branch if VAXCLUSTER = 1
          24BC'CF     D4      0708           CLRL    W^CLSLOA_STAT+STAT_L_NAME ; Set for no CLUSTRLOA lookup
                              070C  32$:
                              070C           ASSUME  VMB$V_LOAD_SCS EQ 0
       08 23E8'CF     E8      070C           BLBS    W^VMB_FLAGS,40$          ; Look it up
          248C'CF     D4      0711           CLRL    W^SCSLOA_STAT+STAT_L_NAME ; Set for no SCSLOA lookup
          24BC'CF     D4      0715           CLRL    W^CLSLOA_STAT+STAT_L_NAME ; Set for no CLUSTRLOA lookup
                              0719  40$:
                              0719  ;
                              0719  ; Check for the presence of the TBCHK processor register.
                              0719  ;
                00'   E5      0719           BBCC    S^#EXE$V_TBCHK,-         ; Clear it out befor we check
       00 0000'CF             071B                   W^EXE$GL_DEFFLAGS,43$
            58  11    DB      071F  43$:     MFPR    #PR$_SCBB,R8             ; Pick up address of SCCB
             18 A8    DD      0722           PUSHL   ^X18(R8)                 ; Save RESERVED OPERAND slot
            57  5E    D0      0725           MOVL    SP,R7                    ; Save stack
     18 A8   38'AF    9E      0728           MOVAB   B^44$,^X18(R8)           ; Put in new handler
            3F  50    DA      072D           MTPR    R0,#PR$_TBCHK            ; Try to write it
                00'   E2      0730           BBSS    S^#EXE$V_TBCHK,-
       02 0000'CF             0732                   W^EXE$GL_DEFFLAGS,44$    ; It worked so set the bit
                00    11      0736           BRB     44$
                              0738           .ALIGN  LONG                     ; Vectors must be on longword boundary
            5E  57    D0      0738  44$:     MOVL    R7,SP                    ; Restore stack
          18 A8 8ED0          073B           POPL    ^X18(R8)                 ; Restore RESERVED OPERAND
                              073F           .ENABL  LSB
                              073F  ;
                              073F  ; Check for the presence of each category of instructions that might
                              073F  ; have to be emulated:  character string, decimal, EDITPC, CRC and the
                              073F  ; floating point data types.  Set an indicator for any that must be
                              073F  ; emulated by software.
                              073F  ;
                              073F  ; WARNING:  This code cannot be debugged with XDELTA, if string instructions
                              073F  ;           are being emulated because this testing replaces the emulation
                              073F  ;           handlers temporarily!
                              073F  ;
                              073F
                              073F           ASSUME ARC$V_DCML_EMUL EQ <ARC$V_CHAR_EMUL + 1>
                              073F           ASSUME ARC$V_EDPC_EMUL EQ <ARC$V_DCML_EMUL + 1>
                              073F           ASSUME ARC$V_CRC_EMUL EQ <ARC$V_EDPC_EMUL + 1>
                              073F           ASSUME ARC$V_DFLT_EMUL EQ <ARC$V_CRC_EMUL + 1>
                              073F           ASSUME ARC$V_FFLT_EMUL EQ <ARC$V_DFLT_EMUL + 1>
                              073F           ASSUME ARC$V_GFLT_EMUL EQ <ARC$V_FFLT_EMUL + 1>
                              073F           ASSUME ARC$V_HFLT_EMUL EQ <ARC$V_GFLT_EMUL + 1>
                              073F
             10 A8    DD      073F           PUSHL   ^X10(R8)                 ; Remember OPCDEC vector contents
           00C8 C8    DD      0742           PUSHL   ^XC8(R8)                 ; Remember emulation vector contents
           00CC C8    DD      0746           PUSHL   ^XCC(R8)                 ; Remember FPD emulation vector contents
           00FF 8F    BB      074A           PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7> ; Save volatile registers
           0000'CF    D4      074E           CLRL    W^EXE$GL_ARCHFLAG        ; Clear all indicators initially
     10 A8 07D4'CF    9E      0752           MOVAB   W^45$,^X10(R8)           ; Put in new OPCDEC handler
   00C8 C8 07E4'CF    9E      0758           MOVAB   W^46$,^XC8(R8)           ; Put in new emulation handler
   00CC C8 07D4'CF    9E      075F           MOVAB   W^45$,^XCC(R8)           ; Put in new FPD emulation handler
            56  04    9A      0766           MOVZBL  #ARC$V_CHAR_EMUL,R6      ; Bit indicator for character string ins
```

```
                                  0769  ;
                                  0769  ; MATCHC was chosen as the sample string instruction to be executed
                                  0769  ; because it is the least likely character instruction to be implemented.
                                  0769  ;
                        00000001  0769          .MDELETE MATCHC
                 57   09'8F   9A  0769          MOVZBL  #MATCHC_INS_SZ,R7         ; Remember size of MATCHC instruction
                                  076D  MATCHC_INS:
 07F2'CF   01   07F2'CF   01  39  076D          MATCHC  #1,W^STRING,#1,W^STRING ; Execute a character string ins
                        00000009  0776  MATCHC_INS_SZ = . - MATCHC_INS
                                  0776
                                  0776  ;
                                  0776  ; DIVP was chose as the sample decimal instruction to be executed
                                  0776  ; because it is the least likely decimal instruction to be implemented.
                                  0776  ;
                          56  D6  0776          INCL    R6                        ; Bit indicator for decimal instruction
                        00000001  0778          .MDELETE DIVP
                 57   0D'8F   9A  0778          MOVZBL  #DIVP_INS_SZ,R7           ; Remember size of DIVP instruction
                                  077C  DIVP_INS:
             07F4'CF   01     27  077C          DIVP    #1,W^DENOMINATOR,-        ; Execute a decimal instruction
             07F3'CF   01         0781                  #1,W^NUMERATOR,-
             07F5'CF   01         0785                  #1,W^QUOTIENT
                        0000000D  0789  DIVP_INS_SZ = . - DIVP_INS
                                  0789
                                  0789  ;
                                  0789  ; This EDITPC instruction does nothing but test if software emulation is
                                  0789  ; needed for this instruction.
                                  0789  ;
                          56  D6  0789          INCL    R6                        ; Bit indicator for EDITPC instruction
                        00000001  078B          .MDELETE EDITPC
                 57   0B'8F   9A  078B          MOVZBL  #EDITPC_INS_SZ,R7         ; Remember size of EDITPC instruction
                                  078F  EDITPC_INS:
     07F9'CF   07F3'CF   01   38  078F          EDITPC  #1,W^NUMERATOR,W^PATTERN,W^QUOTIENT ; Execute an EDITPC instruction
                 07F5'CF          0797
                        0000000B  079A  EDITPC_INS_SZ = . - EDITPC_INS
                                  079A
                                  079A  ;
                                  079A  ; This CRC produces a garbage result but it tests if software emulation is
                                  079A  ; needed for this instruction.
                                  079A  ;
                          56  D6  079A  TBL:    INCL    R6                        ; Bit indicator for CRC instruction
                        00000001  079C          .MDELETE CRC
                          50  D4  079C          CLRL    R0                        ; Set inicrc to zero
                 57   08'8F   9A  079E          MOVZBL  #CRC_INS_SZ,R7            ; Remember size of CRC instruction
                                  07A2  CRC_INS:
                       F5 AF  0B  07A2          CRC     TBL,-                     ; Choose useless address for tbl address
        07F2'CF   01      50      07A5                  R0,#1,W^STRING            ; Use same string as in MATCHC test
                        00000008  07AA  CRC_INS_SZ = . - CRC_INS
                                  07AA
                                  07AA  ;
                                  07AA  ; This is a sample D floating point instruction.  It was chosen because
                                  07AA  ; it is one of the simplest D instructions.
                                  07AA  ;
                          56  D6  07AA          INCL    R6                        ; Bit indicator for D floating point
                        00000001  07AC          .MDELETE MOVD
                 57   03'8F   9A  07AC          MOVZBL  #MOVD_INS_SZ,R7           ; Remember size of MOVD instruction
                                  07B0  MOVD_INS:
               50     00      70  07B0          MOVD    S^#00,R0                  ; Execute an D floating point instruction
```

```
          00000003  07B3  MOVD_INS_SZ = . - MOVD_INS
                    07B3
                    07B3  ;
                    07B3  ; This is a sample F floating point instruction.  It was chosen because
                    07B3  ; it is one of the simplest F instructions.
                    07B3  ;
          56    D6  07B3          INCL    R6                          ; Bit indicator for F floating point
          00000001  07B5          .MDELETE MOVF
    57  03'8F   9A  07B5          MOVZBL  #MOVF_INS_SZ,R7             ; Remember size of MOVF instruction
                    07B9  MOVF_INS:
     50   00    50  07B9          MOVF    S^#00,R0                    ; Execute an F floating point instruction
          00000003  07BC  MOVF_INS_SZ = . - MOVF_INS
                    07BC
                    07BC  ;
                    07BC  ; This is a sample G floating point instruction.  It was chosen because
                    07BC  ; it is one of the simplest G instructions.
                    07BC  ;
          56    D6  07BC          INCL    R6                          ; Bit indicator for G floating point
          00000001  07BE          .MDELETE MOVG
    57  04'8F   9A  07BE          MOVZBL  #MOVG_INS_SZ,R7             ; Remember size of MOVG instruction
                    07C2  MOVG_INS:
     50   00  50FD  07C2          MOVG    S^#00,R0                    ; Execute an G floating point instruction
          00000004  07C6  MOVG_INS_SZ = . - MOVG_INS
                    07C6
                    07C6  ;
                    07C6  ; This is a sample H floating point instruction.  It was chosen because
                    07C6  ; it is one of the simplest H instructions.
                    07C6  ;
          56    D6  07C6          INCL    R6                          ; Bit indicator for H floating point
          00000001  07C8          .MDELETE MOVH
    57  04'8F   9A  07C8          MOVZBL  #MOVH_INS_SZ,R7             ; Remember size of MOVH instruction
                    07CC  MOVH_INS:
     50   00  70FD  07CC          MOVH    S^#00,R0                    ; Execute an H floating point instruction
          00000004  07D0  MOVH_INS_SZ = . - MOVH_INS
                    07D0
          29    11  07D0          BRB     47$                         ; All done, go restore OPCDEC handler
                    07D2
                    07D2  ;
                    07D2  ; Temporary OPCDEC and FPD emulation exception handlers
                    07D2  ;
                    07D2          .ALIGN  LONG
00 0000'CF  56  E2  07D4  45$:    BBSS    R6,W^EXE$GL_ARCHFLAG,145$   ; Indicate data type must be emulated
        6E  57  C0  07DA  145$:   ADDL    R7,(SP)                     ; Point past instruction causing fault
     04 AE  10  CA  07DD          BICL    #PSL$M_TBIT,4(SP)           ; Clear T-bit for use with XDELTA
                02  07E1          REI
                    07E2
                    07E2  ;
                    07E2  ; Temporary emulation exception handler
                    07E2  ;
                    07E2          .ALIGN  LONG
00 0000'CF  56  E2  07E4  46$:    BBSS    R6,W^EXE$GL_ARCHFLAG,146$   ; Indicate data type must be emulated
        5E  28  C0  07EA  146$:   ADDL    #40,SP                      ; Clean off emulation parameters
     04 AE  10  CA  07ED          BICL    #PSL$M_TBIT,4(SP)           ; Clear T-bit for use with XDELTA
                02  07F1          REI
                    07F2
                    07F2  ;
                    07F2  ; The following data areas are used to execute sample instructions
```

```
                              07F2  ; from all the classes that may have to be emulated via software.
                              07F2  ;
                        41    07F2  STRING: .ASCII  "A"
                              07F3  NUMERATOR:
                        3C    07F3          .PACKED 3                          ; Simple dividend
                              07F4  DENOMINATOR:
                        3C    07F4          .PACKED 3                          ; Nonzero divisor
                              07F5  QUOTIENT:
                  000007F9    07F5          .BLKL   1                          ; This cell must be writable
                              07F9
                              07F9  PATTERN:
                              07F9          EO$MOVE 1
                              07FA          EO$END
                              07FB
                              07FB  ;
                              07FB  ; Now check to see what software emulation images must be loaded.
                              07FB  ;
                              07FB  47$:                                       ;
         000000F0 8F    D3    07FB          BITL    #<ARC$M_CHAR_EMUL!ARC$M_DCML_EMUL!ARC$M_CRC_EMUL!ARC$M_EDPC_EMUL>,-
             0000'CF          0801                  W^EXE$GL_ARCHFLAG          ; Any char/dcml/editpc/crc emul needed?
                  04    12    0804          BNEQ    48$                        ; Br if emulation needed
             2504'CF    D4    0806          CLRL    W^VAXEMUL_STAT+STAT_L_NAME ; Set for no lookup of image
                              080A  48$:
         00000F00 8F    D3    080A          BITL    #<ARC$M_DFLT_EMUL!ARC$M_FFLT_EMUL!ARC$M_GFLT_EMUL!ARC$M_HFLT_EMUL>,-
             0000'CF          0810                  W^EXE$GL_ARCHFLAG          ; Any floating emulation needed?
                  04    12    0813          BNEQ    49$                        ; Br if emulation needed
             251C'CF    D4    0815          CLRL    W^FPEMUL_STAT+STAT_L_NAME  ; Set for no lookup of image
                              0819  49$:
             00FF 8F    BA    0819          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7> ; Restore volatile registers
             00CC C8  8ED0    081D          POPL    ^XCC(R8)                   ; Restore FPD emualtion vector contents
             00C8 C8  8ED0    0822          POPL    ^XC8(R8)                   ; Restore emulation vector contents
               10 A8  8ED0    0827          POPL    ^X10(R8)                   ; Restore OPCDEC vector contents
                              082B          .DSABL  LSB
000021C6'EF  0000'CF    B0    082B          MOVW    W^TTY$GW_CLASSNAM,TTNAME+1 ; Set ttdriver 2 letter prefix
                              0834  ;
                              0834  ; Look up the various images that need to be loaded. If there is a name,
                              0834  ; then open the image, derive and store the VBN of the first VBN past
                              0834  ; the image header and size (bytes) of the image excluding the image header.
                              0834  ;
          52  2438'CF   9E    0834          MOVAB   W^DSKDRV_STAT,R2           ; Get adr of first stat block
          55      0B'   D0    0839          MOVL    S^#LOAD_IMAGE_CNT,R5       ; Number of them to do
               0C A2    D5    083C  50$:    TSTL    STAT_L_NAME(R2)            ; Is there a name?
                  08    13    083F          BEQL    60$                        ; No, do not do anything
                0DC1    30    0841          BSBW    IMAGE_INIT                 ; Open image, read 1st block beyond hdr
               08 A6    3C    0844          MOVZWL  DPT$W_SIZE(R6),-
               04 A2          0847                  STAT_L_BYTECNT(R2)         ; Number of bytes in load image.
            52    18    C0    0849  60$:    ADDL    #STAT_C_SIZE,R2            ; Step to next driver
            ED    55    F5    084C          SOBGTR  R5,50$                     ; Do them all
                              084F  ;
                              084F  ; All lookups are now complete, truncate the FIL$OPENFILE cache
                              084F  ;
                              084F  LOOKUP_ALL_DONE:
     00000000'EF  00    FB    084F          CALLS   #0,FIL$CACHE_TRUNC         ; Truncate the FIL$OPENFILE cache
          50  2165'CF   7D    0856          MOVQ    W^FIL$GQ_CACHE,R0          ; Get size and address
             50  7140   9E    085B          MOVAB   -(R1)[R0],R0               ; Form last byte inclusive
23D0'CF   50   F7 8F    78    085F          ASHL    #-9,R0,W^CACHE_HI_PFN      ; Save last PFN in FIL$OPENFILE cache
                              0866  HAVEPRM:                                   ; All parameter values have been acquired
```

```
                          0866        ;
                          0866        ; Acquire some useful information about the hardware/WCS we are running on.
                          0866        ;
0004'CF  23AC'CF     7D   0866                MOVQ    W^CPUDATA,W^EXE$GB_CPUDATA+4     ; First longword contains SID.
                          086D        ;
                          086D        ;       Use only specified number of physical pages to permit the testing
                          086D        ;       of small memory configurations without actually reconfiguring hardware.
                          086D        ;
                          086D                .ENABL  LSB
      50    44 AB    7D   086D                MOVQ    RPB$Q_PFNMAP(R11),R0     ; Get descriptor for PFN bitmap
            50  08   C4   0871                MULL    #8,R0                    ; Scale byte count to bit count
0000'CF     50  01   C3   0874                SUBL3   #1,R0,W^MMG$GL_MAXMEM    ; Assume all of memory is useable
     52   0000'CF    D0   087A                MOVL    W^MMG$GL_PHYPGCNT,R2     ; Get count of pages to use
                53   7C   087F                CLRQ    R3                       ; Initialize bit pointer
                          0881                                                 ; R4 = 0, have not discarded
                          0881                                                 ; an otherwise usable PFN yet
      38 61     53   E1   0881        10$:    BBC     R3,(R1),20$              ; Skip over empty pages
                52   D7   0885                DECL    R2                       ; Count a present page
                34   18   0887                BGEQ    20$                      ; Continue if count not exhausted
      2C 54     00   E2   0889                BBSS    #0,R4,17$                ; Branch if not first discarded PFN
0000'CF     53  01   C3   088D                SUBL3   #1,R3,W^MMG$GL_MAXMEM    ; Save highest useable PFN
           23D4'CF   D5   0893                TSTL    W^PFNMAP_HI_PFN          ; Are we already safe
                0A   13   0897                BEQL    12$                      ; Yes
             0EC7    30   0899                BSBW    MOVE_BITMAP              ; Try to move the bitmap to a safer place
   23D4'CF      53   D1   089C                CMPL    R3,W^PFNMAP_HI_PFN       ; Discarding bitmap pages?
                23   15   08A1                BLEQ    PHYP_ER
   23D0'CF      53   D1   08A3        12$:    CMPL    R3,W^CACHE_HI_PFN        ; Discarding FIL$OPENFILE cache?
                08   14   08A8                BGTR    15$                      ; Branch if not
           2165'CF   7C   08AA                CLRQ    W^FIL$GQ_CACHE           ; Yes, disable it
           23D0'CF   D4   08AE                CLRL    W^CACHE_HI_PFN
   23EC'CF      53   D1   08B2        15$:    CMPL    R3,W^CI_HI_PFN           ; Bumped into the CI micro code?
                0A   15   08B7                BLEQ    21$                      ; Branch if so
      00 61     53   E5   08B9        17$:    BBCC    R3,(R1),20$              ; Remove all pages once count exhausted
      C0 53     50   F2   08BD        20$:    AOBLSS  R0,R3,10$                ; Scan entire bitmap
                41   11   08C1                BRB     SHELLSIZE
                          08C3
             0476    31   08C3        21$:    BRW     BUMP_CI                  ; Give error message
                          08C6        ;
                          08C6        PHYP_ER:
                          08C6                MSG     <-F-PFN bit map conflict - Physical page count set too low>
                     00   0903                HALT
                          0904                .DSABL  LSB
```

```
                                0904  ;
                                0904  ;       COMPUTE VALUES FOR SHELL PROCESS FROM SYSGEN PARAMETERS
                                0904  ;
                                0904  SHELLSIZE:                                                ;
     52  0000'CF    02   78     0904          ASHL    #2,W^SGN$GL_MAXVPGCT,R2  ; 4*VIRTUALPAGE COUNT=BYTES OF PGTBL
         52  01FF   C2   9E     090A          MOVAB   511(R2),R2               ; ROUND UP TO PAGE BOUNDARY
     52  52    F7   8F   78     090F          ASHL    #-9,R2,R2                ; DIVIDE BY 512 TO GET COUNT OF PAGES
         0000'CF    52   D0     0914          MOVL    R2,W^SGN$GL_PTPAGCNT     ; SAVE COUNT OF PAGE TABLES
                                0919  ;
                                0919  ;       COMPUTE SIZE OF WSL PART OF PHD
                                0919  ;
     53  0000'CF    04   C5     0919          MULL3   #WSL$C_LENGTH,W^SGN$GL_MAXWSCNT,R3; SIZE OF WSLIST
     51  0000'CF    04   C5     091F          MULL3   #WSL$C_LENGTH,W^PQL$GL_WSDEFAULT,R1; SIZE OF DEFAULT WSLIST
         51    53   51   C3     0925          SUBL3   R1,R3,R1                         ; FIND EMPTY BYTE COUNT
         53  037B   C3   9E     0929          MOVAB   <PHD$C_LENGTH+511>(R3),R3        ; ADD FIXED SIZE AND ROUNDING
     50  0000'CF    20   A5     092E          MULW3   #SEC$C_LENGTH,W^SGN$GW_MAXPSTCT,R0; GUARANTEED SECTION SPACE
               50   50   3C     0934          MOVZWL  R0,R0                    ; ZERO EXTEND TO LONGWORD
               53   50   C0     0937          ADDL    R0,R3                    ; ADD TO FIXED AND WSL
     53  53    F7   8F   78     093A          ASHL    #-9,R3,R3                ; GET PAGE COUNT
     51  51    F7   8F   78     093F          ASHL    #-9,R1,R1                ; AND EMPTY PAGE COUNT
   0000'CF     53   51   A3     0944          SUBW3   R1,R3,W^SWP$GW_WSLPTE    ; COUNT OF FIXED +WSL PAGES
         0000'CF    51   B0     094A          MOVW    R1,W^SWP$GW_EMPTPTE      ; COUNT OF EMPTY PTE
               50   53   D0     094F          MOVL    R3,R0                    ; COUNT OF ALL WSL+FIXED PAGES
         53    53   09   78     0952          ASHL    #9,R3,R3                 ; BACK TO BYTE COUNT
         53  01FF   C3   9E     0956          MOVAB   511(R3),R3               ; ADD PAGE ROUNDING FACTOR
               50   52   C0     095B          ADDL    R2,R0                    ; GET TOTAL HEADER PAGE COUNT
               50   08   C4     095E          MULL    #8,R0                    ; WSLX+BAK+LCK+VAL = 2+4+1+1
               53   50   C0     0961  10$:    ADDL    R0,R3                    ; HEADER BYTE COUNT
     50  50    F7   8F   78     0964          ASHL    #-9,R0,R0                ; CONVERT ADDED BYTES TO PAGES
               50   08   C4     0969          MULL    #8,R0                    ; ADDED PAGES BACK TO ADDED BYTES
                    F3   12     096C          BNEQ    10$                      ; REPEAT IF SIGNIFICANT
     53  53    F7   8F   78     096E          ASHL    #-9,R3,R3                ; GET TOTAL PAGE COUNT
         0000'CF    53   D0     0973          MOVL    R3,W^SGN$GL_PHDPAGCT     ; SAVE TOTAL
         50    53   09   78     0978          ASHL    #9,R3,R0                 ; GET BYTE COUNT FOR PHD
                    50   C3     097C          SUBL3   R0,-                     ;
                                097E                  #<^X80000000-<<<128*<SWP$C_DBGPTCNT+SWP$C_SHLP1PT>>-
                                097E                  -SWP$C_SHLFPTE>@9>>,-    ;
 0000'CF    80000000'8F         097E                  W^MMG$GL_CTLBASVA        ; SET CONTROL REGION BASE VA
   0000'CF    0000'CF    D0     0986          MOVL    W^MMG$GL_CTLBASVA,W^SWP$GL_PHDBASVA; AND SWAPPER VERSION
   0000'CF     53   51   C3     098D          SUBL3   R1,R3,W^SGN$GL_PHDAPCNT  ; TOTAL ALLOCATED PAGES FOR SHELL PHD
         0000'CF    53   B0     0993          MOVW    R3,W^SWP$GW_BAKPTE       ; SET AS TOTAL BAK PTE
   0000'CF    0000'CF    A2     0998          SUBW    W^SWP$GW_WSLPTE,W^SWP$GW_BAKPTE ;
         0000'CF    51   A2     099F          SUBW    R1,W^SWP$GW_BAKPTE       ; SUBTRACT WSL AND EMPTY  TO GET BAK
   0000'CF     53   07   78     09A4          ASHL    #7,R3,W^SGN$GL_PHDLWCNT  ; COUNT OF LONGWORDS IN PHD
         50    53   52   C1     09AA          ADDL3   R2,R3,R0                 ; TOTAL PAGES
   0000'CF     50   07   78     09AE          ASHL    #7,R0,W^SGN$GL_P1LWCNT   ; LW COUNT TO P1PT END
                    54   D4     09B4          CLRL    R4                       ; ASSUME NO EXTENTION PAGES
 50  0000'CF    0000007F'8F C1  09B6          ADDL3   #<127-SWP$C_SHLFPTE>,W^SGN$GL_PHDPAGCT,R0 ; COMPUTE TOTAL
                    05   15     09C0          BLEQ    15$                      ; BRANCH IF NO EXTENTION PAGES NEEDED
     54  50    F9   8F   78     09C2          ASHL    #-7,R0,R4                ; COUNT OF EXTENSION PAGES
               54   00'  C0     09C7  15$:    ADDL    S^#SWP$C_SHLP1PT,R4      ; TOTAL SHELL P1 PAGE TABLES
         0000'CF    54   D0     09CA          MOVL    R4,W^SWP$GB_SHLP1PT      ; SAVE RESULT
               54   00'  C0     09CF          ADDL    S^#SWP$C_KSTACK,R4       ; ADD STACK
         54  0000'CF     C0     09D2          ADDL    W^SGN$GL_PHDAPCNT,R4     ; ADD ACTUAL PHD PAGE COUNT
               54   01'  C0     09D7          ADDL    S^#SWP$C_NDYN+1,R4       ; VECTOR PAGE + DYNAMIC PAGES
         0000'CF    54   D0     09DA          MOVL    R4,W^SWP$GL_SHELLSIZ     ; SAVE SHELL SIZE TOTAL
                                09DF  ;
```

```
                                  09DF      ;       Check  maximum and default process working set values to ensure
                                  09DF      ;       that they are at least large enough to contain the process header
                                  09DF      ;       plus the specified fluid working set.
                                  09DF      ;
            7E    0000'CF    9A   09DF              MOVZBL  W^SWP$GB_SHLP1PT,-(SP)   ; GET COUNT OF MANDATORY PAGE TABLES
            50    0000'CF    3C   09E4              MOVZWL  W^SGN$GW_MINWSCNT,R0     ; GET MINIMUM FLUID WORKING SET
                  50    50   C0   09E9              ADDL    R0,R0                    ; DOUBLE FLUID REQUIREMENT
            50    0000'CF    C0   09EC              ADDL    W^SGN$GL_PHDPAGCT,R0     ; ADD MIN PROCESS HEADER SIZE
                  50    8E   C0   09F1              ADDL    (SP)+,R0                 ; ADD COUNT OF NECESSARY PAGE TABLES
                  50   01'   C0   09F4              ADDL    S^#SWP$C_KSTACK+1,R0     ; ADD KERNEL STACK AND VECTOR PAGE
          0000'CF       50   D1   09F7              CMPL    R0,W^SGN$GL_MAXWSCNT     ; MUST BE LESS THAN THE TOTAL WORKING SET
                  30    15        09FC              BLEQ    20$                      ; CONTINUE IF SO
          0000'CF       50   D0   09FE              MOVL    R0,W^SGN$GL_MAXWSCNT     ; FORCE TO  AT LEAST MINIMUM VALUE
                                  0A03              MSG     <-W-Maximum WS raised to PHD+MINWSCNT>
                FE38    31        0A2B              BRW     HAVEPRM                  ; RECYCLE TO RECOMPUTE SIZES
      00000000'EF       50   D1   0A2E      20$:    CMPL    R0,PQL$GDWSDEFAULT       ; MUST ALSO HAVE LARGE ENOUGH DEFAULT
                  49    15        0A35              BLEQ    30$                      ; CONTINUE IF SO
          0000'CF       50   D0   0A37              MOVL    R0,W^PQL$GDWSDEFAULT     ; FORCE DEFAULT TO PROPER VALUE
          0000'CF       50   D0   0A3C              MOVL    R0,W^PQL$GMWSDEFAULT     ; ALSO MINIMUM DEFAULT
          0000'CF       50   D0   0A41              MOVL    R0,W^PQL$GDWSQUOTA       ; AND WS QUOTA
          0000'CF       50   D0   0A46              MOVL    R0,W^PQL$GMWSQUOTA       ; AND MINIMUM QUOTA
                                  0A4B              MSG     <-W-WS default and quota raised to PHD+MINWSCNT>
                FDE6    31        0A7D              BRW     HAVEPRM                  ; REPEAT CALCULATION
                                  0A80      30$:                                     ;
            50    0000'CF    D0   0A80              MOVL    W^SGN$GL_MAXWSCNT,R0     ; GET MAXIMUM SIZE OF WS
          0000'CF       50   D1   0A85              CMPL    R0,W^PQL$GDWSQUOTA       ; DEFAULT MUST BE LESS THAN MAX
                  08    18        0A8A              BGEQ    40$                      ; YES, CONTINUE
          0000'CF       50   D0   0A8C              MOVL    R0,W^PQL$GDWSQUOTA       ; LIMIT DEFAULT TO MAXIMUM
                FDD2    31        0A91              BRW     HAVEPRM                  ; RECYCLE THROUGH CALCULATIONS
          0000'CF       50   D1   0A94      40$:    CMPL    R0,W^PQL$GMWSQUOTA       ; SAME FOR MINIMUM DEFAULT WS
                  08    18        0A99              BGEQ    50$                      ; OK, CONTINUE
          0000'CF       50   D0   0A9B              MOVL    R0,W^PQL$GMWSQUOTA       ; LIMIT TO MAXIMUM
                FDC3    31        0AA0              BRW     HAVEPRM                  ; RECYCLE THROUGH CALCULATIONS
     0000'CF     0000'CF     D1   0AA3      50$:    CMPL    W^PQL$GDWSQUOTA,W^PQL$GDWSDEFAULT ; DEFAULT MUST BE LEQ QUOTA
                  0A    18        0AAA              BGEQ    60$                      ; YES, CONTINUE
     0000'CF     0000'CF     D0   0AAC              MOVL    W^PQL$GDWSQUOTA,W^PQL$GDWSDEFAULT; LIMIT DEFAULT TO QUOTA
                FDB0    31        0AB3              BRW     HAVEPRM                  ; RECYCLE THROUG CALCULATIONS
     0000'CF     0000'CF     D1   0AB6      60$:    CMPL    W^PQL$GMWSQUOTA,W^PQL$GMWSDEFAULT; SAME FOR MINIMUMS
                  0A    18        0ABD              BGEQ    70$                      ; OK, CONTINUE
     0000'CF     0000'CF     D0   0ABF              MOVL    W^PQL$GMWSQUOTA,W^PQL$GMWSDEFAULT; LIMIT TO QUOTA
                FD9D    31        0AC6              BRW     HAVEPRM                  ; RECYCLE THROUGH CALCULATIONS
                                  0AC9      70$:
  54   0000'CF   0000'CF     C1   0AC9              ADDL3   W^SGN$GL_PHDPAGCT,W^SGN$GL_PTPAGCNT,R4  ; GET TOTAL BALANCE SLOT PAG
          0000'CF       54   D0   0AD1              MOVL    R4,W^SWP$GL_BSLOTSZ      ; SAVE SIZE OF BALANCE SLOT IN PAGES
            54    0000'CF    C4   0AD6              MULL    W^SGN$GL_BALSETCT,R4     ; TOTAL PAGE REQ FOR BALANCE SLOTS
                                  0ADB      NPAGE_SPT:
 0000'CF   000001FF 8F       CA   0ADB              BICL    #^X1FF,W^SGN$GL_NPAGEDYN         ; ROUND DOWN TO PAGE BOUND
 0000'CF   000001FF 8F       CA   0AE4              BICL    #^X1FF,W^SGN$GL_NPAGEVIR         ; ROUND DOWN TO PAGE BOUND
     0000'CF     0000'CF     D1   0AED              CMPL    W^SGN$GL_NPAGEVIR,W^SGN$GL_NPAGEDYN ; MAXIMIZE WITH DYN VALUE
                  07    18        0AF4              BGEQ    10$                              ; BR IF VIR SUFFICIENT
     0000'CF     0000'CF     D0   0AF6              MOVL    W^SGN$GL_NPAGEDYN,W^SGN$GL_NPAGEVIR ; USE LARGER VALUE
 0000'CF   000001FF 8F       CA   0AFD      10$:    BICL    #^X1FF,W^SGN$GL_PAGEDYN ; ROUND DOWN TO PAGE BOUND
  50   0000'CF   0000'CF     C1   0B06              ADDL3   W^SGN$GL_PAGEDYN,W^SGN$GL_NPAGEVIR,R0 ; TOTAL POOL BYTES
       50    50   F7 8F      78   0B0E              ASHL    #-9,R0,R0                ; CONVERT TO PAGE COUNT
            54      50       C0   0B13              ADDL    R0,R4                    ; TOTAL DYNAMIC SPT SO FAR
                                  0B16      IRP_SPT:
     0000'CF     0000'CF     D1   0B16              CMPL    W^SGN$GL_IRPCNTV,W^SGN$GL_IRPCNT ; CHECK I/O PACKET COUNT
```

```
                 07   18  0B1D  1451        BGEQ    10$                               ; MAXIMIZE WITH IRPCNT
     0000'CF  0000'CF  D0  0B1F  1452        MOVL    W^SGN$GL_IRPCNT,W^SGN$GL_IRPCNTV ; USE COUNT
       000000D0 8F    C5  0B26  1453 10$:   MULL3   #<IRP$C_LENGTH+^XF>&<^C<^XF>>,-
         51  0000'CF      0B2C  1454                W^SGN$GL_IRPCNTV,R1               ; CALCULATE TOTAL BYTES
         51  01FF C1  9E  0B30  1455        MOVAB   511(R1),R1                        ; ROUND TO PAGE BOUND
   51    51    F7 8F  78  0B35  1456        ASHL    #-9,R1,R1                         ; CONVERT TO PAGE COUNT
              54   51  C0  0B3A  1457        ADDL    R1,R4                             ; ADD TO SPT TOTAL
```

```
                                 0B3D  ;
                                 0B3D  ; NOW ESTABLISH THE SPT REQUIRED FOR THE LRP LIST
                                 0B3D  ;
                                 0B3D  LRP_SPT:
          000000D0 8F     D1     0B3D          CMPL    #<<IRP$C_LENGTH+^XF>&<^C<^XF>>>,-
              0000'CF            0B43                  W^SGN$GL_LRPSIZE              ; IS PACKET SIZE TOO SMALL ?
                   06     15     0B46          BLEQ    30$                           ; IF LEQ, NO
                                 0B48
                                 0B48          ASSUME  IRP$C_LENGTH LE <^XF0>
                D0 8F     9A     0B48          MOVZBL  #<<IRP$C_LENGTH+^XF>&<^C<^XF>>>,-
              0000'CF            0B4B                  W^SGN$GL_LRPSIZE              ; FORCE SIZE TO MINIMUM
          0000005B 8F     C1     0B4E  30$:    ADDL3   #CXB$C_OVERHEAD+^XF,-         ; COMPUTE NEW LRP SIZE AND PREPARE
           51 0000'CF            0B54                  W^SGN$GL_LRPSIZE,R1           ; FOR ROUNDING
   00000000'EF  51  0F    CB     0B58          BICL3   #^XF,R1,L^BOO$GL_LRPSIZE; ROUND SIZE TO MULTIPLE OF 16
     0000'CF   0000'CF    D1     0B60          CMPL    W^SGN$GL_LRPCNTV,W^SGN$GL_LRPCNT ; CHECK I/O PACKET COUNT
                   07     18     0B67          BGEQ    40$                           ; MAXIMIZE WITH LRPCNT
     0000'CF   0000'CF    D0     0B69          MOVL    W^SGN$GL_LRPCNT,W^SGN$GL_LRPCNTV ; USE COUNT
51   0000'CF  00000000'EF C5     0B70  40$:    MULL3   L^BOO$GL_LRPSIZE,W^SGN$GL_LRPCNTV,R1      ; CALCULATE TOTAL BYTES
           51  01FF C1    9E     0B7A          MOVAB   511(R1),R1                    ; ROUND TO PAGE BOUND
       51   51   F7 8F    78     0B7F          ASHL    #-9,R1,R1                     ; CONVERT TO PAGE COUNT
              54   51     C0     0B84          ADDL    R1,R4                         ; ADD TO SPT TOTAL
                                 0B87  ;
                                 0B87  ; ALLOCATE SPT FOR THE SMALL PACKET LIST
                                 0B87  ;
        0000'CF    0F     C0     0B87  SRP_SPR:ADDL    #^XF,W^SGN$GL_SRPSIZE         ; ROUND UP TO 16 BYTE BOUNDARY
        0000'CF    0F     CA     0B8C          BICL    #^XF,W^SGN$GL_SRPSIZE         ;
     0000'CF   0000'CF    D1     0B91          CMPL    W^SGN$GL_SRPCNTV,W^SGN$GL_SRPCNT ; CHECK I/O PACKET COUNT
                   07     18     0B98          BGEQ    40$                           ; MAXIMIZE WITH SRPCNT
     0000'CF   0000'CF    D0     0B9A          MOVL    W^SGN$GL_SRPCNT,W^SGN$GL_SRPCNTV ; USE COUNT
51   0000'CF   0000'CF    C5     0BA1  40$:    MULL3   W^SGN$GL_SRPSIZE,W^SGN$GL_SRPCNTV,R1      ; CALCULATE TOTAL BYTES
           51  01FF C1    9E     0BA9          MOVAB   511(R1),R1                    ; ROUND TO PAGE BOUND
       51   51   F7 8F    78     0BAE          ASHL    #-9,R1,R1                     ; CONVERT TO PAGE COUNT
              54   51     C0     0BB3          ADDL    R1,R4                         ; ADD TO SPT TOTAL
                                 0BB6  PFN_SPT:
51   23CC'CF   23C8'CF    C3     0BB6          SUBL3   W^MEM_LO_PFN,W^MEM_HI_PFN,R1 ; PAGES MAPPED BY PFN DATABASE
              23CE'CF     B5     0BBE          TSTW    W^MEM_HI_PFN+2                ; MORE THAN 32 MBYTES PRESENT?
                   09     12     0BC2          BNEQ    80$                           ; BRANCH IF YES
     51   00000000'8F     C4     0BC4          MULL    #PFN$C_WORD_LEN,R1            ; BYTES FOR PFN DATA BASE
                   07     11     0BCB          BRB     90$                           ; JOIN COMMON CODE
     51   00000000'8F     C4     0BCD  80$:    MULL    #PFN$C_LONG_LEN,R1            ; BYTES FOR PFN DATA BASE
           51  01FF C1    9E     0BD4  90$:    MOVAB   511(R1),R1                    ; ROUND TO PAGE BOUND
       51   51   F7 8F    78     0BD9          ASHL    #-9,R1,R1                     ; CONVERT TO PAGE COUNT
              54   51     C0     0BDE          ADDL    R1,R4                         ; ADD TO TOTAL SPT SIZE
           54  0000'CF    C0     0BE1          ADDL    W^SGN$GL_SPTREQ,R4            ; ADD REQUESTED EXTRA PAGES
           54  0000'CF    C0     0BE6          ADDL    W^EXE$GL_RTIMESPT,R4          ; Add SPTs requested for use by
                                 0BEB                                                ; realtime processes using
                                 0BEB                                                ; connect to interrupt.
     54   00000000'8F     C0     0BEB          ADDL    #MMG$C_SPTSKEL,R4             ; ADD SIZE OF SPT SKELETON
           50  0000'CF    3C     0BF2          MOVZWL  W^SGN$GW_SYSDWSCT,R0          ; GET SIZE OF SYSTEM WORKING SET
              50   04     C4     0BF7          MULL    #WSL$C_LENGTH,R0              ; BYTE SIZE OF WORKING SET
           50  037B C0    9E     0BFA          MOVAB   <PHD$C_LENGTH+511>(R0),R0 ; SYSPHD WITH WORKING SET LIST
     51   0000'CF  20     A5     0BFF          MULW3   #SEC$C_LENGTH,W^SGN$GW_GBLSECNT,R1; BYTES FOR GLOBAL SECTIONS
              51   51     3C     0C05          MOVZWL  R1,R1                         ; ZERO EXTEND WORD
              50   51     C0     0C08          ADDL    R1,R0                         ; TOTAL SYSTEM HEADER BYTES
       50   50   F7 8F    78     0C0B          ASHL    #-9,R0,R0                     ; CONVERT TO PAGES
           21C1'CF   50   D0     0C10          MOVL    R0,W^BOO$GL_SYSPHDPG          ; SAVE SYSPHD PAGE COUNT
              54   50     C0     0C15          ADDL    R0,R4                         ; ADD TO SPT REQUIREMENT
```

```
          50  0000'CF   3C  0C18          MOVZWL  W^SGN$GW_ISPPGCT,R0       ; PAGES OF INTERRUPT STACK
                54  50  C0  0C1D          ADDL    R0,R4                     ; ADD TO SPT REQUIREMENT
   51   0000'CF     02  78  0C20          ASHL    #2,W^SGN$GL_MAXGPGCT,R1   ; REQUESTED NUMBER OF GLOBAL PAGES
                            0C26                                            ; TO SIZE OF GLOBAL PAGE TABLE
          51  01FF C1   9E  0C26          MOVAB   511(R1),R1                ; ROUND TO PAGE BOUNDARY
       51   51   F7 8F  78  0C2B          ASHL    #-9,R1,R1                 ; CONVERT TO PAGE COUNT
          21A9'CF   51  D0  0C30          MOVL    R1,W^BOO$GL_GPTPGCT       ; SAVE COUNT OF GLOBAL PAGE TABLES
                54  51  C0  0C35          ADDL    R1,R4                     ; ADD TO SPT TOTAL
                            0C38  SIZ_SCB:                                  ; COMPUTE # PAGES FOR SCB
              50    01  9A  0C38          MOVZBL  #1,R0                     ; ASSUME 1 PAGE SCB
                            0C3B                                            ;
                            0C3B          CPUDISP <<780,SIZ_SCB_780>,-      ; *DISPATCH ON CPU TYPE*
                            0C3B                   <750,SIZ_SCB_750>,-      ;
                            0C3B                   <730,SIZ_SCB_730>,-
                            0C3B                   <790,SIZ_SCB_790>,-
                            0C3B                   <UV1,SIZ_SCB_UV1>,-
                            0C3B                  >,-
                            0C3B                  ENVIRON=VMB
                            0C6E
                            0C6E  SIZ_SCB_790:                              ; 11/790:
              50    04  D0  0C6E          MOVL    #4,R0                     ; ALWAYS ALLOCATE MAXIMUM SCB SIZE
                    0E  11  0C71          BRB     SIZ_SCB_END
                            0C73
                            0C73  SIZ_SCB_730:
                            0C73  SIZ_SCB_UV1:
                    50  D6  0C73          INCL    R0                        ; SCB ALWAYS 2 PAGES (1 UNIBUS)
                    0A  11  0C75          BRB     SIZ_SCB_END
                            0C77                                            ;
                            0C77  SIZ_SCB_750:                              ; 11/750:
                    50  D6  0C77          INCL    R0                        ; ASSUME 2 PAGE SCB (1 UNIBUS)
            0099 CB     95  0C79          TSTB    RPB$B_CONFREG+IO750$C_SL_UB1(R11) ; SECOND UNIBUS CONFIGURED?
                    02  13  0C7D          BEQL    SIZ_SCB_END               ; BRANCH IF NOT
                    50  D6  0C7F          INCL    R0                        ; ELSE 3 PAGE SCB
                            0C81                                            ;
                            0C81  SIZ_SCB_780:                              ; (FOR 11/780, PAGE COUNT=1)
                            0C81                                            ;
                            0C81  SIZ_SCB_END:                              ; *END OF CPU-DEPENDENT CODE*
                            0C81                                            ;
          2384'CF   50  D0  0C81          MOVL    R0,W^SCBPAGCT             ; SAVE SCB SIZE IN PAGES
                54  50  C0  0C86          ADDL    R0,R4                     ; ADD SCB PAGES TO SPT TOTAL
   2388'CF      50  09  78  0C89          ASHL    #9,R0,W^SCBBYTCT          ; SAVE SCB SIZE IN BYTES
            54   7F A4  9E  0C8F          MOVAB   127(R4),R4                ; ROUND TO PAGE BOUND
       51   54   F9 8F  78  0C93          ASHL    #-7,R4,R1                 ; CONVERT TO PAGES
                54  51  C0  0C98          ADDL    R1,R4                     ; ADD PAGES OF SPT TO MAP
       54   54   F9 8F  78  0C9B          ASHL    #-7,R4,R4                 ; PAGE COUNT FOR SPT ALLOCATION
          21B1'CF   54  D0  0CA0          MOVL    R4,W^BOO$GL_SPTPAGCT      ; SAVE COUNT OF PAGES
                            0CA5  ALLOCSPT:                                 ; ALLOCATE SPACE FOR SYSPHD AND SPT
                            0CA5          .ENABLE LSB
   58   21C1'CF     54  C1  0CA5          ADDL3   R4,W^BOO$GL_SYSPHDPG,R8   ; TOTAL SIZE OF PHD+SPT
          58  2384'CF   C0  0CAB          ADDL    W^SCBPAGCT,R8             ;  PLUS SCB
            57    48 AB D0  0CB0          MOVL    RPB$Q_PFNMAP+4(R11),R7    ; GET ADDRESS FOR PFNMAP
          56  23CC'CF   D0  0CB4          MOVL    W^MEM_HI_PFN,R6           ; GET HIGHEST PFN+1 IN MEMORY
                    56  D7  0CB9  10$:    DECL    R6                        ; POINT TO NEXT/HIGHEST PAGE
                    27  14  0CBB          BGTR    20$                       ; CONTINUE IF PAGES AVAILABLE
                            0CBD  15$:    MSG     <-F-Unable to allocate SPT+PHD+SCB.>
                        00  0CE3          HALT                              ; *** FATAL ERROR ***
                            0CE4
```

```
             D1 67   56   E1  0CE4  20$:    BBC     R6,(R7),10$              ; FIND HIGHEST PAGE NUMBER
                21AD'CF   D5  0CE8          TSTL    W^BOO$GL_NEXTPFN         ; HAVE WE MARKED IT?
                     05   12  0CEC          BNEQ    25$                      ; YES
          21AD'CF    56   D0  0CEE          MOVL    R6,W^BOO$GL_NEXTPFN      ; SAVE HIGH WATER MARK
                     50   D4  0CF3  25$:    CLRL    R0                       ; INIT COUNT OF CONTIGUOUS PAGES
                     06   11  0CF5          BRB     35$                      ;
                     56   D7  0CF7  30$:    DECL    R6                       ; NEXT PAGE (LOWER)
             BC 67   56   E1  0CF9          BBC     R6,(R7),10$              ; SKIP IF NO FIT
                     50   D6  0CFD  35$:    INCL    R0                       ; BUMP COUNT OF PAGES
                58   50   D1  0CFF          CMPL    R0,R8                    ; CHECK FOR FIT
                     F3   19  0D02          BLSS    30$                      ; NOT YET
             4C AB   50   C2  0D04          SUBL    R0,RPB$L_PFNCNT(R11)     ; ACCOUNT FOR PAGES REMOVED
                51   56   D0  0D08          MOVL    R6,R1                    ; COPY LOWEST PAGE NUMBER
             02 67   51   E5  0D0B  40$:    BBCC    R1,(R7),50$              ; MARK PAGE ALLOCATED
                     51   D6  0D0F          INCL    R1                       ; NEXT PAGE
                F7   50   F5  0D11  50$:    SOBGTR  R0,40$                   ; ALLOCATE THEM ALL
                              0D14  ;
                              0D14  ;       R6 - PFN FOR START OF SCB
                              0D14  ;       R7 - BASE OF PFNMAP
                              0D14  ;       R8 - SIZE OF SYSPHD+SPT+SCB IN PAGES
                              0D14  ;       R4 - SPT SIZE IN PAGES
                              0D14  ;
          59   56    09   78  0D14          ASHL    #9,R6,R9                 ; COMPUTE PHYSICAL ADDRESS OF SCB
          2198'CF    59   D1  0D18          CMPL    R9,W^BOO$GL_FREEMEM      ; CHECK FOR OVERLAP WITH SYSBOOT
                     9E   19  0D1D          BLSS    15$                      ; CONTINUE IF OK
          23D4'CF    56   D1  0D1F          CMPL    R6,W^PFNMAP_HI_PFN       ; RAN INTO PFN BITMAP?
                     97   15  0D24          BLEQ    15$                      ; BRANCH IF YES
          23D0'CF    56   D1  0D26          CMPL    R6,W^CACHE_HI_PFN        ; RAN INTO FIL$OPENFILE CACHE?
                     08   14  0D2B          BGTR    55$                      ; BRANCH IF NOT
                2165'CF   7C  0D2D          CLRQ    W^FIL$GQ_CACHE           ; YES, DISABLE IT
                23D0'CF   D4  0D31          CLRL    W^CACHE_HI_PFN
          23EC'CF    56   D1  0D35  55$:    CMPL    R6,W^CI_HI_PFN           ; RAN INTO CI MICROCODE?
                     2B   14  0D3A          BGTR    57$                      ; BRANCH IF NOT
                              0D3C  BUMP_CI:MSG     <-F-PFN allocation overwrites CI ucode.>
                     00       0D66          HALT
                              0D67
                50   59   D0  0D67  57$:    MOVL    R9,R0                    ; GET A WORKING COPY OF ADDRESS
          51   58    05   78  0D6A          ASHL    #5,R8,R1                 ; GET COUNT OF DOUBLE QUADWORDS
                     80   7C  0D6E  60$:    CLRQ    (R0)+                    ; CLEAR A
                     80   7C  0D70          CLRQ    (R0)+                    ;  DOUBLE QUADWORD
                F9   51   F5  0D72          SOBGTR  R1,60$                   ; CLEAR THEM ALL
                              0D75          .DISABLE LSB
                              0D75  INIT_SCB:                                ; INITIALIZE SYSTEM SCB
          238C'CF    59   D0  0D75          MOVL    R9,W^SCBPHADDR           ; SAVE PHYSICAL ADDR OF SCB
   69    0000'CF  0200 8F 28  0D7A          MOVC3   #512,W^SCB$AL_BASE,(R9)  ; COPY ARCHITECTURAL PART
                              0D82                                           ;  OF SCB (1 PG) FROM TEMPLATE
                              0D82                                           ;  TO SYSTEM COPY. (RETURNS
                              0D82                                           ;  SCB ADDR+512 IN R3.)
50   2388'CF   00000200 8F   C3  0D82          SUBL3   #512,W^SCBBYTCT,R0       ; GET # BYTES REMAINING IN SCB
                     0F   15  0D8C          BLEQ    20$                      ; BRANCH IF NONE
        50   50   FE 8F   78  0D8E          ASHL    #-2,R0,R0                ; CONVERT # BYTES TO LONGWDS
     83   00000001'8F     D0  0D93  10$:    MOVL    #ERL$UNEXP+1,(R3)+       ; SET NEXT VECTOR TO UNEXPECTED INT
                F6   50   F5  0D9A          SOBGTR  R0,10$                   ; BRANCH IF MORE VECTORS
                              0D9D  ;
                              0D9D  ; CPU specific code for 11/780.  Point SCB vector 0 interrupts and NEXUS
                              0D9D  ; vector 0 interrupts to separate routine in module ERRORLOG.
                              0D9D  ;
```

```
                              0D9D
                              0D9D  20$:    CPUDISP <<780,SCB_VEC_780>,-    ; Dispatch on CPU type
                              0D9D                   <750,SCB_VEC_750>,-
                              0D9D                   <730,SCB_VEC_730>,-
                              0D9D                   <790,SCB_VEC_790>,-
                              0D9D                   <UV1,SCB_VEC_UV1>,-
                              0D9D                  >,-
                              0D9D                  ENVIRON=VMB
                              0DD0
                              0DD0  SCB_VEC_790:
             51   04      D0  0DD0          MOVL    #4,R1                   ; 4 pages of SCB vectors
                  0A      11  0DD3          BRB     FILL_NEXUS0             ; Set address in nexus 0 vectors
                              0DD5
                              0DD5  SCB_VEC_780:
             51   01      D0  0DD5          MOVL    #1,R1                   ; 1 page of SCB vectors
     69   00000000'9F     DE  0DD8          MOVAL   @#ERL$VEC_RETURN,(R9)   ; Dispatches through SCB location 0
                              0DDF                                          ; should be harmless on a 780.
                              0DDF  FILL_NEXUS0:
             50   59      D0  0DDF          MOVL    R9,R0                   ; Copy pointer to SCB
0100 C0   00000001'9F     DE  0DE2  10$:    MOVAL   @#ERL$VEC_RETURN+1,^X100(R0) ; Unibus passive releases can cause
 0140 C0    0100 C0       D0  0DEB          MOVL    ^X100(R0),^X140(R0)     ; interrupts for "TR #0"; make them
 0180 C0    0100 C0       D0  0DF2          MOVL    ^X100(R0),^X180(R0)     ; harmless (at all 4 IPLs).
 01C0 C0    0100 C0       D0  0DF9          MOVL    ^X100(R0),^X1C0(R0)
     50   00000200 8F     C0  0E00          ADDL    #^X200,R0               ; Step to next page of SCB
               D8 51      F5  0E07          SOBGTR  R1,10$                  ; Fill in vectors in next SCB page
                              0E0A
                              0E0A  SCB_VEC_750:
                              0E0A  SCB_VEC_730:                            ; Do nothing for 11/750 and 11/730
                              0E0A  SCB_VEC_UV1:
                              0E0A
                              0E0A  ; End of CPU dependent code
                              0E0A
                              0E0A  INIT_SYSPHD:                            ; INITIALIZE SYSTEM PHD
         59   2388'CF     C0  0E0A          ADDL    W^SCBBYTCT,R9           ; STEP PAST SCB TO PHD
  69   0000'CF   017C 8F  28  0E0F          MOVC3   #PHD$C_LENGTH,W^BOO$A_SYSPHD,(R9)       ; COPY PHD TO REAL PLACE
         50   0000'CF     B0  0E17          MOVW    W^SGN$GW_SYSDWSCT,R0    ; GET SIZE OF SYSTEM WORKING SET
           50 A9   50     B0  0E1C          MOVW    R0,PHD$W_WSSIZE(R9)     ; SET SYSTEM WORKING SET SIZE
           50   10 A9     A0  0E20          ADDW    PHD$W_WSNEXT(R9),R0     ; GET POINTER TO WORKING SET
           12 A9   50     B0  0E24          MOVW    R0,PHD$W_WSLAST(R9)     ; SET END OF SYSTEM WORKING SET LIST
           18 A9   50     B0  0E28          MOVW    R0,PHD$W_WSQUOTA(R9)    ; SET QUOTA VALUES
           0A A9   50     B0  0E2C          MOVW    R0,PHD$W_WSAUTH(R9)     ; FOR CONSISTENCY
           1A A9   50     B0  0E30          MOVW    R0,PHD$W_DFWSCNT(R9)    ; SAKE
           16 A9   50     B0  0E34          MOVW    R0,PHD$W_WSEXTENT(R9)   ; ...
           14 A9   50     B0  0E38          MOVW    R0,PHD$W_WSAUTHEXT(R9)  ; ...
        42 A9   0000'CF   B0  0E3C          MOVW    W^SGN$GL_BALSETCT,PHD$W_PHVINDEX(R9)    ; SET HEADER NUMBER
  1F A9   0000'CF   01    81  0E42          ADDB3   #1,W^SGN$GW_SWPFILES,PHD$B_PAGFIL(R9) ; SET SYSTEM PAGING FILE
00CC C9   21B1'CF   07    78  0E49          ASHL    #7,W^BOO$GL_SPTPAGCT,PHD$L_POLRASTL(R9) ; SET SPT LENGTH
   0000'CF   00CC C9      D0  0E51          MOVL    PHD$L_POLRASTL(R9),W^MMG$GL_SPTLEN      ; AND SAVE FOR REFERENCE
     51   21C1'CF   09    78  0E58          ASHL    #9,W^BOO$GL_SYSPHDPG,R1 ; BYTE OFFSET TO SPT
   0000'CF   59   51      C1  0E5E          ADDL3   R1,R9,W^MMG$GL_SBR      ; SAVE PHYSICAL ADDRESS OF SPT
  20 A9   21C1'CF   09    78  0E64          ASHL    #9,W^BOO$GL_SYSPHDPG,PHD$L_PSTBASOFF(R9); SAVE AS BYTE OFFSET
     50   00CC C9   09    78  0E6B          ASHL    #9,PHD$L_POLRASTL(R9),R0; COMPUTE MAX SYSTEM ADDRESS+1
          52   01   1F    78  0E71          ASHL    #VA$V_SYSTEM,#1,R2      ; SYSTEM MASK
       28 A9   50   52    C9  0E75          BISL3   R2,R0,PHD$L_FREPOVA(R9) ; SAVE AS MAX ADDRESS
   0000'CF   50   52      C9  0E7A          BISL3   R2,R0,W^MMG$GL_FRESVA   ;
   0000'CF   50   52      C9  0E80          BISL3   R2,R0,W^MMG$GL_MAXSYSVA ;
   0000'CF   50   52      C9  0E86          BISL3   R2,R0,W^MMG$GL_MAXGPTE  ;
```

```
          21BD'CF      59   D0  0E8C         MOVL    R9,W^BOO$GL_SYSPHD          ; SAVE PHYSICAL PHD ADDRESS
 51  00CC C9   21A9'CF      C3  0E91         SUBL3   W^BOO$GL_GPTPGCT,PHD$L_P0LRASTL(R9),R1   ; VPN OF GPT
          50      51   09   78  0E99         ASHL    #9,R1,R0                    ; CONVERT TO BYTE ADDRESS
     0000'CF      50   52   C9  0E9D         BISL3   R2,R0,W^MMG$GL_GPTE         ; BASE OF GLOBAL PAGE TABLE ENTRIES
          51   21B1'CF      C2  0EA3         SUBL    W^BOO$GL_SPTPAGCT,R1        ; VPN OF SPT
          50      51   09   78  0EA8         ASHL    #9,R1,R0                    ; BYTE ADDRESS
     0000'CF      50   52   C9  0EAC         BISL3   R2,R0,W^MMG$GL_GPTBASE      ; SAVE AS GPT BASE
     0000'CF      50   52   C9  0EB2         BISL3   R2,R0,W^MMG$GL_SPTBASE      ; AND SPT BASE
     00C8 C9      50   52   C9  0EB8         BISL3   R2,R0,PHD$L_P0BR(R9)        ; AND VIRTUAL SBR
          51   21C1'CF      C2  0EBE         SUBL    W^BOO$GL_SYSPHDPG,R1        ; VPN OF SYSPHD
          50      51   09   78  0EC3         ASHL    #9,R1,R0                    ; BYTE ADDRESS
     0000'CF      50   52   C9  0EC7         BISL3   R2,R0,W^MMG$GL_SYSPHD       ; SAVE VIRTUAL POINTER TO PHD
0000'CF    21C1'CF     09   78  0ECD         ASHL    #9,W^BOO$GL_SYSPHDPG,W^MMG$GL_SYSPHDLN  ; BYTES IN SYSTEM PHD
50   0000'CF    0000'CF     C5  0ED5         MULL3   W^SWP$GL_BSLOTSZ,W^SGN$GL_BALSETCT,R0    ; BALANCE SET MAP SIZE
          50      51   50   C3  0EDD         SUBL3   R0,R1,R0                    ; VPN OF BAL BASE
   0000'CF    0000'DF40     DE  0EE1         MOVAL   @W^MMG$GL_SPTBASE[R0],W^SWP$GL_BALSPT    ; SET BASE OF BALANCE MAP
          50      50   09   78  0EE9         ASHL    #9,R0,R0                    ; CONVERT VPN TO VA
     0000'CF      50   52   C9  0EED         BISL3   R2,R0,W^SWP$GL_BALBASE      ; SAVE VA BASE OF BALANCE SLOTS
          50   2388'CF      C2  0EF3         SUBL    W^SCBBYTCT,R0               ; COMPUTE VA OF SCB
 00000000'EF      50   52   C9  0EF8         BISL3   R2,R0,EXE$GL_SCB            ; SAVE SYSTEM SPACE VA OF SCB
          50    FE00 C0     9E  0F00         MOVAB   -512(R0),R0                 ; SKIP ONE PAGE FOR ERROR DETECTION
     0000'CF      50   52   C9  0F05         BISL3   R2,R0,W^EXE$GL_INTSTK       ; SET BASE OF INTERRUPT STACK
        53    0000'DF41     DE  0F0B         MOVAL   @W^MMG$GL_SBR[R1],R3        ; POINTER TO SPTE
     50   58   2384'CF      C3  0F11         SUBL3   W^SCBPAGCT,R8,R0            ; GET # PAGES IN PHD+SPT ONLY
          56   2384'CF      C0  0F17         ADDL    W^SCBPAGCT,R6               ; GET PFN OF START OF SYS PHD
 83   56   B0000000 8F      C9  0F1C 10$:    BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,R6,(R3)+    ; MAP A PAGE
                       56   D6  0F24         INCL    R6                          ; NEXT PFN
                  F3   50   F5  0F26         SOBGTR  R0,10$                      ; MAP ENTIRE SYSPHD+SPT
          50   21A9'CF      D0  0F29         MOVL    W^BOO$GL_GPTPGCT,R0         ; COUNT OF GLOBAL PAGE TABLE PAGES
     83    70000000 8F      D0  0F2E 20$:    MOVL    #PTE$C_URKW,(R3)+           ; FILL THEM AS DEMAND ZERO
                  F6   50   F5  0F35         SOBGTR  R0,20$                      ;
                                0F38 MAPSCB:                                     ; MAP SCB
               15      09   EF  0F38         EXTZV   #VA$V_VPN,#VA$S_VPN,-       ; GET VPN OF
          51   0000'CF          0F3B                 W^EXE$GL_SCB,R1             ; SCB
        53    0000'DF41     DE  0F3F         MOVAL   @W^MMG$GL_SBR[R1],R3        ; GET ADDR OF SPTE TO FILL
          51   2384'CF      D0  0F45         MOVL    W^SCBPAGCT,R1               ; GET # PAGES OF SCB
 50    238C'CF     F7 8F    78  0F4A         ASHL    #-9,W^SCBPHADDR,R0          ; GET PFN OF START OF SCB
       B0000000 8F          C9  0F51 10$:    BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,- ; MAP NEXT SPTE
               83      50       0F57                 R0,(R3)+
                       50   D6  0F59         INCL    R0                          ; STEP TO NEXT PFN
                  F3   51   F5  0F5B         SOBGTR  R1,10$                      ; MAP WHOLE SCB
                                0F5E MAPISTK:                                    ; MAP INTERRUPT STACK
 51   0000'CF     15   09   EF  0F5E         EXTZV   #VA$V_VPN,#VA$S_VPN,W^EXE$GL_INTSTK,R1   ; GET VPN FOR INT STK
        53    0000'DF41     DE  0F65         MOVAL   @W^MMG$GL_SBR[R1],R3        ; SVASPT + 4
          51   0000'CF      3C  0F6B         MOVZWL  W^SGN$GW_ISPPGCT,R1         ; GET PAGE COUNT FOR INTERRUPT STACK
                     077E   30  0F70 10$:    BSBW    ALLOCPFN                    ; ALLOCATE A PAGE
 73   50   B0000000 8F      C9  0F73         BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,R0,-(R3); FILL VALID MAP ENTRY
                  F2   51   F5  0F7B         SOBGTR  R1,10$                      ; MAP ENTIRE INTERRUPT STACK
          51   0000'CF      3C  0F7E         MOVZWL  W^SGN$GW_ISPPGCT,R1         ; GET SIZE OF INTERRUPT STACK
                       51   D6  0F83         INCL    R1                          ; SKIP ONE PAGE FOR ERROR DETECTION
                       73   D5  0F85         TSTL    -(R3)                       ; ALSO BUMP SPT POINTER DOWN
          51      51   09   78  0F87         ASHL    #9,R1,R1                    ; CONVERT TO BYTES
     51   0000'CF      51   C3  0F8B         SUBL3   R1,W^EXE$GL_INTSTK,R1       ; FORM ADDRESS OF BASE
                                0F91 ;
                                0F91 ; NOW ESTABLISH THE SIZE AND BASE ADDRESS OF THE SMALL REQUEST PACKET
                                0F91 ; LOOK ASIDE LIST
                                0F91 ;
```

```
                                 0F91  1744 ALLOC_SRP:
   00000000'EF   0000'CF    D0   0F91  1745         MOVL    W^SGN$GL_SRPCNT,L^BOO$GL_SRPCNT ; COUNT OF SRP'S TO INITIALIZE
 52   0000'CF    0000'CF    C5   0F9A  1746         MULL3   W^SGN$GL_SRPSIZE,W^SGN$GL_SRPCNT,R2 ; CALCULATE TOTAL BYTES
            52   01FF C2    9E   0FA2  1747         MOVAB   511(R2),R2              ; ROUND TO PAGE
       52   000001FF 8F     CA   0FA7  1748         BICL    #^X1FF,R2               ; BOUNDARY
 50   0000'CF    0000'CF    C5   0FAE  1749         MULL3   W^SGN$GL_SRPSIZE,W^SGN$GL_SRPCNTV,R0; CALCULATE VIRTUAL BYTES
            50   01FF C0    9E   0FB6  1750         MOVAB   511(R0),R0              ; AND ROUND TO PAGE BOUNDARY
       50   000001FF 8F     CA   0FBB  1751         BICL    #^X1FF,R0               ;
   00000000'EF   51   50    C3   0FC2  1752         SUBL3   R0,R1,-
                                 0FCA  1753                 L^BOO$GL_SRPSPLIT       ; LOOKASIDE LIST SPLIT ADDRESS
      00000000'EF   52      C1   0FCA  1754         ADDL3   R2,L^BOO$GL_SRPSPLIT,-
                 0000'CF         0FD1  1755                 W^MMG$GL_SRPNEXT        ; NEXT PAGE TO ALLOCATE FOR SRP AREA
                 50   52    C2   0FD4  1756         SUBL    R2,R0                   ; COMPUTE SIZE OF GAP
      50   50    F9 8F      78   0FD7  1757         ASHL    #-7,R0,R0               ; CONVERT TO 4*PAGES
                 53   50    C2   0FDC  1758         SUBL    R0,R3                   ; SKIP SPT ENTRIES FOR GAP
      51   52    F7 8F      78   0FDF  1759         ASHL    #-9,R2,R1               ; COMPUTE PAGE COUNT TO ALLOCATE
                      0E    13   0FE4  1760         BEQL    20$                     ; BR IF NONE
                    0708    30   0FE6  1761 10$:    BSBW    ALLOCPFN                ; ALLOCATE A PAGE
 73   50   B0000000 8F      C9   0FE9  1762         BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,R0,-(R3); FILL VALID MAP ENTRY
                   F2 51    F5   0FF1  1763         SOBGTR  R1,10$                  ; MAP ACTIVE SRP AREA
                                 0FF4  1764 20$:
                                 0FF4  1765 ALLOC_IRP:
          52   0000'CF      D0   0FF4  1766         MOVL    W^SGN$GL_IRPCNT,R2      ; USE COUNT
      00000000'EF   52      D0   0FF9  1767         MOVL    R2,L^BOO$GL_IRPCNT      ; COUNT OF IRP'S TO INITIALIZE
      52   000000D0 8F      C4   1000  1768         MULL    #<IRP$C_LENGTH+^XF>&<^C<^XF>>,R2 ; CALCULATE TOTAL BYTES
            52   01FF C2    9E   1007  1769         MOVAB   511(R2),R2              ; ROUND TO PAGE BOUNDARY
      52   000001FF 8F      CA   100C  1770         BICL    #^X1FF,R2               ; SIZE OF ALLOCATED IRP AREA
           000000D0 8F      C5   1013  1771         MULL3   #<IRP$C_LENGTH+^XF>&<^C<^XF>>,-
            50   0000'CF         1019  1772                 W^SGN$GL_IRPCNTV,R0     ; CALCULATE TOTAL BYTES
            50   01FF C0    9E   101D  1773         MOVAB   511(R0),R0              ; ROUND TO PAGE BOUNDARY
      50   000001FF 8F      CA   1022  1774         BICL    #^X1FF,R0               ; VIRTUAL SIZE OF IRP AREA
                                 1029  1775
 51   00000000'EF   50      C3   1029  1776         SUBL3   R0,L^BOO$GL_SRPSPLIT,R1 ; BASE ADDRESS OF IRP AREA
      00000000'EF   51      D0   1031  1777         MOVL    R1,L^BOO$GL_SPLITADR    ; LOOKASIDE LIST SPLIT ADDRESS
    0000'CF   52   51       C1   1038  1778         ADDL3   R1,R2,W^MMG$GL_IRPNEXT  ; SET ADDRESS FOR NEXT ALLOCATION
                 50   52    C2   103E  1779         SUBL    R2,R0                   ; COMPUTE SIZE OF GAP
      50   50    F9 8F      78   1041  1780         ASHL    #-7,R0,R0               ; CONVERT TO PAGES*4
                 53   50    C2   1046  1781         SUBL    R0,R3                   ; SKIP SPT ENTRIES FOR GAP
      51   52    F7 8F      78   1049  1782         ASHL    #-9,R2,R1               ; CONVERT ACTIVE ARE TO PAGE COUNT
                      0E    13   104E  1783         BEQL    20$                     ; BR IF NONE
                    069E    30   1050  1784 10$:    BSBW    ALLOCPFN                ; ALLOCATE A PAGE
 73   50   B0000000 8F      C9   1053  1785         BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,R0,-(R3); FILL VALID MAP ENTRY
                   F2 51    F5   105B  1786         SOBGTR  R1,10$                  ; MAP ACTIVE IRP AREA
                                 105E  1787 20$:                                    ;
                                 105E  1788 ;
                                 105E  1789 ; NOW ESTABLISH THE SIZE AND BASE ADDRESS OF THE LARGE REQUEST PACKET
                                 105E  1790 ; LOOK ASIDE LIST
                                 105E  1791 ;
                                 105E  1792 ALLOC_LRP:
00000000'EF   0000'CF   0F  CB   105E  1793         BICL3   #^XF,W^SGN$GL_LRPMIN,L^BOO$GL_LRPMIN ; ROUND THE MINIMUM SIZE
   00000000'EF   0000'CF    D0   1068  1794         MOVL    W^SGN$GL_LRPCNT,L^BOO$GL_LRPCNT ; COUNT OF LRP'S TO INITIALIZE
 52   0000'CF   00000000'EF C5   1071  1795         MULL3   L^BOO$GL_LRPSIZE,W^SGN$GL_LRPCNT,R2 ; CALCULATE TOTAL BYTES
            52   01FF C2    9E   107B  1796         MOVAB   511(R2),R2              ; ROUND TO PAGE
       52   000001FF 8F     CA   1080  1797         BICL    #^X1FF,R2               ; BOUNDARY
 50   0000'CF   00000000'EF C5   1087  1798         MULL3   L^BOO$GL_LRPSIZE,W^SGN$GL_LRPCNTV,R0; CALCULATE VIRTUAL BYTES
            50   01FF C0    9E   1091  1799         MOVAB   511(R0),R0              ; AND ROUND TO PAGE BOUNDARY
       50   000001FF 8F     CA   1096  1800         BICL    #^X1FF,R0               ;
```

```
      00000000'EF   50   C3  109D          SUBL3   R0,L^BOO$GL_SPLITADR,-
            00000000'EF      10A4                  L^BOO$GL_LRPSPLIT           ; LOOKASIDE LIST SPLIT ADDRESS
      00000000'EF   52   C1  10A9          ADDL3   R2,L^BOO$GL_LRPSPLIT,-
                0000'CF      10B0                  W^MMG$GL_LRPNEXT            ; NEXT PAGE TO ALLOCATE FOR LRP AREA
                 50   52 C2  10B3          SUBL    R2,R0                       ; COMPUTE SIZE OF GAP
        50    50   F9 8F 78  10B6          ASHL    #-7,R0,R0                   ; CONVERT TO 4*PAGES
                 53   50 C2  10BB          SUBL    R0,R3                       ; SKIP SPT ENTRIES FOR GAP
        51    52   F7 8F 78  10BE          ASHL    #-9,R2,R1                   ; COMPUTE PAGE COUNT TO ALLOCATE
                      0E 13  10C3          BEQL    20$                         ; BR IF NONE
                    0629 30  10C5 10$:     BSBW    ALLOCPFN                    ; ALLOCATE A PAGE
   73    50  B0000000 8F C9  10C8          BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,R0,-(R3); FILL VALID MAP ENTRY
                   F2 51 F5  10D0          SOBGTR  R1,10$                      ; MAP ACTIVE LRP AREA
                             10D3 20$:                                         ;
                             10D3 MAPNPAGDYN:                                  ; MAP NON-PAGED DYNAMIC POOL
        51   00000000'EF D0  10D3          MOVL    L^BOO$GL_LRPSPLIT,R1        ; GET ADDRESS OF PREVIOUS AREA
   50   0000'CF   0000'CF C3  10DA          SUBL3   W^SGN$GL_NPAGEVIR,W^SGN$GL_NPAGEDYN,R0 ; GET NEG SIZE OF GAP
            51    0000'CF C2  10E2          SUBL    W^SGN$GL_NPAGEVIR,R1        ; AND COMPUTE BASE OF POOL
            0000'CF   51  D0  10E7          MOVL    R1,W^MMG$GL_NPAGEDYN        ; SAVE VA OF BASE
 0000'CF    0000'CF   51  C1  10EC          ADDL3   R1,W^SGN$GL_NPAGEDYN,W^MMG$GL_NPAGNEXT  ; SET VA OF NEXT PAGE
 0000'CF    51    0000'CF C3  10F4          SUBL3   W^SGN$GL_PAGEDYN,R1,W^MMG$GL_PAGEDYN    ; COMPUTE BASE OF PAGED POOL
        50    50   F7 8F 78  10FC          ASHL    #-9,R0,R0                   ; CONVERT GAP TO PAGE COUNT
               53   6340 DE  1101          MOVAL   (R3)[R0],R3                 ; ADVANCE SPT POINTER PAST GAP
            51    0000'CF D0  1105          MOVL    W^SGN$GL_NPAGEDYN,R1        ; GET BYTES OF POOL
        51    51   F7 8F 78  110A          ASHL    #-9,R1,R1                   ; AND CONVERT TO PAGE COUNT
                    05DF 30  110F 10$:     BSBW    ALLOCPFN                    ; ALLOCATE A PAGE
   73    50  B0000000 8F C9  1112          BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,R0,-(R3); FILL A VALID MAP ENTR
                   F2 51 F5  111A          SOBGTR  R1,10$                      ; MAP ALL POOL PAGES
                             111D ;
                             111D ; PREALLOCATE THE TOP OF POOL AND PLACE THE FIL$OPENFILE CACHE UP THERE
                             111D ; START BY ESTBALISHING THE SIZE AND BASE ADDRESS OF THE IRP LOOKASIDE LIST
                             111D ;
        50   00000000'EF DE  111D          MOVAL   L^BOO$GL_NPAGEDYN,R0        ; ADDRESS OF POOL SIZE WITH
            60    0000'CF D0  1124          MOVL    W^SGN$GL_NPAGEDYN,(R0)      ; INIT TO SIZE OF NON-PAGED POOL
                             1129                                              ; PREALLOCATED PIECES TAKEN OUT
                             1129 ;
                             1129 ; Now allocate pool for:
                             1129 ;        Boot Driver for System Disk
                             1129 ;         and any ucode needed for it
                             1129 ;        System Disk Driver
                             1129 ;        Port Driver (if any)
                             1129 ;        Terminal Driver - TTDRIVER.EXE
                             1129 ;        CPU dependent code - SYSLOA.EXE
                             1129 ;        SCS loadable code - SCSLOA.EXE
                             1129 ;
                             1129 ALLOC_DRIVERS:
            52    2420'CF DE  1129          MOVAL   W^BOODRV_STAT,R2            ; STAT BLOCK FOR BOOT DRIVER
               54     0C' D0  112E          MOVL    S^#ALLOC_POOL_CNT,R4        ; NO. OF DRIVERS TO ALLOCATE SPACE FOR
                    0C38 30  1131 10$:     BSBW    ALLOC_POOL
                  52   18 C0  1134          ADDL    #STAT_C_SIZE,R2             ; POINT TO NEXT STAT BLOCK
                   F7 54 F5  1137          SOBGTR  R4,10$                      ; ALLOCATE SPACE FOR NEXT DRIVER
                             113A ;
                             113A ; The following code calculates the amount of space needed to map the
                             113A ; FIL$OPENFILE cache.  The cache will be released sometime later by SYSINIT.
                             113A ;
                             113A ALLOC_FILCACHE:
            53    2165'CF D0  113A          MOVL    W^FIL$GQ_CACHE,R3           ; GET SIZE IN R3
                      27 13  113F          BEQL    80$                         ; BRANCH IF DISABLED
```

```
          53   0F   C0  1141          ADDL    #^XF,R3                    ; ROUND TO QUAD WORD
          53   0F   CA  1144          BICL    #^XF,R3                    ; FOR POOL ALLOCATION
                        1147     ;
                        1147     ; IF THE FIL$OPENFILE CACHE IS LARGE WITH RESPECT TO THE AMOUNT OF
                        1147     ; SPACE THAT IS LEFT IN THE POOL, THEN SIMPLY DISABLE IT.
                        1147     ;
          55   60   D0  1147          MOVL    (R0),R5                    ; SPACE IN POOL
          55   02   C6  114A          DIVL    #2,R5                      ; 50% OF THE SPACE LEFT
          55   53   D1  114D          CMPL    R3,R5                      ; CACHE TOO BIG?
               02   15  1150          BLEQ    70$                        ; BRANCH IF NOT
               53   D4  1152     50$: CLRL    R3                         ; DISABLE THE FILE CACHE
               53   D5  1154     70$: TSTL    R3                         ; IS THE CACHE ENABLED?
               10   13  1156          BEQL    80$                        ; BRANCH IF NOT
          60   53   C2  1158          SUBL    R3,(R0)                    ; ALLOCATE FIL$OPENFILE CACHE
 54    0000'CF 60   C1  115B          ADDL3   (R0),W^MMG$GL_NPAGEDYN,R4  ; ADDRESS OF FILE CACHE IN POOL
   00000000'EF 53   7D  1161          MOVQ    R3,L^BOO$GQ_FILCACHE       ; SET SIZE AND ADDRESS OF FILE CACHE
                        1168     80$:
                        1168     ;
                        1168     ;       There are two separate dependencies on the FIL$OPENFILE cache
                        1168     ; being allocated as the next higher address after INILOA.  First the
                        1168     ; initial pool fragmentation is eliminated by the orderly deallocation
                        1168     ; of these 2 pieces, INILOA by INIT and the FIL$OPENFILE cache by SYSINIT.
                        1168     ; Second, the code at MOVEFILECACHE below depends on being able to
                        1168     ; disable the FIL$OPENFILE cache by clearing the descriptor and adding
                        1168     ; its size to the length of INILOA for deallocation by the INIT code.
                        1168     ;
                        1168     ; At this point non-paged pool look like this:
                        1168     ;
                        1168     ;          +-----------------------------------------+
                        1168     ;          |                                         |  :MMG$GL_NPAGEDYN
                        1168     ;          |                                         |
                        1168     ;          |             Available Pool              |
                        1168     ;          |                                         |
                        1168     ;          |                                         |
                        1168     ;          +-----------------------------------------+
                        1168     ;          |                                         |
                        1168     ;          |           FIL$OPENFILE Cache            |  :BOO$GQ_CACHE
                        1168     ;          |                                         |   (deallocated by SYSINIT)
                        1168     ;          +-----------------------------------------+
                        1168     ;          |                                         |
                        1168     ;          |              MTACCESS.EXE               |  :BOO$GL_MTACCESSLOA
                        1168     ;          |               (if loaded)               |
                        1168     ;          |                                         |
                        1168     ;          +-----------------------------------------+
                        1168     ;          |                                         |
                        1168     ;          |              VAXEMUL.EXE                |  :BOO$GL_VAXEMUL
                        1168     ;          |                                         |
                        1168     ;          +-----------------------------------------+
                        1168     ;          |                                         |
                        1168     ;          |               FPEMUL.EXE                |  :BOO$GL_FPEMUL
                        1168     ;          |                                         |
                        1168     ;          +-----------------------------------------+
                        1168     ;          |                                         |
                        1168     ;          |             CHKPRTLOA.EXE               |  :BOO$GL_CHKPRTLOA
                        1168     ;          |              (if loaded)                |
                        1168     ;          |                                         |
                        1168     ;          +-----------------------------------------+
```

```
                 1168  ;            |                                       |
                 1168  ;            |            ERAPATLOA.EXE              |   :BOO$GL_ERAPATLOA
                 1168  ;            |            (if loaded)                |
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |            CLUSTRLOA.EXE              |   :BOO$GL_CLSLOA
                 1168  ;            |            (if loaded)                |
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |            SYSLOAxxx.EXE              |   :BOO$GL_SYSLOA
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |             SCSLOA.EXE                |   :BOO$GL_SCSLOA
                 1168  ;            |             (if loaded)               |
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |             TTDRIVER.EXE              |   :BOO$GL_TRMDRV
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |             PxDRIVER.EXE              |   :BOO$GL_PRTDRV
                 1168  ;            |             (if loaded)               |
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |          System Disk Driver           |   :BOO$GL_DSKDRV
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |               BOOTCB                  |   :BOO$GL_BOOTCB
                 1168  ;            |        Bootstrap Disk Driver          |   :RPB$L_IOVEC(RPB_ADR)
                 1168  ;            |           Microcode file              |   :BOO$GL_UCODE
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |        LRP Look-aside List            |   :BOO$GL_LRPSPLIT
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |        IRP Look-aside List            |   :BOO$GL_SPLITADR
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;            |                                       |
                 1168  ;            |        SRP Look-aside List            |   :BOO$GL_SRPSPLIT
                 1168  ;            |                                       |
                 1168  ;            +---------------------------------------+
                 1168  ;
                 1168  ;            MOVE SKELETON SPT TO ACTUAL LOCATION
                 1168  ;
0000'8F     28   1168               MOVC3   #<MMG$C_SPTSKEL@2>,-           ;
0000'CF          116C                       W^MMG$AL_SYSPAGTB,-            ;
0000'DF          116F                       @W^MMG$GL_SBR                  ;
                 1172       MAPRESEXEC:                                    ; MAP RESIDENT EXEC
```

```
      09 0000'CF     00'  E1  1172       BBC     S^#EXE$V_SYSPAGING,W^EXE$GL_DEFFLAGS,10$ ; BRANCH IF NOT PAGING
51    0000'CF   15   09   EF  1178       EXTZV   #VA$V_VPN,#VA$S_VPN,W^MMG$GL_PGDCOD,R1   ; GET START OF PAGED EXEC
                     05   11  117F       BRB     20$                             ; JOIN COMMON CODE
           51    0000'8F  3C  1181 10$:  MOVZWL  #<<MMG$AL_PGDCODEN-^X80000000>@-9>,R1    ; GET END OF PAGED EXEC
        53    0000'DF41   DE  1186 20$:  MOVAL   @W^MMG$GL_SBR[R1],R3            ; GET SVASPT
                   0562   30  118C 30$:  BSBW    ALLOCPFN                        ; ALLOCATE A PAGE
     73    001FFFFF  8F   CA  118F       BICL    #PTE$M_PFN,-(R3)                ; CLEAR PFN FIELD
                63   50   C8  1196       BISL    R0,(R3)                         ; ATTEMPT SIMPLE MERGE OF PAGE
                     08   19  1199       BLSS    40$                             ; SUCCESS
63   50    F0000000  8F   C9  119B       BISL3   #<PTE$C_URKW!PTE$M_VALID!PTE$C_KOWN>,R0,(R3); MAP A VALID PAGE
                E6   51   F5  11A3 40$:  SOBGTR  R1,30$                          ; MAP ALL RESIDENT EXEC
                              11A6 MAPPFNDAT:                                    ; MAP PFN DATA BASE
        00000000'EF       B4  11A6       CLRW    MMG$GW_BIGPFN                   ; ASSUME WORK LINKS IN PFN DATABASE
  0000'CF    23C8'CF      D0  11AC       MOVL    W^MEM_CO_PFN,W^MMG$GL_MINPFN    ; INITIALIZE SMALLEST PFN
             0000'CF      D4  11B3       clrl    w^mmg$gl_minpfn                 ; *************** TEMP ***************
   53    21AD'CF     02   C1  11B7       ADDL3   #2,W^BOO$GL_NEXTPFN,R3          ; START WITH CURRENT TOP PFN + FUDGE
         54    0000'CF    CE  11BD       MNEGL   W^MMG$GL_MINPFN,R4              ; R4 CONTAINS NEGATIVE OF BASE PFN NUMBER
                53   54   C0  11C2       ADDL2   R4,R3                           ; R3 NOW CONTAINS PFN COUNT
55   53    00000000'8F    C5  11C5       MULL3   #PFN$C_WORD_LEN,R3,R5           ; PFN DATA BASE SIZE (ASSUME SHORT FORM)
   55    55      F7  8F   78  11CD       ASHL    #-9,R5,R5                       ; PFN DATA BASE PAGES - 1
  55   21AD'CF       55   C3  11D2       SUBL3   R5,W^BOO$GL_NEXTPFN,R5          ; PAGES AFTER ALLOCATING PFN DATA BASE
   55    55      F0  8F   78  11D8       ASHL    #-16,R5,R5                      ; IS IT BIGGER THAN 65 K PAGES
                     06   13  11DD       BEQL    1$                              ; NO
        00000000'EF       B6  11DF       INCW    MMG$GW_BIGPFN                   ; CONDITION FOR IF MACROS
                              11E5 1$:   PFN_DISP_IF_BIGPFN_THEN         END_BIGPFN_CODE=3$
                              11ED
                              11ED       ;This code executes if the PFN link arrays are longword arrays.
     0000'CF     00'8F    90  11ED               MOVB    #PFN$C_LONG_LEN,W^PFN$GB_LENGTH
                              11F3       PFN_DISP_ELSE                   ELSE_CODE=3$,COMMON_CODE=6$
                              11F5
                              11F5       ;This code executes if the PFN link arrays are word arrays.
     0000'CF     00'8F    90  11F5               MOVB    #PFN$C_WORD_LEN,W^PFN$GB_LENGTH
                              11FB       PFN_DISP_ENDIF                  COMMON_CODE=6$
                              11FB
                              11FB       ;End of code that depends on size of PFN link arrays
          55    0000'CF   9A  11FB       MOVZBL  W^PFN$GB_LENGTH,R5              ; R5 CONTAINS NUMBER OF BYTES PER PAGE
          52    53   55   C5  1200       MULL3   R5,R3,R2                        ; COMPUTE BYTES OF PFN DATA BASE NEEDED
          52    01FF C2   9E  1204       MOVAB   511(R2),R2                      ; ROUND UP TO PAGE BOUND
   50    52      F7  8F   78  1209       ASHL    #-9,R2,R0                       ; AND CONVERT TO PAGES
                50   55   C4  120E 10$:  MULL    R5,R0                           ; GET BYTES OF PFN DATA
                52   50   C2  1211       SUBL    R0,R2                           ; REMOVE FROM REQUIREMENT
   50    50      F7  8F   78  1214       ASHL    #-9,R0,R0                       ; GET PAGES REMOVED
                     F3   12  1219       BNEQ    10$                             ; BR IF WHOLE PAGES REMOVED
   52    52      F7  8F   78  121B       ASHL    #-9,R2,R2                       ; CONVERT TO PAGE COUNT
                53   52   C2  1220       SUBL    R2,R3                           ; SUBTRACT PAGES OF PFN DATA
                53   55   C4  1223       MULL    R5,R3                           ; COMPUTE BYTE OF PFN DATA
          52    01FF C3   9E  1226       MOVAB   511(R3),R2                      ; ROUND TO NEXT PAGE
   51    52      F7  8F   78  122B       ASHL    #-9,R2,R1                       ; COMPUTE PAGES OF PFN DATA BASE
          52    51   09   78  1230       ASHL    #9,R1,R2                        ; BACK TO BYTES
          53    52   55   C7  1234       DIVL3   R5,R2,R3                        ; COMPUTE NUMBER OF ENTRIES
   52    0000'CF     52   C3  1238       SUBL3   R2,W^MMG$GL_PAGEDYN,R2          ; SUBTRACT FROM PAGEDYN TO GET PFN BASE
         0000'CF     52   D0  123E       MOVL    R2,W^PFN$AL_PTE                 ; SAVE AS BASE OF FIRST W^PFN ARRAY
0000'CF    0000'DF43      DE  1243       MOVAL   @W^PFN$AL_PTE[R3],W^PFN$AL_BAK  ; ADD SIZE * PAGE COUNT
0000'CF    0000'DF43      DE  124B       MOVAL   @W^PFN$AL_BAK[R3],W^PFN$AW_REFCNT       ; FOR EACH OF THE W^PFN ARRA
0000'CF    0000'DF43      3E  1253       MOVAW   @W^PFN$AW_REFCNT[R3],W^PFN$Ax_FLINK;
                              125B       PFN_DISP_IF_BIGPFN_THEN         END_BIGPFN_CODE=20$
                              1263
```

```
                              1263         ;This code executes if the PFN link arrays are longword arrays.
 0000'CF    0000'DF43    DE   1263                 MOVAL   @W^PFN$Ax_FLINK[R3],W^PFN$Ax_BLINK      ; TO GET ADDRESS OF
 0000'CF    0000'DF43    DE   126B                 MOVAL   @W^PFN$Ax_BLINK[R3],W^PFN$AW_SWPVBN; ARRAY BASE
                              1273         PFN_DISP_ELSE                       ELSE_CODE=20$,COMMON_CODE=25$
                              1275
                              1275         ;This code executes if the PFN link arrays are word arrays.
 0000'CF    0000'DF43    3E   1275                 MOVAW   @W^PFN$Ax_FLINK[R3],W^PFN$Ax_BLINK      ; TO GET ADDRESS OF
 0000'CF    0000'DF43    3E   127D                 MOVAW   @W^PFN$Ax_BLINK[R3],W^PFN$AW_SWPVBN; ARRAY BASE
                              1285         PFN_DISP_ENDIF                      COMMON_CODE=25$
                              1285
                              1285         ;End of code that depends on size of PFN link arrays
 0000'CF    0000'DF43    3E   1285         MOVAW   @W^PFN$AW_SWPVBN[R3],W^PFN$AB_STATE;
 0000'CF    0000'DF43    9E   128D         MOVAB   @W^PFN$AB_STATE[R3],W^PFN$AB_TYPE          ;
                              1295  ;
                              1295  ; RESET BASE OF EACH ARRAY TO ACCOUNT FOR NONZERO MINPFN.
                              1295  ; R4 CONTAINS THE NEGATIVE OF THE SMALLEST PFN IN THE PFN DATA BASE.
                              1295  ;
 0000'CF    0000'DF44    DE   1295         MOVAL   @W^PFN$AL_PTE[R4],W^PFN$AL_PTE
 0000'CF    0000'DF44    DE   129D         MOVAL   @W^PFN$AL_BAK[R4],W^PFN$AL_BAK
 0000'CF    0000'DF44    3E   12A5         MOVAW   @W^PFN$AW_REFCNT[R4],W^PFN$AW_REFCNT
                              12AD         PFN_DISP_IF_BIGPFN_THEN             END_BIGPFN_CODE=30$
                              12B5
                              12B5         ;This code executes if the PFN link arrays are longword arrays.
 0000'CF    0000'DF44    DE   12B5                 MOVAL   @W^PFN$Ax_FLINK[R4],W^PFN$Ax_FLINK
 0000'CF    0000'DF44    DE   12BD                 MOVAL   @W^PFN$Ax_BLINK[R4],W^PFN$Ax_BLINK
                              12C5         PFN_DISP_ELSE                       ELSE_CODE=30$,COMMON_CODE=35$
                              12C7
                              12C7         ;This code executes if the PFN link arrays are word arrays.
 0000'CF    0000'DF44    3E   12C7                 MOVAW   @W^PFN$Ax_FLINK[R4],W^PFN$Ax_FLINK
 0000'CF    0000'DF44    3E   12CF                 MOVAW   @W^PFN$Ax_BLINK[R4],W^PFN$Ax_BLINK
                              12D7         PFN_DISP_ENDIF                      COMMON_CODE=35$
                              12D7
                              12D7         ;End of code that depends on size of PFN link arrays
 0000'CF    0000'DF44    3E   12D7         MOVAW   @W^PFN$AW_SWPVBN[R4],W^PFN$AW_SWPVBN
 0000'CF    0000'DF44    9E   12DF         MOVAB   @W^PFN$AB_STATE[R4],W^PFN$AB_STATE
 0000'CF    0000'DF44    9E   12E7         MOVAB   @W^PFN$AB_TYPE[R4],W^PFN$AB_TYPE
                   52    DD   12EF         PUSHL   R2                          ; SAVE PFN DATA AREA BASE ADDRESS
    52    52   15  09    EF   12F1         EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R2       ; COMPUTE STARTING PAGE NUMBER
      53    0000'DF42    DE   12F6         MOVAL   @W^MMG$GL_SBR[R2],R3        ; AND CONVERT TO SVAPTE
              53  6341   DE   12FC         MOVAL   (R3)[R1],R3                 ; POINTING TO HIGH VPN+1
                  03EE   30   1300  40$:   BSBW    ALLOCPFN                    ; ALLOCATE A PAGE
 73   50   B0000000 8F   C9   1303         BISL3   #<PTE$C_ERKW!PTE$M_VALID!PTE$C_KOWN>,R0,-(R3); MAP A VALID ENTRY
          50   50  09    78   130B         ASHL    #9,R0,R0                    ; CONVERT TO PHYSICAL ADDRESS
          52   0040 8F   3C   130F         MOVZWL  #512/8,R2                   ; SET COUNT TO CLEAR
                   80    7C   1314  45$:   CLRQ    (R0)+                       ; CLEAR A QUAD
                FB 52    F5   1316         SOBGTR  R2,45$                      ; CLEAR ENTIRE PAGE
                E4 51    F5   1319         SOBGTR  R1,40$                      ; DO ALL OF THE PFN DATA BASE PAGES
                   04    BA   131C         POPR    #^M<R2>                     ; RESTORE BASE ADDRESS
                              131E  MAPRPB:                                    ;
    52    52   15  09    EF   131E         EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R2  ; FREE HIGH VPN+1
                   52    D7   1323         DECL    R2                          ; VPN FOR RPB
     50    5B   F7 8F    78   1325         ASHL    #-9,R11,R0                  ; PFN FOR FPB
      00 48 BB     50    E5   132A         BBCC    R0,@RPB$Q_PFNMAP+4(R11),10$      ; MARK RPB ALLOCATED
 50     F0000000 8F      C9   132F  10$:   BISL3   #<PTE$C_URKW!PTE$M_VALID!PTE$C_KOWN>,R0,@MMG$GL_SBR[R2] ; MAP PAGE
      00000000'FF42           1336
                4C AB    D7   133C         DECL    RPB$L_PFNCNT(R11)           ; ONE LESS AVAILABLE PAGE
      50    52     09    78   133F         ASHL    #9,R2,R0                    ; CONVERT VPN TO VA
```

```
0000'CF   50   80000000 8F   C9 1343          BISL3   #^X80000000,R0,W^EXE$GL_RPB      ; SAVE VA OF RPB
          0000'CF     72     9E 134D          MOVAB   -(R2),W^BOO$GL_SPTFREH   ; SAVE FOR SUBSEQUENT USE
     0000'CF   0000'8F       3C 1352          MOVZWL  #MMG$C_SPTSKEL,W^BOO$GL_SPTFREL ; SAVE LOW VPN ALSO
     0000'CF   21AD'CF       D0 1359          MOVL    W^BOO$GL_NEXTPFN,W^MMG$GL_MAXPFN ; SAVE MAX PFN VALUE
                                1360
                                1360 MAP_XDELTA_INIT:                         ; MAP XDELTA AND INIT
          53   FFC00000'8F   D0 1360          MOVL    #<<MMG$A_SYS_END-^X80000000>@-9>,R3     ; VPN+1 OF END OF EXEC
             51   0000'8F    3C 1367          MOVZWL  #<<MMG$A_SYS_END-MMG$AL_PGDCODEN>@-9>,R1; COUNT OF XDELTA+INIT
          53    0000'DF43    DE 136C          MOVAL   @W^MMG$GL_SBR[R3],R3     ; SVAPTE+4
                     037C    30 1372 10$:     BSBW    ALLOCPFN                 ; ALLOCATE A PHYSICAL PAGE
     73   50   F0000000 8F   C9 1375          BISL3   #<PTE$C_URKW!PTE$M_VALID!PTE$C_KOWN>,R0,-(R3); MAP A VALID PAGE
                    F2 51    F5 137D          SOBGTR  R1,10$                   ; MAP ALL POTENTIALLY PAGED EXEC
                                1380
                                1380 ;
                                1380 ;
                                1380 ;        SET MAP INFORMATION FOR BOOT DRIVER
                                1380 ;
                                1380 INITBOOTDRV:                             ;
             59   0000'8F    3C 1380          MOVZWL  #<<MMG$A_SYS_END-^X80000000>@-9>,R9     ; SET NUMBER OF PAGES
       00000000'EF   59      D0 1385          MOVL    R9,BOO$GL_SPTFREL        ; SAVE NEXT FREE SPT ENTRY
             54   34 AB      D0 138C          MOVL    RPB$L_IOVEC(R11),R4      ; GET POINTER TO I/O VECTOR
                       5B    DD 1390          PUSHL   R11                      ; ADDRESS OF RPB
               00000000'9F   DF 1392          PUSHAL  @#MMG$A_SYS_END          ; VA FOR I/O WINDOW
                  0000'CF    DD 1398          PUSHL   W^MMG$GL_SBR             ; PHYSICAL ADDRESS OF SPT
          04 B444     03     FB 139C          CALLS   #3,@BOO$L_MAP(R4)[R4]    ; CALL BOO$MAP
                                13A1 ;
                                13A1 ; Mapping information has now been provided to permit the bootstrap
                                13A1 ; I/O driver to read the EXEC into non-contiguous physical pages.
                                13A1 ;
                                13A1 ;
                                13A1 ; Read the drivers and other loadable pieces of code into their
                                13A1 ; allocated slots in non-paged pool.
                                13A1 ;
          52   2438'CF       DE 13A1          MOVAL   W^DSKDRV_STAT,R2         ; SYSTEM DISK DRIVER STAT BLOCK
             55   0B'        D0 13A6          MOVL    S^#LOAD_IMAGE_CNT,R5     ; NO. OF LOAD IMAGES TO READ
                  0425       30 13A9 10$:     BSBW    LOAD_CODE                ; LOAD THIS IMAGE INTO POOL
             52   18         C0 13AC          ADDL    #STAT_C_SIZE,R2          ; ADDRESS OF NEXT STAT BLOCK
                F7 55        F5 13AF          SOBGTR  R5,10$                   ; LOAD NEXT IMAGE
                                13B2 ;
                                13B2 ; Assuming system paging is enabled (SYSPAGING = 1), the EXEC is read
                                13B2 ; in two pieces.  The first contains all the resident code, the second
                                13B2 ; contains XDELTA, INIT, and the BUGCHECK code.  The pages of BUGCHECK
                                13B2 ; messages are not read in any longer, though they were in release 2.
                                13B2 ; When SYSPAGING = 0, the entire system is read in (including the paged
                                13B2 ; code) and in this case too, the read stops short of bringing in the
                                13B2 ; BUGCHECK messages.  The BUGCHECK code is still required because the
                                13B2 ; salutation put out by INIT uses the CONSOLIO code in BUGCHECK.  There
                                13B2 ; may or may not be other needs for this code.
                                13B2 ;
                                13B2 ; If we are operating Standalone (i.e. boot device is console device) allow
                                13B2 ; for a volume switch at this point.
                                13B2 ;
                                13B2 ; If a volume switch occurs:
                                13B2 ;        - All "open" files will be closed by zeroing the STAT_L_VBN, STAT_L_BYTECNT
                                13B2 ;          and STAT_L_MAP fields.  The file cache will be deallocated.  All file
                                13B2 ;          mapping information in the BOOTCB will be zeroed.
                                13B2 ;        - The SYS_STAT block will contain a dummy retrieval pointer which maps those
```

```
                              13B2  ;       blocks on the previous volume(s).
                              13B2  ;     - The flag RPB$V_NOSYSDISK in RPB$B_FLAGS will be set so that any code with
                              13B2  ;       access to the RPB will be able to tell if a volume switch has occurred.
                              13B2  ;
        2394'CF    01    D0   13B2          MOVL    #1,W^CONEOF_ENABLE        ; SET A FLAG THAT VOLUME SWITCH IS ALLOWED
     59  80000000'8F     D0   13B7          MOVL    #<<<BUG$T_MESSAGES+511>&^C511>-^X80000000>,R9
                              13BE                                            ; NUMBER OF BYTES OF SYS TO READ
     54     2400'CF      D0   13BE          MOVL    W^SYS_STAT+STAT_L_MAP,R4  ; VIRTUAL TO LOGICAL MAP
     53     23F0'CF      D0   13C3          MOVL    W^SYS_STAT+STAT_L_VBN,R3  ; STARTING VBN IN IMAGE
     56     01     1F    78   13C8          ASHL    #VA$V_SYSTEM,#1,R6        ; SET ADDRESS OF "BUFFER"
     28 0000'CF    00'   E1   13CC          BBC     S^#EXE$V_SYSPAGING,W^EXE$GL_DEFFLAGS,20$ ; SINGLE READ IF NOT PAGING
59   0000'CF   80000000 8F   C3   13D2          SUBL3   #^X80000000,W^MMG$GL_PGDCOD,R9   ; SET NUMBER OF BYTES
                 04AE    30   13DC          BSBW    READ_VIRTUAL              ; READ NONPAGED EXEC CODE
                22 50    E9   13DF          BLBC    R0,READ_SYS_ERR           ; BRANCH IF ERROR
     56  00000000'9F     DE   13E2          MOVAL   @#MMG$AL_PGDCODEN,R6      ; SET NEW BUFFER ADDRESS
     59  00000000'8F     D0   13E9          MOVL    #<<<BUG$T_MESSAGES+511>&^C511>-MMG$AL_PGDCODEN>,R9
                              13F0                                            ; SET SIZE OF INIT
              23F0'CF    C1   13F0          ADDL3   W^SYS_STAT+STAT_L_VBN,-   ; COMPUTE VBN FOR INIT
     53  FFC00000'8F          13F4                  #<<MMG$AL_PGDCODEN-^X80000000>@-9>,R3
                              13FA  20$:                                      ;
                 0490    30   13FA          BSBW    READ_VIRTUAL              ; READ EXEC IMAGE
              2394'CF    D4   13FD          CLRL    W^CONEOF_ENABLE           ; CLOSE WINDOW ON VOLUME SWITCHES
                09 50    E8   1401          BLBS    R0,MOVSYSPARAM            ; BRANCH IF SUCCESSFULLY READ
                              1404  ;
                              1404  ; Error reading SYS.EXE - fatal
                              1404  ;
                              1404  READ_SYS_ERR:
     51     2190'CF      DE   1404          MOVAL   W^BOO$GT_SYS,R1           ; APPEND "SYS.EXE"
                 09C5    30   1409          BSBW    BOO$TYPE_ASCIC            ; TO THE READ ERROR MESSAGE
                   00    00   140C          HALT                              ; ***** FATAL ERROR *****
                              140D  ;
                              140D  ; EXEC image has now been read. The correct copy of SYSPARAM
                              140D  ; information will now be moved into the system image.
                              140D  ;
                              140D  MOVSYSPARAM:
                              140D  ;
                              140D  ; Assume the first 12 bytes of the SYSGEN parameter area is EXE$GQ_TODCBASE and
                              140D  ; EXE$GL_TODR and is not to be overwritten.
                              140D  ;
     56  FFFFFFF4'8F     D0   140D          MOVL    #BOO$C_SYSPARSZ-12,R6     ; Size of SYSPARAM in bytes
     57  0000000C'8F     D0   1414          MOVL    #MMG$A_SYSPARAM+12,R7     ; Get System VA of SYSPARAM
     51     000C'CF      DE   141B          MOVAL   W^EXE$A_SYSPARAM+12,R1    ; Address of SYSBOOT copy
                 090B    30   1420          BSBW    MOVEMAPPED                ; Move SYSBOOT copy into sys copy.
                              1423  ;
                              1423  ; Move the FIL$OPENFILE cache into its allocated place in non-paged pool.
                              1423  ; If this cache was allocated space in pool, but before it could be
                              1423  ; safely moved into pool, its pages were allocated and potentially
                              1423  ; written into, then we must disable the cache here.  To do that
                              1423  ; we clear the desriptor for the FIL$OPENFILE cache and
                              1423  ; tack on its size to the INILOA piece.  This is assumed to be allocated
                              1423  ; just before the FIL$OPENFILE cache and the size field of the INILOA
                              1423  ; descriptor is only used for deallocating the INILOA piece after INIT
                              1423  ; has executed that portion of the initialization code.
                              1423  ;
                              1423  MOVFILECACHE:
     56     0000'CF      7D   1423          MOVQ    W^BOO$GQ_FILCACHE,R6      ; R6 = size, R7 = pool address
                   28    13   1428          BEQL    5$                        ; Branch if no cache enabled
```

```
    23D0'CF    21AD'CF   D1  142A          CMPL    W^BOO$GL_NEXTPFN,W^CACHE_HI_PFN ; Alloc over FIL$OPENFILE cache?
                    0B   18  1431          BGEQ    2$                      ; Branch if not
               0000'CF   7C  1433          CLRQ    W^BOO$GQ_FILCACHE       ; No FIL$OPENFILE cache
       0004'CF      56   C0  1437          ADDL    R6,W^BOO$GQ_INILOA+4    ; Deallocate this with INILOA
                    14   11  143C          BRB     5$
               21AD'CF   D1  143E  2$:     CMPL    W^BOO$GL_NEXTPFN,-      ; Alloc over the CI ucode?
               23EC'CF       1442                  W^CI_HI_PFN
                    03   18  1445          BGEQ    3$                      ; Branch if not
                  FBF2   31  1447          BRW     BUMP_CI                 ; Yes, fatal error
                             144A
            51 2169'CF   D0  144A  3$:     MOVL    W^FIL$GQ_CACHE+4,R1     ; Physical location of cache
                  08DC   30  144F          BSBW    MOVEMAPPED              ; Move the cache
                             1452  ;
                             1452  ; If a top level system directory is in use, move its name for INIT
                             1452  ;
               216D'CF   95  1452  5$:     TSTB    W^FIL$GT_TOPSYS         ; Is there a TOPSYS directory?
                    08   13  1456          BEQL    10$                     ; Branch if not
  0000'CF      216D'CF   0A   28  1458     MOVC3   #10,W^FIL$GT_TOPSYS,W^BOO$GT_TOPSYS ; Yes, make a copy for INIT
                             1460  10$:
                             1460  ;
                             1460  ; Move the Boot Control Block to its place in non-paged pool.  Finish
                             1460  ; initializing it and the system disk boot driver for use in INIT.
                             1460  ; This boot control block contains retrieval pointer virtual to logical
                             1460  ; maps for SYS.EXE and SYSDUMP.DMP.  After moving the boot control block,
                             1460  ; the boot driver, and its associated microcode file - if any - , are
                             1460  ; moved adjacent to it.  The space allocated for the boot control block
                             1460  ; includes the space for the boot driver and the microcode.  This code
                             1460  ; moves the 3 pieces into one area of pool.  BOO$W_SIZE of the Boot
                             1460  ; Control Block contains the size without the boot driver.  BOODRV_STAT
                             1460  ; + STAT_L_BYTECNT contains the size with the boot driver and microcode.
                             1460  ;
                             1460  MOVBOODRIVER:
            51 23C4'CF   D0  1460          MOVL    W^BOOTCB,R1             ; Physical adr of boot control block
            57 0000'CF   D0  1465          MOVL    W^BOO$GL_BOOTCB,R7      ; Pool adr to put boot control block
      0F 00A3 CB    00   E1  146A          BBC     #RPB$V_NOSYSDISK, -     ; If we have removed the system disk,
                             1470                  RPB$B_FLAGS(R11),10$    ;  make a single invalid map ptr for sys
      50   14 A1    51   C1  1470          ADDL3   R1,BOO$L_SYS_MAP(R1),R0 ; R0 -> list of rtrv pointers
              60    08   D0  1475          MOVL    #8,(R0)                 ; One pointer, eight bytes long
                 04 A0   D4  1478          CLRL    4(R0)                   ; Zero blocks in the window
           08 A0    01   CE  147B          MNEGL   #1,8(R0)                ; Window starts at LBN -1
           14 A1    57   C0  147F  10$:    ADDL    R7,BOO$L_SYS_MAP(R1)    ; Relocate adr of virtual to logical
                 1C A1   D5  1483          TSTL    BOO$L_DMP_SIZE(R1)      ; If dump file is empty
                    04   13  1486          BEQL    20$                     ;  then leave map address 0
           20 A1    57   C0  1488          ADDL    R7,BOO$L_DMP_MAP(R1)    ; maps for SYS and SYSDUMP
      08 00A3 CB    00   E0  148C  20$:    BBS     #RPB$V_NOSYSDISK, -     ; If we have removed the system disk,
                             1492                  RPB$B_FLAGS(R11),30$    ;  skip adjustments of param and bug maps
           04 A1    57   C0  1492          ADDL    R7,BOO$L_PARAM_MAP(R1)  ; and for SYSPARAM
           24 A1    57   C0  1496          ADDL    R7,BOO$L_BUG_MAP(R1)    ; and for non-resident BUGCHECK
           56    08 A1   3C  149A  30$:    MOVZWL  BOO$W_SIZE(R1),R6       ; Size of BOOTCB not including boot driver
           7E  23E0'CF   7D  149E          MOVQ    W^UCODE_LEN,-(SP)       ; Save room for ucode file
                  6746   9F  14A3          PUSHAB  (R7)[R6]                ; Save SYS virtual adr to put boot driver
           50  2424'CF   D0  14A6          MOVL    W^BOODRV_STAT+STAT_L_BYTECNT,R0 ; Size of BOOTCB + boot driver
           08 A1    50   B0  14AB          MOVW    R0,BOO$W_SIZE(R1)       ; Set size of BOOTCB+driver+ucode
           50  23E0'CF   C2  14AF          SUBL2   W^UCODE_LEN,R0          ; Size of driver+BOOTCB
           7E    50    56   C3  14B4       SUBL3   R6,R0,-(SP)             ; Save size of boot driver to move
      0C AE   04 AE    6E   C1  14B8       ADDL3   (SP),4(SP),12(SP)       ; Set sys virtual adr of ucode
                  086D   30  14BE          BSBW    MOVEMAPPED              ; Move BOOTCB to pool
```

```
         56     8E    7D  14C1         MOVQ    (SP)+,R6                    ; R6 = Size of boot driver to move
                          14C4                                             ; R7 = SYS virtual adr to put it
      51    34  AB    D0  14C4         MOVL    RPB$L_IOVEC(R11),R1         ; Physical address of boot driver
      34 AB     57    D0  14C8         MOVL    R7,RPB$L_IOVEC(R11)         ; Set new SYS virtual adr of boot driver
         58     57    D0  14CC         MOVL    R7,R8                       ; Save it away for later
              085C    30  14CF         BSBW    MOVEMAPPED                  ; Move the boot driver adjacent to BOOTCB
            0000'CF   D0  14D2         MOVL    W^MMG$GL_SPTBASE,-          ; Set SPT virtual address as
              50 AB       14D6                 RPB$L_SVASPT(R11)           ;  adr of SYS page table for BOOTDRIVR
         56     8E    7D  14D8         MOVQ    (SP)+,R6                    ; R6 = Size of ucode to move
                          14DB                                             ; R7 = SYS virtual adr to put it
      51    23E4'CF   D0  14DB         MOVL    W^UCODE_ADR,R1              ; Physical address of ucode
                1E    13  14E0         BEQL    40$                         ; There isn't any
   0000'CF      57    D0  14E2         MOVL    R7,W^BOO$GL_UCODE           ; Stuff the virtual adr away
              0844    30  14E7         BSBW    MOVEMAPPED                  ; Move the ucode adjacent to boot driver
      08    239C'CF   B1  14EA         CMPW    W^VMB_VERSION,#8            ; If the version is >= 8
                0F    1F  14EF         BLSSU   40$                         ;  then update the cell
      51    0000'CF   DE  14F1         MOVAL   W^BOO$GL_UCODE,R1           ; Pick up physical addr of cell w/VA
          56    04    D0  14F6         MOVL    #4,R6                       ; 4 bytes to move
       57   28  A8    DE  14F9         MOVAL   BQO$L_UCODE(R8),R7          ; VA of where to stick it
              082E    30  14FD         BSBW    MOVEMAPPED                  ; Move the VA
                          1500 40$:
                          1500 ;
                          1500 ; Move the copy of BOOPARAM that SYSBOOT has filled with data
                          1500 ;  into INIT's copy of BOOPARAM.
                          1500 ;
                          1500 MOVBOOPARAM:
   56   00000000'8F   D0  1500         MOVL    #BOO$C_BOOPARSZ,R6          ; Size of BOOPARAM in bytes
   57   00000000'8F   D0  1507         MOVL    #EXE$A_BOOPARAM,R7          ; Get System VA of BOOPARAM
      51    0000'CF   DE  150E         MOVAL   W^BOO$A_BOOPARAM,R1         ; Address of SYSBOOT copy
              0818    30  1513         BSBW    MOVEMAPPED                  ; Move SYSBOOT copy into SYS copy.
                          1516 ;
                          1516 ; Now locate INIT and transfer control to it
                          1516 ;
   53   FFC00000'8F   D0  1516         MOVL    #<<EXE$INIT-^X80000000>@-9>,R3   ; VPN OF INIT
     50    0000'DF43  D0  151D         MOVL    @W^MMG$GL_SBR[R3],R0        ; GET PHYSICAL PAGE NUMBER
 53   50    15    00  EF  1523         EXTZV   #PTE$V_PFN,#PTE$S_PFN,R0,R3 ; ISOLATE PFN AND SAVE IN R3
      50    53    09  78  1528         ASHL    #9,R3,R0                    ; CONVERT TO PHYS ADDRESS
   51   00000000'8F   D0  152C         MOVL    #EXE$INIT,R1                ; VIRTUAL ADDRESS OF INIT
 50   09    00    51  F0  1533         INSV    R1,#0,#9,R0                 ; INSERT BYTE OFFSET TO GET PHYSICAL
      0C    0000'CF   DA  1538         MTPR    W^MMG$GL_SBR,#PR$_SBR       ; SET SYSTEM BASE REGISTER
      0D    0000'CF   DA  153D         MTPR    W^MMG$GL_SPTLEN,#PR$_SLR    ; AND LENGTH REGISTER
          53    02    C0  1542         ADDL    #2,R3                       ; P0 LENGTH=PFN+1+1
          09    53    DA  1545         MTPR    R3,#PR$_P0LR                ; SET P0 LENGTH TO PFN OF EXE$INIT
 52   80000000'8F 50  C3  1548         SUBL3   R0,#<EXE$INIT-^X80000000>,R2      ; DELTA PFN-VPN
    52    52   F7 8F  78  1550         ASHL    #-9,R2,R2                   ; CONVERT TO PAGE COUNT
     51    0000'DF42  DE  1555         MOVAL   @W^MMG$GL_SPTBASE[R2],R1    ; COMPUTE BASE FOR P0 PT
          08    51    DA  155B         MTPR    R1,#PR$_P0BR                ; AND SET AS BASE REGISTER
                          155E         INVALID                             ; INVALIDATE TRANSLATION BUFFER
                60    17  1561         JMP     (R0)                        ; TRANSFER TO INIT
                          1563 ;
                          1563 ;       R0 = PHYSICAL ADDRESS OF EXE$INIT
                          1563 ;       R1 = ** UNDEFINED **
                          1563 ;       R2 = ** UNDEFINED **
                          1563 ;       R3 = ** UNDEFINED **
                          1563 ;       R4 = ** UNDEFINED **
                          1563 ;       R5 = ** UNDEFINED **
                          1563 ;       R6 = ** UNDEFINED **
```

```
1563 ;        R7 = ** UNDEFINED **
1563 ;        R8 = ** UNDEFINED **
1563 ;        R9 = ** UNDEFINED **
1563 ;        R10 = ** UNDEFINED **
1563 ;        R11 = PHYSICAL ADDRESS OF RESTART PARAMETER BLOCK (RPB)
1563 ;        AP = ** UNDEFINED **
1563 ;        FP = ** UNDEFINED **
1563 ;        SP = PHYSICAL ADDRESS OF A 3 PAGE STACK
1563 ;        PR$_SBR/PR$_SLR - SET TO DESCRIBE SPT
1563 ;        PR$_P0BR/PR$_P0LR - SET TO MAP EXE$INIT VIRTUAL = REAL
1563 ;
```

```
                                   1563        .SBTTL  ADD MEMORY DESCRIPTORS TO RPB
                                   1563  ;++
                                   1563  ; FUNCTIONAL DESCRIPTION:
                                   1563  ;
                                   1563  ;       THIS ROUTINE IS EXECUTED OR NOT DEPENDING UPON WHAT VERSION OF
                                   1563  ;       VMB BUILT THE RESTART PARAMETER BLOCK.  VERSION 1 AND 2 OF VMB
                                   1563  ;       DO NOT INCLUDE THE PAGE COUNT, TR NUMBER, AND BASE PFN TO DESCRIBE
                                   1563  ;       EACH MEMORY ON THE SYSTEM.  VERSION 3 AND SUBSEQUENT VERSIONS INCLUDE
                                   1563  ;       THIS INFORMATION.  THIS INFORMATION IS USED BY BUG CHECK IN DUMPING
                                   1563  ;       MEMORY TO THE SYSTEM DUMP FILE.  THIS ROUTINE ADDS A MEMORY
                                   1563  ;       DESCRIPTOR TO THE RPB.  THIS DESCRIPTOR IS BUILT USING THE
                                   1563  ;       ASSUMPTIONS THAT PRE-RELEASE 2.0 BUG CHECKS USED, I.E., THAT
                                   1563  ;       MEMORY STARTS AT PFN 0 AND IS BOUNDED AT THE OTHER END BY THE
                                   1563  ;       SYSTEM PAGE TABLE.
                                   1563  ;
                                   1563  ; CALLING SEQUENCE:
                                   1563  ;
                                   1563  ;       BSBW    ADD_RPB_MEMDSC
                                   1563  ;
                                   1563  ; INPUT PARAMETERS:
                                   1563  ;
                                   1563  ;       R11 - ADR OF REBOOT PARAMETER BLOCK
                                   1563  ;
                                   1563  ; OUTPUT PARAMETERS:
                                   1563  ;
                                   1563  ;       NONE
                                   1563  ;
                                   1563  ; IMPLICIT INPUTS:
                                   1563  ;
                                   1563  ;       PR$_SBR AND PR$SLR ARE USED TO DETERMINE THE SIZE OF MEMORY.
                                   1563  ;
                                   1563  ; IMPLICIT OUTPUTS:
                                   1563  ;
                                   1563  ;       NONE
                                   1563  ;
                                   1563  ; COMPLETION CODES:
                                   1563  ;
                                   1563  ;       NONE
                                   1563  ;
                                   1563  ; SIDE EFFECTS:
                                   1563  ;
                                   1563  ;       A MEMORY DESCRIPTOR IS ADDED TO THE RPB IF ONE IS NOT INCLUDED.
                                   1563  ;--
                                   1563
                                   1563  ADD_RPB_MEMDSC:
                         3F    BB  1563        PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; SAVE REGISTERS
             54    00BC  CB    9E  1565        MOVAB   RPB$L_MEMDSC(R11),R4    ; GET ADR OF FIRST MEMORY DESCRIPTOR
                   53    0C    DB  156A        MFPR    #PR$_SBR,R3             ; GET PHYSICAL ADR OF SYS PAGE TBL
                   52    0D    DB  156D        MFPR    #PR$_SLR,R2             ; GET LENGTH OF SPT IN BYTES
             52    6342        DE  1570        MOVAL   (R3)[R2],R2             ; GET # OF BYTES OF MEMORY TOTAL
        84   52    F7    8F    78  1574        ASHL    #-9,R2,(R4)+            ; SET # OF PAGES (NO TR #) INTO MEMDSC
64  0049 8F  00    6B    00    2C  1579        MOVC5   #0,(R11),#0, -          ; SET THE BASE PFN TO 0 AND FILL THE
                                   1581                #<<RPB$C_LENGTH-RPB$L_MEMDSC>-4>,(R4) ; OTHER MEMDSC'S WITH 0'S
                         3F    BA  1581        POPR    #^M<R0,R1,R2,R3,R4,R5>  ; RESTORE REGISTERS
                         05        1583        RSB                             ;
```

```
                                1584         .SBTTL  IMAGE_OPEN - Look up image file
                                1584 ;
                                1584 ; Input parameters:
                                1584 ;       R2                = Statistics block for file to open
                                1584 ;       STAT_L_NAME(R2) = base relative adr of ASCIC file name
                                1584 ;
                                1584 ; Output parameters:
                                1584 ;       R0 = Status
                                1584 ;       R1 = Address of file header buffer
                                1584 ;       R7,R8,R9 altered
                                1584 ;       R2-R5 preserved
                                1584 ;       STAT_L_MAP(R2)       = Data to map virtual block to logical block
                                1584 ;       STAT_L_BYTECNT(R2) = Size of file in bytes
                                1584 ;
                                1584 IMAGE_OPEN:
51   0C A2    23A0'CF      C1   1584         ADDL3   W^SYSBOOT_BASE,STAT_L_NAME(R2),R1 ; Address of ASCIC name string
              50    81     9A   158B         MOVZBL  (R1)+,R0                ; R0 = size, R1 = address
              7E    50     7D   158E         MOVQ    R0,-(SP)                ; Save file name descriptor
              57    5E     D0   1591         MOVL    SP,R7                   ; Address of file name descriptor
         59   23B4'CF      DE   1594         MOVAL   W^RTRV_BUF_DSC,R9       ; Address of rtrv ptr buffer descriptor
              50    69     7D   1599         MOVQ    (R9),R0                 ; R0 = size, R1 = address of buffer
              0C    50     D1   159C         CMPL    R0,#3*4                 ; At least 3 long words required
                                159F                                         ; to express a contiguous file
                    29     19   159F         BLSS    110$                    ; Branch if out of space, fatal
              58    81     DE   15A1         MOVAL   (R1)+,R8                ; R8 = loc to store byte count returned
                                15A4                                         ; R1 = R1 + 4
              68    04     CE   15A4         MNEGL   #4,(R8)                 ; Init count returned for error path
              50    04     C2   15A7         SUBL    #4,R0                   ; Account for byte count long word
              69    50     7D   15AA         MOVQ    R0,(R9)                 ; Size and address of buffer
                                15AD                                         ; to store retrieval pointers in
                  0119     30   15AD         BSBW    FILOPEN                 ; Call FIL$OPENFILE
              5E    08     C0   15B0         ADDL    #8,SP                   ; Clean off name descriptor
              89    68     C2   15B3         SUBL    (R8),(R9)+              ; Allocate retrieval pointers used
                    12     19   15B6         BLSS    110$                    ; Branch if not enough space
              69    68     C0   15B8         ADDL    (R8),(R9)               ; Update next available address
                 0B 50     E9   15BB         BLBC    R0,20$                  ; Leave bytecnt 0 if error
04 A2    21B9'CF    09     78   15BE         ASHL    #9,W^BOO$GQ_STATBLK+4,STAT_L_BYTECNT(R2) ; Size of file in bytes
           10 A2    58     D0   15C5         MOVL    R8,STAT_L_MAP(R2)       ; Save adr of virtual to logical map
                           05   15C9 20$:    RSB                             ; Otherwise return with status in R0
                                15CA ;
                                15CA ; Not enough space for retrieval pointers, the boot time files were
                                15CA ; far more fragmented than was planned for.  Only SYS.EXE is big
                                15CA ; enough to cause such a problem.  All the drivers are so small that
                                15CA ; a few pointers each is adequate.
                                15CA ;
                                15CA 110$:   MSG     <-F-Boot time files have too many non-contiguous pieces>
                           00   1604         HALT                            ; ***** Fatal error *****
```

```
                                 1605          .SBTTL  IMAGE_INIT - Look up and start read of image file
                                 1605   ;
                                 1605   ; Input parameters:
                                 1605   ;       R2              = Statistics block for file to open
                                 1605   ;       STAT_L_NAME(R2) = base relative adr of ASCIC file name
                                 1605   ;
                                 1605   ; Output parameters:
                                 1605   ;       Returns to caller only if successful.
                                 1605   ;       On error, prints diagnostic and halts
                                 1605   ;       R0,R1,R3,R4,R7,R8,R9 altered
                                 1605   ;       R2,R5 preserved
                                 1605   ;       R6              = Adr of first block read in past image header
                                 1605   ;       STAT_L_MAP(R2)  = Data to map virtual block to logical block
                                 1605   ;       STAT_L_VBN(R2)  = Starting VBN in image after image header block(s)
                                 1605   ;       STAT_L_VBN2ADR(R2) = Address of 1st block after image header block(s)
                                 1605   ;
                                 1605   IMAGE_INIT:
                     FF7C   30   1605          BSBW    IMAGE_OPEN                      ; Open the file
                    49 50   E9   1608          BLBC    R0,90$                          ; Branch if failed to find file
                                 160B   ;
                                 160B   ; Read 2 blocks on the assumption that the image header will only take
                                 160B   ; up the first block.  This is essentially always correct.
                                 160B   ;
   56   23C0'CF   23BC'CF   C1   160B          ADDL3   W^VBN2_BUF_DSC,W^VBN2_BUF_DSC+4,R6 ; Adr of last+1 byte of
                                 1613                                                  ; block buffer for reading and saving
                                 1613                                                  ; the 1st block after image header
             56   FC00 C6   9E   1613          MOVAB   -1024(R6),R6                    ; Back off 2 pages, first is image
                                 1618                                                  ; header, 2nd is first block after
   23BC'CF   00000200 8F    C2   1618          SUBL    #512,W^VBN2_BUF_DSC             ; Record another page used
          59     02   09    9C   1621          ROTL    #9,#2,R9                        ; Set to read 2 VBN's
                 53   01    D0   1625          MOVL    #1,R3                           ; Starting with VBN 1
             54    10 A2    D0   1628          MOVL    STAT_L_MAP(R2),R4               ; Adr of virtual to logical map
                     025E   30   162C          BSBW    READ_VIRTUAL                    ; Read the desired VBN's
                    3F 50   E9   162F          BLBC    R0,100$                         ; Branch if read error
             53    10 A6    9A   1632          MOVZBL  IHD$B_HDRBLKCNT(R6),R3          ; Get # header blks
                       53   D6   1636          INCL    R3                              ; Starting VBN of image
                 62    53   D0   1638          MOVL    R3,STAT_L_VBN(R2)               ; Save for reading image later
             56   0200 C6   9E   163B          MOVAB   512(R6),R6                      ; Address of VBN 2
                 02    53   D1   1640          CMPL    R3,#2                           ; One image header block?
                       0A   13   1643          BEQL    20$                             ; Branch if yes, the norm
                                 1645   ;
                                 1645   ; Lost a good bet, now read one block after the image header blocks
                                 1645   ;
          59     01   09    9C   1645          ROTL    #9,#1,R9                        ; Set up to read one block
                     0241   30   1649          BSBW    READ_VIRTUAL                    ; Read the desired VBN
                    22 50   E9   164C          BLBC    R0,100$                         ; Branch if read error
             14 A2     56   D0   164F   20$:   MOVL    R6,STAT_L_VBN2ADR(R2)           ; Note where VBN2 was read
                            05   1653          RSB
                                 1654   ;
                                 1654   ; Failed to open file
                                 1654   ;
                                 1654   90$:   MSG     <-E-Unable to locate file >
                                 1671   ;
                                 1671   ; Error opening or reading the file, error message already given,
                                 1671   ; add on the file name to complete the diagnostic
                                 1671   ;
   51    0C A2   23A0'CF    C1   1671   100$:  ADDL3   W^SYSBOOT_BASE,STAT_L_NAME(R2),R1 ; Address of ASCIC file name
```

```
0756  30  1678  2468        BSBW    BOO$TYPE_ASCIC          ; Type the name
      00  167B  2469        HALT                            ; ***** Fatal Error *****
```

```
                              167C         .SBTTL  BOO$FILOPEN - Lookup a specified file
                              167C  ;
                              167C  ; Input Parameters:
                              167C  ;       R7 - Address of file name descriptor
                              167C  ;
                              167C  ; Output Parameters:
                              167C  ;       R0 - Status, if error, message (without file name) already typed
                              167C  ;       R8 - Starting LBN of file
                              167C  ;       R9 - Size of file in blocks
                              167C  ;
                              167C  BOO$FILOPEN::                               ; File open Routine
                    58    7C  167C         CLRQ    R8                          ; No retrieval pointer info needed
                    49    10  167E         BSBB    FILOPEN                     ; Call FIL$OPENFILE
               1F   50    E8  1680         BLBS    R0,10$                      ; Branch if successful
                              1683         MSG     <-E-Unable to locate file > ; Report the failure
                    24    11  16A0         BRB     20$
          58    21B5'CF   7D  16A2  10$:   MOVQ    W^BOO$GQ_STATBLK,R8         ; Get file statistics, LBN/Size in blocks
                    58    D5  16A7         TSTL    R8                          ; Check for contiguous
                    1D    12  16A9         BNEQ    30$                         ; Yes, continue
                              16AB         MSG     <-E-File not contiguous >   ; Give error
                    50    D4  16C6  20$:   CLRL    R0                          ; Indicate error
                          05  16C8  30$:   RSB                                 ; Return to caller
                              16C9  ;
                              16C9  ; Inputs:
                              16C9  ;       R7 - Address of file name descriptor
                              16C9  ;       R8 - Address to return byte of retrieval pointer data stored
                              16C9  ;            0 if not used
                              16C9  ;       R9 - Address of descriptor of retrieval pointer buffer
                              16C9  ;            0 if not used
                              16C9  ;
                              16C9  ; Outputs:
                              16C9  ;       R0 - Status
                              16C9  ;       R1 - Address of file header buffer
                              16C9  ;       All other registers preserved
                              16C9  ;       BOO$GQ_STATBLK     = Starting LBN if contiguous, 0 if not
                              16C9  ;       BOO$GQ_STATBLK + 4 = Size of file in blocks
                              16C9  ;
                              16C9  FILOPEN:
               50   7E    DE  16C9         MOVAL   -(SP),R0                    ; Save pointer to phony channel
               7E   58    7D  16CC         MOVQ    R8,-(SP)                    ; Push adr of rtrv buf descriptor
                              16CF                                             ; Push adr to return byte count stored
               21B5'CF    7F  16CF         PUSHAQ  W^BOO$GQ_STATBLK            ; Address of file statistics block
               2198'CF    DD  16D3         PUSHL   W^BOO$GL_FREEMEM            ; file header buffer
  7E   6E   00000200 8F   C1  16D7         ADDL3   #512,(SP),-(SP)             ; Index file header buffer
                    67    DF  16DF         PUSHAL  (R7)                        ; Address of file name descriptor
                    60    DF  16E1         PUSHAL  (R0)                        ; Address of phony channel
          0000'CF   07    FB  16E3         CALLS   #7,W^FIL$OPENFILE           ; Call FILEREAD to locate file
          51    2198'CF   D0  16E8         MOVL    W^BOO$GL_FREEMEM,R1         ; Return address of file header
               5E   04    C0  16ED         ADDL2   #4,SP                       ; Clean off phony channel
                          05  16F0         RSB                                 ;  and return
```

```
                          16F1          .SBTTL  ALLOCPFN - Allocate physical page
                          16F1  ;
                          16F1  ; Output parameters:
                          16F1  ;       R0 - PFN of allocated page
                          16F1  ;
                          16F1  ALLOCPFN:                                     ;
    50   21AD'CF     D0   16F1          MOVL    W^BOO$GL_NEXTPFN,R0           ; Get PFN to start scan
              09     12   16F6          BNEQ    20$                           ; Branch if not null
  50    44 AB        03   16F8  10$:    ASHL    #3,RPB$Q_PFNMAP(R11),R0       ; Else start at end of map
              50     D7   16FD  15$:    DECL    R0                            ; Next/first PFN
              2F     15   16FF          BLEQ    30$                           ; Branch if none left
     F7 48 BB 50     E5   1701  20$:    BBCC    R0,@RPB$Q_PFNMAP+4(R11),15$        ; Allocate it if possible
  21AD'CF     50     01   1706          SUBL3   #1,R0,W^BOO$GL_NEXTPFN        ; Save pointer for next time
           4C AB     D7   170C          DECL    RPB$L_PFNCNT(R11)             ; Account for page removed
     23D4'CF  50     D1   170F          CMPL    R0,W^PFNMAP_HI_PFN            ; Ran into PFN bitmap?
              03     1A   1714          BGTRU   22$
            F1AD     31   1716          BRW     PHYP_ER                       ; Branch if yes
     23D0'CF  50     D1   1719  22$:    CMPL    R0,W^CACHE_HI_PFN             ; Ran into FIL$OPENFILE cache?
              08     14   171E          BGTR    25$                           ; Branch if not
         2165'CF     7C   1720          CLRQ    W^FIL$GQ_CACHE                ; Yes, disable it
         23D0'CF     D4   1724          CLRL    W^CACHE_HI_PFN
     23EC'CF  50     D1   1728  25$:    CMPL    R0,W^CI_HI_PFN                ; Ran into CI microcode?
              2B     1B   172D          BLEQU   40$                           ; Yes
              05          172F          RSB                                   ;  and return
                          1730  30$:    MSG     <-F-Unable to allocate physical memory>
              00          1759          HALT                                  ; **** FATAL ERROR ****
            F5DF     31   175A  40$:    BRW     BUMP_CI
                          175D  ;
                          175D  ; Functional Description:
                          175D  ;       DALLOCPFN marks the specified page available in the PFN bitmap
                          175D  ;       but does not reposition the scan pointer.
                          175D  ;
                          175D  ; Calling Sequence:
                          175D  ;       BSBW    DALLOCPFN
                          175D  ; Input Parameters:
                          175D  ;       R0 - PFN
                          175D  ;
                          175D  DALLOCPFN:                                    ; Deallocate pfn
     00 48 BB 50     E2   175D          BBSS    R0,@RPB$Q_PFNMAP+4(R11),10$        ; Mark it available
              05          1762  10$:    RSB                                   ; And return
                          1763
```

```
                              1763          .SBTTL  MOVE_BITMAP - move the PFN bit map if PHYSICAL PAGES reduced
                              1763
                              1763   ;++
                              1763   ; Functional Description:
                              1763   ;       An attempt is made to either move the PFN bitmap to the preallocated
                              1763   ;       small bit map or if that fails, to unused physical memory.
                              1763   ;
                              1763   ; Inputs:
                              1763   ;       R0 - Highest page represented in current bitmap
                              1763   ;       R1 - Base of current bitmap
                              1763   ;       R3 - Required size of new bit map (bits)
                              1763   ;
                              1763   ; Outputs:
                              1763   ;       R0 -> highest page for new bitmap
                              1763   ;       R1 -> New base of bitmap
                              1763   ;       all otherregisters preserved
                              1763   ;
                              1763   ;--
                              1763
                              1763   MOVE_BITMAP:
            007C 8F     BB    1763          PUSHR   #^M<R2,R3,R4,R5,R6>      ; Get some scratch registers
   56    53   FD 8F     78    1767          ASHL    #-3,R3,R6                ; Full bytes to copy from current bitmap
                 56     D6    176C          INCL    R6                       ; One more for last byte
        23D8'CF  56     D1    176E          CMPL    R6,W^SMALL_PFNMAP        ; Can it hold everything we intend to use
                 13     1A    1773          BGTRU   10$                      ; No
           48 BB 56     2C    1775          MOVC5   R6,@RPB$Q_PFNMAP+4(R11),- ; Copy the bitmap
 23DC'DF 23D8'CF 00           1779                  #0,W^SMALL_PFNMAP,@W^SMALL_PFNMAP+4
   44 AB   23D8'CF      7D    1780          MOVQ    W^SMALL_PFNMAP,RPB$Q_PFNMAP(R11) ; Use the small bitmap
                 34     11    1786          BRB     50$
                              1788   ;
 55    53   00001000 8F C1    1788   10$:   ADDL3   #4096,R3,R5              ; Round up the bit map pages needed
    55    55    F4 8F   78    1790          ASHL    #-12,R5,R5               ; Make it pages
                 54     D4    1795   20$:   CLRL    R4                       ; Count of contiguous pages found
            52   53     D0    1797          MOVL    R3,R2                    ; Save base page
            50   53     D1    179A   30$:   CMPL    R3,R0                    ; Time to give up?
                 21     1E    179D          BGEQU   60$                      ; Yes - not enough contiguous pages
                 53     D6    179F          INCL    R3                       ; Next page
         F0 61   53     E1    17A1          BBC     R3,(R1),20$              ; Is page good - no - start again
                 54     D6    17A5          INCL    R4                       ; One more
            55   54     D1    17A7          CMPL    R4,R5                    ; Found enough?
                 EE     19    17AA          BLSS    30$                      ; No - try for more
       52   52   09     78    17AC          ASHL    #9,R2,R2                 ; Base of new bitmap
         48 AB   52     D0    17B0          MOVL    R2,RPB$Q_PFNMAP+4(R11)
         44 AB   56     D0    17B4          MOVL    R6,RPB$Q_PFNMAP(R11)     ; New size of bitmap
       62   61   56     28    17B8          MOVC3   R6,(R1),(R2)             ; Copy the bitmap
                              17BC   ;
        23D4'CF         D4    17BC   50$:   CLRL    W^PFNMAP_HI_PFN          ; No longer need to check for conflicts
        007C 8F         BA    17C0   60$:   POPR    #^M<R2,R3,R4,R5,R6>
        23CC'CF  53     D0    17C4          MOVL    R3,W^MEM_HI_PFN          ; Forget about discarded pages
       50    44 AB      7D    17C9          MOVQ    RPB$Q_PFNMAP(R11),R0     ; New base and limit
            50   08     C4    17CD          MULL    #8,R0
                        05    17D0          RSB
                              17D1
```

```
                                 17D1          .SBTTL  LOAD_CODE - load the code from the specified file
                                 17D1
                                 17D1  ;++
                                 17D1  ; Functional Description:
                                 17D1  ;        Given a statistics block for a file, load the file into pool
                                 17D1  ;        The first block after the image header was previously read
                                 17D1  ;        in to determine the size of the image.  This buffer was
                                 17D1  ;        saved to avoid reading it again.  Move this page and
                                 17D1  ;        read the rest.
                                 17D1  ;
                                 17D1  ; Inputs:
                                 17D1  ;        STAT_L_VBN2ADR(R2) = Address of 1st block after image header
                                 17D1  ;                             It was already read in, just move this page
                                 17D1  ;        STAT_L_VBN(R2)     = Starting virtual block number to read
                                 17D1  ;        STAT_L_BYTECNT(R2) = Number of bytes in image
                                 17D1  ;        STAT_L_MAP(R2)     = Mapping information to map VBN to LBN
                                 17D1  ;        STAT_L_SYSVA(R2)   = Base relative address of pool address
                                 17D1  ;        STAT_L_NAME(R2)    = Base relative address of name for error msg
                                 17D1  ;
                                 17D1  ; Outputs:
                                 17D1  ;        Returns in line only if successful
                                 17D1  ;        Types diagnostic and halts if error
                                 17D1  ;        R0,R1,R3,R4,R6,R9 altered
                                 17D1  ;        All other registers preserved
                                 17D1  ;--
                                 17D1
                                 17D1  LOAD_CODE:
                        57   DD  17D1          PUSHL   R7
              51     14 A2   D0  17D3          MOVL    STAT_L_VBN2ADR(R2),R1    ; Adr of 1st block after image hdr
    57   08 A2     23A0'CF   C1  17D7          ADDL3   W^SYSBOOT_BASE,STAT_L_SYSVA(R2),R7 ; Adr of pool adr for image
                   57   67   D0  17DE          MOVL    (R7),R7                  ; Adr in pool to move 1st page of image
                   0200 C7   9F  17E1          PUSHAB  512(R7)                  ; Adr in pool to read into
             56   01    09   9C  17E5          ROTL    #9,#1,R6                 ; Move 1 page or less
              04 A2     56   D1  17E9          CMPL    R6,STAT_L_BYTECNT(R2)    ; Image less than a page?
                        04   15  17ED          BLEQ    10$                      ; Branch if not
             56   04 A2      D0  17EF          MOVL    STAT_L_BYTECNT(R2),R6    ; Yes, use the image size
      59   04 A2        56   C3  17F3  10$:    SUBL3   R6,STAT_L_BYTECNT(R2),R9 ; Byte count left to be read
                        34   BB  17F8          PUSHR   #^M<R2,R4,R5>
                      0531   30  17FA          BSBW    MOVEMAPPED               ; Move the 1st image block after the
                                 17FD                                           ; image hdr, avoid reading it again
                   00F4 8F   BA  17FD          POPR    #^M<R2,R4,R5,R6,R7>      ; R6 = Adr to read into
             53   62    01   C1  1801          ADDL3   #1,STAT_L_VBN(R2),R3     ; Starting VBN to read
                        59   D5  1805          TSTL    R9                       ; Anything left to read?
                        01   14  1807          BGTR    20$                      ; Branch if yes
                             05  1809          RSB                              ; Move was all that was needed
             54     10 A2    D0  180A  20$:    MOVL    STAT_L_MAP(R2),R4        ; Adr of virtual to logical map
                        7D   10  180E          BSBB    READ_VIRTUAL             ; Read the specified VBN's
                     01 50   E9  1810          BLBC    R0,30$                   ; Branch if failed to read them
                             05  1813          RSB
    51   0C A2     23A0'CF   C1  1814  30$:    ADDL3   W^SYSBOOT_BASE,STAT_L_NAME(R2),R1 ; Adr of ASCIC file name
                      05B3   30  181B          BSBW    BOO$TYPE_ASCIC           ; Append the file name to error msg
                             00  181E          HALT                             ; ***** Fatal Error *****
```

```
                        181F          .SBTTL  BOO$SETMAP - Set up Virtual to Logical Map
                        181F    ;++
                        181F    ; Functional Description:
                        181F    ;       This routine produces a virtual to logical map for a specified
                        181F    ;       virtual segment of a file.  Give a desired virtual block and
                        181F    ;       block count and a map of the file, it produces a new map for
                        181F    ;       just that segment.
                        181F    ;
                        181F    ; Calling Sequence:
                        181F    ;       BSBW    BOO$SETMAP
                        181F    ;
                        181F    ; Inputs:
                        181F    ;       R3 = Desired VBN
                        181F    ;       R4 = Virtual to logical map
                        181F    ;       R5 = Desired number of blocks to map
                        181F    ;       R6 = Size in bytes of region to store the map being created
                        181F    ;       R7 = Address of region to store the map being created
                        181F    ;
                        181F    ; Outputs:
                        181F    ;       R0,R1,R2,R3,R5 altered
                        181F    ;       R4 preserved
                        181F    ;       R6,R7 updated to reflect space used
                        181F    ;       others preserved
                        181F    ;--
                        181F    BOO$SETMAP:
             54    DD   181F            PUSHL   R4                      ; Save input map
             87    DF   1821            PUSHAL  (R7)+                   ; Save Adr of output byte count
                        1823                                            ; R7 points to first rtrv ptr to store
             28    10   1823            BSBB    BOO$MAPVBN              ; Find starting LBN
             50    D5   1825            TSTL    R0                      ; If EOF, then done
             16    13   1827            BEQL    60$
       55    50    D1   1829    20$:    CMPL    R0,R5                   ; More block mapped than needed?
             03    15   182C            BLEQ    40$                     ; Branch if not
       50    55    D0   182E            MOVL    R5,R0                   ; Yes, just use what is needed
       87    50    7D   1831    40$:    MOVQ    R0,(R7)+                ; Store this retrieval pointer
       55    50    C2   1834            SUBL    R0,R5                   ; Count blocks mapped
             06    15   1837            BLEQ    60$                     ; Branch if all done
       50    84    7D   1839            MOVQ    (R4)+,R0                ; Get next retrieval pointer
          EA 52    F5   183C            SOBGTR  R2,20$                  ; If there are any more
             14    BA   183F    60$:    POPR    #^M<R2,R4>              ; R2 = Adr of byte count field
                        1841                                            ; R4 = saved input map
 50    57    52    C3   1841            SUBL3   R2,R7,R0                ; No. of bytes used
       56    50    C2   1845            SUBL    R0,R6                   ; Adjust size of buffer remaining
 62    50    04    C3   1848            SUBL3   #4,R0,(R2)              ; No. of bytes of retrieval pointers
             05         184C            RSB
```

```
                          184D        .SBTTL  BOO$MAPVBN - Map Virtual to Logical Block
                          184D  ;++
                          184D  ; Functional Description:
                          184D  ;       Map the specified virtual block to its associated logical block
                          184D  ;
                          184D  ; Calling Sequence:
                          184D  ;       BSBW    BOO$MAPVBN
                          184D  ;
                          184D  ; Inputs:
                          184D  ;       R3 = Virtual Block Number
                          184D  ;       R4 = Address of virtual to logical map
                          184D  ;               # of bytes of retrieval pointers following
                          184D  ;               count of LBN's in first rtrv ptr
                          184D  ;               starting LBN in first rtrv ptr
                          184D  ;               count of LBN's in second rtrv ptr
                          184D  ;               starting LBN in second rtrv ptr
                          184D  ;
                          184D  ;               ...
                          184D  ;
                          184D  ;               count of LBN's in last rtrv ptr
                          184D  ;               starting LBN in last rtrv ptr
                          184D  ;
                          184D  ; Outputs:
                          184D  ;       R0 = Number of contiguous blocks starting at LBN in R1
                          184D  ;       R1 = Starting LBN for the specified VBN
                          184D  ;       R2 = number of retrieval pointers (quad words) not yet used in the map
                          184D  ;       R3 = VBN preserved
                          184D  ;       R4 = pointer to first retrieval pointer not yet used in map
                          184D  ;       all other registers preserved
                          184D  ;--
                          184D  BOO$MAPVBN:
                  53  DD  184D          PUSHL   R3                      ; Save desired VBN
  52  84    FD 8F     78  184F          ASHL    #-3,(R4)+,R2            ; Get count of quad words
                          1854                                          ; Point at first retrieval pointer
           50     84  7D  1854  10$:    MOVQ    (R4)+,R0                ; R0 = block count, R1 = LBN
           50     53  D1  1857          CMPL    R3,R0                   ; Desired VBN in this rtrv ptr?
                  0A  15  185A          BLEQ    20$                     ; Branch if yes
           53     50  C2  185C          SUBL    R0,R3                   ; Pass over that many VBN's
              F2  52  F5  185F          SOBGTR  R2,10$                  ; Try the next retrieval pointer
                  50  7C  1862          CLRQ    R0                      ; EOF, no blocks mapped
                  08  11  1864          BRB     30$
                  53  D7  1866  20$:    DECL    R3                      ; Make VBN base 0
           50     53  C2  1868          SUBL    R3,R0                   ; No. of blocks left
           51     53  C0  186B          ADDL    R3,R1                   ; Starting LBN for desired VBN
                  08  BA  186E  30$:    POPR    #^M<R3>                 ; Recover desired VBN
                      05  1870          RSB
```

```
                          1871          .SBTTL  BOO$READFILE - Routine to read specified piece of file
                          1871  ;++
                          1871  ; Functional Description:
                          1871  ;       BOO$READFILE reads the specified piece of the file
                          1871  ;
                          1871  ; Calling Sequence:
                          1871  ;       BSBW    BOO$READFILE
                          1871  ;
                          1871  ; Inputs:
                          1871  ;       R6 - Buffer address (updated)
                          1871  ;       R8 - Logical block number (updated)
                          1871  ;       R9 - Blocks in file (updated)
                          1871  ;
                          1871  ; Outputs:
                          1871  ;       R0 - Status, diagnostic issued (without file name) if error
                          1871  ;       R1,R6-R9 altered
                          1871  ;       R2-R5 preserved
                          1871  ;--
                          1871  BOO$READFILE::                                  ;
           7E    5A  7D   1871          MOVQ    R10,-(SP)                       ; Save additional registers
           5A    21  3C   1874          MOVZWL  #IC$_READLBLK,R10               ; Read function
      5B  23A4'CF    D0   1877          MOVL    W^BOO$GL_RPBBASE,R11            ; RPB address
      59   59    09  78   187C          ASHL    #9,R9,R9                        ; Form byte count to transfer
               034D  30   1880          BSBW    QIO_RWLB                        ; Read the specified blocks
  59   59    F7 8F   78   1883          ASHL    #-9,R9,R9                       ; Pages not transferred
           5A    8E  7D   1888          MOVQ    (SP)+,R10                       ; Restore additional registers
                 4B  11   188B          BRB     READ_COMPLETE                   ; Use common exit path
```

```
                     188D          .SBTTL  READ_VIRTUAL - Read specified virtual blocks of a file
                     188D  ;++
                     188D  ; Functional Description:
                     188D  ;       This routine maps the specified virtual blocks to logical blocks
                     188D  ;       and reads the desired number of bytes into the specified location
                     188D  ;       in memory.
                     188D  ;
                     188D  ; Inputs:
                     188D  ;       R3 = Virtual Block Number
                     188D  ;       R4 = Mapping info for virtual to logical mapping:
                     188D  ;               # of bytes of retrieval pointers following
                     188D  ;               count of LBN's in first rtrv ptr
                     188D  ;               starting LBN in first rtrv ptr
                     188D  ;               count of LBN's in second rtrv ptr
                     188D  ;               starting LBN in second rtrv ptr
                     188D  ;
                     188D  ;               ...
                     188D  ;
                     188D  ;               count of LBN's in last rtrv ptr
                     188D  ;               starting LBN in last rtrv ptr
                     188D  ;       R6 = Buffer Address to read into
                     188D  ;            if system bit is set, then the system map is used
                     188D  ;       R9 = Byte count to read (quad word aligned)
                     188D  ;
                     188D  ; Outputs:
                     188D  ;       R0 = Status, diagnostic issued if error (without file name)
                     188D  ;       R1 altered
                     188D  ;       All other registers preserved
                     188D  ;
                     188D  ;--
                     188D
                     188D  READ_VIRTUAL:
      0FFC 8F    BB  188D          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
 5B  23A4'CF     D0  1891          MOVL    W^BOO$GL_RPBBASE,R11    ; Address of RPB
 5A      21      3C  1896          MOVZWL  #IO$_READLBLK,R10       ; Read logical block function
         B2      10  1899          BSBB    BOO$MAPVBN              ; Convert starting VBN to LBN
         50      D5  189B          TSTL    R0                      ; End of file?
         05      12  189D          BNEQ    40$                     ; Branch if not
         25      11  189F          BRB     60$                     ; Yes, report the error
 50      84      7D  18A1  30$:    MOVQ    (R4)+,R0                ; Get the next rtrv ptr
                     18A4  ;
                     18A4  ; R0 = number of blocks that can be read in this portion
                     18A4  ; R1 = starting LBN to read from
                     18A4  ;
         59      DD  18A4  40$:    PUSHL   R9                      ; Save desired byte count
 50  50  09      78  18A6          ASHL    #9,R0,R0                ; # of bytes that can be read
     50  59      D1  18AA          CMPL    R9,R0                   ; If fewer are needed
         03      15  18AD          BLEQ    50$                     ; Then read the smaller number
     59  50      D0  18AF          MOVL    R0,R9                   ; Otherwise read all we can
     6E  59      C2  18B2  50$:    SUBL    R9,(SP)                 ; Note how much is left to be read
     58  51      D0  18B5          MOVL    R1,R8                   ; Starting LBN of read request
       0315      30  18B8          BSBW    QIO_RWLB                ; Read or write the file
 59      8E      D0  18BB          MOVL    (SP)+,R9                ; Recover byte left to be read
         11      15  18BE          BLEQ    90$                     ; Branch if all done
      0E 50      E9  18C0          BLBC    R0,90$                  ; Branch if read error
      DB 52      F5  18C3          SOBGTR  R2,30$                  ; Get the next retrieval pointer
 50  0000'8F     3C  18C6  60$:    MOVZWL  #SS$_ENDOFFILE,R0       ; Indicate EOF error
```

```
              56  10  18CB  2849            BSBB    CONEOF_CHECK            ; Check EOF to see if booting from console
              50  D5  18CD  2850            TSTL    R0                      ; R0 will be cleared if read should continue
              D0  13  18CF  2851            BEQL    30$
         0FFC 8F  BA  18D1  2852 90$:       POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                      18D5  2853 ;
                      18D5  2854 ; R0 = status from read
                      18D5  2855 ;
                      18D5  2856 READ_COMPLETE:
            4A 50  E8  18D5  2857            BLBS    R0,30$                  ; Branch if read successful
    50  0000'8F  B1  18D8  2858            CMPW    #SS$_ENDOFFILE,R0       ; If it is an EOF error
              23  12  18DD  2859            BNEQ    10$
                      18DF  2860            MSG     <-E-End of file error reading > ; Report EOF error
              1E  11  1900  2861            BRB     20$
                      1902  2862 10$:       MSG     <-E-I/O error reading file > ; Otherwise catchall error
              50  D4  1920  2863 20$:       CLRL    R0                      ; Indicate error occurred
                  05  1922  2864 30$:       RSB
```

```
                            1923          .SBTTL  CONEOF_CHECK
                            1923  ;++
                            1923  ; Functional Description:
                            1923  ;       This routine checks for a specific end-of-file situation.  If we get an EOF
                            1923  ;       while reading SYS.EXE from the console device, then we will prompt for the
                            1923  ;       next volume to be mounted in the console drive.  After the next volume is
                            1923  ;       loaded, we will open the next piece of SYS.EXE.  We will add a dummy retriev
                            1923  ;       pointer which will map the previous piece(s) of SYS.EXE to an invalid VBN.
                            1923  ;       Finally, we will modify the registers so that READ_VIRTUAL can continue to
                            1923  ;       read in the next piece of SYS.EXE.
                            1923  ;
                            1923  ; Calling Sequence:
                            1923  ;       BSBx    CONEOF_CHECK
                            1923  ;
                            1923  ; Inputs:
                            1923  ;       R0  = SS$_ENDOFFILE
                            1923  ;       R1  = (trash)
                            1923  ;       R2  = count of rtrv ptr remaining (should be zero since we are in EOF)
                            1923  ;       R3  = VBN
                            1923  ;       R4  = pointer to next rtrv ptr (during eof points past end)
                            1923  ;       R5  = (trash)
                            1923  ;       R6  = buffer address to read to
                            1923  ;       R7  = (trash)
                            1923  ;       R8  = highest lbn in previous read
                            1923  ;       R9  = Byte count left to read
                            1923  ;       R10 = #IO$_READLBLK
                            1923  ;       R11 = Address of RPB
                            1923  ;
                            1923  ; Outputs:
                            1923  ;       R0  = 0 if read should be continued, unchanged if EOF error should stand.
                            1923  ;       If R0 = 0, then
                            1923  ;               R1 = (altered)
                            1923  ;               R2 = Count of valid retrieval pointers in the new section
                            1923  ;               R4 = Points to first valid pointer
                            1923  ;       All other registers preserved
                            1923  ;--
                            1923  CONEOF_CHECK:
        66 AB     91        1923          CMPB    RPB$B_DEVTYP(R11),-     ; Booting from the console block
        40 8F               1926                  #BTD$K_CONSOLE          ;  storage device?
           06     12        1928          BNEQ    10$                     ; Not booting from console, let EOF stand
      2394'CF     D5        192A          TSTL    W^CONEOF_ENABLE         ; Have console EOF's been enabled?
           01     12        192E          BNEQ    20$                     ; Looks like a valid EOF, prepare for switch
                  05        1930  10$:    RSB                             ; Let the EOF error stand
                            1931  20$:                                    ; Commit to either switch volumes or halt
                            1931  ;
                            1931  ; Perform a couple of sanity checks.  Make sure that syspaging is off and that we
                            1931  ; are reading SYS.EXE.  If we are reading SYS, then the next map pointer should
                            1931  ; be at the end of the map pointer described by SYS_STAT.
                            1931  ;
 43 0000'CF      00'  E1        1931          BBC     S^#EXE$V_SYSPAGING,W^EXE$GL_DEFFLAGS,30$
                            1937          MSG     <-F-Console volume switch attempted with SYSPAGING=1, must be 0>
                  00        1979          HALT
     50   2400'CF     D0        197A  30$:    MOVL    W^SYS_STAT+STAT_L_MAP,R0        ; Get address of map for SYS.EXE
          50   60     C0        197F          ADDL2   (R0),R0                         ; Add length of map to address
          50   04     C0        1982          ADDL2   #4,R0                           ; Adjust for length longword
          50   54     D1        1985          CMPL    R4, R0                          ; Is it the same?
               3D     13        1988          BEQL    40$                             ; If same, this is SYS.EXE
```

```
                          198A        MSG     <-F-Console EOF on wrong file, supported for SYS.EXE only>
                    00    19C6        HALT
              52    D5    19C7 40$:   TSTL    R2                                  ; R2 (ptr cnt) should be zero
              35    13    19C9        BEQL    50$
                          19CB        MSG     <-F-Console EOF before end of map, internal error>
                    00    19FF        HALT
                          1A00 50$:
                          1A00 ;
                          1A00 ; Now we are convinced that we really want to try to do this.  First, disable all
                          1A00 ; open files by clearing several fields in the stat blocks.  We do not zero the
                          1A00 ; entire block in case we need to reopen the file later.  We also disable and
                          1A00 ; deallocate the file open cache.
                          1A00 ;
 50     2408'CF     DE    1A00        MOVAL   W^CONEOF_DISABLE_STAT,R0           ; First stat block to disable
     51     0D'     D0    1A05        MOVL    S^#CONEOF_DISABLE_CNT,R1           ; Number of blocks to disable
         10 A0      D4    1A08 60$:   CLRL    STAT_L_MAP(R0)                     ; Clear rtrv pointer address
     50     18      C0    1A0B        ADDL    #STAT_C_SIZE,R0                    ; Point to next stat block
         F7 51      F5    1A0E        SOBGTR  R1,60$                             ; Do all of them
 50     0000'CF     7D    1A11        MOVQ    W^BOO$GQ_FILCACHE,R0               ; R0 = size, R1 = pool address
            09      13    1A16        BEQL    70$                                ; Branch if no cache enabled
        0000'CF     7C    1A18        CLRQ    W^BOO$GQ_FILCACHE                  ; No FIL$OPENFILE cache
 0004'CF    50      C0    1A1C        ADDL    R0,W^BOO$GQ_INILOA+4               ; Deallocate this with INILOA
        2165'CF     7C    1A21 70$:   CLRQ    W^FIL$GQ_CACHE                     ; Blast the other file cache too
                          1A25 ;
                          1A25 ; Zero mapping information in the boot control block so that references to these
                          1A25 ; files will be trapped.  We do not zero the BOO$L_SYS_MAP, since INIT will turn
                          1A25 ; it into a WCB for the system.  Note that we also have to check these fields
                          1A25 ; near the label MOVBOODRIVER, where the addresses are adjusted.
                          1A25 ;
 50     23C4'CF     D0    1A25        MOVL    W^BOOTCB,R0                        ; Address of boot control block
         04 A0      D4    1A2A        CLRL    BOO$L_PARAM_MAP(R0)                ; Clear the parameter area map
         18 A0      D4    1A2D        CLRL    BOO$L_DMP_VBN(R0)                  ; Clear dump file info
         1C A0      D4    1A30        CLRL    BOO$L_DMP_SIZE(R0)
         20 A0      D4    1A33        CLRL    BOO$L_DMP_MAP(R0)
         24 A0      D4    1A36        CLRL    BOO$L_BUG_MAP(R0)                  ; Clear the bugcheck area map
                          1A39 ;
                          1A39 ; Now, ask the human to switch console volumes.  Tell him to remove the old volume (
                          1A39 ; rewind the volume at the same time).  Wait for a "Y" that says that he is ready, a
                          1A39 ; tell him that we are resuming on the new volume.
                          1A39 ;
        2296'CF     9F    1A39        PUSHAB  W^CONEOF_REMOVE2                   ; Second part of remove message
        2277'CF     9F    1A3D        PUSHAB  W^CONEOF_REMOVE1                   ; First part of remove message
 1B50'CF    02      FB    1A41        CALLS   #2,W^CONEOF_MESSAGE                ; Print the message that we are goin
        22F6'CF     DF    1A46 75$:   PUSHAL  W^CONEOF_BUFFER                    ; We must have a buffer
            04      DD    1A4A        PUSHL   #4                                 ; Even if only one longword
        222E'CF     9F    1A4C        PUSHAB  W^CONEOF_PROMPT                    ; And the prompt
 0000'CF    03      FB    1A50        CALLS   #3,W^BOO$READPROMPT                ; Type the string
 22F7'CF    20      8A    1A55        BICB    #32,W^CONEOF_BUFFER+1              ; Force to upper case (ASCIC string
 59 8F  22F7'CF     91    1A5A        CMPB    W^CONEOF_BUFFER+1,#^A"Y"           ; He has to say Yes
            E4      12    1A60        BNEQ    75$                                ; Loop until we see the Yes
        22DA'CF     9F    1A62        PUSHAB  W^CONEOF_RESUME2                   ; Second part of resume message
        22B4'CF     9F    1A66        PUSHAB  W^CONEOF_RESUME1                   ; First part of resume message
 1B50'CF    02      FB    1A6A        CALLS   #2,W^CONEOF_MESSAGE                ; Print the message that we are goin
                          1A6F ;
                          1A6F ; Reopen SYS.EXE on the next volume.  First save the current stat block and rtrv buf
                          1A6F ; on the stack, so that we can adjust things so that it conceals the fact
                          1A6F ; that SYS.EXE is in two pieces.
```

```
                          1A6F  ;
          0FFC 8F    BB   1A6F          PUSHR   #R2_R11                         ; Save registers
            5E 20    C2   1A73          SUBL2   #STAT_C_SIZE+8,SP               ; Allocate the space
    6E   23F0'CF 18  28   1A76          MOVC3   #STAT_C_SIZE,W^SYS_STAT,(SP)    ; Move the stat block to the stack
    63   23B4'CF     7D   1A7C          MOVQ    W^RTRV_BUF_DSC,(R3)             ; Save the current retrieval buffer
                          1A81  ;
                          1A81  ; Everything has been saved, now try to open the next piece of SYS.EXE
                          1A81  ;
       52   23F0'CF  9E   1A81          MOVAB   W^SYS_STAT,R2                   ; Get address of stat block
              FAFB   30   1A86          BSBW    IMAGE_OPEN                      ; Attempt to open the image
             30 50   E8   1A89          BLBS    R0,80$
                          1A8C          MSG     <-F-Unable to open SYS.EXE on the new volume>
                     00   1ABB          HALT
                          1ABC  80$:
                          1ABC  ;
                          1ABC  ; File is now open, fix up the stat block and retrieval pointers
                          1ABC  ;
       58   2400'CF  D0   1ABC          MOVL    W^SYS_STAT+STAT_L_MAP,R8        ; R8 -> the new map
  23F0'CF   6E 18    28   1AC1          MOVC3   #STAT_C_SIZE,(SP),W^SYS_STAT    ; Restore the original stat block
       57   2400'CF  D0   1AC7          MOVL    W^SYS_STAT+STAT_L_MAP,R7        ; R7 -> the original map
     23B4'CF   83    7D   1ACC          MOVQ    (R3)+,W^RTRV_BUF_DSC            ; Restore the original retrieval buf
                          1AD1  ;
                          1AD1  ; R7 -> the original stat block, R8 -> the new block.  We need to create a dummy rtr
                          1AD1  ; pointer which maps all of the blocks of the previous mapping list.  We then append
                          1AD1  ; our new pointers to the list.  The modified new list is stored over the original
                          1AD1  ; list.  Space for several additional map pointers was allocated with the original,
                          1AD1  ; make sure that the next piece isn't too fragmented.
                          1AD1  ;
    50   68   FD 8F  78   1AD1          ASHL    #-3,(R8),R0                     ; Count of new pointers
     2398'CF   50    D1   1AD6          CMPL    R0,W^CONEOF_MAX_PTR             ; How many have we allocated?
               31    15   1ADB          BLEQ    90$                             ; Enough room
                          1ADD          MSG     <-F-Continuation of SYS.EXE is too fragmented>
                     00   1B0D          HALT
                          1B0E  90$:
                          1B0E  ;
                          1B0E  ; First, get number of blocks mapped by previous map
                          1B0E  ;
    50   67   FD 8F  78   1B0E          ASHL    #-3,(R7),R0                     ; Count of quadword pointers
       51   04 A7    DE   1B13          MOVAL   4(R7),R1                        ; R1 -> first map pointer
               52    D4   1B17          CLRL    R2                              ; R2 will accumulate blocks
          54   81    7D   1B19  100$:   MOVQ    (R1)+,R4                        ; R4 is count, R5 vbn
          52   54    C0   1B1C          ADDL2   R4,R2                           ; Add this count to total
          F7   50    F5   1B1F          SOBGTR  R0,100$                         ; Do every old map pointer
                          1B22  ;
                          1B22  ; Now create the modified mapping list
                          1B22  ;
    87   68    08    C1   1B22          ADDL3   #8,(R8),(R7)+                   ; Use new length plus dummy ptr
         87    52    D0   1B26          MOVL    R2,(R7)+                        ; Move block count of dummy
         87    01    CE   1B29          MNEGL   #1,(R7)+                        ; Set an invalid LBN
    67   04 A8  68   28   1B2C          MOVC3   (R8),4(R8),(R7)                 ; Append the new pointers
                          1B31
            5E 20    C0   1B31          ADDL2   #STAT_C_SIZE+8,SP               ; Deallocate stack storage
          0FFC 8F    BA   1B34          POPR    #R2_R11                         ; Restore everything
                          1B38  ;
                          1B38  ; Almost done, all we have to do is set up the registers so that READ_VIRTUAL can
                          1B38  ; continue with the read.
                          1B38  ;
```

```
         54  2400'CF   D0  1B38          MOVL    W^SYS_STAT+STAT_L_MAP,R4           ; Get the new, modified map address
   52    64     FD 8F  78  1B3D          ASHL    #-3,(R4),R2                        ; Map byte count to quadword ptr cnt
                   52  D7  1B42          DECL    R2                                 ; Ignore the dummy retrieval pointer
         54        0C  C0  1B44          ADDL2   #12,R4                             ; Move over len and dummy to good pt
                           1B47  ;
                           1B47  ; All done with the switch, set a flag in the RPB so that anybody can check at any
                           1B47  ; later time whether the system disk is still there.
                           1B47  ;
   00 00A3 CB      00  E2  1B47          BBSS    #RPB$V_NOSYSDISK,RPB$B_FLAGS(R11),110$  ; Set flag in RPB
                   50  D4  1B4D  110$:   CLRL    R0                                 ; Tell READ_VIRTUAL to resume
                   05  1B4F              RSB
                           1B50
                           1B50  ;+
                           1B50  ; Print a two piece message including the volume label.
                           1B50  ; Inputs:         4(AP)   Address of first part of string
                           1B50  ;                 8(AP)   Address of second part
                           1B50  ;-
                           1B50  CONEOF_MESSAGE:
                 0FFC      1B50          .WORD   R2,R11                             ; Save all registers
         56  2198'CF   D0  1B52          MOVL    W^BOO$GL_FREEMEM,R6                ; Get a buffer for volume header
         55        56  D0  1B57          MOVL    R6,R5                              ; Save a second copy
         58        01  D0  1B5A          MOVL    #1,R8                              ; Read lbn 1
         59   0200 8F  3C  1B5D          MOVZWL  #512,R9                            ; Read a single block
         5A        21  3C  1B62          MOVZWL  #IO$_READLBLK,R10                  ; Move the read function code
                   69  10  1B65          BSBB    QIO_RWLB                           ; Read the block
                2A 50  E8  1B67          BLBS    R0,10$                             ; Branch if successful
                           1B6A          MSG     <-F-Unable to read continuation volume>
                   00      1B93          HALT
                   7E  7C  1B94  10$:    CLRQ    -(SP)                              ; Null buffer is print only
                04 AC  DD  1B96          PUSHL   4(AP)                              ; Pass address of first string
      0000'CF      03  FB  1B99          CALLS   #3,W^BOO$READPROMPT                ; Type the first string
         55   01D8 C5  9E  1B9E          MOVAB   472(R5),R5                         ; Point R5 at the volume label
         50        0C  D0  1BA3          MOVL    #12,R0                             ; Length of volume label
         51      6540  9E  1BA6          MOVAB   (R5)[R0],R1                        ; Point R1 one past the end
         52        71  90  1BAA  20$:    MOVB    -(R1),R2                           ; R2 contains the byte
                   05  13  1BAD          BEQL    30$                                ; Byte is null, skip it
         52        20  91  1BAF          CMPB    #^A" ",R2                          ; Is it a space
                   05  12  1BB2          BNEQ    40$                                ; End of name found, print it
                F3 50  F5  1BB4  30$:    SOBGTR  R0,20$                             ; Do the whole string
                   0C  11  1BB7          BRB     50$                                ; No name, don't bother not printing
                           1BB9
                01 A1  94  1BB9  40$:    CLRB    1(R1)                              ; Put a null after the string
                   7E  7C  1BBC          CLRQ    -(SP)                              ; Null buffer is print only
                   55  DD  1BBE          PUSHL   R5                                 ; R5 -> front of volume label
      0000'CF      03  FB  1BC0          CALLS   #3,W^BOO$READPROMPT                ; Type the first string
                   7E  7C  1BC5  50$:    CLRQ    -(SP)                              ; Null buffer is print only
                08 AC  DD  1BC7          PUSHL   8(AP)                              ; Finish the message
      0000'CF      03  FB  1BCA          CALLS   #3,W^BOO$READPROMPT                ; Type the second string
                   04  1BCF              RET
```

```
                              1BD0       .SBTTL  QIO_RWLB - Read or Write Logical Block
                              1BD0 ;++
                              1BD0 ; Functional Description:
                              1BD0 ;       This routine reads/writes the specified logical block numbers
                              1BD0 ;       from/to the boot disk.
                              1BD0 ;
                              1BD0 ; Calling Sequence:
                              1BD0 ;       BSBW    QIO_RWLB
                              1BD0 ;
                              1BD0 ; Inputs:
                              1BD0 ;       R6  = Buffer address (updated)
                              1BD0 ;       R8  = Logical block number (updated)
                              1BD0 ;       R9  = Byte count to transfer (up to 31 bits)
                              1BD0 ;       R10 = #IO$_READLBLK or #IO$_WRITELBLK
                              1BD0 ;       R11 = RPB address
                              1BD0 ; Outputs:
                              1BD0 ;       R0 = Status
                              1BD0 ;       R1,R6-R9 altered
                              1BD0 ;       All other registers preserved
                              1BD0 ;--
                              1BD0 QIO_RWLB:
        57   FE00 8F    3C    1BD0 10$:   MOVZWL  #IOSIZE*512,R7              ; Assume maximum transfer
               59 57    D1    1BD5        CMPL    R7,R9                       ; Minimize with file size
                  03    15    1BD8        BLEQ    20$                         ; Smaller than remaining file size
               57 59    D0    1BDA        MOVL    R9,R7                       ; Set to remaining file size
                  5B    DD    1BDD 20$:   PUSHL   R11                         ; Base of RPB
   7E     56    01 1F   EF    1BDF        EXTZV   #VA$V_SYSTEM,#1,R6,-(SP)    ; Set mode to virtual if system space
             05C0 8F    BB    1BE4        PUSHR   #^M<R6,R7,R8,R10>           ; R/W, LBN, size in bytes, buffer adr
           50   34 AB   D0    1BE8        MOVL    RPB$L_IOVEC(R11),R0         ; GET POINTER TO I/O VECTOR
          00 B040       06 FB 1BEC        CALLS   #6,@BQO$L_QIO(R0)[R0]       ; Perform read
                              1BF1 ;
                              1BF1 ; R0 = status of transfer, if successful, see if there is any more to do.
                              1BF1 ;
               10 50    E9    1BF1        BLBC    R0,30$                      ; Branch if error
     51    57   F7 8F   78    1BF4        ASHL    #-9,R7,R1                   ; Block count
               58 51    C0    1BF9        ADDL    R1,R8                       ; Starting LBN for next piece
               56 57    C0    1BFC        ADDL    R7,R6                       ; Starting Buf Adr for next piece
               59 57    C2    1BFF        SUBL    R7,R9                       ; Count bytes tranferred
                  CC    14    1C02        BGTR    10$                         ; Branch if another transfer to do
                        05    1C04 30$:   RSB
```

```
                                1C05          .SBTTL  BOO$USEFILE - Use parameter file
                                1C05  ;++
                                1C05  ; Functional description:
                                1C05  ;       BOO$USEFILE reads the specified file in response to the USE
                                1C05  ;       command and merges all of the values specified in that file into
                                1C05  ;       the working copy of the parameter values.  This is accomplished
                                1C05  ;       by looking up each value specified and merging the associated
                                1C05  ;       value.
                                1C05  ;
                                1C05  ; Calling sequence:
                                1C05  ;       CALLG   arglist,BOO$USEFILE
                                1C05  ;
                                1C05  ; Input Parameters:
                                1C05  ;       TPA$L_TOKENCNT(AP) - Length of file name string
                                1C05  ;       TPA$L_TOKENPTR(AP) - Address fo file name string
                                1C05  ; Output Parameters:
                                1C05  ;       R0 - Completion status code
                                1C05  ;
                                1C05  ;--
                                1C05  BOO$USEFILE::
                         03FC   1C05          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>     ; Entry mask
                                1C07
            00000000'8F    E2   1C07          BBSS    #EXE$V_WRITESYSPARAMS,-  ; Use a file => write current needed
         00 00000000'GF         1C0D                  G^EXE$GL_DYNAMIC_FLAGS,1$;
                                1C13  1$:
          57      10 AC    9E   1C13          MOVAB   TPA$L_TOKENCNT(AP),R7   ; Set address of file name descriptor
                   FA62    30   1C17          BSBW    BOO$FILOPEN             ; Open specified file
                  04 50    E8   1C1A          BLBS    R0,20$                  ; Continue if success
             50      01    3C   1C1D  10$:    MOVZWL  #1,R0                   ; Force success
                           04   1C20          RET                             ;
          56    2198'CF    D0   1C21  20$:    MOVL    W^BOO$GL_FREEMEM,R6     ; Set address of parameter buffer
                   FC48    30   1C26          BSBW    BOO$READFILE            ; Read file content into parameter buffer
                  F1 50    E9   1C29          BLBC    R0,10$                  ; Exit if error
      58    2198'CF    20  C1   1C2C          ADDL3   #32,W^BOO$GL_FREEMEM,R8 ; Init pointer to parameter buffer
00000000'EF 2198'DF    20  28   1C32          MOVC3   #32,@W^BOO$GL_FREEMEM,EXE$GT_STARTUP    ; Set name of startup file
                2390'CF    D4   1C3C          CLRL    W^VALID_PAR_FILE        ; Initialize valid parameter file flag
                     68    D5   1C40  30$:    TSTL    (R8)                    ; Check for end of list
                     50    13   1C42          BEQL    DONE                    ; Branch if yes
           10 AC     68    9A   1C44          MOVZBL  (R8),TPA$L_TOKENCNT(AP) ; Set token count for search
       14 AC     01 A8     9E   1C48          MOVAB   1(R8),TPA$L_TOKENPTR(AP); And address of string
             58      10    C0   1C4D          ADDL    #16,R8                  ; Advance to value
           1C AC     88    D0   1C50          MOVL    (R8)+,TPA$L_NUMBER(AP)  ; Set number
           0000'CF   6C    FA   1C54          CALLG   (AP),W^BOO$SEARCH       ; Search for parameter
                  E4 50    E9   1C59          BLBC    R0,30$                  ; Next parameter if not found
           2390'CF   01    D0   1C5C          MOVL    #1,W^VALID_PAR_FILE     ; Indicate valid parameter file
             54   20 AC    D0   1C61          MOVL    TPA$L_PARAM(AP),R4      ; Get a pointer to the parameter descripttor
          22 10 A4   10    E1   1C65          BBC     #PRM$V_ASCII,PRM$L_FLAGS(R4),40$; Branch if not an ascii parameter
           14 AC     78    DE   1C6A          MOVAL   -(R8),TPA$L_TOKENPTR(AP); Get a pointer to the parameter value
             50   14 A4    9A   1C6E          MOVZBL  PRM$B_SIZE(R4),R0       ; Get parameter size in bits
       50    50   FD 8F    78   1C72          ASHL    #-3,R0,R0               ; Set parameter size
           10 AC     50    9A   1C77          MOVZBL  R0,TPA$L_TOKENCNT(AP)   ;
             50      03    C0   1C7B          ADDL2   #3,R0                   ; Round size up to the next longword
             50      03    CA   1C7E          BICL2   #3,R0                   ;
             58      50    C0   1C81          ADDL2   R0,R8                   ; Advance past value
           0000'CF   6C    FA   1C84          CALLG   (AP),W^BOO$SETASCII     ; Set the value of the parameter
                   FFB4    31   1C89          BRW     30$                     ; Continue with the next parameter
           0000'CF   6C    FA   1C8C  40$:    CALLG   (AP),W^BOO$SETVALUE     ; Set value of parameter
```

```
          FFAC   31  1C91  3185          BRW     30$                      ; Continue with next parameter
     1B 2390'CF  E8  1C94  3186 DONE:    BLBS    W^VALID_PAR_FILE,10$     ; If LBS, valid parameter file
                     1C99  3187          MSG     <-E-Not a parameter file>
        50    01 3C  1CB4  3188 10$:     MOVZWL  #1,R0                    ; Return success
                 04  1CB7  3189          RET                              ;
```

```
                             1CB8          .SBTTL  GETCONLOC_780 - Routine to read console information location
                             1CB8  ;++
                             1CB8  ; Functional Description:
                             1CB8  ;         GETCONLOC_780 is used to access the locations in 11/780 console
                             1CB8  ;         memory containing values such as WCS and FPLA version numbers.
                             1CB8  ;
                             1CB8  ; Input Parameters:
                             1CB8  ;         R1 - Location code
                             1CB8  ;
                             1CB8  ; Output Parameters:
                             1CB8  ;         R0 - Value contained in console cell
                             1CB8  ;--
                             1CB8  GETCONLOC_780:
      51   0300 C1     9E    1CB8          MOVAB   ^X300(R1),R1                    ; Set code to read console memory
           50    22    DB    1CBD  10$:    MFPR    #PR$_TXCS,R0                    ; Get transmit status register
        F9 50    07    E1    1CC0          BBC     #7,R0,10$                       ; Wait for done
           23    51    DA    1CC4          MTPR    R1,#PR$_TXDB                    ; Request data from console
           50    22    DB    1CC7  20$:    MFPR    #PR$_TXCS,R0                    ; Read transmit status register
        F9 50    07    E1    1CCA          BBC     #7,R0,20$                       ; Wait for done
           50    20    DB    1CCE  30$:    MFPR    #PR$_RXCS,R0                    ; Get receiver status
        F9 50    07    E1    1CD1          BBC     #7,R0,30$                       ; And wait for done
           50    21    DB    1CD5          MFPR    #PR$_RXDB,R0                    ; Now read data value
 03   50   04    08    ED    1CD8          CMPZV   #8,#4,R0,#3                     ; Is this a valid response?
                             1CDD  ;       BNEQ    10$                             ; No, try again
                 01          1CDD          NOP                                     ;****** TEMP
                 01          1CDE          NOP                                     ;****** TEMP
      50   50    9A          1CDF          MOVZBL  R0,R0                           ; Zero extend data
                 05          1CE2          RSB                                     ;
                             1CE3  ;
                             1CE3  ; Table of microcode revision levels for the 11/780.
                             1CE3  ;
                             1CE3  VERSVECT_780:                                   ; Vector of version offsets
                 6D          1CE3          .BYTE   FPLA_VLOC_780                   ; FPLA Version offset
                 6A          1CE4          .BYTE   PCS_VLOC_780                    ; PCS Version offset
                 6C          1CE5          .BYTE   WCSS_VLOC_780                   ; WCS Secondary version offset
                 6B          1CE6          .BYTE   WCSP_VLOC_780                   ; WCS Primary version offset
           00000004          1CE7  VERSVEC780LEN=.-VERSVECT_780
                 00          1CE7          .BYTE   0                               ; End of list
                             1CE8  VERSNUM_780:                                    ; Vector of required version numbers
                 0C          1CE8          .BYTE   FPLA_780                        ; FPLA minimum
                 01          1CE9          .BYTE   PCS_780                         ; PCS minimum
                 12          1CEA          .BYTE   WCSS_780                        ; WCS Secondary minimum
                 0C          1CEB          .BYTE   WCSP_780                        ; WCS Primary minimum
                             1CEC
                             1CEC  ;
                             1CEC  ; Table of microcode revision levels for the 11/785.
                             1CEC  ;
                             1CEC  VERSVECT_785:                                   ; Vector of version offsets
                 6A          1CEC          .BYTE   PCS_VLOC_785                    ; PCS Version offset
                 6B          1CED          .BYTE   MTCH_VLOC_785                   ; PCS/WCS match version offset
                 6D          1CEE          .BYTE   WCSS_VLOC_785                   ; WCS secondary version offset
                 6C          1CEF          .BYTE   WCSP_VLOC_785                   ; WCS primary version offset
           00000004          1CF0  VERSVEC785LEN=.-VERSVECT_785
                 00          1CF0          .BYTE   0                               ; End of list
                             1CF1  VERSNUM_785:                                    ; Vector of required version numbers
                 04          1CF1          .BYTE   PCS_785                         ; PCS minimum
                 04          1CF2          .BYTE   MTCH_785                        ; PCS/WCS match
```

```
00  1CF3  3248          .BYTE   WCSS_785                        ; WCS secondary minimum
01  1CF4  3249          .BYTE   WCSP_785                        ; WCS primary minimum
    1CF5  3250  ;
    1CF5  3251  ; The code paragraph CHKVERS_780 assumes that both the 780 and the 785 have
    1CF5  3252  ; 4 console locations to check.
    1CF5  3253  ;
    1CF5  3254          ASSUME  VERSVEC780LEN EQ VERSVEC785LEN
    1CF5  3255          ASSUME  VERSVEC780LEN EQ 4
    1CF5  3256
```

```
                        1CF5          .SBTTL  GETCONLOC_790
                        1CF5  ;++
                        1CF5  ; Functional Description:
                        1CF5  ;       GETCONLOC_790 is used to read data stored in console memory,
                        1CF5  ;       such as microcode revision levels.
                        1CF5  ;
                        1CF5  ; Inputs:
                        1CF5  ;       R1 = code for console data requested
                        1CF5  ;       R2 = # of bytes of data expected
                        1CF5  ;       R3 = address of buffer to store requested data in
                        1CF5  ;
                        1CF5  ; Outputs:
                        1CF5  ;       Data is stored in the buffer.
                        1CF5  ;       Input registers destroyed.
                        1CF5  ;--
                        1CF5
                        1CF5  GETCONLOC_790:
              7E    D4  1CF5          CLRL    -(SP)                   ; Longword of temporary storage.
22  00088000  8F    DA  1CF7  1$:     MTPR    #^X88000,#PR$_TXCS      ; Disable console terminal transmits
                        1CFE                                          ; and request logical console.
         6E   22    DB  1CFE  10$:    MFPR    #PR$_TXCS,(SP)          ; Get transmit status.
      F9 6E   07    E1  1D01          BBC     #7,(SP),10$             ; Loop until ready bit is set.
      03   01 AE    91  1D05          CMPB    1(SP),#3                ; ID = logical console data?
              EC    12  1D09          BNEQ    1$                      ; If not, try again.
         23   51    DA  1D0B          MTPR    R1,#PR$_TXDB            ; Send request code to console.
                        1D0E
         6E   20    DB  1D0E  20$:    MFPR    #PR$_RXCS,(SP)          ; Get receiver status.
      F9 6E   07    E1  1D11          BBC     #7,(SP),20$             ; Loop until done bit is set.
         6E   21    DB  1D15          MFPR    #PR$_RXDB,(SP)          ; Get received data.
      03   01 AE    91  1D18          CMPB    1(SP),#3                ; ID = logical console data?
              F0    12  1D1C          BNEQ    20$                     ; If not, throw data away and try again.
                        1D1E  ;       CMPB    (SP),R1                 ; Code = requested data returned?
                        1D1E  ;       BNEQ    CONSOLE_ERROR           ; No recovery from protocol error.
         83   6E    90  1D1E          MOVB    (SP),(R3)+              ; Put data byte in user's buffer.
         EA   52    F5  1D21          SOBGTR  R2,20$                  ; Branch back to get another byte.
                        1D24
              8E    D5  1D24          TSTL    (SP)+                   ; Get rid of temporary buffer.
22  00018000  8F    DA  1D26          MTPR    #^X18000,#PR$_TXCS      ; Disable logical console and
                        1D2D                                          ; enable local terminal line.
                    05  1D2D          RSB
                        1D2E
```

```
                                 1D2E          .SBTTL  MOVEMAPPED - MOVE FROM PHYSICAL TO MAPPED MEMORY
                                 1D2E  ;++
                                 1D2E  ; Functional Description:
                                 1D2E  ;       MOVEMAPPED moves the specified number of bytes from the
                                 1D2E  ;       given physical address to a specified system virtual address
                                 1D2E  ;
                                 1D2E  ;       R1 = physical address to move data from
                                 1D2E  ;       R6 = number of bytes to move
                                 1D2E  ;       R7 = system virtual address to move them to
                                 1D2E  ;
                                 1D2E  ; Output Parameters:
                                 1D2E  ;       R0 - R7 altered
                                 1D2E  ;
                                 1D2E  ;--
                                 1D2E  MOVEMAPPED:
                 7E   58   7D    1D2E          MOVQ    R8,-(SP)
       58   57   09   00   EF    1D31          EXTZV   #VA$V_BYTE,#VA$S_BYTE,R7,R8 ; STARTING BYTE OFFSET
       57   57   15   09   EF    1D36          EXTZV   #VA$V_VPN,#VA$S_VPN,R7,R7 ; GET VIRTUAL PAGE NUMBER
     57   00000000'FF47   DE     1D3B          MOVAL   @MMG$GL_SBR[R7],R7        ; GET SPT ENTRY ADDRESS
                      50   D4    1D43          CLRL    R0                        ; BACKGROUND THIS FOR INSV IN LOOP
                                 1D45                                            ; MOVC WILL ZERO IT EACH TIME
                      02   11    1D45          BRB     20$
                      58   D4    1D47  10$:    CLRL    R8                        ; NO BYTE OFFSET
       50   15   09   87   F0    1D49  20$:    INSV    (R7)+,#VA$V_VPN,#PTE$S_PFN,R0 ; DESTINATION PAGE ADDRESS
  59   00000200  8F   58   C3    1D4E          SUBL3   R8,#512,R9                ; NO. OF BYTES TO MOVE
                 56   59   D1    1D56          CMPL    R9,R6                     ; DON'T MOVE MORE THAN IS LEFT
                 03   15         1D59          BLEQ    30$                       ; BRANCH IF MORE TO DO
                 59   56   D0    1D5B          MOVL    R6,R9                     ; OTHERWISE USE WHAT IS LEFT
          6048   61   59   28    1D5E  30$:    MOVC3   R9,(R1),(R0)[R8]          ; MOVE THE NEXT PIECE
                 56   59   C2    1D63          SUBL    R9,R6                     ; COUNT BYTES MOVED
                      DF   14    1D66          BGTR    10$                       ; BRANCH IF MORE TO MOVE
                 58   8E   7D    1D68          MOVQ    (SP)+,R8                  ; RESTORE SAVED REGISTERS
                           05    1D6B          RSB                               ; OTHERWISE RETURN
```

```
                                1D6C          .SBTTL  ALLOC_POOL - Allocate Pool From the top
                                1D6C  ;++
                                1D6C  ; Functional Description:
                                1D6C  ;       This routine allocates the requested amount of pool from
                                1D6C  ;       the current top of non-paged pool and returns its address
                                1D6C  ;
                                1D6C  ; Calling Sequence:
                                1D6C  ;       BSBW    ALLOC_POOL
                                1D6C  ;
                                1D6C  ; Inputs:
                                1D6C  ;       R0                  = Address of remaining size of pool
                                1D6C  ;       STAT_L_BYTECNT(R2) = Number of bytes to allocate
                                1D6C  ;       STAT_L_SYSVA(R2)   = Base relative address of location to
                                1D6C  ;                             store allocated address
                                1D6C  ;       STAT_L_NAME(R2)    = Base relative adr of ASCIC name of file for error msg
                                1D6C  ;
                                1D6C  ; Outputs:
                                1D6C  ;       R0,R2 preserved
                                1D6C  ;       R1 = rounded up byte count allocated
                                1D6C  ;       STAT_L_BYTECNT(R2) = rounded up byte count allocated
                                1D6C  ;       Returns to caller only if pool is successfully allocated
                                1D6C  ;       If there is not enough room a diagnostic is issued
                                1D6C  ;       and a HALT is executed.
                                1D6C  ;--
                                1D6C
                                1D6C  ALLOC_POOL:
     51   04 A2     0F     C1   1D6C          ADDL3   #^XF,STAT_L_BYTECNT(R2),R1 ; Round desired size up
               51   0F     CA   1D71          BICL    #^XF,R1                    ; to quad boundary
                    16     13   1D74          BEQL    10$                        ; None needed
          04 A2     51     D0   1D76          MOVL    R1,STAT_L_BYTECNT(R2)      ; Save allocated size
               60   51     C2   1D7A          SUBL    R1,(R0)                    ; Allocate the pool
                    0E     19   1D7D          BLSS    20$                        ; Branch if no pool left
  7E   08 A2   23A0'CF     C1   1D7F          ADDL3   W^SYSBOOT_BASE,STAT_L_SYSVA(R2),-(SP) ; Adr to store pool adr
    9E  0000'CF     60     C1   1D86          ADDL3   (R0),W^MMG$GL_NPAGEDYN,@(SP)+ ; Set adr allocated
                           05   1D8C  10$:    RSB
                                1D8D
                                1D8D  20$:    MSG     <-F-Not enough non-paged pool to map >
  51   0C A2   23A0'CF     C1   1DB5          ADDL3   W^SYSBOOT_BASE,STAT_L_NAME(R2),R1 ; Adr of file name (ASCIC)
                  0012     30   1DBC          BSBW    BOO$TYPE_ASCIC             ; Type the name
                           00   1DBF          HALT                               ; ***** Fatal Error *****
```

```
                            1DC0       .SBTTL  BOO$FACMSG - Output facility error message
                            1DC0 ;++
                            1DC0 ; Functional Description:
                            1DC0 ;        BOO$FACMSG outputs an error message preceded by a new line
                            1DC0 ;        and facility name string.
                            1DC0 ;
                            1DC0 ; Calling Sequence:
                            1DC0 ;        BSBW    BOO$FACMSG
                            1DC0 ;        .ASCIZ  message-text
                            1DC0 ;--
                            1DC0 BOO$FACMSG::                              ;
                            1DC0 ERROUT:
                 E23D'  30  1DC0          BSBW    BOO$MSGOUT               ; Output prefix message
                    0A  0D  1DC3          .ASCII  <CR><LF>                 ; With a new line first
00 54 4F 4F 42 53 59 53 25  1DC5          .ASCIZ  /%SYSBOOT/               ; Then facility name
                 E22F'  31  1DCE          BRW     BOO$MSGOUT               ; And finally the meat of the message
                            1DD1
```

```
                        1DD1  3393          .SBTTL  BOO$TYPE_ASCIC - Type a counted ASCII string
                        1DD1  3394  ;++
                        1DD1  3395  ; Functional Description:
                        1DD1  3396  ;       This routine accepts a string descriptor and types the message
                        1DD1  3397  ;       on the console terminal
                        1DD1  3398  ;
                        1DD1  3399  ; Calling Sequence:
                        1DD1  3400  ;       BSBW    BOO$TYPE_ASCIC
                        1DD1  3401  ;
                        1DD1  3402  ; Inputs:
                        1DD1  3403  ;       R1 = Address of ASCIC string
                        1DD1  3404  ;
                        1DD1  3405  ; Outputs:
                        1DD1  3406  ;       R0,R1 altered
                        1DD1  3407  ;       All other registers preserved
                        1DD1  3408  ;--
                        1DD1  3409
                        1DD1  3410  BOO$TYPE_ASCIC:
      50    81     9A   1DD1  3411          MOVZBL  (R1)+,R0                ; R0 = Size, R1 = Address of string
            3C     BB   1DD4  3412          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
 53   2198'CF     D0   1DD6  3413          MOVL    W^BOO$GL_FREEMEM,R3     ; Address of scratch storage
            7E     7C   1DDB  3414          CLRQ    -(SP)                   ; No read for PROMPTREAD call
            53     DD   1DDD  3415          PUSHL   R3                      ; Address of ASCIZ string
 63   61    50     28   1DDF  3416          MOVC3   R0,(R1),(R3)            ; Move the string to scratch storage
            83     94   1DE3  3417          CLRB    (R3)+                   ; and make it ASCIZ
 0000'CF    03     FB   1DE5  3418          CALLS   #3,W^BOO$READPROMPT     ; Type the string
            3C     BA   1DEA  3419          POPR    #^M<R2,R3,R4,R5>        ; Recover saved registers
                   05   1DEC  3420          RSB
```

```
                  1DED          .SBTTL  Unexpected Machine Check Handler
                  1DED  ;
                  1DED  ;       Unexpected Machine Check Handler
                  1DED  ;
                  1DED          .ALIGN  LONG                    ; Exception handlers must be longword aligne
                  1DF0          .IF     DF,DEBUG                ;
                  1DF0  EXE$ACVIOLAT::                          ; Access violation vector
                  1DF0  EXE$BREAK::                             ; Breakpoint vector
                  1DF0  EXE$ROPRAND::                           ; Reserved operand vector
                  1DF0  EXE$TBIT::                              ; TBIT vector
                  1DF0  MMG$PAGEFAULT::                         ; Pagefault exception vector
                  1DF0          .ENDC                           ;
                  1DF0  BOOT_FAULT:                             ;
                  1DF0          MSG     <-F-Unexpected Exception>; Output error message
                  1E0B
                  1E0B          .ALIGN  LONG                    ;
                  1E0C  UNEXP_MCHK:                             ;
                  1E0C          MSG     <-F-Unexpected Machine Check>; Output error message
                  1E2B
                  1E2B          .IF     DF,DEBUG                ;
                  1E2B  INI$RDONLY::                            ; Dummy change protection routines
                  1E2B  INI$WRITABLE::                          ;
                  1E2B  SYSL$CLRSBIA::                          ; Dummy routine to clear SBIA errors
              05  1E2B          RSB                             ;
                  1E2C  EXE$GL_FLAGS::                          ; Dummy
                  1E2C  XDS$GT_WORD_PFN::
        00000000  1E2C          .LONG   0                       ;
                  1E30          .ENDC                           ;
                  1E30  ;
                  1E30  ; DUMMY SYS$FAO LINKAGE TO EXE$FAO
                  1E30  ;
                  1E30  SYS$FAO::                               ;
            0FFC  1E30          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>     ; ALL REGISTERS
       E1CD'  31  1E32          BRW     EXE$FAO+2               ; ENTER FAO
                  1E35
```

```
                                          1E35        .SBTTL  BOO$GIVEHELP - Print Help information
                                          1E35   ;
                                          1E35   ;       Print Help Information
                                          1E35   ;
                                   003C   1E35   BOO$GIVEHELP::  .WORD   ^M<R2,R3,R4,R5>  ;
                              7E     7C   1E37          CLRQ    -(SP)                     ; Null input buffer
                          45'AF      9F   1E39          PUSHAB  B^HELPMSG                 ; Set address of help message
                   0000'CF    03     FB   1E3C          CALLS   #3,W^BOO$READPROMPT       ; Write help message
                          50    01   D0   1E41          MOVL    #1,R0                     ; Return success
                                     04   1E44          RET                               ;
                                          1E45
                                          1E45
                                          1E45   HELPMSG:                                 ; ASCIZ Help message string
4F 4F 42 53 59 53 20 72 6F 6A 61 4D   1E45          .ASCII  /Major SYSBOOT Commands are:/<CR><LF><LF>
61 20 73 64 6E 61 6D 6D 6F 43 20 54   1E51
                  0A 0A 0D 3A 65 72   1E5D
20 2D 20 45 55 4E 49 54 4E 4F 43 09   1E63          .ASCII  /       CONTINUE - Continue with boot process/<CR><LF>
74 69 77 20 65 75 6E 69 74 6E 6F 43   1E6F
65 63 6F 72 70 20 74 6F 6F 62 20 68   1E7B
                        0A 0D 73 73   1E87
6E 6F 43 20 2D 20 09 54 49 58 45 09   1E8B          .ASCII  /       EXIT     - Continue with boot process/<CR><LF>
62 20 68 74 69 77 20 65 75 6E 69 74   1E97
0D 73 73 65 63 6F 72 70 20 74 6F 6F   1EA3
                                  0A  1EAF
20 74 65 53 20 2D 20 09 54 45 53 09   1EB0          .ASCII  /       SET      - Set parameter value/<CR><LF>
61 76 20 72 65 74 65 6D 61 72 61 70   1EBC
                     0A 0D 65 75 6C   1EC8
72 61 70 20 54 45 53 20 20 20 20 09   1ECD          .ASCII  /           SET parameter_name value/<CR><LF>
20 65 6D 61 6E 5F 72 65 74 65 6D 61   1ED9
               0A 0D 65 75 6C 61 76   1EE5
54 53 2F 20 54 45 53 20 20 20 20 09   1EEC          .ASCII  \           SET /STARTUP file_spec\<CR><LF>
73 5F 65 6C 69 66 20 50 55 54 52 41   1EF8
                  0A 0D 63 65 70      1F04
6F 68 53 20 2D 20 09 57 4F 48 53 09   1F09          .ASCII  /       SHOW     - Show parameter value(s)/<CR><LF>
20 72 65 74 65 6D 61 72 61 70 20 77   1F15
      0A 0D 29 73 28 65 75 6C 61 76   1F21
61 70 20 57 4F 48 53 20 20 20 20 09   1F2B          .ASCII  /           SHOW parameter_name/<CR><LF>
65 6D 61 6E 5F 72 65 74 65 6D 61 72   1F37
                              0A 0D   1F43
68 53 20 2D 09 20 50 43 41 2F 09 09   1F45          .ASCII  \               /ACP     - Show ACP parameters\<CR><LF>
6D 61 72 61 70 20 50 43 41 20 77 6F   1F51
               0A 0D 73 72 65 74 65   1F5D
68 53 20 2D 09 20 4C 4C 41 2F 09 09   1F64          .ASCII  \               /ALL     - Show ALL parameters\<CR><LF>
6D 61 72 61 70 20 4C 4C 41 20 77 6F   1F70
               0A 0D 73 72 65 74 65   1F7C
68 53 20 2D 09 20 4E 45 47 2F 09 09   1F83          .ASCII  \               /GEN     - Show generative parameters\<CR><LF>
76 69 74 61 72 65 6E 65 67 20 77 6F   1F8F
73 72 65 74 65 6D 61 72 61 70 20 65   1F9B
                              0A 0D   1FA7
20 2D 09 20 52 4F 4A 41 4D 2F 09 09   1FA9          .ASCII  \               /MAJOR   - Show MAJOR parameters\<CR><LF>
70 20 52 4F 4A 41 4D 20 77 6F 68 53   1FB5
   0A 0D 73 72 65 74 65 6D 61 72 61   1FC1
20 2D 09 20 53 45 4D 41 4E 2F 09 09   1FCC          .ASCII  \               /NAMES   - Show parameter names\<CR><LF>
74 65 6D 61 72 61 70 20 77 6F 68 53   1FD8
      0A 0D 73 65 6D 61 6E 20 72 65   1FE4
68 53 20 2D 09 20 4C 51 50 2F 09 09   1FEE          .ASCII  \               /PQL     - Show Process Quota List values\<CR><LF>
51 20 73 73 65 63 6F 72 50 20 77 6F   1FFA
```

```
61 76 20 74 73 69 4C 20 61 74 6F 75   2006
                  0A 0D 73 65 75 6C   2012
68 53 20 2D 09 20 53 4D 52 2F 09 09   2018  3485          .ASCII  \           /RMS     - Show RMS parameters\<CR><LF>
6D 61 72 61 70 20 53 4D 52 20 77 6F   2024
               0A 0D 73 72 65 74 65   2030
6F 68 53 20 2D 09 53 43 53 2F 09 09   2037  3486          .ASCII  \           /SCS     - Show SCS parameters\<CR><LF>
65 6D 61 72 61 70 20 53 43 53 20 77   2043
                  0A 0D 73 72 65 74   204F
2D 20 50 55 54 52 41 54 53 2F 09 09   2055  3487          .ASCII  \           /STARTUP - Show Startup command file name\<CR><LF>
75 74 72 61 74 53 20 77 6F 68 53 20   2061
69 66 20 64 6E 61 6D 6D 6F 63 20 70   206D
         0A 0D 65 6D 61 6E 20 65 6C   2079
68 53 20 2D 09 20 53 59 53 2F 09 09   2082  3488          .ASCII  \           /SYS     - Show SYSTEM parameters\<CR><LF>
61 70 20 4D 45 54 53 59 53 20 77 6F   208E
      0A 0D 73 72 65 74 65 6D 61 72   209A
68 53 20 2D 09 20 59 54 54 2F 09 09   20A4  3489          .ASCII  \           /TTY     - Show terminal parameters\<CR><LF>
20 6C 61 6E 69 6D 72 65 74 20 77 6F   20B0
0A 0D 73 72 65 74 65 6D 61 72 61 70   20BC
20 74 65 53 20 2D 20 09 45 53 55 09   20C8  3490          .ASCII  /       USE       - Set parameter file name /<CR><LF>
69 66 20 72 65 74 65 6D 61 72 61 70   20D4
      0A 0D 20 65 6D 61 6E 20 65 6C   20E0
6C 69 66 20 45 53 55 20 20 20 20 09   20EA  3491          .ASCII  \           USE file_spec.PAR \<CR><LF>
0D 20 52 41 50 2E 63 65 70 73 5F 65   20F6
                                  0A   2102
66 20 64 65 76 72 65 73 65 52 09 09   2103  3492          .ASCII  /           Reserved filespecs are:/<CR><LF>
65 72 61 20 73 63 65 70 73 65 6C 69   210F
                           0A 0D 3A   211B
2D 20 54 4C 55 41 46 45 44 09 09 09   211E  3493          .ASCII  /                   DEFAULT - Use permanent defaults/<CR><LF>
65 6E 61 6D 72 65 70 20 65 73 55 20   212A
0D 73 74 6C 75 61 66 65 64 20 74 6E   2136
                                  0A   2142
2D 20 54 4E 45 52 52 55 43 09 09 09   2143  3494          .ASCIZ  /                   CURRENT - Use current values/<CR><LF>
74 6E 65 72 72 75 63 20 65 73 55 20   214F
      00 0A 0D 73 65 75 6C 61 76 20   215B
```

```
                                   2165  3496            .SBTTL  Miscellaneous constants and temps
                                   2165  3497 FIL$GQ_CACHE::
                         0000216D  2165  3498            .BLKQ   1                        ; FIL$OPENFILE cache descriptor
                                   216D  3499 FIL$GT_TOPSYS::
              30 53 59 53 00'      216D  3500            .ASCIC  /SYS0/                   ; Default top level system dir name
                          04       216D
                         00002177  2172  3501            .BLKB   10-<.-FIL$GT_TOPSYS>     ; Fill to 10 bytes
                                   2177  3502 BOO$GT_SYSDUMP:                             ;
50 4D 44 2E 50 4D 55 44 53 59 53 00'  2177  3503         .ASCIC  /SYSDUMP.DMP/            ; Name of dump file
                          0B       2177
                                   2183  3504 BOO$GT_PAGEFILE:                            ;
59 53 2E 45 4C 49 46 45 47 41 50 00'  2183  3505         .ASCIC  /PAGEFILE.SYS/           ; Name of page file
                          53       218F
                          0C       2183
                                   2190  3506
                                   2190  3507 BOO$GT_SYS::                                ;
        45 58 45 2E 53 59 53 00'   2190  3508            .ASCIC  /SYS.EXE/                ; Name of system image
                          07       2190
                                   2198  3509 BOO$GL_FREEMEM::                            ; Base of free memory after boot image
                         00000000  2198  3510            .LONG   0                        ;
                                   219C  3511 BOO$GT_PROMPT::                             ; Prompt string
20 20 3E 54 4F 4F 42 53 59 53 0A 0D  219C  3512          .ASCIZ  <CR><LF>%SYSBOOT>  %     ;
                          00       21A8
                                   21A9  3513 BOO$GL_GPTPGCT::                            ; Count of global page table pages
                         00000000  21A9  3514            .LONG   0                        ;
                                   21AD  3515 BOO$GL_NEXTPFN::                            ; Starting index for PFN scan
                         00000000  21AD  3516            .LONG   0                        ;
                                   21B1  3517 BOO$GL_SPTPAGCT::                           ; Count of pages allocated for SPT
                         00000000  21B1  3518            .LONG   0                        ;
                                   21B5  3519 BOO$GQ_STATBLK::                            ; Statistics block for file open
                00000000 00000000  21B5  3520            .LONG   0,0                      ;
                                   21BD  3521 BOO$GL_SYSPHD::                             ; Physical address of system header
                         00000000  21BD  3522            .LONG   0                        ;
                                   21C1  3523 BOO$GL_SYSPHDPG::                           ; Count of pages for system PHD
                         00000000  21C1  3524            .LONG   0                        ;
58 45 2E 52 45 56 49 52 44 54 54 00'  21C5  3525 TTNAME: .ASCIC  /TTDRIVER.EXE/           ; NAME OF TERMINAL SERVICE CODE
                          45       21D1
                          0C       21C5
                                   21D2  3526
58 45 2E 52 45 56 49 52 44 42 44 00'  21D2  3527 DBNAME: .ASCIC  /DBDRIVER.EXE/           ; Name of RP06 Driver
                          45       21DE
                          0C       21D2
58 45 2E 52 45 56 49 52 44 4D 44 00'  21DF  3528 DMNAME: .ASCIC  /DMDRIVER.EXE/           ; Name of RK07 Driver
                          45       21EB
                          0C       21DF
58 45 2E 52 45 56 49 52 44 52 44 00'  21EC  3529 DRNAME: .ASCIC  /DRDRIVER.EXE/           ; Name of RM03/RP07 Driver
                          45       21F8
                          0C       21EC
58 45 2E 52 45 56 49 52 44 4C 44 00'  21F9  3530 DLNAME: .ASCIC  /DLDRIVER.EXE/           ; Name of RL01/RL02 driver
                          45       2205
                          0C       21F9
58 45 2E 52 45 56 49 52 44 58 44 00'  2206  3531 DXNAME: .ASCIC  /DXDRIVER.EXE/           ; Name of console RX01 driver
                          45       2212
                          0C       2206
58 45 2E 52 45 56 49 52 44 44 44 00'  2213  3532 DDNAME: .ASCIC  /DDDRIVER.EXE/           ; Name of console TU58 driver
                          45       221F
                          0C       2213
```

```
45 2E 52 45 56 49 52 44 4B 4E 55 00' 2220  3533 UNKNAME:.ASCIC  /UNKDRIVER.EXE/          ; Name of Driver for unknown devices
                               45 58 222C
                                  0D 2220
                                     222E  3534
68 74 20 74 72 65 73 6E 49 07 0A 0D  222E  3535 CONEOF_PROMPT:  .ASCII  <CR><LF><BELL>/Insert the next standalone system volume /
64 6E 61 74 73 20 74 78 65 6E 20 65  223A
6D 65 74 73 79 73 20 65 6E 6F 6C 61  2246
         20 65 6D 75 6C 6F 76 20     2252
59 22 20 72 65 74 6E 65 20 64 6E 61  225A  3536                 .ASCIZ  /and enter "YES" when ready: /
61 65 72 20 6E 65 68 77 20 22 53 45  2266
                  00 20 3A 79 64     2272
                                     2277  3537
65 72 20 65 73 61 65 6C 50 07 0A 0D  2277  3538 CONEOF_REMOVE1: .ASCIZ  <CR><LF><BELL>/Please remove the volume "/<BELL>
6C 6F 76 20 65 68 74 20 65 76 6F 6D  2283
         00 07 22 20 65 6D 75        228F
63 20 65 68 74 20 6D 6F 72 66 20 22  2296  3539 CONEOF_REMOVE2: .ASCIZ  /" from the console device./<BELL><CR><LF>
63 69 76 65 64 20 65 6C 6F 73 6E 6F  22A2
               00 0A 0D 07 2E 65     22AE
                                     22B4  3540
6C 20 67 6E 69 6D 75 73 65 52 0A 0D  22B4  3541 CONEOF_RESUME1: .ASCIZ  <CR><LF>/Resuming load operation on volume "/
6F 69 74 61 72 65 70 6F 20 64 61 6F  22C0
20 65 6D 75 6C 6F 76 20 6E 6F 20 6E  22CC
                              00 22  22D8
74 73 20 65 73 61 65 6C 70 20 2C 22  22DA  3542 CONEOF_RESUME2: .ASCIZ  /", please stand by . . ./<CR><LF><LF>
2E 20 2E 20 2E 20 79 62 20 64 6E 61  22E6
                  00 0A 0A 0D        22F2
                                     22F6  3543
                         00000000    22F6  3544 CONEOF_BUFFER:  .LONG   0
                                     22FA  3545 ;
                                     22FA  3546 ; DATA FOR LOADABLE CPU-DEPENDENT CODE:
                                     22FA  3547 ;
                                     22FA  3548
                                     22FA  3549 NAME_SYSLOA:                             ; FILENAME OF LOADABLE
45 2E 58 58 58 41 4F 4C 53 59 53 00' 22FA  3550         .ASCIC  /SYSLOAXXX.EXE/          ;  CPU-DEPENDENT IMAGE
                               45 58 2306
                                  0D 22FA
                         00002301    2308  3551 NAME_XXX=NAME_SYSLOA+1+6                 ; ADDR OF XXX FIELD IN
                                     2308  3552                                          ;  IMAGE FILENAME
                                     2308  3553 MODEL_TABLE:                             ; TABLE OF LONGWDS:  EACH LONGWD
                                     2308  3554                                          ;  HAS 3 CHAR MODEL NUMBER+ 1 SPACE
                                     2308  3555 MODEL_780:                               ; VAX 11/780:
                      20 30 38 37    2308  3556         .ASCII  /780 /
                                     230C  3557 MODEL_750:                               ; VAX 11/750:
                      20 30 35 37    230C  3558         .ASCII  /750 /
                                     2310  3559 MODEL_730:                               ; VAX 11/730:
                      20 30 33 37    2310  3560         .ASCII  /730 /
                                     2314  3561 MODEL_790:                               ; VAX 11/790:
                      20 30 39 37    2314  3562         .ASCII  /790 /
                                     2318  3563 MODEL_5:                                 ; cpu type = 5
                      20 35 35 35    2318  3564         .ASCII  /555 /                   ;
                                     231C  3565 MODEL_6:                                 ; cpu type = 6
                      20 36 36 36    231C  3566         .ASCII  /666 /                   ;
                                     2320  3567 MODEL_UV1:                               ; MicroVAX I
                      20 31 56 55    2320  3568         .ASCII  /UV1 /                   ;
                                     2324  3569 MODEL_UV2:                               ; MicroVAX II
                      20 32 56 55    2324  3570         .ASCII  /UV2 /                   ;
                                     2328  3571 NAME_SCSLOA:                             ; Filename of loadable
```

```
   45 58 45 2E 41 4F 4C 53 43 53 00' 2328  3572          .ASCIC  /SCSLOA.EXE/          ;  SCS code image
                                0A 2328
                                   2333  3573 NAME_CLSLOA:                          ; Filename of loadable cluster
45 2E 41 4F 4C 52 54 53 55 4C 43 00' 2333  3574          .ASCIC  /CLUSTRLOA.EXE/       ;  code image
                             45 58 233F
                                0D 2333
                                   2341  3575 NAME_ERAPATLOA:                       ; Filename of loadable $ERAPAT
45 2E 41 4F 4C 54 41 50 41 52 45 00' 2341  3576          .ASCIC  /ERAPATLOA.EXE/       ;  code image
                             45 58 234D
                                0D 2341
                                   234F  3577 NAME_CHKPRTLOA:                       ; Filename of loadable $CHKPRT
45 2E 41 4F 4C 54 52 50 4B 48 43 00' 234F  3578          .ASCIC  /CHKPRTLOA.EXE/       ;  code image
                             45 58 235B
                                0D 234F
                                   235D  3579 NAME_VAXEMUL:                         ; Filename of loadable character and
45 58 45 2E 4C 55 4D 45 58 41 56 00' 235D  3580          .ASCIC  /VAXEMUL.EXE/         ;  decimal instruction emul code image
                                0B 235D
                                   2369  3581 NAME_FPEMUL:                          ; Filename of loadable FP emulation
   45 58 45 2E 4C 55 4D 45 50 46 00' 2369  3582          .ASCIC  /FPEMUL.EXE/          ;  code image
                                0A 2369
                                   2374  3583 NAME_MTACCESSLOA:                     ; Filename of loadable $MTACCESS
58 45 2E 53 53 45 43 43 41 54 4D 00' 2374  3584          .ASCIC  /MTACCESS.EXE/        ; code image
                                45 2380
                                0C 2374
                                   2381  3585 ;
                                   2381  3586 ; MISC. LOCAL STORAGE:
                                   2381  3587 ;
                                   2381  3588
                                   2381  3589          .ALIGN  LONG
                                   2384  3590 SCBPAGCT:                             ; # pages for system SCB
                          00000000 2384  3591          .LONG   0                     ;
                                   2388  3592 SCBBYTCT:                             ; # bytes for system SCB
                          00000000 2388  3593          .LONG   0                     ;
                                   238C  3594 SCBPHADDR:                            ; Physical address of SCB
                          00000000 238C  3595          .LONG   0                     ;
                                   2390  3596 VALID_PAR_FILE:                       ; Valid parameter file flag
                          00000000 2390  3597          .LONG   0
                                   2394  3598 CONEOF_ENABLE:                        ; Request volume switch on console EOF
                          00000000 2394  3599          .LONG   0                     ; Initially disabled, do not allow EOF
                                   2398  3600 CONEOF_MAX_PTR:                       ; Reserved number of retrieval pointers
                          00000000 2398  3601          .LONG   0                     ;  for SYS.EXE, less one for the dummy ptr
                                   239C  3602 VMB_VERSION:
                          00000000 239C  3603          .LONG   0                     ; VMB version number
                                   23A0  3604 SYSBOOT_BASE:
                          00000000 23A0  3605          .LONG   0                     ; Base address of SYSBOOT
                                   23A4  3606 BOO$GL_RPBBASE::
                          000023A8 23A4  3607          .BLKL   1                     ; Base address of RPB
                                   23A8  3608 CPUTYPE:
                          00000000 23A8  3609          .LONG   0                     ; Copy of cpu type not in SYSPARAMS
                                   23AC  3610 ASSUME  VERSVEC780LEN LE 8            ; The following space must be large
                                   23AC  3611                                       ;  enough to accomodate all of the
                                   23AC  3612 CPUDATA:                              ;  VERVECT info
                 00000000 00000000 23AC  3613          .LONG   0,0                   ; Copy of cpu data not in SYSPARAMS
                                   23B4  3614 RTRV_BUF_DSC:                         ; Size and address of buffer for
                          000023BC 23B4  3615          .BLKQ   1                     ; retrieval pointer information
                                   23BC  3616 VBN2_BUF_DSC:                         ; Size and address of buffer for
                          000023C4 23BC  3617          .BLKQ   1                     ; 2nd block of each driver
```

```
          23C4  BOOTCB:
00000000  23C4          .LONG   0                       ; Address of boot control block
          23C8                                          ; Contains BOOTDRIVR, QIO_RWVB,
          23C8                                          ; maps for SYS and SYSDUMP
          23C8  MEM_LO_PFN:                             ; Lowest PFN found by VMB (inclusive)
000023D0  23C8          .BLKQ   1
000023CC  23D0          MEM_HI_PFN = MEM_LO_PFN+4       ; Highest PFN found by VMB (exclusive)
          23D0  CACHE_HI_PFN:
000023D4  23D0          .BLKL   1                       ; Highest PFN in FIL$OPENFILE cache
          23D4  PFNMAP_HI_PFN:
000023D8  23D4          .BLKL   1                       ; Highest PFN in PFN bitmap pages
          23D8  SMALL_PFNMAP::                          ; Saved descriptor for backwards
000023E0  23D8          .BLKQ   1                       ; compatible 8mb (max) PFN bitmap
          23E0  UCODE_LEN:                              ; Length of loadable ucode
00000000  23E0          .LONG   0
          23E4  UCODE_ADR:                              ; Physical addr of it
00000000  23E4          .LONG   0
          23E8  ;
          23E8  ; Following two cells must be contiguous!!
          23E8  ;
          23E8  VMB_FLAGS:                              ; Flags passed from VMB
00000000  23E8          .LONG   0
          23EC  CI_HI_PFN:
00000000  23EC          .LONG   0                       ; Highest PFN for CI support
          23F0  ;
          23F0  ; Statistics Blocks for various files that are placed in non-paged pool
          23F0  ; ***** Do Not Reorder or separate these Statistics Blocks *****
          23F0  ;
          23F0  SYS_STAT:
000023F8  23F0          .BLKQ   1                       ; Starting VBN, size in bytes
00100000  23F8          .LONG   1@20                    ; not used
00002190' 23FC          .LONG   BOO$GT_SYS              ; Adr of ASCIC name of system image
00002408  2400          .BLKB   STAT_C_SIZE-16
          2408  ;
          2408  ; The following stat blocks will be disabled if the console volume has been
          2408  ; switched during a boot from the console.
          2408  ;
          2408  CONEOF_DISABLE_STAT:
          2408  SYSDUMP_STAT:
00002410  2408          .BLKQ   1                       ; Starting VBN, size in bytes
00100000  2410          .LONG   1@20                    ; not used
00000000  2414          .LONG   0                       ; Adr of ASCIC name of dump file
          2418                                          ; now filled in at run time
00002420  2418          .BLKB   STAT_C_SIZE-16
          2420  BOODRV_STAT:
00002424  2420          .BLKL   1                       ; VBN (not used)
00002428  2424          .BLKL   1                       ; Size in bytes of boot driver
00000000' 2428          .LONG   BOO$GL_BOOTCB           ; Adr to store adr of boot driver
00002540' 242C          .LONG   BOODRV_MAP_ERR          ; Adr of ASCIC string for alloc err msg
00002438  2430          .BLKB   STAT_C_SIZE-16
          2438  DSKDRV_STAT:
00002440  2438          .BLKQ   1                       ; VBN and size in bytes of disk driver
00000000' 2440          .LONG   BOO$GL_DSKDRV           ; Adr to store adr of disk driver
00000000  2444          .LONG   0                       ; Adr of ASCIC name of disk driver
00002450  2448          .BLKB   STAT_C_SIZE-16
          2450  PRTDRV_STAT:
00002458  2450          .BLKQ   1                       ; VBN and size in bytes of port driver
```

```
                  00000000' 2458  3675            .LONG    BOO$GL_PRTDRV               ; Adr to store adr of port driver
                  00000000  245C  3676            .LONG    0                           ; Adr of ASCIC name of port driver
                  00002468  2460  3677            .BLKB    STAT_C_SIZE-16
                            2468  3678 TRMDRV_STAT:
                  00002470  2468  3679            .BLKQ    1                           ; VBN and size in bytes of terminal driver
                  00000000' 2470  3680            .LONG    BOO$GL_TRMDRV               ; Adr to store adr of terminal driver
                  000021C5' 2474  3681            .LONG    TTNAME                      ; Adr of ASCIC name of terminal driver
                  00002480  2478  3682            .BLKB    STAT_C_SIZE-16
                            2480  3683 SCSLOA_STAT:
                  00002488  2480  3684            .BLKQ    1                           ; VBN and size in bytes of SCS code
                  00000000' 2488  3685            .LONG    BOO$GL_SCSLOA               ; Adr to store adr of SCS code
                  00002328' 248C  3686            .LONG    NAME_SCSLOA                 ; Adr of ASCIC name of SCS code
                  00002498  2490  3687            .BLKB    STAT_C_SIZE-16
                            2498  3688 SYSLOA_STAT:
                  000024A0  2498  3689            .BLKQ    1                           ; VBN and size in bytes of CPU dependent cod
                  00000000' 24A0  3690            .LONG    BOO$GL_SYSLOA               ; Adr to store adr of CPU dependent code
                  000022FA' 24A4  3691            .LONG    NAME_SYSLOA                 ; Adr of ASCIC name of CPU dependent code
                  000024B0  24A8  3692            .BLKB    STAT_C_SIZE-16
                            24B0  3693 CLSLOA_STAT:
                  000024B8  24B0  3694            .BLKQ    1                           ; VBN and size in bytes of cluster code
                  00000000' 24B8  3695            .LONG    BOO$GL_CLSLOA               ; Adr to store adr of cluster code
                  00002333' 24BC  3696            .LONG    NAME_CLSLOA                 ; Adr of ASCIC name of cluster code
                  000024C8  24C0  3697            .BLKB    STAT_C_SIZE-16
                            24C8  3698 ERAPATLOA_STAT:
                  000024D0  24C8  3699            .BLKQ    1                           ; VBN and size in bytes of $ERAPAT code
                  00000000' 24D0  3700            .LONG    BOO$GL_ERAPATLOA            ; Adr to store adr of $ERAPAT code
                  00002341' 24D4  3701            .LONG    NAME_ERAPATLOA              ; Adr of ASCIC name of $ERAPAT code
                  000024E0  24D8  3702            .BLKB    STAT_C_SIZE-16
                            24E0  3703 CHKPRTLOA_STAT:
                  000024E8  24E0  3704            .BLKQ    1                           ; VBN and size in bytes of $CHKPRT code
                  00000000' 24E8  3705            .LONG    BOO$GL_CHKPRTLOA            ; Adr to store adr of $CHKPRT code
                  0000234F' 24EC  3706            .LONG    NAME_CHKPRTLOA              ; Adr of ASCIC name of $CHKPRT code
                  000024F8  24F0  3707            .BLKB    STAT_C_SIZE-16
                            24F8  3708 VAXEMUL_STAT:
                  00002500  24F8  3709            .BLKQ    1                           ; VBN & size in bytes of char emul code
                  00000000' 2500  3710            .LONG    BOO$GL_VAXEMUL              ; Adr of char/decimal emulation code
                  0000235D' 2504  3711            .LONG    NAME_VAXEMUL                ; Adr of ASCIC name of char emul code
                  00002510  2508  3712            .BLKB    STAT_C_SIZE-16
                            2510  3713 FPEMUL_STAT:
                  00002518  2510  3714            .BLKQ    1                           ; VBN and size in bytes of FP emul code
                  00000000' 2518  3715            .LONG    BOO$GL_FPEMUL               ; Adr to store adr of FP emulation code
                  00002369' 251C  3716            .LONG    NAME_FPEMUL                 ; Adr of ASCIC name of FP emulation code
                  00002528  2520  3717            .BLKB    STAT_C_SIZE-16
                            2528  3718 MTACCESSLOA_STAT:
                  00002530  2528  3719            .BLKQ    1                           ; VBN and size in bytes of $MTACCESS code
                  00000000' 2530  3720            .LONG    BOO$GL_MTACCESSLOA          ; Adr to store adr of $MTACCESS code
                  00002374' 2534  3721            .LONG    NAME_MTACCESSLOA            ; Adr of ASCIC name of $MTACCESS code
                  00002540  2538  3722            .BLKB    STAT_C_SIZE-16
                            2540  3723
                  0000000D  2540  3724 CONEOF_DISABLE_CNT = <.-CONEOF_DISABLE_STAT>/STAT_C_SIZE        ; Files to "close" o
                            2540  3725
                  0000000C  2540  3726 ALLOC_POOL_CNT = <.-BOODRV_STAT>/STAT_C_SIZE    ; Things to alloc real pool for
                            2540  3727
                  0000000B  2540  3728 LOAD_IMAGE_CNT = <.-DSKDRV_STAT>/STAT_C_SIZE    ; Images to load
                            2540  3729
                            2540  3730 BOODRV_MAP_ERR:
 52 56 49 52 44 54 4F 4F 42 00' 2540  3731            .ASCIC   /BOOTDRIVR/
```

```
        09 2540
           254A 3732
           254A 3733 BOO$GL_PGDCOD:
 00000000' 254A 3734          .LONG   PAT$A_NONPGD_CODE_END
           254E 3735 ;
           254E 3736 ; This is patch area for SYSBOOT.EXE .
           254E 3737 ;
           254E 3738 PATCH_DESC::                               ; Patch area descriptor
  00000064 254E 3739          .LONG   100                       ;   size
 00002556' 2552 3740          .LONG   PATCH_AREA                ;   next free byte
           2556 3741 PATCH_AREA:                                ; Patch area
  000025BA 2556 3742          .BLKB   100                       ;   100 bytes
           25BA 3743
           25BA 3744          .END                              ;
```

```
$$BASE               = 00000001                 BOO$GL_GPTPGCT            000021A9 RG    03
$$DISPL              = 00000008                 BOO$GL_IRPCNT             ********    X  03
$$GENSW              = 00000001                 BOO$GL_LRPCNT             ********    X  03
$$HIGH               = 00000007                 BOO$GL_LRPMIN             ********    X  03
$$LIMIT              = 00000006                 BOO$GL_LRPSIZE            ********    X  03
$$LOW                = 00000001                 BOO$GL_LRPSPLIT           ********    X  03
$$MNSW               = 00000001                 BOO$GL_MTACCESSLOA        ********    X  03
$$MXSW               = 00000001                 BOO$GL_NEXTPFN            000021AD RG    03
ADD_CACHE              000001EA R     03        BOO$GL_NPAGEDYN           ********    X  03
ADD_RPB_MEMDSC         00001563 R     03        BOO$GL_PGDCOD             0000254A R     03
ALLOCPFN               000016F1 R     03        BOO$GL_PRTDRV             ********    X  03
ALLOCSPT               00000CA5 R     03        BOO$GL_RPBBASE            000023A4 RG    03
ALLOC_DRIVERS          00001129 R     03        BOO$GL_SCSLOA             ********    X  03
ALLOC_FILCACHE         0000113A R     03        BOO$GL_SPLITADR           ********    X  03
ALLOC_IRP              00000FF4 R     03        BOO$GL_SPTFREH            ********    X  03
ALLOC_LRP              0000105E R     03        BOO$GL_SPTFREL            ********    X  03
ALLOC_POOL             00001D6C R     03        BOO$GL_SPTPAGCT           000021B1 RG    03
ALLOC_POOL_CNT       = 0000000C                 BOO$GL_SRPCNT             ********    X  03
ALLOC_SRP              00000F91 R     03        BOO$GL_SRPSPLIT           ********    X  03
ARC$M_CHAR_EMUL      = 00000010                 BOO$GL_SYSLOA             ********    X  03
ARC$M_CRC_EMUL       = 00000080                 BOO$GL_SYSPHD             000021BD RG    03
ARC$M_DCML_EMUL      = 00000020                 BOO$GL_SYSPHDPG           000021C1 RG    03
ARC$M_DFLT_EMUL      = 00000100                 BOO$GL_TRMDRV             ********    X  03
ARC$M_EDPC_EMUL      = 00000040                 BOO$GL_UCODE              ********    X  03
ARC$M_FFLT_EMUL      = 00000200                 BOO$GL_VAXEMUL            ********    X  03
ARC$M_GFLT_EMUL      = 00000400                 BOO$GQ_FILCACHE           ********    X  03
ARC$M_HFLT_EMUL      = 00000800                 BOO$GQ_INILOA             ********    X  03
ARC$V_CHAR_EMUL      = 00000004                 BOO$GQ_STATBLK            000021B5 RG    03
ARC$V_CRC_EMUL       = 00000007                 BOO$GT_PAGEFILE           00002183 R     03
ARC$V_DCML_EMUL      = 00000005                 BOO$GT_PROMPT             0000219C RG    03
ARC$V_DFLT_EMUL      = 00000008                 BOO$GT_SYS                00002190 RG    03
ARC$V_EDPC_EMUL      = 00000006                 BOO$GT_SYSDUMP            00002177 R     03
ARC$V_FFLT_EMUL      = 00000009                 BOO$GT_TOPSYS             ********    X  03
ARC$V_GFLT_EMUL      = 0000000A                 BOO$IMAGE_ATT             ********    X  03
ARC$V_HFLT_EMUL      = 0000000B                 BOO$L_BUG_MAP           = 00000024
BELL                 = 00000007                 BOO$L_CHECKSUM          = 00000000
BOO$A_BOOPARAM         ********    X  03        BOO$L_DMP_MAP           = 00000020
BOO$A_SYSPHD           ********    X  03        BOO$L_DMP_SIZE          = 0000001C
BOO$B_TYPE           = 0000000A                 BOO$L_DMP_VBN           = 00000018
BOO$CACHE_INIT         ********    X  03        BOO$L_PARAM_MAP         = 00000004
BOO$C_BOOPARSZ         ********    X  03        BOO$L_SYS_MAP           = 00000014
BOO$C_LENGTH         = 00000028                 BOO$L_SYS_SIZE          = 00000010
BOO$C_SYSPARSZ         ********    X  03        BOO$L_SYS_VBN           = 0000000C
BOO$FACMSG             00001DC0 RG    03        BOO$MAPVBN                0000184D R     03
BOO$FILOPEN            0000167C RG    03        BOO$MSGOUT                ********    X  03
BOO$GB_NODENAME        ********    X  03        BOO$READFILE              00001871 RG    03
BOO$GB_SYSTEMID        ********    X  03        BOO$READPROMPT            ********    X  03
BOO$GETPARAM           ********    X  03        BOO$SEARCH                ********    X  03
BOO$GIVEHELP           00001E35 RG    03        BOO$SETASCII              ********    X  03
BOO$GL_BOOTCB          ********    X  03        BOO$SETMAP                0000181F R     03
BOO$GL_CHKPRTLOA       ********    X  03        BOO$SETVALUE              ********    X  03
BOO$GL_CLSLOA          ********    X  03        BOO$TYPE_ASCIC            00001DD1 R     03
BOO$GL_DEVNAME         ********    X  03        BOO$USECUR                ********    X  03
BOO$GL_DSKDRV          ********    X  03        BOO$USEFILE               00001C05 RG    03
BOO$GL_ERAPATLOA       ********    X  03        BOO$W_SIZE              = 00000008
BOO$GL_FPEMUL          ********    X  03        BOODRV_MAP_ERR            00002540 R     03
BOO$GL_FREEMEM         00002198 RG    03        BOODRV_STAT               00002420 R     03
```

```
BOOTCB                            000023C4 R       03
BOOTHIGH                          00000000 R       02
BOOTNDT_TST                       00000137 R       03
BOOT_FAULT                        00001DF0 R       03
BOO_HALT1                         000003E4 R       03
BQO$L_AUXDRNAME                 = 00000020
BQO$L_DEVNAME                   = 00000030
BQO$L_DRIVRNAME                 = 0000000C
BQO$L_MAP                       = 00000004
BQO$L_QIO                       = 00000000
BQO$L_UCODE                     = 00000028
BQO$W_VERCHECK                  = 00000012
BQO$W_VERSION                   = 00000010
BTD$K_CONSOLE                   = 00000040
BTD$K_MB                        = 00000000
BUG$A_PAGED                       ********    X    03
BUG$A_PAGEDEND                    ********    X    03
BUG$T_MESSAGES                    ********    X    03
BUMP_CI                           00000D3C R       03
CACHE_HI_PFN                      000023D0 R       03
CHECK_SID                         00000288 R       03
CHKPRTLOA_STAT                    000024E0 R       03
CHKVERSION                        0000022F R       03
CHKVERS_730                       00000282 R       03
CHKVERS_750                       0000027A R       03
CHKVERS_780                       0000034D R       03
CHKVERS_785                       00000342 R       03
CHKVERS_790                       000002E3 R       03
CHKVERS_END                       000003E4 R       03
CHKVERS_UCODE                     00000290 R       03
CHKVERS_UV1                       00000339 R       03
CHK_780_785                       00000356 R       03
CHK_CPU_TYPE                      00000034 R       03
CI_HI_PFN                         000023EC R       03
CLSLOA_STAT                       000024B0 R       03
CLU$GB_VAXCLUSTER                 ********    X    03
CON$GETCHAR                       ********    X    03
CON$PUTCHAR                       ********    X    03
CONEOF_BUFFER                     000022F6 R       03
CONEOF_CHECK                      00001923 R       03
CONEOF_DISABLE_CNT              = 0000000D
CONEOF_DISABLE_STAT               00002408 R       03
CONEOF_ENABLE                     00002394 R       03
CONEOF_MAX_PTR                    00002398 R       03
CONEOF_MESSAGE                    00001B50 R       03
CONEOF_PROMPT                     0000222E R       03
CONEOF_REMOVE1                    00002277 R       03
CONEOF_REMOVE2                    00002296 R       03
CONEOF_RESUME1                    000022B4 R       03
CONEOF_RESUME2                    000022DA R       03
CPUDATA                           000023AC R       03
CPUTYPE                           000023A8 R       03
CR                              = 0000000D
CRC_INS                           000007A2 R       03
CRC_INS_SZ                      = 00000008
CXB$C_OVERHEAD                  = 0000004C
DALLOCPFN                         0000175D R       03
DBNAME                            000021D2 R       03
DDNAME                            00002213 R       03
DEBUG                           = 00000001
DENOMINATOR                       000007F4 R       03
DIR...                          = 00000001
DIVP_INS                          0000077C R       03
DIVP_INS_SZ                     = 0000000D
DLNAME                            000021F9 R       03
DMNAME                            000021DF R       03
DONE                              00001C94 R       03
DPT$W_SIZE                      = 00000008
DRNAME                            000021EC R       03
DSKDRV_STAT                       00002438 R       03
DXNAME                            00002206 R       03
DYN$C_BOOTCB                    = 00000006
DYN$C_INIT                      = 00000063
EDITPC_INS                        0000078F R       03
EDITPC_INS_SZ                   = 0000000B
END_BOOTNDT_TST                   00000157 R       03
END_LOOKUP                        000001ED R       03
ERAPATLOA_STAT                    000024C8 R       03
ERL$UNEXP                         ********    X    03
ERL$VEC_RETURN                    ********    X    03
ERROUT                            00001DC0 R       03
EXE$ACVIOLAT                      00001DF0 RG      03
EXE$A_BOOPARAM                    ********    X    03
EXE$A_SYSPARAM                    ********    X    03
EXE$BREAK                         00001DF0 RG      03
EXE$FAO                           ********    X    03
EXE$GB_CPUDATA                    ********    X    03
EXE$GB_CPUTYPE                    ********    X    03
EXE$GL_ARCHFLAG                   ********    X    03
EXE$GL_DEFFLAGS                   ********    X    03
EXE$GL_DYNAMIC_FLAGS              ********    X    03
EXE$GL_FLAGS                      00001E2C RG      03
EXE$GL_INTSTK                     ********    X    03
EXE$GL_RPB                        ********    X    03
EXE$GL_RTIMESPT                   ********    X    03
EXE$GL_SCB                        ********    X    03
EXE$GT_STARTUP                    ********    X    03
EXE$GW_PGFL_FID                   ********    X    03
EXE$INIT                          ********    X    03
EXE$ROPRAND                       00001DF0 RG      03
EXE$SYSBOOT                       00000000 RG      03
EXE$TBIT                          00001DF0 RG      03
EXE$V_PAGFILDMP                   ********    X    03
EXE$V_SYSPAGING                   ********    X    03
EXE$V_TBCHK                       ********    X    03
EXE$V_WRITESYSPARAMS              ********    X    03
FH2$W_FID_NUM                   = 00000008
FH2$W_FID_RVN                   = 0000000C
FIL$CACHE_TRUNC                   ********    X    03
FIL$GQ_CACHE                      00002165 RG      03
FIL$GT_TOPSYS                     0000216D RG      03
FIL$OPENFILE                      ********    X    03
FILL_NEXUSO                       00000DDF R       03
FILOPEN                           000016C9 R       03
```

FPEMUL_STAT	00002510	R	03	MMG$GL_GPTE	********	X	03
FPLA_780	= 0000000C			MMG$GL_IRPNEXT	********	X	03
FPLA_VLOC_780	= 0000006D			MMG$GL_LRPNEXT	********	X	03
GETCONLOC_780	00001CB8	R	03	MMG$GL_MAXGPTE	********	X	03
GETCONLOC_790	00001CF5	R	03	MMG$GL_MAXMEM	********	X	03
GETCURRENT	000003E4	R	03	MMG$GL_MAXPFN	********	X	03
GHOPT_VLOC	= 00000064			MMG$GL_MAXSYSVA	********	X	03
HAVEPRM	00000866	R	03	MMG$GL_MINPFN	********	X	03
HELPMSG	00001E45	R	03	MMG$GL_NPAGEDYN	********	X	03
HWREV_730	= 00000000			MMG$GL_NPAGNEXT	********	X	03
HWREV_750	= 00000000			MMG$GL_PAGEDYN	********	X	03
IHD$B_HDRBLKCNT	= 00000010			MMG$GL_PGDCOD	********	X	03
IMAGE_INIT	00001605	R	03	MMG$GL_PHYPGCNT	********	X	03
IMAGE_OPEN	00001584	R	03	MMG$GL_SBR	********	X	03
INI$BRK	00000107	RG	03	MMG$GL_SPTBASE	********	X	03
INI$RDONLY	00001E2B	RG	03	MMG$GL_SPTLEN	********	X	03
INI$WRITABLE	00001E2B	RG	03	MMG$GL_SRPNEXT	********	X	03
INITBOOTDRV	00001380	R	03	MMG$GL_SYSPHD	********	X	03
INIT_SCB	00000D75	R	03	MMG$GL_SYSPHDLN	********	X	03
INIT_SYSPHD	00000E0A	R	03	MMG$GW_BIGPFN	********	X	03
IO$_READLBLK	= 00000021			MMG$PAGEFAULT	00001DF0	RG	03
IO750$C_SL_UB1	= 00000009			MODEL_5	00002318	R	03
IOSIZE	= 0000007F			MODEL_6	0000231C	R	03
IRP$C_LENGTH	= 000000C4			MODEL_730	00002310	R	03
IRP_SPT	00000B16	R	03	MODEL_750	0000230C	R	03
LF	= 0000000A			MODEL_780	00002308	R	03
LOAD_CODE	000017D1	R	03	MODEL_790	00002314	R	03
LOAD_IMAGE_CNT	= 0000000B			MODEL_TABLE	00002308	R	03
LOOKUPDRIVER	000005C9	R	03	MODEL_UV1	00002320	R	03
LOOKUP_ALL_DONE	0000084F	R	03	MODEL_UV2	00002324	R	03
LOOKUP_LOADABLE_CODE	000006E1	R	03	MOVBOODRIVER	00001460	R	03
LRP_SPT	00000B3D	R	03	MOVBOOPARAM	00001500	R	03
MAPISTK	00000F5E	R	03	MOVD_INS	000007B0	R	03
MAPNPAGDYN	000010D3	R	03	MOVD_INS_SZ	= 00000003		
MAPPFNDAT	000011A6	R	03	MOVEMAPPED	00001D2E	R	03
MAPRESEXEC	00001172	R	03	MOVE_BITMAP	00001763	R	03
MAPRPB	0000131E	R	03	MOVFILECACHE	00001423	R	03
MAPSCB	00000F38	R	03	MOVF_INS	000007B9	R	03
MAP_XDELTA_INIT	00001360	R	03	MOVF_INS_SZ	= 00000003		
MATCHC_INS	0000076D	R	03	MOVG_INS	000007C2	R	03
MATCHC_INS_SZ	= 00000009			MOVG_INS_SZ	= 00000004		
MAXPGS	= 00004000			MOVH_INS	000007CC	R	03
MEM_HI_PFN	= 000023CC	R	03	MOVH_INS_SZ	= 00000004		
MEM_LO_PFN	000023C8	R	03	MOVSYSPARAM	0000140D	R	03
MICREV_730	= 00000033			MTACCESSLOA_STAT	00002528	R	03
MICREV_750	= 0000003E			MTCH_785	= 00000004		
MICREV_790	= 00000000			MTCH_VLOC_785	= 0000006B		
MICREV_REQ_790	= 00000012			NAME_CHKPRTLOA	0000234F	R	03
MICREV_UV1	= 00000004			NAME_CLSLOA	00002333	R	03
MMG$AL_PGDCODEN	********	X	03	NAME_ERAPATLOA	00002341	R	03
MMG$AL_SYSPAGTB	********	X	03	NAME_FPEMUL	00002369	R	03
MMG$A_SYSPARAM	********	X	03	NAME_MTACCESSLOA	00002374	R	03
MMG$A_SYS_END	********	X	03	NAME_SCSLOA	00002328	R	03
MMG$C_SPTSKEL	********	X	03	NAME_SYSLOA	000022FA	R	03
MMG$GL_CTLBASVA	********	X	03	NAME_VAXEMUL	0000235D	R	03
MMG$GL_FRESVA	********	X	03	NAME_XXX	= 00002301	R	03
MMG$GL_GPTBASE	********	X	03	NOBRK	00000108	R	03

```
NOCONVER                000004ED R      03      OP$_CVTLG               = 00004EFD
NPAGE_SPT               00000ADB R      03      OP$_CVTLH               = 00006EFD
NUMERATOR               000007F3 R      03      OP$_CVTLP               = 000000F9
OP$_ACBD              = 0000006F                OP$_CVTPL               = 00000036
OP$_ACBF              = 0000004F                OP$_CVTPS               = 00000008
OP$_ACBG              = 00004FFD                OP$_CVTPT               = 00000024
OP$_ACBH              = 00006FFD                OP$_CVTRDL              = 0000006B
OP$_ADDD2             = 00000060                OP$_CVTRFL              = 0000004B
OP$_ADDD3             = 00000061                OP$_CVTRGL              = 00004BFD
OP$_ADDF2             = 00000040                OP$_CVTRHL              = 00006BFD
OP$_ADDF3             = 00000041                OP$_CVTSP               = 00000009
OP$_ADDG2             = 000040FD                OP$_CVTTP               = 00000026
OP$_ADDG3             = 000041FD                OP$_CVTWD               = 0000006D
OP$_ADDH2             = 000060FD                OP$_CVTWF               = 0000004D
OP$_ADDH3             = 000061FD                OP$_CVTWG               = 00004DFD
OP$_ADDP4             = 00000020                OP$_CVTWH               = 00006DFD
OP$_ADDP6             = 00000021                OP$_DIVD2               = 00000066
OP$_ASHP              = 000000F8                OP$_DIVD3               = 00000067
OP$_CLRD              = 0000007C                OP$_DIVF2               = 00000046
OP$_CLRF              = 000000D4                OP$_DIVF3               = 00000047
OP$_CLRG              = 0000007C                OP$_DIVG2               = 000046FD
OP$_CLRH              = 00007CFD                OP$_DIVG3               = 000047FD
OP$_CMPD              = 00000071                OP$_DIVH2               = 000066FD
OP$_CMPF              = 00000051                OP$_DIVH3               = 000067FD
OP$_CMPG              = 000051FD                OP$_DIVP                = 00000027
OP$_CMPH              = 000071FD                OP$_EDITPC              = 00000038
OP$_CMPP3             = 00000035                OP$_EMODD               = 00000074
OP$_CMPP4             = 00000037                OP$_EMODF               = 00000054
OP$_CRC               = 0000000B                OP$_EMODG               = 000054FD
OP$_CVTBD             = 0000006C                OP$_EMODH               = 000074FD
OP$_CVTBF             = 0000004C                OP$_MATCHC              = 00000039
OP$_CVTBG             = 00004CFD                OP$_MNEGD               = 00000072
OP$_CVTBH             = 00006CFD                OP$_MNEGF               = 00000052
OP$_CVTDB             = 00000068                OP$_MNEGG               = 000052FD
OP$_CVTDF             = 00000076                OP$_MNEGH               = 000072FD
OP$_CVTDH             = 000032FD                OP$_MOVD                = 00000070
OP$_CVTDL             = 0000006A                OP$_MOVF                = 00000050
OP$_CVTDW             = 00000069                OP$_MOVG                = 000050FD
OP$_CVTFB             = 00000048                OP$_MOVH                = 000070FD
OP$_CVTFD             = 00000056                OP$_MOVP                = 00000034
OP$_CVTFG             = 000099FD                OP$_MOVTC               = 0000002E
OP$_CVTFH             = 000098FD                OP$_MOVTUC              = 0000002F
OP$_CVTFL             = 0000004A                OP$_MULD2               = 00000064
OP$_CVTFW             = 00000049                OP$_MULD3               = 00000065
OP$_CVTGB             = 000048FD                OP$_MULF2               = 00000044
OP$_CVTGF             = 000033FD                OP$_MULF3               = 00000045
OP$_CVTGH             = 000056FD                OP$_MULG2               = 000044FD
OP$_CVTGL             = 00004AFD                OP$_MULG3               = 000045FD
OP$_CVTGW             = 000049FD                OP$_MULH2               = 000064FD
OP$_CVTHB             = 000068FD                OP$_MULH3               = 000065FD
OP$_CVTHD             = 0000F7FD                OP$_MULP                = 00000025
OP$_CVTHF             = 0000F6FD                OP$_POLYD               = 00000075
OP$_CVTHG             = 000076FD                OP$_POLYF               = 00000055
OP$_CVTHL             = 00006AFD                OP$_POLYG               = 000055FD
OP$_CVTHW             = 000069FD                OP$_POLYH               = 000075FD
OP$_CVTLD             = 0000006E                OP$_SCANC               = 0000002A
OP$_CVTLF             = 0000004E                OP$_SKPC                = 0000003B
```

```
OP$_SPANC                      = 0000002B
OP$_SUBD2                      = 00000062
OP$_SUBD3                      = 00000063
OP$_SUBF2                      = 00000042
OP$_SUBF3                      = 00000043
OP$_SUBG2                      = 000042FD
OP$_SUBG3                      = 000043FD
OP$_SUBH2                      = 000062FD
OP$_SUBH3                      = 000063FD
OP$_SUBP4                      = 00000022
OP$_SUBP6                      = 00000023
OP$_TSTD                       = 00000073
OP$_TSTF                       = 00000053
OP$_TSTG                       = 000053FD
OP$_TSTH                       = 000073FD
OVERLAY_VMB                      00000201 R      03
PAT$A_NONPGD_CODE_END            ********   X    03
PATCH_AREA                       00002556 R      03
PATCH_DESC                       0000254E RG     03
PATTERN                          000007F9 R      03
PCS_780                        = 00000001
PCS_785                        = 00000004
PCS_VLOC_780                   = 0000006A
PCS_VLOC_785                   = 0000006A
PFN$AB_STATE                     ********   X    03
PFN$AB_TYPE                      ********   X    03
PFN$AL_BAK                       ********   X    03
PFN$AL_PTE                       ********   X    03
PFN$AW_REFCNT                    ********   X    03
PFN$AW_SWPVBN                    ********   X    03
PFN$AX_BLINK                     ********   X    03
PFN$AX_FLINK                     ********   X    03
PFN$C_LONG_LEN                   ********   X    03
PFN$C_WORD_LEN                   ********   X    03
PFN$GB_LENGTH                    ********   X    03
PFNMAP_HI_PFN                    000023D4 R      03
PFN_SPT                          00000BB6 R      03
PHD$B_PAGFIL                   = 0000001F
PHD$C_LENGTH                   = 0000017C
PHD$L_FREPOVA                  = 00000028
PHD$L_P0BR                     = 000000C8
PHD$L_P0LRASTL                 = 000000CC
PHD$L_PSTBASOFF                = 00000020
PHD$W_DFWSCNT                  = 0000001A
PHD$W_PHVINDEX                 = 00000042
PHD$W_WSAUTH                   = 0000000A
PHD$W_WSAUTHEXT                = 00000014
PHD$W_WSEXTENT                 = 00000016
PHD$W_WSLAST                   = 00000012
PHD$W_WSNEXT                   = 00000010
PHD$W_WSQUOTA                  = 00000018
PHD$W_WSSIZE                   = 00000050
PHYP_ER                          000008C6 R      03
PQL$GDWSDEFAULT                  ********   X    03
PQL$GDWSQUOTA                    ********   X    03
PQL$GMWSDEFAULT                  ********   X    03
PQL$GMWSQUOTA                    ********   X    03
PR$V_SID_TYPE                  = 00000018
PR$_P0BR                       = 00000008
PR$_P0LR                       = 00000009
PR$_RXCS                       = 00000020
PR$_RXDB                       = 00000021
PR$_SBR                        = 0000000C
PR$_SCBB                       = 00000011
PR$_SID                        = 0000003E
PR$_SID_TYP730                 = 00000003
PR$_SID_TYP750                 = 00000002
PR$_SID_TYP780                 = 00000001
PR$_SID_TYP785                 = 00000009
PR$_SID_TYP790                 = 00000004
PR$_SID_TYPMAX                 = 00000008
PR$_SID_TYPUV1                 = 00000007
PR$_SLR                        = 0000000D
PR$_TBCHK                      = 0000003F
PR$_TBIA                       = 00000039
PR$_TXCS                       = 00000022
PR$_TXDB                       = 00000023
PR$_WCSD                       = 0000002D
PR780$_ACCS                    = 00000028
PRM$B_SIZE                     = 00000014
PRM$L_FLAGS                    = 00000010
PRM$V_ASCII                    = 00000010
PRTDRV_STAT                      00002450 R      03
PSL$M_TBIT                     = 00000010
PTE$C_ERKW                     = 30000000
PTE$C_KOWN                     = 00000000
PTE$C_URKW                     = 70000000
PTE$M_PFN                      = 001FFFFF
PTE$M_VALID                    = 80000000
PTE$S_PFN                      = 00000015
PTE$V_PFN                      = 00000000
QIO_RWLB                         00001BD0 R      03
QUOTIENT                         000007F5 R      03
QVSS$INPUT                       ********   X    03
QVSS$KEY                         ********   X    03
QVSS$KEYTABLE                    ********   X    03
QVSS$OUTPUT                      ********   X    03
R2_R11                         = 00000FFC
READ_COMPLETE                    000018D5 R      03
READ_SYS_ERR                     00001404 R      03
READ_VIRTUAL                     0000188D R      03
RPB$B_CONFREG                  = 00000090
RPB$B_CTRLLTR                  = 00000108
RPB$B_DEVTYP                   = 00000066
RPB$B_FLAGS                    = 000000A3
RPB$C_LENGTH                   = 00000109
RPB$L_ADPPHY                   = 0000005C
RPB$L_BADPGS                   = 00000104
RPB$L_BOOTR1                   = 00000020
RPB$L_BOOTR5                   = 00000030
RPB$L_IOVEC                    = 00000034
RPB$L_IOVECSZ                  = 00000038
RPB$L_MEMDSC                   = 000000BC
RPB$L_PFNCNT                   = 0000004C
```

```
RPB$L_SVASPT              = 00000050
RPB$Q_PFNMAP              = 00000044
RPB$T_FILE                = 00000068
RPB$V_BOOBPT              = 00000005
RPB$V_CONV                = 00000000
RPB$V_NOSYSDISK           = 00000000
RPB$W_BOOTNDT             = 000000A1
RPB$W_UNIT                = 00000064
RP_DT                     = 00000018
RTRV_BUF_DSC                00002384 R      03
RTRV_PAG_CNT              = 00000003
SAVABS...                 = 00000018
SCB$AL_BASE                 ********   X    03
SCBBYTCT                    00002388 R      03
SCBPAGCT                    00002384 R      03
SCBPHADDR                   0000238C R      03
SCB_VEC_730                 00000E0A R      03
SCB_VEC_750                 00000E0A R      03
SCB_VEC_780                 00000DD5 R      03
SCB_VEC_790                 00000DD0 R      03
SCB_VEC_UV1                 00000E0A R      03
SCSLOA_STAT                 00002480 R      03
SEC$C_LENGTH              = 00000020
SGN$GL_BALSETCT             ********   X    03
SGN$GL_IRPCNT               ********   X    03
SGN$GL_IRPCNTV              ********   X    03
SGN$GL_LOADFLAGS            ********   X    03
SGN$GL_LRPCNT               ********   X    03
SGN$GL_LRPCNTV              ********   X    03
SGN$GL_LRPMIN               ********   X    03
SGN$GL_LRPSIZE              ********   X    03
SGN$GL_MAXGPGCT             ********   X    03
SGN$GL_MAXVPGCT             ********   X    03
SGN$GL_MAXWSCNT             ********   X    03
SGN$GL_NPAGEDYN             ********   X    03
SGN$GL_NPAGEVIR             ********   X    03
SGN$GL_P1LWCNT              ********   X    03
SGN$GL_PAGEDYN              ********   X    03
SGN$GL_PHDAPCNT             ********   X    03
SGN$GL_PHDLWCNT             ********   X    03
SGN$GL_PHDPAGCT             ********   X    03
SGN$GL_PTPAGCNT             ********   X    03
SGN$GL_SPTREQ               ********   X    03
SGN$GL_SRPCNT               ********   X    03
SGN$GL_SRPCNTV              ********   X    03
SGN$GL_SRPSIZE              ********   X    03
SGN$GW_GBLSECNT             ********   X    03
SGN$GW_ISPPGCT              ********   X    03
SGN$GW_MAXPSTCT             ********   X    03
SGN$GW_MINWSCNT             ********   X    03
SGN$GW_SWPFILES             ********   X    03
SGN$GW_SYSDWSCT             ********   X    03
SGN$V_LOADCHKPRT            ********   X    03
SGN$V_LOADERAPAT            ********   X    03
SGN$V_LOADMTACCESS          ********   X    03
SHELLSIZE                   00000904 R      03
SID_785                   = 00000017

SIZ_SCB                     00000C38 R      03
SIZ_SCB_730                 00000C73 R      03
SIZ_SCB_750                 00000C77 R      03
SIZ_SCB_780                 00000C81 R      03
SIZ_SCB_790                 00000C6E R      03
SIZ_SCB_END                 00000C81 R      03
SIZ_SCB_UV1                 00000C73 R      03
SMALL_PFNMAP                000023D8 RG     03
SRP_SPR                     00000B87 R      03
SS$_ENDOFFILE               ********   X    03
STAT_C_SIZE                 00000018
STAT_L_BYTECNT              00000004
STAT_L_MAP                  00000010
STAT_L_NAME                 0000000C
STAT_L_SYSVA                00000008
STAT_L_VBN                  00000000
STAT_L_VBN2ADR              00000014
STRING                      000007F2 R      03
SWP$C_DBGPTCNT              ********   X    03
SWP$C_KSTACK                ********   X    03
SWP$C_NDYN                  ********   X    03
SWP$C_SHLFPTE               ********   X    03
SWP$C_SHLP1PT               ********   X    03
SWP$GB_SHLP1PT              ********   X    03
SWP$GL_BALBASE              ********   X    03
SWP$GL_BALSPT               ********   X    03
SWP$GL_BSLOTSZ              ********   X    03
SWP$GL_PHDBASVA             ********   X    03
SWP$GL_SHELLSIZ             ********   X    03
SWP$GW_BAKPTE               ********   X    03
SWP$GW_EMPTPTE              ********   X    03
SWP$GW_WSLPTE               ********   X    03
SYS$FAO                     00001E30 RG     03
SYSBOOT_BASE                000023A0 R      03
SYSDUMP_STAT                00002408 R      03
SYSL$CLRSBIA                00001E2B RG     03
SYSLOA_STAT                 00002498 R      03
SYS_STAT                    000023F0 R      03
TBL                         0000079A R      03
TPA$L_NUMBER              = 0000001C
TPA$L_PARAM               = 00000020
TPA$L_TOKENCNT            = 00000010
TPA$L_TOKENPTR            = 00000014
TRMDRV_STAT                 00002468 R      03
TTNAME                      000021C5 R      03
TTY$GW_CLASSNAM             ********   X    03
UCODE_ADR                   000023E4 R      03
UCODE_LEN                   000023E0 R      03
UNEXP_MCHK                  00001E0C R      03
UNKNAME                     00002220 R      03
VA$S_BYTE                 = 00000009
VA$S_VPN                  = 00000015
VA$V_BYTE                 = 00000000
VA$V_SYSTEM               = 0000001F
VA$V_VPN                  = 00000009
VALID_PAR_FILE              00002390 R      03
VAX$EMULATE                 ********   X    03
```

```
VAX$EMULATE_FPD                  ********    X    03
VAXEMUL_STAT                     000024F8 R       03
VBN2_BUF_DSC                     000023BC R       03
VERSNUM_780                      00001CE8 R       03
VERSNUM_785                      00001CF1 R       03
VERSVEC780LEN                  = 00000004
VERSVEC785LEN                  = 00000004
VERSVECT_780                     00001CE3 R       03
VERSVECT_785                     00001CEC R       03
VMB$B_SYSTEMID                   00000024
VMB$C_ARGBYTCNT                  0000003C
VMB$L_CI_HIPFN                   00000030
VMB$L_FLAGS                      0000002C
VMB$L_HI_PFN                     00000010
VMB$L_LO_PFN                     0000000C
VMB$Q_FILECACHE                  00000004
VMB$Q_NODENAME                   00000034
VMB$Q_PFNMAP                     00000014
VMB$Q_UCODE                      0000001C
VMB$V_LOAD_SCS                 = 00000000
VMB_FLAGS                        000023E8 R       03
VMB_VERSION                      0000239C R       03
WCB$W_P1_COUNT                 = 00000030
WCSP_780                       = 0000000C
WCSP_785                       = 00000001
WCSP_VLOC_780                  = 0000006B
WCSP_VLOC_785                  = 0000006C
WCSS_780                       = 00000012
WCSS_785                       = 00000000
WCSS_VLOC_780                  = 0000006C
WCSS_VLOC_785                  = 0000006D
WSL$C_LENGTH                   = 00000004
XDELBPT                          ********    X    03
XDELIBRK                         ********    X    03
XDELTBIT                         ********    X    03
XDS$GT_WORD_PFN                  00001E2C RG      03

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

PSECT name                       Allocation           PSECT No.  Attributes
----------                       ----------           ---------  ----------
.  ABS  .                        00000000  (    0.)   00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            0000003C  (   60.)   01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
 Z99BOOT                         00000000  (    0.)   02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC PAGE
$$$$00BOOT                       000025BA  ( 9658.)   03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   36     00:00:00.08    00:00:00.62
Command processing              139     00:00:00.81    00:00:04.34
Pass 1                         1110     00:00:49.53    00:01:31.71
```

```
Symbol table sort               0     00:00:04.20    00:00:06.91
Pass 2                       1312     00:00:13.54    00:00:29.34
Symbol table output             3     00:00:00.57    00:00:00.86
Psect synopsis output           3     00:00:00.02    00:00:00.02
Cross-reference output          0     00:00:00.00    00:00:00.00
Assembler run totals         2605     00:01:08.76    00:02:13.81

The working set limit was 1350 pages.
229177 bytes (448 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2469 non-local and 247 local symbols.
6496 source lines were read in Pass 1, producing 61 object records in Pass 2.
174 pages of virtual memory were used to define 170 macros.

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

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[BOOTS.OBJ]BOOTS.MLB;1                   1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      26
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   15
TOTALS (all libraries)                              42

2372 GETS were required to define 42 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSBOOT/OBJ=OBJ$:SYSBOOT MASD$:[EMULAT.SRC]MISSING/UPDATE=(MASD$:[EMULAT.ENH]MISSING)+MASD$:[BOOTS.SRC]SYSBOOT/UPDATE
```

0040 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSBOOCMD
LIS

SHOWADAP
LIS

STANDCONF
LIS

SHODEV
LIS

STAPUTERR
LIS

STALOCK
LIS

STASGNMSG
LIS

STACONFIG
LIS

STASYSGEN
LIS

SYSBOOT
LIS

STARDDRIV
LIS

0041 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY

SYSGENMN
LIS

UMB
LIS

SYSGEN
LIS

SYSGTTSTR
LIS

SYSGENCMD
LIS

SYSGTERM
LIS

TS8BOOTIO
LIS