```
   SSSSSSSSSSS  YYY         YYY    SSSSSSSSSSS  LLL              OOOOOOOOO       AAAAAAAAA
   SSSSSSSSSSS  YYY         YYY    SSSSSSSSSSS  LLL              OOOOOOOOO       AAAAAAAAA
   SSSSSSSSSSS  YYY         YYY    SSSSSSSSSSS  LLL              OOOOOOOOO       AAAAAAAAA
SSS             YYY         YYY SSS             LLL           OOO         OOO   AAA         AAA
SSS             YYY         YYY SSS             LLL           OOO         OOO   AAA         AAA
SSS             YYY         YYY SSS             LLL           OOO         OOO   AAA         AAA
SSS                YYY   YYY    SSS             LLL           OOO         OOO   AAA         AAA
SSS                YYY   YYY    SSS             LLL           OOO         OOO   AAA         AAA
SSS                YYY   YYY    SSS             LLL           OOO         OOO   AAA         AAA
   SSSSSSSSS          YYY          SSSSSSSSS     LLL           OOO         OOO   AAA         AAA
   SSSSSSSSS          YYY          SSSSSSSSS     LLL           OOO         OOO   AAA         AAA
   SSSSSSSSS          YYY          SSSSSSSSS     LLL           OOO         OOO   AAA         AAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAAAAAAAAAAAAAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAAAAAAAAAAAAAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAAAAAAAAAAAAAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAA         AAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAA         AAA
            SSS       YYY                   SSS  LLL           OOO         OOO   AAA         AAA
SSSSSSSSSSS           YYY SSSSSSSSSSS     LLLLLLLLLLLLLLL        OOOOOOOOO     AAA         AAA
SSSSSSSSSSS           YYY SSSSSSSSSSS     LLLLLLLLLLLLLLL        OOOOOOOOO     AAA         AAA
SSSSSSSSSSS           YYY SSSSSSSSSSS     LLLLLLLLLLLLLLL        OOOOOOOOO     AAA         AAA
```

FILEID**INIADP750

```
IIIIII    NN      NN    IIIIII      AAAAAA    DDDDDDDD    PPPPPPPP    77777777    5555555555    000000
IIIIII    NN      NN    IIIIII      AAAAAA    DDDDDDDD    PPPPPPPP    77777777    5555555555    000000
  II      NN      NN      II      AA      AA  DD      DD  PP      PP          77    55            00    00
  II      NN      NN      II      AA      AA  DD      DD  PP      PP          77    55            00    00
  II      NNNN    NN      II      AA      AA  DD      DD  PP      PP          77    555555        00  0000
  II      NNNN    NN      II      AA      AA  DD      DD  PP      PP          77    555555        00  0000
  II      NN  NN  NN      II      AA      AA  DD      DD  PPPPPPPP        77        55    00  00  00
  II      NN  NN  NN      II      AA      AA  DD      DD  PPPPPPPP        77        55    00  00  00
  II      NN    NNNN      II      AAAAAAAAAA  DD      DD  PP            77            55  0000    00
  II      NN    NNNN      II      AAAAAAAAAA  DD      DD  PP            77            55  0000    00
  II      NN      NN      II      AA      AA  DD      DD  PP          77      55      55  00      00
  II      NN      NN      II      AA      AA  DD      DD  PP          77      55      55  00      00
IIIIII    NN      NN    IIIIII    AA      AA  DDDDDDDD    PP          77        555555      000000
IIIIII    NN      NN    IIIIII    AA      AA  DDDDDDDD    PP          77        555555      000000
```

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

Table of contents

```
0000     1          .NLIST  CND
0000     5
0000     7          .TITLE  INIADP750 - ADAPTER INITIALIZATION FOR VAX 11/750
0000     9
0000    13
0000    17
0000    21
0000    25
0000    26          .IDENT  'V04-002'
0000    27 ;
0000    28 ;*****************************************************************************
0000    29 ;*                                                                           *
0000    30 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    31 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    32 ;*  ALL RIGHTS RESERVED.                                                     *
0000    33 ;*                                                                           *
0000    34 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    35 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    36 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    37 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    38 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    39 ;*  TRANSFERRED.                                                             *
0000    40 ;*                                                                           *
0000    41 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    42 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    43 ;*  CORPORATION.                                                             *
0000    44 ;*                                                                           *
0000    45 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    46 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    47 ;*                                                                           *
0000    48 ;*                                                                           *
0000    49 ;*****************************************************************************
0000    50 ;
0000    51 ; Facility: System bootstrapping and initialization
0000    52 ;
0000    53 ; Abstract: This module contains initialization routines that are loaded
0000    54 ;           during system initialization (rather than linked into the system).
0000    55 ;
0000    56 ; Environment: Mode = KERNEL, Executing on INTERRUPT stack, IPL=31
0000    57 ;
0000    58 ; Author:  Trudy C. Matthews              Creation date: 22-Jan-1981
0000    59 ;
0000    60 ; Modification history:
0000    61 ;
0000    62 ;       V04-002 TCM0013         Trudy C. Matthews       10-Sep-1984
0000    63 ;               Add $BQODEF missing from TCM0012.
0000    64 ;
0000    65 ;       V04-001 TCM0012         Trudy C. Matthews       07-Sep-1984
0000    66 ;               For venus processor:  turn on cache before calibrating
0000    67 ;               TIMEDWAIT cells (routine EXE$INI_TIMWAIT).  Store the TIMEDWAIT
0000    68 ;               values calculated after cache is enabled in the boot driver's
0000    69 ;               TIMEDWAIT cells.  This is because the boot driver initially
0000    70 ;               has to run with cache off, but after booting will run with
0000    71 ;               cache on.
0000    72 ;
0000    73 ;       V03-024 TCM0011         Trudy C. Matthews       31-Jul-1984
0000    74 ;               Change venus's CRD interrupt vector back to ^X54 in the SCB,
```

```
0000     75 ;                  and its SBIA Fail vector to ^X64.
0000     76 ;
0000     77 ;          V03-023 WMC0001         Wayne Cardoza           30-Jul-1984
0000     78 ;                  Add H memory to 780 list.
0000     79 ;
0000     80 ;          V03-022 TCM0010         Trudy C. Matthews       25-Jul-1984
0000     81 ;                  Fix a bug in INI$UBSPACE for the 11/790 that caused second
0000     82 ;                  and subsequent unibus adapter spaces to be mapped incorrectly.
0000     83 ;                  Fix bugs in INI$SCB for the 11/790.  Fix conditional
0000     84 ;                  assembly flags in INI$CONSOLE for the 11/790.
0000     85 ;
0000     86 ;          V03-021 KDM0100         Kathleen D. Morse       01-May-1984
0000     87 ;                  Correct address of memory CSRs to be past the 8 missing
0000     88 ;                  Qbus adapter pages that do not exist.
0000     89 ;
0000     90 ;          V03-020 KDM0099         Kathleen D. Morse       27-Apr-1984
0000     91 ;                  On a MicroVAX I, if the sysgen parameter TIMEDWAIT is set
0000     92 ;                  to request no time-prompting, then use the last recorded
0000     93 ;                  system time instead.  This is found in EXE$GQ_TODCBASE
0000     94 ;                  which can be updated with a SET TIME command.
0000     95 ;
0000     96 ;          V03-019 RLRSCORPIO      Robert L. Rappaport     16-Mar-1984
0000     97 ;                  Begin additions (to INI$IOMAP) for Scorpio support.
0000     98 ;                  Also move ADAPDESC to SYSMAR.MAR, changing it to remove
0000     99 ;                  the ADAP_GENERAL array.
0000    100 ;
0000    101 ;          V03-018 RLRINIADP       Robert Rappaport        28-Feb-1984
0000    102 ;                  Add refinements to previous update that introduces
0000    103 ;                  longword array CONFREG.  Mainly add logic to allow for
0000    104 ;                  independently assembled invocations of ADAPDESC macro
0000    105 ;                  to be linked into this code.  This provides possible
0000    106 ;                  support of BI as a public bus, with user defined nodes.
0000    107 ;
0000    108 ;          V03-017 KPL0100         Peter Lieberwirth       30-Jan-1984
0000    109 ;                  Implement first step towards a longword-array CONFREG to
0000    110 ;                  replace current byte array CONFREG.  INIADP will construct
0000    111 ;                  two confregs, CONFREG and CONFREGL.  CONFREGL will be
0000    112 ;                  a longword array.  The high byte will be a VMS-bus
0000    113 ;                  designation, and the low word will contain the 16-bit
0000    114 ;                  device type.  The BI introduces 16 bit device types.
0000    115 ;
0000    116 ;                  When all references to CONFREG have been modified to touch
0000    117 ;                  CONFREGL, INIADP will be modified again to stop creating
0000    118 ;                  the byte array.
0000    119 ;
0000    120 ;                  While here, map 9 pages of CI register space, up from 8.
0000    121 ;
0000    122 ;          V03-016 KPL0001         Peter Lieberwirth       17-Jan-1984
0000    123 ;                  Fix bug in V03-015 that caused a failure to boot on 750s.
0000    124 ;                  Specifically, add NDT$_MEM1664NI to ADAPDESC macro.
0000    125 ;
0000    126 ;          V03-015 TCM0009         Trudy C. Matthews       12-Dec-1983
0000    127 ;                  Add support for booting from VENUS console device to
0000    128 ;                  INI$CONSOLE.  When mapping I/O space on VENUS, use the
0000    129 ;                  PAMM to determine if any adaptors are present on the
0000    130 ;                  ABUS.
0000    131 ;
```

```
0000   132 ;       V03-014 KDM0081         Kathleen D. Morse       13-Sep-1983
0000   133 ;               Create version for Micro-VAX I.
0000   134 ;
0000   135 ;       V03-013 DWT0126         David W. Thiel          30-Aug-1983
0000   136 ;               Modify EXE$INIT_TODR to set internal time without
0000   137 ;               modifying the contents of the system disk.
0000   138 ;
0000   139 ;       V03-012 KDM0062         Kathleen D. Morse       18-Jul-1983
0000   140 ;               Add loadable, cpu-dependent routine for initializing
0000   141 ;               the time-wait loop data cells, EXE$INI_TIMWAIT.
0000   142 ;
0000   143 ;       V03-011 KDM0057         Kathleen D. Morse       15-Jul-1983
0000   144 ;               Added loadable, cpu-dependent routine for initializing
0000   145 ;               the system time, EXE$INIT_TODR.
0000   146 ;
0000   147 ;       V03-010 KTA3071         Kerbey T. Altmann       12-Jul-1983
0000   148 ;               Include CPU-specific console init code.
0000   149 ;
0000   150 ;       V03-009 TCM0008         Trudy C. Matthews       10-Jan-1983
0000   151 ;               Change PSECT of 11/790 data that must stick around after
0000   152 ;               INIADP is deleted.  Build arrays ABUS_VA, ABUS_TYPE, and
0000   153 ;               ABUS_INDEX that describe the 11/790 ABUS configuration.
0000   154 ;
0000   155 ;       V03-008 MSH0002         Maryann Hinden          08-Dec-1982
0000   156 ;               Add powerfail support for DW750.
0000   157 ;
0000   158 ;       V03-007 ROW0142         Ralph O. Weber          24-NOV-1982
0000   159 ;               Change UBA interrupt services routines prototype so that
0000   160 ;               UBAERRADR is correctly computed as an offset from UBAINTBASE.
0000   161 ;
0000   162 ;       V03-006 TCM0007         Trudy C. Matthews       10-Nov-1982
0000   163 ;               Add 11/790-specific initialization of SCB.
0000   164 ;
0000   165 ;       V03-005 TCM0006         Trudy C. Matthews       8-Nov-1982
0000   166 ;               Initialize field ADP$L_AVECTOR with the address of
0000   167 ;               each adapter's first SCB vector.
0000   168 ;
0000   169 ;       V03-004 KTA3018         Kerbey T. Altmann       30-Oct-1982
0000   170 ;               Move from INILOA facility, rename from INITADP,
0000   171 ;               put in conditional assembly, rewrite some routines.
0000   172 ;
0000   173 ;       V03-003 MSH0001         Maryann Hinden          24-Sep-1982
0000   174 ;               Change EXE$DW780_INT to EXE$UBAERR_INT.
0000   175 ;
0000   176 ;       V03-002 TCM0005         Trudy C. Matthews       10-Aug-1982
0000   177 ;               Added support for 11/790 processor.
0000   178 ;
0000   179 ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000   180 ;               Added $DCDEF.
0000   181 ;
0000   182 ;--
```

```
                     ;
                     ; MACRO LIBRARY CALLS
                     ;
                             $ADPDEF                 ; Define ADP offsets.
                             $BIICDEF                ; Define BIIC offsets.
                             $BOOTDEF                ; Define boot vector offsets.
                             $BTDDEF                 ; Define boot devices
                             $BUADEF                 ; Define BUA Register offsets.
                             $CRBDEF                 ; Define CRB offsets.
                             $DCDEF                  ; Define adapter types
                             $DDBDEF                 ; Define DDB offsets
                             $DYNDEF                 ; Define data structure type codes.
                             $IDBDEF                 ; Define interrupt dispatcher offsets.
                             $IO750DEF               ; Define 11/750 I/O space.
                             $UASDEF                 ; Define DW750 IPEC registers.
                             $MCHKDEF                ; Define machine check masks.
                             $NDTDEF                 ; Define nexus device types.
                             $PRDEF                  ; Define IPR numbers.
                     ;
                     ;
                             $PR750DEF               ; Define 11/750 specific IPR numbers.
                     ;
                     ;
                     ;
                     ;
                     ;
                             $PTEDEF                 ; Define Page Table Entry bits.
                             $RPBDEF                 ; Define Restart Parameter Block fields.
                             $UBADEF                 ; Define UBA register offsets.
                             $UCBDEF                 ; Define UCB offsets.
                             $VADEF                  ; Define virtual address fields.
                             $VECDEF                 ; Define vec offsets.
```

```
0000    254          .SBTTL  Macros to describe nexus configurations
0000    255 ;
0000    256 ;       The macros FLOAT_NEXUS and FIXED_NEXUS add one or more entries to a
0000    257 ;       nexus descriptor table.  Each entry is of the form:
0000    258 ;               +-------------------------------+
0000    259 ;               |   PFN of nexus I/O space      |
0000    260 ;               +-------+-------+---------------+
0000    261 ;               |  bus  |   0   |     type      |
0000    262 ;               +-------+-------+---------------+
0000    263 ;       type = 0 -> floating nexus
0000    264 ;       type = non-zero -> fixed nexus; type = fixed adapter type
0000    265 ;       bus  = 0, if SBI; %x80 if BI  (this is a VMS-only designation)
0000    266 ;
0000    267 ;
0000    268 ;       device_type:    SBI adapters have 8-bit device type codes.  These
0000    269 ;                       device types are simple integers.
0000    270 ;
0000    271 ;                       BI adapters have 16-bit device type codes, that are
0000    272 ;                       subject to the following interpretation:
0000    273 ;
0000    274 ;                       - the MSB of the device-type field will be 0 for DEC
0000    275 ;                       devices and 1 for non-DEC devices,
0000    276 ;
0000    277 ;                       - DEC memory devices will have 0s in the high-order
0000    278 ;                       byte of the device type,
0000    279 ;
0000    280 ;                       - non-DEC supplied memory devices will have a 1 in the
0000    281 ;                       MSB of the high-order byte, and the rest of the high
0000    282 ;                       order byte will contain 0s.
0000    283 ;
0000    284 ;                       - The "all 0s" and "all 1s" device-type codes are
0000    285 ;                       reserved for DEC.
0000    286 ;
0000    287 ; If SBI type codes were simply expanded to a word for purposes of the routines
0000    288 ; in this module, there would be possible conflicts between SBI devices and
0000    289 ; BI memory adapters supplied by DEC.  Voila:  the bus type.
0000    290 ;
0000    291 ; Macro FLOAT_NEXUS.
0000    292 ; INPUTS:
0000    293 ;       PHYSADR -- physical address of 1 or more contiguous floating nexus
0000    294 ;                  slots
0000    295 ;       NUMNEX -- number of contiguous floating nexuses, default = 1
0000    296 ;       PERNEX -- amount of address space per nexus (does not have to be
0000    297 ;                 specified if NUMNEX = 1)
0000    298 ;
0000    299          .MACRO  FLOAT_NEXUS     PHYSADR,NUMNEX=1,PERNEX=0
0000    300          PA = PHYSADR
0000    301          .REPEAT NUMNEX          ; For each nexus...
0000    302          .LONG   <PA/^X200>      ; Store PFN.
0000    303          .LONG   0               ; Store floating nexus type.
0000    304          PA = PA + PERNEX        ; Increment to physical address of next nexus.
0000    305          .ENDR
0000    306          .ENDM   FLOAT_NEXUS
0000    307
0000    308 ;
0000    309 ; Macro FIXED_NEXUS.
0000    310 ;
```

```
                     0000    311 ; INPUTS:
                     0000    312 ;       PHYSADR - physical address of 1 or more contiguous fixed nexus slots
                     0000    313 ;       PERNEX - amount of address space per nexus
                     0000    314 ;       NEXUSTYPES - a list of fixed nexus types, enclosed in <>
                     0000    315 ;
                     0000    316         .MACRO  FIXED_NEXUS     PHYSADR,PERNEX=0,NEXUSTYPES
                     0000    317         PA = PHYSADR
                     0000    318         .IRP    TYPECODE,NEXUSTYPES     ; For each fixed nexus type...
                     0000    319         .LONG   <PA/^X200>              ; Store PFN.
                     0000    320         .LONG   TYPECODE                ; Store fixed nexus type.
                     0000    321         PA = PA + PERNEX                ; Increment to address of next nexus.
                     0000    322         .ENDR
                     0000    323         .ENDM   FIXED_NEXUS
                     0000    324
                     0000    325 ;
                     0000    326 ; Macro NEXUSDESC_TABLE - declare the beginning of a NEXUS descriptor table
                     0000    327 ;
                     0000    328 ;       1st byte in table (at offset -5 from label) contains length of
                     0000    329 ;       adapter type code field in CSR's on this bus. [Note for SBI like
                     0000    330 ;       busses, this is 1.]  The next longword (at offset -4) in the
                     0000    331 ;       table contains the Software defined bus type byte defined in the
                     0000    332 ;       high order byte of the longword.  [Note for SBI like busses, this
                     0000    333 ;       value is 0, for the BI it is ^x80.]
                     0000    334 ;
                     0000    335
                     0000    336 ; Define parameters that may be specified or used in macro invocation.
                     0000    337
00000000             0000    338 BI_LIKE  = 0                    ; BI like bus.
00000001             0000    339 SBI_LIKE = 1                    ; SBI like bus.
                     0000    340
00000001             0000    341 SBI_CSR_LEN = 1                 ; Length of type code field in adapter CSR's
                     0000    342                                 ;  on SBI, CMI, etc.
00000002             0000    343 BI_CSR_LEN  = 2                 ; Length of type code field in adapter CSR's
                     0000    344                                 ;  on BI.
                     0000    345
00000000             0000    346 SBI_BUS_CODE = 0                ; Software defined bus code for SBI like busses.
80000000             0000    347 BI_BUS_CODE  = ^x80000000       ; Software defined bus code for the BI.
                     0000    348
                     0000    349         .MACRO  NEXUSDESC_TABLE LABEL,BUS_TYPE=SBI_LIKE
                     0000    350         .IF     EQ,BUS_TYPE-SBI_LIKE
                     0000    351                                 .BYTE   SBI_CSR_LEN
                     0000    352                                 .LONG   SBI_BUS_CODE
                     0000    353         .IFF
                     0000    354                 .IF     EQ,BUS_TYPE-BI_LIKE
                     0000    355                                 .BYTE   BI_CSR_LEN
                     0000    356                                 .LONG   BI_BUS_CODE
                     0000    357                 .IFF
                     0000    358                                 .ERROR  ; UNRECOGNIZED BUS TYPE, NEXUSDESC_TABLE;
                     0000    359                 .ENDC
                     0000    360         .ENDC
                     0000    361
                     0000    362 LABEL:
                     0000    363         .ENDM   NEXUSDESC_TABLE
                     0000    364
FFFFFFFB             0000    365 CSR_LEN_OFFSET  = -5                        ; Offset before nexus descriptor of
                     0000    366                                             ;  byte containing length of adapter
                     0000    367                                             ;  type field in adapter CSR.
```

```
FFFFFFFC  0000   368 BUS_CODE_OFFSET = -4                    ; Offset before nexus descriptor table
          0000   369                                        ;  of longword containing software
          0000   370                                        ;  defined bus type to be or'ed with
          0000   371                                        ;  adapter type to produce NDT$_ value.
          0000   372 ;
          0000   373 ; Macro END_NEXUSDESC.
          0000   374 ;
          0000   375         .MACRO  END_NEXUSDESC
          0000   376         .LONG   0                      ; PFN=0 -> end of nexus descriptors.
          0000   377         .ENDM   END_NEXUSDESC
```

```
                        .SBTTL  Adapter-specific data structures
                ;
                ; Put a symbol for arrays built by macros in the correct psects.
                ;
                ;****************** ADAPTERS array *************
00000000                .PSECT  $$$INIT$DATA0
0000            ADAPTERS:                               ; Build adapter type code arrays here.
                
00000000                .PSECT  $$$INIT$DATA1           ; User contributions in this .PSECT.
                                                        ; End of ADAPTERS array.
                ;****************** End of ADAPTERS array *************
                
                ;****************** NUM_PAGES array *************
00000000                .PSECT  $$$INIT$DATA2
0000            NUM_PAGES:                              ; Build "number of pages to map" array.
00000000                .PSECT  $$$INIT$DATA3           ; User contributions in this .PSECT.
                ;****************** End of NUM_PAGESarray *************
                
                ;****************** INIT_ROUTINES array *************
00000000                .PSECT  $$$INIT$DATA4
0000            INIT_ROUTINES:                          ; Build "address of init routine" array.
00000000                .PSECT  $$$INIT$DATA5           ; User contributions in this .PSECT.
                ;****************** End of INIT_ROUTINES array *************
                
                ;
                ; To add a new adapter type:
                ;       1) Add a new ADAPDESC macro invocation to the end of this list.
                ;
00000000                .PSECT  $$$INIT$DATA,LONG
                
                ;
                ; Default interupt vectors for UNIBUS system devices
                ; (This array is indexed by the RPB field RPB$B_DEVTYP, if the RPB field
                ; RPB$W_ROUBVEC is zero.  If RPB$W_ROUBVEC is not zero, then RPB$W_ROUBVEC
                ; is used and this array is not referenced at all.  RPB$W_ROUBVEC is set up
                ; by PQDRIVER.  RPB$L_BOOTR0 is set by VMB to contain the device name in
                ; ASCII, not the vector number and device type, as it does on full
                ; architecture VAX machines.
                ;
0000            BOOTVECTOR:
0088                    .WORD   ^X88                    ; RK06/7 Interrupt vector
0070                    .WORD   ^X70                    ; RL01/2 Interrupt vector
                
0004            BUS_CSR_LEN:                            ; Static byte containing the length (in bytes)
00                      .BYTE   0                       ;  of the adapter type field in the CSR's of
                                                        ;  the bus currently being configured.  The
                                                        ;  proper value for the bus of interest is
                                                        ;  copied here, from the current nexus
                                                        ;  descriptor table, when we enter subroutine
                                                        ;  CONFIG_IOSPACE.
                
0005            SW_BUS_CODE:                            ; Static longword containing the software
00000000                .LONG   0                       ;  defined bus type, of the bus currently being
                                                        ;  configured, in the high order byte.  The
                                                        ;  proper value for the bus of current interest
                                                        ;  is copied here, from the nexus descriptor
0009                                                    ;  table, when we enter subroutine
```

```
                     0009   436                                ;  CONFIG_IOSPACE.
                     0009   437
                     0009   438 DIRECT_VEC_NODE_CNT:           ; Static longword that counts the number of
                     0009   439                                ;  direct vectoring adpater nodes that we have
00000000             0009   440         .LONG   0              ;  run across so far.
                     000D   441
00000001             000D   442 $$$VMSDEFINED = 1              ; Define symbol that means VMS system software.
00000080             000D   443 NUMUBAVEC = 128                ; ALLOW FOR 128 UNIBUS VECTORS
                     000D   444
                     000D   445         ADAPDESC -             ; Memory. ** MUST BE 1ST IN DESCRIPTOR LIST **
                     000D   446         ADPTYPES=<NDT$_MEM1664NI,NDT$_MEM4NI,NDT$_MEM4I,NDT$_MEM16NI, -
                     000D   447                 NDT$_MEM16I, -
                     000D   448                 NDT$_MEM64NIL,NDT$_MEM64EIL,NDT$_MEM64NIU,NDT$_MEM64EIU,  -
                     000D   449                 NDT$_MEM64I, -
                     000D   450                 NDT$_MEM256NIL,NDT$_MEM256EIL,NDT$_MEM256NIU,NDT$_MEM256EIU,  -
                     000D   451                 NDT$_MEM256I, -
                     000D   452                 NDT$_SCORMEM> -
                     000D   453                 NUMPAGES=1
                     000D   454
                     000D   455         ADAPDESC -             ; MASSbus.
                     000D   456                 ADPTYPES=NDT$_MB, -
                     000D   457                 NUMPAGES=8, -
                     000D   458                 INITRTN=INI$MBADP
                     000D   459
                     000D   460         ADAPDESC -             ; UNIbus.
                     000D   461                 ADPTYPES=<NDT$_UB0,NDT$_UB1,NDT$_UB2,NDT$_UB3,NDT$_BUA>, -
                     000D   462                 NUMPAGES=8, -
                     000D   463                 INITRTN=INI$UBSPACE
                     000D   464
                     000D   465         ADAPDESC -             ; Multi-port memory.
                     000D   466                 ADPTYPES=<NDT$_MPM0,NDT$_MPM1,NDT$_MPM2,NDT$_MPM3>, -
                     000D   467                 NUMPAGES=1, -
                     000D   468                 INITRTN=INI$MPMADP
                     000D   469
                     000D   470         ADAPDESC -             ; DR32.
                     000D   471                 ADPTYPES=NDT$_DR32, -
                     000D   472                 NUMPAGES=4, -
                     000D   473                 INITRTN=INI$DRADP
                     000D   474
                     000D   475         ADAPDESC -             ; CI780
                     000D   476                 ADPTYPES=NDT$_CI, -
                     000D   477                 NUMPAGES=9, -
                     000D   478                 INITRTN=INI$CIADP
                     000D   479
                     000D   480         ADAPDESC -             ; KDZ11 Processor
                     000D   481                 ADPTYPES=NDT$_KDZ11, -
                     000D   482                 NUMPAGES=1, -
                     000D   483                 INITRTN=INI$KDZ11
                     000D   484
```

```
                     000D   488 ;
                     000D   489 ; TABLES OF ADAPTER-DEPENDENT INFORMATION
                     000D   490 ;
                     000D   491 ; THE TABLE OFFSETS ARE:
                     000D   492 ;
                     000D   493         $DEFINI ADPTAB
                     0000   494
00000001             0000   495 ADPTAB_IDBUNITS:.BLKB   1              ; # UNITS TO SET IN IDB
00000003             0001   496 ADPTAB_ADPLEN:  .BLKW   1              ; LENGTH OF ADP
00000004             0003   497 ADPTAB_ATYPE:   .BLKB   1              ; ADP TYPE
                     0004   498
                     0004   499         $DEFEND ADPTAB
                     000D   500
                     000D   501 ;
                     000D   502 ; TABLES THEMSELVES:
                     000D   503 ;
                     000D   504
                     000D   505 MBATAB:                              ; TABLE OF MBA CONSTANTS
                  08 000D   506         .BYTE   8                    ; # UNITS IN MBA IDB
                0030 000E   507         .WORD   ADP$C_MBAADPLEN      ; # BYTES IN MBA ADP
                  00 0010   508         .BYTE   AT$_MBA              ; MBA ADAPTER TYPE
                     0011   509
                     0011   510 DRTAB:                               ; TABLE OF DR32 CONSTANTS
                  01 0011   511         .BYTE   1                    ; # UNITS IN DR IDB
                0030 0012   512         .WORD   ADP$C_DRADPLEN       ; # BYTES IN DR ADP
                  02 0014   513         .BYTE   AT$_DR               ; DR ADAPTER TYPE
                     0015   514
                     0015   515 CITAB:                               ; TABLE OF CI CONSTANTS
                  01 0015   516         .BYTE   1                    ; # UNITS IN CI IDB
                0030 0016   517         .WORD   ADP$C_CIADPLEN       ; # BYTES IN CI ADP
                  04 0018   518         .BYTE   AT$_CI               ; CI ADAPTER TYPE
                     0019   519
```

```
                     0019   523          .SBTTL  CPU-specific data structures
                     0019   524 ;
                     0019   525 ; To add a new CPU type:
                     0019   526 ;       1) Create a new nexus descriptor table, using FLOAT_NEXUS and
                     0019   527 ;          FIXED_NEXUS macros.  Put an END_NEXUSDESC macro at the end.
                     0019   528 ;
                     0019   529
                     0019   552
                     0019   554
                     0019   555 CPU_ADPSIZE:
                0258 0019   556          .WORD   ADP$C_UBAADPLEN
                     001B   557
                     001B   558
                     001B   559 ;
                     001B   560 ; Declare the beginning of a nexus-descriptor table.
                     001B   561 ;
                     001B   562          NEXUSDESC_TABLE LABEL=NEXUSDESC
                     0020   563
                     0020   564
                     0020   565 ;
                     0020   566 ; Describe all possible nexuses on an 11/750 (the first 10 have fixed adapter
                     0020   567 ; assignments).
                     0020   568 ;
            00000000 0020   569          SBI_CPU = 0
            00000000 0020   570          BI_CPU  = 0
                     0020   571          FIXED_NEXUS -
                     0020   572                  PHYSADR=I0750$AL_IOBASE, -
                     0020   573                  PERNEX=I0750$AL_PERNEX, -
                     0020   574                  NEXUSTYPES=<NDT$_MEM1664NI, -
                     0020   575                              NDT$_MPM0, -
                     0020   576                              NDT$_MPM1, -
                     0020   577                              NDT$_MPM2, -
                     0020   578                              NDT$_MB, -
                     0020   579                              NDT$_MB, -
                     0020   580                              NDT$_MB, -
                     0020   581                              NDT$_MB, -
                     0020   582                              NDT$_UB0, -
                     0020   583                              NDT$_UB1>
                     0070   584          FLOAT_NEXUS -
                     0070   585                  PHYSADR=I0750$AL_IOBASE+<10*I0750$AL_PERNEX>, -
                     0070   586                  NUMNEX=6, -
                     0070   587                  PERNEX=I0750$AL_PERNEX
                     00A0   588          END_NEXUSDESC
                     00A4   590
                     00A4   617
                     00A4   659
                     00A4   660
                     00A4   682
                     00A4   706
                     00A4   707 ;
                     00A4   708 ; Nexus "descriptor" arrays -- these arrays hold the nexus-device type and
                     00A4   709 ; virtual address of every adapter on the system.  The arrays, CONFREGL and
                     00A4   710 ; SBICONF, are allocated enough space to hold the maximum number of adapters
                     00A4   711 ; that can be attached to any CPU.  When the code discovers how many adapters
                     00A4   712 ; actually exist on the system, it will allocate space from non-paged pool
                     00A4   713 ; and move a permanent copy of these arrays into that space.
                     00A4   714 ;
```

```
00000040  00A4    715 MAXNEXUS = 64
          00A4    716 CONFREG:
000000E4  00A4    717         .BLKB   MAXNEXUS                ; Byte array of nexus-device type codes..
          00E4    718 SBICONF:
000001E4  00E4    719         .BLKL   MAXNEXUS                ; Longword array of VAs of adapter space.
          01E4    720 CONFREGL:
000002E4  01E4    721         .BLKL   MAXNEXUS                ; Longword array of nexus-device type codes
```

```
                                                02E4    723          .SBTTL  Message strings
                                                02E4    724
                                     0000000D   02E4    725 CR = 13
                                     0000000A   02E4    726 LF = 10
                                                02E4    727 NOSPT:
2D 54 49 4E 49 43 45 58 45 25 0A 0D             02E4    728          .ASCIZ  <CR><LF>/%EXECINIT-F-Insufficient SPT entries/<CR><LF>
65 69 63 69 66 66 75 73 6E 49 2D 46             02F0
69 72 74 6E 65 20 54 50 53 20 74 6E             02FC
                  00 0A 0D 73 65                0308
                                                030D    730 BADUMR:
2D 54 49 4E 49 43 45 58 45 25 0A 0D             030D    731          .ASCIZ  <CR><LF>/%EXECINIT-F-UNIBUS memory does not start at 0/<CR><LF>
6D 65 6D 20 53 55 42 49 4E 55 2D 46             0319
74 6F 6E 20 73 65 6F 64 20 79 72 6F             0325
0D 30 20 74 61 20 74 72 61 74 73 20             0331
                        00 0A                   033D
```

```
                                        .SBTTL  INI$IOMAP, Initialize and map nexuses
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;       This routine is executed only once, during system initialization.
                                ;       It loops through all nexuses on the system, testing for
                                ;       adapters.  When it finds an adapter, it maps its I/O space and
                                ;       initializes it.
                                ;
                                ; INPUTS:
                                ;       BOO$GL_SPTFREL  - next free VPN
                                ;       MMG$GL_SPTVASE  - base of system page table
                                ;       EXE$GL_RPB      - address of reboot parameter block
                                ;       RPB$L_ADPPHY(RPB) - PFN of boot adapter space
                                ;
                                ; OUTPUTS:
                                ;       R0 - SS$_NORMAL
                                ;
                                ;       For each adapter found, its accessible I/O space is mapped to virtual
                                ;       addresses.  An ADP (Adapter Control Block) is built, and the hardware
                                ;       adapter is initialized.
                                ;
                                ;       The arrays CONFREG (a byte array of nexus-device type codes, defined
                                ;       by NDT$_ symbols) and SBICONF (a longword array of
                                ;       virtual addresses that map adapter space) are initialized.  Pointers
                                ;       to these arrays are stored in EXE$GL_CONFREG  and
                                ;       MMG$GL_SBICONF.  The number of entries in these two parallel arrays is
                                ;       stored in EXE$GL_NUMNEXUS.
                                ;
                                ;       Since BI devices have a 16-bit device type code, a new CONFREG array is
                                ;       constructed.  This is a longword array called CONFREGL.
                                ;
                                ;       Several locations in the RPB that describe the boot device are init'ed:
                                ;       RPB$L_BOOTR1    - holds index into CONFREG and SBICONF for the boot
                                ;                         adapter
                                ;       RPB$L_ADPVIR    - holds VA of boot device adapter's register space
                                ;       RPB$L_CSRVIR    - holds VA of boot device's register space
                                ;--
                                
                        00000000        .PSECT  $$$INIT$CODE,QUAD
                        0000    INI$IOMAP::
                        0000
                0FFF 8F BB      0000            PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0004
                        0004    ; Set up common inputs to CONFIG_IOSPACE subroutine for the CPU-specific code.
                        0004    ;
    52  00000000'GF     D0  0004            MOVL    G^BOO$GL_SPTFREL,R2     ; Get next available VPN.
    53  00000000'GF     D0  000B            MOVL    G^MMG$GL_SPTBASE,R3     ; Get base of System Page Table.
      53   6342         DE  0012            MOVAL   (R3)[R2],R3             ; Compute SVASPT.
    52   52   09         78  0016            ASHL    #9,R2,R2                ; Convert VPN to VA.
    52  80000000 8F      C8  001A            BISL    #VA$M_SYSTEM,R2         ; Set system bit.
              54         D4  0021            CLRL    R4                      ; Clear index into CONFREG and SBICONF.
    59  00000000'GF     D0  0023            MOVL    G^EXE$GL_RPB,R9         ; Get address of RPB.
 5A  5C A9   F7 8F      78  002A            ASHL    #-9,RPB$L_ADPPHY(R9),R10; Get PFN of boot adapter space.
00000000'GF  00E4'CF    DE  0030            MOVAL   W^SBICONF,G^MMG$GL_SBICONF  ; Set pointers to local copies
00000000'GF  00A4'CF    DE  0039            MOVAL   W^CONFREG,G^EXE$GL_CONFREG  ; of these arrays for init routines.
00000000'GF  01E4'CF    DE  0042            MOVAL   W^CONFREGL,G^EXE$GL_CONFREGL ; ...
```

```
                        004B   899          .SBTTL  INITADP_780, _750, _730, and _UV1
                        004B   900 ;
                        004B   901 ; I/O address space for the 11/780, 11/750, 11/730, and Micro-VAX I cpus
                        004B   902 ; is statically defined in their respective nexus descriptor tables.
                        004B   903 ;
56   0020'CF   DE       004B   904          MOVAL   W^NEXUSDESC,R6          ; Get address of nexus table.
        5B   D4    0050   905          CLRL    R11                     ; Signal use 1st page of SCB.
        0B   10    0052   906          BSBB    CONFIG_IOSPACE          ; Configure processor I/O space.
                   0054   907
                   0054   909
      00C3   30    0054   910          BSBW    CREATE_ARRAYS           ; Create CONFREG and SBICONF arrays.
  0FFF  8F   BA    0057   911          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
  50    01   D0    005B   912          MOVL    #1,R0                   ; Set success status
             05    005E   913          RSB                             ; Return.
```

```
                                G 5
005F   916          .SBTTL  CONFIG_IOSPACE
005F   917 ;
005F   918 ; CONFIG_IOSPACE
005F   919 ;       Given a nexus descriptor table, which describes what "nexuses" or
005F   920 ;       "slots" are available on a system to hold I/O adapters, find and
005F   921 ;       initialize all adapters on the system.
005F   922 ;
005F   923 ; Inputs:
005F   924 ;       R2 - next available virtual address, to be used for mapping I/O space
005F   925 ;       R3 - address of PTE associated with VA in R2
005F   926 ;       R4 - Current index into CONFREG and SBICONF arrays (should be 0 the
005F   927 ;            first time CONFIG_IOSPACE is called)
005F   928 ;       R6 - address of nexus descriptor table
005F   929 ;       R9 - address of Restart Parameter Block (RPB)
005F   930 ;       R10 - PFN of boot adapter space
005F   931 ;       R11- page offset from beginning of SCB; tells which page of the SCB
005F   932 ;            to use for this set of nexuses (passed to routines that init ADP)
005F   933 ;
005F   934 ; Outputs:
005F   935 ;       R2,R3,R4 - updated
005F   936 ;       R9,R10,R11 - preserved; all other registers potentially modified
005F   937 ;       CONFREG - initialized with adapter NDT$ code for each nexus
005F   938 ;       SBICONF - initialized with adapter space VA for each nexus
005F   939 ;
005F   940 CONFIG_IOSPACE:
005F   942 ;
005F   943 ; Main loop.  Map and initialize all adapters on system.
005F   944 ;
005F   950
FB A6   90 005F   951          MOVB    CSR_LEN_OFFSET(R6),-    ; Move length of adapter type field
0004'CF       0062   952          W^BUS_CSR_LEN           ;  in CSR's to static location.
FC A6   D0 0065   953          MOVL    BUS_CODE_OFFSET(R6),-   ; Move software defined bus type code
0005'CF       0068   954          W^SW_BUS_CODE           ;  to static longword.
              006B   955
              006B   956 NXT_NEXUS:                          ; For each nexus...
58   86   D0 006B   957          MOVL    (R6)+,R8                ; Get PFN of nexus.
     01   12 006E   959          BNEQ    TEST_NEXUS              ; If PFN non-zero, go test the slot.
          05 0070   960          RSB                             ; If 0, we've found all nexuses.
             0071   961 ;
             0071   962 ; Read configuration register to determine if anything is present at this
             0071   963 ; nexus.
             0071   964 ;
             0071   965 TEST_NEXUS:
90000000 8F   C9 0071   966          BISL3   #PTE$M_VALID!PTE$C_KW,- ; Temporarily associate VA in R2 with
   63   58       0077   967          R8,(R3)                 ; PFN in R8 via SPTE in R3.
             0079   968          $PRTCTINI B^10$, -              ; Protect following code from non-
             0079   969                  #<MCHK$M_NEXM!MCHK$M_LOG>; existent memory machine checks.
51   62   D0 0085   970          MOVL    (R2),R1                 ; Read adapter configuration register.
             0088   971          $PRTCTEND 10$                   ; End of protected code.
             0089   972          INVALID R2                      ; Clear TB of temporary mapping.
  11 50   E8 008C   973          BLBS    R0,GET_TYPE             ; Branch if no machine check occurred.
             008F   974 ;
             008F   975 ; No adapter present at this nexus.
             008F   976 ;
00A4'CF44   94 008F   977          CLRB    W^CONFREG[R4]           ; Store "unknown" type in CONFREG
01E4'CF44   D4 0094   978          CLRL    W^CONFREGL[R4]          ; and in CONFREGL also.
     55   D4 0099   979          CLRL    R5                      ; Use general memory type to map
```

```
                              009B                                                 ; one page of I/O space.
          56    04     C0     009B             ADDL2   #4,R6                       ; Step past type code in nexus table.
                59     11     009E             BRB     MAP_NEXUS                   ; Go map I/O space for this nexus.
                              00A0       ;
                              00A0       ; Execution continues here if adapter was present.
                              00A0       ;
                              00A0       GET_TYPE:
          57    86     D0     00A0             MOVL    (R6)+,R7                    ; Get nexus-device type from nexus table.
                14     12     00A3             BNEQ    GET_GEN_TYPE                ; Branch if fixed slot.
                              00A5       ;
                              00A5       ; Floating-type slot.  Use type from configuration register.
                              00A5       ; Determine if type in configuration register is 8-bits or 16-bits.
                              00A5       ;
                              00A5
     0004'CF    01     91     00A5             CMPB    #1,W^BUS_CSR_LEN            ; Determine length of adapter type
                              00AA                                                 ;   field in CSR contained in R7.
                05     13     00AA             BEQL    10$                         ; EQL implies 1 byte (8-bit) field.
          57    51     3C     00AC             MOVZWL  R1,R7                       ; BI_LIKE, so use word instruction.
                03     11     00AF             BRB     20$                         ; Skip byte instruction.
          57    51     9A     00B1       10$:  MOVZBL  R1,R7                       ; Use byte instruction to get type.
                              00B4       20$:                                      ;
   57   0005'CF        C8     00B4             BISL    W^SW_BUS_CODE,R7            ; Or in software bus code.
                              00B9       ;
                              00B9       ; Here R7 has hardware adapter code or'ed with software bus code.
                              00B9       ; Translate specific nexus device type code into general adapter type code.
                              00B9       ;
                              00B9       GET_GEN_TYPE:
   00A4'CF44    57     90     00B9             MOVB    R7,W^CONFREG[R4]            ; Save nexus-device type in CONFREG.
   01E4'CF44    57     D0     00BF             MOVL    R7,W^CONFREGL[R4]           ; CONFREGL also filled in.
                55     D4     00C5             CLRL    R5                          ; Clear loop index.
                              00C7       30$:                                      ;
   50   0000'CF45      DE     00C7             MOVAL   W^ADAPTERS[R5],R0           ; Get address of adapter type code.
        0000'CF        9F     00CD             PUSHAB  W^NUM_PAGES                 ; Push addr of end of ADAPTERS array.
          8E    50     D1     00D1             CMPL    R0,(SP)+                    ; See if we went beyond array.
                3F     1E     00D4             BGEQU   END_NEXUS                   ; unrecognized adapter, do not map.
          60    57     D1     00D6             CMPL    R7,(R0)                     ; Adapter type match?
                04     13     00D9             BEQL    40$                         ; If EQL yes, adapter type match.
                55     D6     00DB             INCL    R5                          ; Increment loop index.
                E8     11     00DD             BRB     30$                         ; Look at next adapter.
                              00DF       40$:
                              00DF
                              00DF       ;
                              00DF       ; Store boot parameters.
                              00DF       ;
          5A    58     D1     00DF             CMPL    R8,R10                      ; Does PFN match boot adapter's PFN?
                15     12     00E2             BNEQ    MAP_NEXUS                   ; No; continue.
       60 A9    52     D0     00E4             MOVL    R2,RPB$L_ADPVIR(R9)         ; Store VA of boot adapter space.
       20 A9    54     D0     00E8             MOVL    R4,RPB$L_BOOTR1(R9)         ; Store boot adapter nexus number.
  51   54 A9    0D  00 EF     00EC             EXTZV   #0,#13, -                   ; Get offset into UNIBUS/QBUS I/O page.
                              00F2                     RPB$L_CSRPHY(R9),R1         ;
  58 A9   1000 C241    9E     00F2             MOVAB   <8*512>(R2)[R1], -          ; Set VA of UNIBUS/QBUS registers.
                              00F9                     RPB$L_CSRVIR(R9)            ;
                              00F9
                              00F9
                              00F9       ;
                              00F9       ; R5/ general adapter type; index into "general" adapter arrays.
                              00F9       ; For each adapter -
```

```
                     00F9  1042 ;       Map the # of pages specified in ADAPDESC macro
                     00F9  1043 ;       JSB to initialization routine specified in ADAPDESC macro
                     00F9  1044 ;
                     00F9  1045 MAP_NEXUS:
00E4'CF44    52   D0 00F9  1050           MOVL    R2,W^SBICONF[R4]          ; Save VA of adapter space in SBICONF.
51   0000'CF45   3C 00FF  1051           MOVZWL  W^NUM_PAGES[R5],R1        ; Get number of pages to map.
          6C   10 0105  1052           BSBB    MAP_PAGES                 ; Map the I/O pages.
51   0000'CF45   DE 0107  1053           MOVAL   W^INIT_ROUTINES[R5],R1    ; Get address of initialization routine.
          61   D5 010D  1054           TSTL    (R1)                      ; Initialization routine specified?
          04   13 010F  1055           BEQL    END_NEXUS                 ; Branch if none.
       00 B141 16 0111  1056           JSB     @(RT)[R1]                 ; Call initialization routine.
                0115  1057 END_NEXUS:
          54   D6 0115  1058           INCL    R4                        ; Increment CONFREG and SBICONF index.
        FF51   31 0117  1060           BRW     NXT_NEXUS                 ; Go do next nexus.
                011A  1064
```

```
                                011A  1066          .SBTTL  CREATE_ARRAYS
                                011A  1067 ;
                                011A  1068 ; CREATE_ARRAYS
                                011A  1069 ;
                                011A  1070 ;       Move the local CONFREG and SBICONF arrays into non-paged pool.
                                011A  1071 ;
                                011A  1072 ; Inputs:
                                011A  1073 ;       R4 - Number of nexuses on the system.
                                011A  1074 ;       CONFREG and SBICONF have been initialized.
                                011A  1075 ;
                                011A  1076 ; Outputs:
                                011A  1077 ;       R0 - R5 destroyed
                                011A  1078 ;       EXE$GL_CONFREG points to a copy of the CONFREG array in non-paged pool
                                011A  1079 ;       MMG$GL_SBICONF points to a copy of the SBICONF array in non-paged pool
                                011A  1080 ;       EXE$GL_NUMNEXUS contains the number of nexuses on the system
                                011A  1081 ;
                                011A  1082 ;
                                011A  1083 CREATE_ARRAYS:
00000000'GF    54    D0         011A  1084          MOVL    R4,G^EXE$GL_NUMNEXUS    ; Store number of nexuses on system.
     51    0C A444    DE         0121  1085          MOVAL   12(R4)[R4],R1           ; Allocate n bytes for CONFREG plus
                                0126  1086                                          ; 4n bytes for SBICONF + header
           51   6144    DE      0126  1087          MOVAL   (R1)[R4],R1             ; Another 4n bytes for CONFREGL.
                 01F9    30      012A  1088          BSBW    ALONPAGD                ; Get pool for CONFREG and SBICONF.
                   82    7C      012D  1089          CLRQ    (R2)+                   ; Clear out unused
              82   51    B0      012F  1090          MOVW    R1,(R2)+                ; Set in size
        82   0763 8F    B0      0132  1091          MOVW    #<DYN$C_CONF@8>!DYN$C_INIT,(R2)+ ; Set type and subtype
00000000'GF    62    9E         0137  1092          MOVAB   (R2),G^EXE$GL_CONFREG   ; Store address of system CONFREG.
           51   6244    9E      013E  1093          MOVAB   (R2)[R4],R1             ; Two steps to CONFREGL, 1st, SBICONF,
00000000'GF    51    D0         0142  1094          MOVL    R1,G^MMG$GL_SBICONF     ; Store address of system SBICONF.
00000000'GF    6144    DE       0149  1095          MOVAL   (R1)[R4],G^EXE$GL_CONFREGL ; And address of system CONFREGL.
                 14    BB      0151  1096          PUSHR   #^M<R2,R4>              ; Save pool address and nexus count.
     62   00A4'CF    54    28   0153  1097          MOVC3   R4,W^CONFREG,(R2)       ; Copy CONFREG to pool.
                 14    BA      0159  1098          POPR    #^M<R2,R4>              ; Retrieve pool address and nexus count.
           51   54    04    C5   015B  1099          MULL3   #4,R4,R1                ; Number of bytes in SBICONF.
                   7E  51    D0   015F  1100          MOVL    R1,-(SP)                ; Save, SBICONF size = CONFREGL size
   6244    00E4'CF    51    28   0162  1101          MOVC3   R1,W^SBICONF,(R2)[R4]   ; Copy SBICONF to pool.
                   51  8E    D0   0169  1102          MOVL    (SP)+,R1                ; Restore size of SBICONF and CONFREGL.
     63    01E4'CF    51    28   016C  1103          MOVC3   R1,W^CONFREGL,(R3)      ; Copy CONFREGL to pool.  R3 is output
                                0172  1104                                          ; from SBICONF MOVC3, so SBICONF and
                                0172  1105                                          ; CONFREGL must be adjacent.
                                0172  1106
                          05    0172  1107          RSB
```

```
                                0173  1109          .SBTTL  MAP_PAGES
                                0173  1110 ;++
                                0173  1111 ; INPUTS:
                                0173  1112 ;       R1/ Number of pages to map.
                                0173  1113 ;       R2/ VA of page to map.
                                0173  1114 ;       R3/ VA of system page table entry to be used.
                                0173  1115 ;       R8/ PFN of page(s) to map.
                                0173  1116 ;
                                0173  1117 ; OUTPUTS:
                                0173  1118 ;       R2,R3 updated; R1,R8 destroyed; all other registers preserved
                                0173  1119 ;
                                0173  1120 ;--
                                0173  1121
                                0173  1122 MAP_PAGES:
                                0173  1123
    83    58    90000000 8F     0173  1124          BISL3   #<PTE$M_VALID!PTE$C_KW>,R8,(R3)+
                                017B  1125                                          ; Map a page.
                          58    D6  017B  1126          INCL    R8                  ; Next PFN.
                    52    0200  C2  017D  1127          MOVAB   512(R2),R2          ; Next VA.
                     00000000'GF  D6  0182  1128          INCL    G^BOO$GL_SPTFREL    ; Next free entry.
    00000000'GF      00000000'GF  D1  0188  1129          CMPL    G^BOO$GL_SPTFREH, - ; Check for no more system page
                                0193  1130                  G^BOO$GL_SPTFREL    ; table entries.
                          04    15  0193  1131          BLEQ    ERROR_HALT          ; Branch if out of SPTEs.
                       DB 51    F5  0195  1132          SOBGTR  R1,MAP_PAGES        ; Map another page.
                          05    05  0198  1133          RSB                         ; All done.
                                0199  1134
                                0199  1135 ERROR_HALT:
                    51    02E4'CF  9E  0199  1136          MOVAB   W^NOSPT,R1          ; Set error message.
                                019E  1137 ERROR_HALT_1:
                          5B    D4  019E  1138          CLRL    R11                 ; Indicate console terminal.
                     00000000'GF  16  01A0  1139          JSB     G^EXE$OUTZSTRING    ; Output error message.
                                00  01A6  1140          HALT                        ; ***** FATAL ERROR *******
```

```
                                                                                .SBTTL  INI$UBSPACE
                                                                        ;++
                                                                        ;       Map UNIBUS space; initialize UNIBUS ADP.
                                                                        ;
                                                                        ; INPUTS:
                                                                        ;       R2 - VA of next free system page
                                                                        ;       R3 - VA of system page table entry to be used to map VA in R2
                                                                        ;       R4 - nexus identification number of this adapter
                                                                        ;     -8(R6) - PFN of this UNIBUS adapter's register space
                                                                        ;
                                                                        ; OUTPUTS:
                                                                        ;       UNIBUS space is mapped.
                                                                        ;       INI$UBADP is called to build an ADP block and initialize UNIBUS
                                                                        ;       adapter hardware.
                                                                        ;
                                                                        ;--

                                                                        INI$UBSPACE:

    58    01E4'CF44    DE                                               MOVAL   W^CONFREGL[R4],R8               ; R8 => CONFREGL slot.
 58   68     02    00   EF                                               EXTZV   #0,#2,(R8),R8                   ; Get UBA number.
    58     58    09    78                                               ASHL    #9,R8,R8                        ; Position UB number.

 58   00007FF0 8F    58   C3                                               SUBL3   R8,#<IO750$AL_UB0SP+^0760000/^X200>,R8
                                                                                                                ; Get PFN of UB I/O page.

51     10    D0                                                           MOVL    #16,R1                          ; Number of pages to map (UB/Qbus space).
   FFAF    30                                                           BSBW    MAP_PAGES                       ; Map I/O pages.
                                                                        ;
                                                                        ; Call adapter initialization routine.
                                                                        ;
                                                                        ;       BSBW    INI$UBADP               ; Init ADP block.
                                                                        ;       RSB
```

```
                                01C4  1339          .SBTTL  INI$UBADP - BUILD ADP AND INITIALIZE UBA
                                01C4  1340 ;+
                                01C4  1341 ; INI$UBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
                                01C4  1342 ; DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS.  A CALL IS
                                01C4  1343 ; THEN MADE TO UBA$INITIAL TO INITIALIZE THE ADAPTER HARDWARE.
                                01C4  1344 ;
                                01C4  1345 ; INPUT:
                                01C4  1346 ;       R4 - nexus identification number of this adapter
                                01C4  1347 ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                                01C4  1348 ;-
                                01C4  1349
                                01C4  1350 INI$UBADP:
                                01C4  1351
        01FF 8F     BB          01C4  1352          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; SAVE R0-R8
                                01C8  1353 ;
                                01C8  1354 ; Allocate and initialize Adapter Control Block (ADP).
                                01C8  1355 ;
  51    0019'CF     3C          01C8  1356          MOVZWL  W^CPU_ADPSIZE,R1        ; PICK UP LENGTH OF ADP
              0156  30          01CD  1357          BSBW    ALONPAGD                ; ALLOCATE SPACE FOR ADP
     08 A2    51   B0          01D0  1358          MOVW    R1,ADP$W_SIZE(R2)       ; SET SIZE INTO ADP BLOCK
     0A A2    01   90          01D4  1359          MOVB    #DYN$C_ADP,-            ; AND SET TYPE OF BLOCK
                                01D8  1360                  ADP$B_TYPE(R2)
     0E A2    01   B0          01D8  1361          MOVW    #ATS_UBA,-              ; SET TYPE OF ADAPTER
                                01DC  1362                  ADP$W_ADPTYPE(R2)
  62   00E4'CF44   D0          01DC  1363          MOVL    W^SBICONF[R4],-         ; SET VA OF CONFIGURATION REG
                                01E2  1364                  ADP$L_CSR(R2)
     0C A2    54   B0          01E2  1365          MOVW    R4,ADP$W_TR(R2)         ; SET TR NUMBER FOR ADAPTER
                                01E6  1366
     50    14 A2   DE          01E6  1367          MOVAL   ADP$L_DPQFL(R2),R0      ; ADDRESS OF DATA PATH WAIT QUEUE
        60   50   D0          01EA  1368          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
     04 A0   50   D0          01ED  1369          MOVL    R0,4(R0)                ;
                               01F1  1370
     50    30 A2   DE          01F1  1371          MOVAL   ADP$L_MRQFL(R2),R0      ; ADDRESS OF MAP WAIT Q'JEUE
        60   50   D0          01F5  1372          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
     04 A0   50   D0          01F8  1373          MOVL    R0,4(R0)                ;
           04 A2   D4          01FC  1374          CLRL    ADP$L_LINK(R2)          ; ZAP ADAPTER CHAIN LINK
            FDFE'  30          01FF  1375          BSBW    ADPLINK                 ; LINK ADP TO END OF LIST
                               0202  1376 ;
                               0202  1377 ; Initialize adapter interrupt vectors in System Control Block.
                               0202  1378 ;
  58    00000000'GF   D0          0202  1379          MOVL    G^EXE$GL_SCB,R8         ; GET SCB ADDRESS
                               0209  1380
                               0209  1387
                               0209  1447
                               0209  1449
                 28   DD          0209  1450          PUSHL   #NDT$_UB0               ; ASSUME UB0
           0200 C8   DE          020B  1451          MOVAL   ^X200(R8),-             ; GET VECTOR SPACE FOR UB0
              10 A2              020F  1452                  ADP$L_VECTOR(R2)        ;
     28   01E4'CF44   D1          0211  1453          CMPL    W^CONFREGL[R4],#NDT$_UB0; IS DEVICE TYPE = UB0?
                 0B   13          0217  1454          BEQL    10$                     ; BRANCH IF SO
                 6E   29          0219  1455          MOVL    #NDT$_UB1,(SP)          ; INDICATE UB1
10 A2    00000200 8F   C0          021C  1456          ADDL    #^X200,ADP$L_VECTOR(R2) ; STEP TO ITS VECTOR SPACE
        60 A2    0E   B0          0224  1457 10$:     MOVW    #^XE,ADP$W_DPBITMAP(R2) ; MARK DATAPATHS 1-3 AVAILABLE
           53    62   D0          0228  1458          MOVL    ADP$L_CSR(R2),R3        ; VIRTUAL ADDRESS OF ADAPTER
        53    0800 C3   9E          022B  1459          MOVAB   UBA$L_MAP(R3),R3        ; POINT TO MAPPING REGISTERS
        54    01F0 8F   3C          0230  1460          MOVZWL  #496,R4                 ; NUMBER OF UMR TO DISABLE
                 83   D4          0235  1461 20$:     CLRL    (R3)+                   ; DISABLE A UNIBUS MAP REGISTER
```

```
                FB 54    F5   0237  1462          SOBGTR  R4,20$                  ; LOOP THRU THEM ALL
      53  00000001'GF    DE   023A  1463          MOVAL   G^UBA$UNEXINT+1,R3      ; GET ADDR OF UNEXP INT SERVICE
                                0241  1464                                          ;  (+1 MEANS HANDLE ON INT STACK)
          54    0001'CF    DE   0241  1465          MOVAL   W^UBA$INT0+1,R4         ; SPECIAL CASE TO COUNT PASSIVE RELEASE
                                0246  1466
                                0246  1467 ;
                                0246  1468 ; INIT UB VECTORS TO UNEXPECTED INTERRUPT SERVICE
                                0246  1469 ;
           50    10 A2    D0   0246  1470          MOVL    ADP$L_VECTOR(R2),R0     ; GET ADDRESS OF VECTORS
              80    54    D0   024A  1471          MOVL    R4,(R0)+                ; SPECIAL CASE FOR VECTOR 0
           51    7F 8F    9A   024D  1472          MOVZBL  #<NUMUBAVEC-1>,R1       ; REST OF VECTORS
              80    53    D0   0251  1473 30$:     MOVL    R3,(R0)+                ; FILL VECTOR WITH UNEXP INT
                 FA 51    F5   0254  1474          SOBGTR  R1,30$                  ; FILL ALL VECTORS
              6E    29    91   0257  1475          CMPB    #NDT$_UB1,(SP)          ; IS THIS UB1?
                    3C    12   025A  1476          BNEQ    40$                     ; IF NOT, SKIP CODE
                                025C  1477 ;
                                025C  1478 ; SAVE CONTENTS OF SPTE'S MAPPING UB SPACE
                                025C  1479 ;
              54    52    D0   025C  1480          MOVL    R2,R4                   ; SAVE ADP ADDRESS
              52    62    D0   025F  1481          MOVL    ADP$L_CSR(R2),R2        ; GET VA OF ADAPTER
        00000000'GF    16   0262  1482          JSB     G^MMG$SVAPTECHK         ; GET ADDRESS OF SPTE MAPPING ADAPTER
           54 A4    63    D0   0268  1483          MOVL    (R3),ADP$L_UBASPTE(R4)  ; STORE CONTENTS OF SPTE IN ADP
        58 A4    20 A3    D0   026C  1484          MOVL    <8*4>(R3),ADP$L_UBASPTE+4(R4) ; SAME FOR I/O SPACE
                                0271  1485 ;
                                0271  1486 ; CALCULATE AND STORE VA OF IPEC REGISTER, WHICH CONTAINS BITS NEEDED
                                0271  1487 ; TO PROCESS POWERFAIL
                                0271  1488 ;
        00002464 8F    C1   0271  1489          ADDL3   #<8*^X200> + UAS$W_IP_CR1,- ; VA OF ADAPTER + OFFSET TO
           50 A4    52        0277  1490                  R2,ADP$L_UBASCB+12(R4)  ; I/O SPACE + OFFSET TO IPEC REGISTER
                                027A  1491 ;
                                027A  1492 ; STORE INTERRUPT CODE IN ADP, STORE ITS ADDRESS IN POWERFAIL INTERRUPT
                                027A  1493 ; VECTOR IN SCB, AND SAVE ITS ADDRESS IN ADP
                                027A  1494 ;
     48 A4    031E'CF    7D   027A  1495          MOVQ    W^UBA1INT,ADP$L_UBASCB+4(R4)
     4A A4    0000'CF    9E   0280  1496          MOVAB   W^EXE$UBAERR_INT,ADP$L_UBASCB+6(R4)
  01E4 C8      48 A4    DE   0286  1497          MOVAL   ADP$L_UBASCB+4(R4),^X1E4(R8)
           01E4 C8       D6   028C  1498          INCL    ^X1E4(R8)               ; USE INTERRUPT STACK
        44 A4    48 A4    DE   0290  1499          MOVAL   ADP$L_UBASCB+4(R4),ADP$L_UBASCB(R4)
                                0295  1500 ;
                                0295  1501 ; DONE WITH ADAPTER-SPECIFIC CODE
                                0295  1502 ;
              52    54    D0   0295  1503          MOVL    R4,R2                   ; RESTORE R2
              5E    04    C0   0298  1504 40$:     ADDL    #4,SP                   ; CLEAN STACK
                                029B  1505
                                029B  1507
                                029B  1508
                                029B  1536
                                029B  1537
                                029B  1558
                                029B  1559
                                029B  1601
                                029B  1602
                                029B  1651 ;
                                029B  1652 ; Now check for any UNIBUS memory that may be on the adapter. First we must
                                029B  1653 ; disable all the UNIBUS Map Registers so that there is no conflict in
                                029B  1654 ; which memory will respond.  Then we check all 248Kb of potential memory in
                                029B  1655 ; 8Kb chunks, since each disable bit on the 780 UBA represents 16 UMR's or
```

```
                                029B  1656 ; 8Kb of memory.  The number of registers is stored in the ADP and the
                                029B  1657 ; corresponding number withdrawn from the UMR map in the ADP.
                                029B  1658 ;
                                029B  1659
                     56   62    D0  029B  1661          MOVL    ADP$L_CSR(R2),R6        ; Pick up adapter pointer
                          51    D4  029E  1662          CLRL    R1                      ; Zero out number of UMR to disable
      57   08 AE  00000200  8F   C3  02A0  1664          SUBL3   #512,8(SP),R7           ; R7 = VA of last page of UNIBUS
           58   0C AE    04   C3  02A9  1665          SUBL3   #4,12(SP),R8            ; R8 = VA of SPTE mapping (R7)
      54   20 AE  00000200  8F   C3  02AE  1666          SUBL3   #512,32(SP),R4          ; R4 = PFN of first page of UNIBUS
                          68   DD  02B7  1667          PUSHL   (R8)                    ; Save contents of SPTE
                     53   54   D0  02B9  1668          MOVL    R4,R3                   ; Copy starting PFN
                     55   1F   D0  02BC  1669          MOVL    #31,R5                  ; 31 8Kb chunks to test
                                02BF  1670 50$:     INVALID R7                      ; Invalidate TB
                90000000  8F   C9  02C2  1671          BISL3   #<PTE$M_VALID!PTE$C_KW>,-
                     68   54        02C8  1672                   R4,(R8)             ; Map each page of UNIBUS
                     50   57   D0  02CA  1673          MOVL    R7,R0                   ; Address to check
                        FD30'  30  02CD  1674          BSBW    EXE$TEST_CSR            ; Validate it
                       0D   50   E9  02D0  1675          BLBC    R0,70$                  ; Not there
                     54   53   D1  02D3  1676          CMPL    R3,R4                   ; First time in?
                          04   13  02D6  1677          BEQL    60$                     ; Yes, skip next test
                          51   D5  02D8  1678          TSTL    R1                      ; Any registers already?
                          3A   13  02DA  1679          BEQL    80$                     ; No, memory not start at 0
                51   10  A1   9E  02DC  1680 60$:     MOVAB   16(R1),R1               ; Yes, up the count
                54   10  A4   9E  02E0  1681 70$:     MOVAB   16(R4),R4               ; Map Next 8Kb (16*512)
                     D8  55   F5  02E4  1682          SOBGTR  R5,50$                  ; Loop until done
                         68 8ED0  02E7  1683          POPL    (R8)                    ; Restore old contents of SPTE
                                02EA  1684          INVALID R7                      ; Invalidate TB
           0256 C2   51   B0  02ED  1686          MOVW    R1,ADP$W_UMR_DIS(R2)    ; Record number disabled
                                02F2  1688 ;
                                02F2  1689 ; Initialize fields for new UBA map register allocation.  Make it appear
                                02F2  1690 ;       that we have one contiguous array of 496 available map registers.
                                02F2  1691 ;       To do this we set ADP$L_MRACTMDRS to one (the number of active
                                02F2  1692 ;       map register descriptors for distinct contiguous areas),
                                02F2  1693 ;       ADP$W_MRNREGARY(0) to 496 (i.e the number of registers in this
                                02F2  1694 ;       contiguous range) and ADP$FREGARY(0) to 0 (i.e. the first register
                                02F2  1695 ;       in the range is register 0).
                                02F2  1696 ;
             5C A2    01   D0  02F2  1697          MOVL    #1,ADP$L_MRACTMDRS(R2)  ; 1 active map descriptor
  64 A2    01F0 8F    51   A3  02F6  1698          SUBW3   R1,#496,ADP$W_MRNREGARY(R2);  for a range of 496 registers
           015E C2    51   B0  02FD  1710          MOVW    R1,ADP$W_MRFREGARY(R2)  ;  starting at register zero.
             62 A2    01   AE  0302  1711          MNEGW   #1,ADP$W_MRNFENCE(R2)   ; Also init "fences" which preceed
           015C C2    01   AE  0306  1712          MNEGW   #1,ADP$W_MRFFENCE(R2)   ;  the two descriptor arrays.
                                030B  1713 ;
                                030B  1714 ; Initialize adapter hardware.
                                030B  1715 ;
                     54   62   D0  030B  1716          MOVL    ADP$L_CSR(R2),R4        ; Get CSR address to init
                        FCEF'  30  030E  1717          BSBW    UBA$INITIAL             ; And initialize adapter
                     01FF 8F   BA  0311  1718          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; Restore registers
                          05   05  0315  1719          RSB                             ; Return
                                0316  1720
                                0316  1722 ;
                                0316  1723 ; Error if UNIBUS memory not start at location 0
                                0316  1724 ;
                51  030D'CF    9E  0316  1725 80$:     MOVAB   W^BADUMR,R1             ; Set error message
                       FE80    31  031B  1726          BRW     ERROR_HALT_1            ; Put it out
                                031E  1728
```

```
                     031E  1802 ;
                     031E  1803 ; UBA INTERRUPT SERVICE HANDLER FOR 11/750.  THIS CODE IS PLACED
                     031E  1804 ; IN THE ADP, AND IT IS POINTED TO BY THE SCB VECTOR WHICH
                     031E  1805 ; HANDLES UBA POWERFAIL INTERRUPTS.  USING A JSB TO DISPATCH
                     031E  1806 ; TO THE ADAPTER POWERFAIL INTERRUPT CODE ALLOWS A POINTER WITH A
                     031E  1807 ; KNOWN OFFSET INTO THE ADP TO BE PUSHED ON THE STACK AND USED
                     031E  1808 ; BY THE CODE TO FIND THE ADP.
                     031E  1809 ;
                     031E  1810 UBA1INT:
00000000 9F   16     031E  1811          JSB     @#0                     ; ERROR ROUTINE IN ADPERR750
              0000   0324  1812          .WORD   0                       ; ZERO OUT REST OF QUADWORD
```

```
                                0326  1815          .SBTTL  INI$MBADP - BUILD ADP AND INITIALIZE MBA
                                0326  1816          .SBTTL  INI$DRADP - BUILD ADP AND INITIALIZE DR32
                                0326  1817          .SBTTL  INI$CIADP - BUILD ADP AND INITIALIZE CI
                                0326  1818 ;+
                                0326  1819 ; INI$MBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                                0326  1820 ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                                0326  1821 ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                                0326  1822 ; BY CALLING MBA$INITIAL.
                                0326  1823 ;
                                0326  1824 ; INI$DRADP IS CALLED AFTER MAPPING THE REGISTERS FOR THE DR32
                                0326  1825 ; ADAPTER.  THE ADAPTER CONTROL BLOCK, CRB, AND IDB ARE ALLOCATED
                                0326  1826 ; AND INITIALIZED.  THE ADAPTER HARDWARE IS THEN INITIALIZED BY
                                0326  1827 ; CALLING DR$INITIAL.
                                0326  1828 ;
                                0326  1829 ; INI$MBADP AND INI$DRADP SHARE COMMON CODE AFTER THE TABLE OF ADAPTER
                                0326  1830 ; SPECIFIC CONSTANTS IS SELECTED AND STORED IN R8.
                                0326  1831 ;
                                0326  1832 ; INPUT:
                                0326  1833 ;       R4 - nexus identification number of this adapter
                                0326  1834 ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                                0326  1835 ;
                                0326  1836 ; OUTPUTS:
                                0326  1837 ;       ALL REGISTERS PRESERVED
                                0326  1838 ;-
                                0326  1839
00000000'GF     17              0326  1840 ALONPAGD:JMP    G^INI$ALONONPAGED
                                032C  1841
                                032C  1842          .ENABL  LSB
                                032C  1843
                                032C  1844 INI$DRADP:                               ; INITIALIZE DR32 DATA STRUCTURES
                                032C  1845
        07FF  8F        BB      032C  1848          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
58      0011'CF         DE      0330  1849          MOVAL   W^DRTAB,R8              ; GET DR32 TABLE OF CONSTANTS
59      0000'CF         9E      0335  1850          MOVAB   W^DR$INT,R9             ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A      0000'CF         9E      033A  1851          MOVAB   W^DR$INITIAL,R10        ; ADDRESS OF DEVICE INITIALIZATION
              28        11      033F  1852          BRB     10$                     ; JOIN COMMON CODE
                                0341  1855
                                0341  1856 INI$CIADP:                               ; INITIALIZE CI DATA STRUCTURES
                                0341  1857
        07FF  8F        BB      0341  1860          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
58      0015'CF         DE      0345  1861          MOVAL   W^CITAB,R8              ; GET CI TABLE OF CONSTANTS
59      0000'CF         9E      034A  1862          MOVAB   W^CI$INT,R9             ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A      0000'CF         9E      034F  1863          MOVAB   W^CI$INITIAL,R10        ; ADDRESS OF DEVICE INITIALIZATION
              13        11      0354  1864          BRB     10$                     ; JOIN COMMON CODE
                                0356  1867
                                0356  1868 INI$MBADP:                               ; INIT MBA DATA STRUCTURES
                                0356  1869
        07FF  8F        BB      0356  1872          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ;
58      000D'CF         DE      035A  1873          MOVAL   W^MBATAB,R8             ; GET MBA TABLE OF CONSTANTS
59      0000'CF         9E      035F  1874          MOVAB   W^MBA$INT,R9            ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A      0000'CF         9E      0364  1875          MOVAB   W^MBA$INITIAL,R10       ; ADDRESS OF DEVICE INITIALIZATION
                                0369  1876 10$:                                    ;
                                0369  1877 ;
                                0369  1878 ; Allocate and initialize Channel Request Block.
                                0369  1879 ;
51      0048  8F        3C      0369  1880          MOVZWL  #CRB$C_LENGTH,R1        ; SET SIZE OF CRB
              B6        10      036E  1881          BSBB    ALONPAGD                ; ALLOCATE SPACE FOR CRB
```

```
                                                E 6
                08 A2    51    B0  0370  1882          MOVW    R1,CRB$W_SIZE(R2)           ; SET CORRECT SIZE
                0A A2    05    90  0374  1883          MOVB    #DYN$C_CRB,CRB$B_TYPE(R2) ; SET CORRECT TYPE
                   62    62    DE  0378  1884          MOVAL   CRB$L_WQFL(R2),CRB$L_WQFL(R2)   ; INITIALIZE WAIT QUEUE HEADER
                04 A2    62    DE  037B  1885          MOVAL   CRB$L_WQFL(R2),CRB$L_WQBL(R2)   ; FLINK AND BLINK
                50    24 A2    9E  037F  1886          MOVAB   CRB$L_INTD(R2),R0           ; SET ADDRESS OF INTD AREAD
          80    9F163CBB 8F    D0  0383  1887          MOVL    #^X9F163CBB,(R0)+           ; "PUSHR ^M<R2,R3,R4,R5>,JSB @#"
                   80    59    D0  038A  1888          MOVL    R9,(R0)+                    ; ADDR OF XXX$INT ROUTINE
                   80    D4    038D  1889          CLRL    (R0)+                       ; CLEAR OUT UNNEEDED AREA
                   60    5A    D0  038F  1890          MOVL    R10,(R0)                    ; ADDR OF XXX$INITIAL ROUTINE
                   5A    52    D0  0392  1891          MOVL    R2,R10                      ; SAVE CRB ADDRESS
                                   0395  1892 ;
                                   0395  1893 ; Allocate and initialize Interrupt Dispatch Block.
                                   0395  1894 ;
                   51    68    9A  0395  1895          MOVZBL  ADPTAB_IDBUNITS(R8),R1  ; GET # OF IDB UNITS
          51    00000038 9F41    DE  0398  1896          MOVAL   @#IDB$C_LENGTH[R1],R1   ; GET TOTAL SIZE OF IDB
                      84    10  03A0  1897          BSBB    ALONPAGD                    ; ALLOCATE SPACE FOR CRB
                08 A2    51    B0  03A2  1898          MOVW    R1,IDB$W_SIZE(R2)           ; SET STRUCTURE SIZE
                0A A2    09    90  03A6  1899          MOVB    #DYN$C_IDB, -               ; AND TYPE CODE
                                   03AA  1900                  IDB$B_TYPE(R2)
                      68    9B  03AA  1901          MOVZBW  ADPTAB_IDBUNITS(R8),-   ; SET COUNT OF UNITS
                   0C A2          03AC  1902                  IDB$W_UNITS(R2)
                62    00E4'CF44    D0  03AE  1903          MOVL    W^SBICONF[R4], -            ; SET CSR ADDRESS TO
                                   03B4  1904                  IDB$L_CSR(R2)               ;  START OF ADAPTER REG SPACE
                2C AA    52    D0  03B4  1905          MOVL    R2, -                       ; SET ADDRESS OF IDB INTO CRB
                                   03B8  1906                  CRB$L_INTD+VEC$L_IDB(R10)
                   59    52    D0  03B8  1907          MOVL    R2,R9                       ; SAVE ADDRESS OF IDB
                                   03BB  1908 ;
                                   03BB  1909 ; Allocate and initialize Adapter Control Block (ADP).
                                   03BB  1910 ;
                51    01 A8    3C  03BB  1911          MOVZWL  ADPTAB_ADPLEN(R8),R1    ; GET SIZE OF ADAPTER
                      FF64    30  03BF  1912          BSBW    ALONPAGD                    ; ALLOCATE SPACE FOR CRB
                08 A2    51    B0  03C2  1913          MOVW    R1,ADP$W_SIZE(R2)           ; SET SIZE OF STRUCTURE
                0A A2    01    90  03C6  1914          MOVB    #DYN$C_ADP,ADP$B_TYPE(R2); AND TYPE CODE
                   62    69    D0  03CA  1915          MOVL    IDB$L_CSR(R9),ADP$L_CSR(R2); SET ADDRESS OF CONFIGURATION REGISTER
                0C A2    54    B0  03CD  1916          MOVW    R4,ADP$W_TR(R2)             ; SET TR/SLOT-16 NUMBER OF ADAPTER
                      03 A8    9B  03D1  1917          MOVZBW  ADPTAB_ATYPE(R8),-      ; SET THE ADAPTER TYPE
                   0E A2          03D4  1918                  ADP$W_ADPTYPE(R2)
                10 A2    5A    D0  03D6  1919          MOVL    R10,ADP$L_CRB(R2)           ; POINT ADP TO CRB
                                   03DA  1920 ;       CMPW    ADP$W_ADPTYPE(R2),#AT$_CI ; CI?
                                   03DA  1921 ;       BEQL    20$                         ; YES, DO NOT CONNECT UP VECTORS
                                   03DA  1922 ;
                                   03DA  1923 ; Initialize adapter interrupt vectors in System Control Block.
                                   03DA  1924 ;
          50    00000000'GF      D0  03DA  1925          MOVL    G^EXE$GL_SCB,R0             ; GET ADDRESS OF SCB
             55    5B    09    78  03E1  1926          ASHL    #9,R11,R5                   ; Turn SCB page offset into byte offset.
                   50    55    C0  03E5  1927          ADDL    R5,R0                       ; set to beginning of correct SCB page.
          54    54    04    00    EF  03E8  1928          EXTZV   #0,#4,R4,R4                 ; Use low 4 bits of nexus number.
             50    0100 C044      DE  03ED  1929          MOVAL   ^X100(R0)[R4],R0            ; COMPUTE ADDR OF 1ST VECTOR
                1C A2    50    D0  03F3  1930          MOVL    R0,ADP$L_AVECTOR(R2)        ; SAVE ADDR OF ADAPTER'S SCB VECTORS
                60    25 AA    DE  03F7  1931          MOVAL   CRB$L_INTD+1(R10),(R0)      ; CONNECT VECTOR TO CRB CODE
             40 A0    25 AA    DE  03FB  1932          MOVAL   CRB$L_INTD+1(R10),64(R0); SAME FOR
          0080 C0     25 AA    DE  0400  1933          MOVAL   CRB$L_INTD+1(R10),128(R0); ALL FOUR
          00C0 C0     25 AA    DE  0406  1934          MOVAL   CRB$L_INTD+1(R10),192(R0); VECTORS
                                   040C  1935 ;
                                   040C  1936 ; Continue with ADP initialization.
                                   040C  1937 ;
             14 A2    25 AA    DE  040C  1938 20$:     MOVAL   CRB$L_INTD+1(R10), -    ; SAVE SCB VECTOR CONTENTS IN ADP
```

```
                     0411  1939                       ADP$L_MBASCB(R2)
          0C   BB    0411  1940             PUSHR     #^M<R2,R3>                      ; SAVE SOME REGISTERS
       55 52   D0    0413  1941             MOVL      R2,R5                           ; COPY ADP ADDRESS
       52 62   D0    0416  1942             MOVL      ADP$L_CSR(R2),R2                ; VIRTUAL ADDRESS OF ADAPTER
00000000'GF   16    0419  1943             JSB       G^MMG$SVAPTECHK                 ; ADDRESS OF SPTE THAT MAPS ADAPTER
 18 A5    63   D0    041F  1944             MOVL      (R3),ADP$L_MBASPTE(R5)          ; SAVE CONTENTS OF SPTE
          0C   BA    0423  1945             POPR      #^M<R2,R3>                      ; RESTORE REGISTERS
 38 AA    52   D0    0425  1946             MOVL      R2,CRB$L_INTD+VEC$L_ADP(R10)    ; SET CRB POINTER TO ADP
 14 A9    52   D0    0429  1947             MOVL      R2,IDB$L_ADP(R9)                ; AND INTO IDB
       FBD0'   30    042D  1948             BSBW      ADPLINK                         ; LINK ADP TO END OF CHAIN
                     0430  1949 ;
                     0430  1950 ; Initialize adapter hardware.
                     0430  1951 ;
       55 59   D0    0430  1952             MOVL      R9,R5                           ; ADDRESS OF IDB
       54 65   D0    0433  1953             MOVL      IDB$L_CSR(R5),R4                ; ADDRESS OF CONFIGURATION REGISTER 0
          30   BA    0436  1954             JSB       @CRB$L_INTD+VEC$L_INITIAL(R10) ; INIT ADAPTER
     07FF 8F   BA    0439  1955             POPR      #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10>; RESTORE ALL REGISTERS
          05   05    043D  1956             RSB                                       ; RETURN
                     043E  1957
                     043E  1958             .DSABL    LSB
```

```
                                G 6
043E  1997          .SBTTL  INI$KDZ11
043E  1998 ;++
043E  1999 ;
043E  2000 ; INPUTS:
043E  2001 ;        R2 - VA of next free system page
043E  2002 ;        R3 - VA of system page table entry to be used to map VA in R2
043E  2003 ;        R4 - nexus identification number of this adapter
043E  2004 ;
043E  2005 ; OUTPUTS:
043E  2006 ;
043E  2007 ;--
043E  2008
043E  2009 INI$KDZ11:
043E  2010
05  043E  2029          RSB                             ; Return to caller.
```

```
                                043F  2031          .SBTTL  INI$CONSOLE, init data structures for console
                                043F  2032 ;++
                                043F  2033 ; FUNCTIONAL DESCRIPTION:
                                043F  2034 ;
                                043F  2035 ;       This routine is executed only once, during system initialization.
                                043F  2036 ;       It initializes the CRB and IDB for boot/console device.
                                043F  2037 ;
                                043F  2038 ;       This routine is called from INIT.
                                043F  2039 ;
                                043F  2040 ; INPUTS:
                                043F  2041 ;
                                043F  2042 ;       R3 -->  DISK [CLASS] DRIVER DDB
                                043F  2043 ;       R4 -->  DISK [CLASS] DRIVER DPT
                                043F  2044 ;       R5 -->  DISK [CLASS] DRIVER UCB
                                043F  2045 ;       R6 -->  RPB
                                043F  2046 ;       R7 -->  ADP FOR EITHER A REAL DISK OR A PORT
                                043F  2047 ;       R9 -->  PORT DRIVER DPT (IF PRESENT)
                                043F  2048 ;       R10-->  PORT DIRVER UCB (IF PRESENT)
                                043F  2049 ;
                                043F  2050 ;--
                                043F  2051
                                043F  2052 INI$CONSOLE::
                                043F  2053          .ENABL  LSB
                                043F  2054
          66 A6    91           043F  2056          CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
          40 8F                 0442  2057                  #BTD$K_CONSOLE          ; STORAGE DEVICE?
             10    12           0444  2058          BNEQ    BLD_CRB                 ; NO
41534303 8F    D0              0446  2059          MOVL    #^A7CSA/@8+3,-          ; YES, SET DEVICE NAME
          14 A3                044C  2060                  DDB$T_NAME(R3)          ; COUNTED STRING
                               044E  2062
                               044E  2067
             57    D4           044E  2070          CLRL    R7                      ; CLEAR ADP POINTER
          54 A5    01    B0     0450  2071          MOVW    #1,UCB$W_UNIT(R5)       ; SET UNIT NUMBER TO 1
                   0D    11     0454  2072          BRB     FILL_CRB
                               0456  2075
                               0456  2076 ;
                               0456  2077 ; NOW BUILD THE AUXILIARY DATA BLOCKS (CRB,IDB)
                               0456  2078 ;
                               0456  2079 BLD_CRB:
          58    10 A7    D0     0456  2080          MOVL    ADP$L_CRB(R7),R8        ; GET ADDRESS OF CRB IF IT EXISTS
          0E A7    01    B1     045A  2081          CMPW    #AT$_UBA,ADP$W_ADPTYPE(R7); IS THIS A UNIBUS ADAPTER?
                   03    13     045E  2082          BEQL    FILL_CRB                ; YES, ALLOCATE CRB
                 008A    31     0460  2083          BRW     100$                    ; NO, CRB/IDB ALREADY ALLOCATED
                               0463  2084
                               0463  2085 FILL_CRB:
          00000000'9F    16     0463  2086          JSB     @#INI$ALLOC_CRB         ; GO ALLOCATE AND SETUP CRB
24 A2     9F163FBB 8F    D0     0469  2087          MOVL    #^X9F163FBB,CRB$L_INTD(R2) ; SET PUSHR #^M<R0,...R5>
                               0471  2088                                          ;  JSB @#0 INTO INTERRUPT DISPATCH
          38 A2    57    D0     0471  2089          MOVL    R7,CRB$L_INTD+VEC$L_ADP(R2)     ; SET POINTER TO ADP
             58    52    D0     0475  2090          MOVL    R2,R8                   ; SAVE CRB POINTER
       51    0058 8F    3C     0478  2091          MOVZWL  #<IDB$C_LENGTH+<8*4>>,R1; SIZE TO ALLOCATE FOR IDB
          00000000'9F    16     047D  2092          JSB     @#INI$ALLONONPAGED      ; ALLOCATE IDB
          08 A2    51    B0     0483  2093          MOVW    R1,IDB$W_SIZE(R2)       ; SET SIZE OF IDB
          0A A2    09    90     0487  2094          MOVB    #DYN$C_IDB,IDB$B_TYPE(R2); AND STRUCTURE TYPE CODE
          2C A8    52    D0     048B  2095          MOVL    R2,CRB$L_INTD+VEC$L_IDB(R8) ; SET IDB INTO CRB
                               048F  2096
                66 A6    91     048F  2099          CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
```

```
                        40 8F      0492  2100                 #BTD$K_CONSOLE          ; STORAGE DEVICE?
                           26  12  0494  2101         BNEQ    10$                     ; NO
50   000000F0 8F   00000000'9F  C1  0496  2102         ADDL3   @#EXE$GL_SCB,#^XF0,R0   ; YES, GET ADDRESS OF VECTOR IN SCB
                        80   25 A8  DE  04A2  2103         MOVAL   CRB$L_INTD+1(R8),(R0)+  ; SET ADDR IN 1ST VECTOR
                        60   49 A8  DE  04A6  2104         MOVAL   CRB$L_INTD2+1(R8),(R0)  ; SET ADDR IN 2ND VECTOR
              48 A8   9F163FBB 8F   D0  04AA  2105         MOVL    #^X9F163FBB,CRB$L_INTD2(R8) ; STORE PUSHR #^M<R0...R5>
                                    04B2  2106                                         ; JMP @# IN 2ND INT. DISPATCH
                        50 A8   52  D0  04B2  2107         MOVL    R2,CRB$L_INTD2+VEC$L_IDB(R8); STORE ADDRESS OF IDB IN CRB
                        2C B8   1F  D0  04B6  2108         MOVL    #PR$_CSTD, -            ; STORE IPR NUMBER OF CONSOLE INTERFACE
                                    04BA  2109                 @CRB$L_INTD+8(R8)       ; REGISTER AS DEVICE CSR ADDRESS
                                 31 11  04BA  2110         BRB     100$
                                    04BC  2113
                        62   58 A6  D0  04BC  2114 10$:     MOVL    RPB$L_CSRVIR(R6), -     ; SAVE BOOT DEVICE CSR ADDRESS
                                    04C0  2115                 IDB$L_CSR(R2)           ; IN INTERRUPT DISPATCH BLOCK
                                 11 91  04C0  2116         CMPB    #BTD$K_UDA,-            ; LOW ORDER BYTE OF ORIGINAL R0 TELLS
                              66 A6     04C2  2117                 RPB$B_DEVTYP(R6)        ;  BOOT DEVICE TYPE.
                                 08 12  04C4  2118         BNEQ    20$                     ; IF NOT BOOTING FROM A UDA BRANCH
                                    04C6  2119                                         ;  AROUND.
              00000000'9F   58 A6  D0  04C6  2120         MOVL    RPB$L_CSRVIR(R6), -     ; COPY VIRTUAL ADDRESS OF UDA PORT CSR
                                    04CE  2121                 @#BOO$GB_SYSTEMID       ;  TO LOW ORDER LONGWORD OF SYSTEMID
                                    04CE  2122 20$:
                        14 A2   57  D0  04CE  2123         MOVL    R7,IDB$L_ADP(R2)        ; POINT IDB TO ADP
                        50   1E A6  3C  04D2  2124         MOVZWL  RPB$W_R00BVEC(R6),R0    ; GET USER SPECIFIED VECTOR
                                 0A 12  04D6  2125         BNEQ    30$                     ; BRANCH IF VECTOR SPECIFIED
                        50   66 A6  9A  04D8  2126         MOVZBL  RPB$B_DEVTYP(R6),R0     ; ELSE GET DEVICE TYPE CODE
                  50    FFFE'CF40  3C  04DC  2127         MOVZWL  W^BOOTVECTOR-2[R0],R0   ; GET DEFAULT INTERRUPT VECTOR
                     50   10 B740  9E  04E2  2128 30$:     MOVAB   @ADP$L_VECTOR(R7)[R0],R0; COMPUTE ADDRESS OF VECTOR
                        60   26 A8  9E  04E7  2129         MOVAB   CRB$L_INTD+2(R8),(R0)   ; SET ADDR OF INTERRUPT VECTOR
                                    04EB  2130
                                 60 D7  04EB  2133         DECL    (R0)                    ;  BACK TWO BYTES TO PUSHR, +1 TO
                                    04ED  2136                                         ;
                                    04ED  2137 100$:
                                 05 04ED  2138                 RSB                     ; RETURN
                                    04EE  2139                 .DISABLE LSB
```

```
                                        .SBTTL  EXE$INI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ; EXE$INI_TIMWAIT initializes EXE$GL_TENUSEC and EXE$GL_UBDELAY, cells used
                                ; in the time-wait macros.  The first data cell, EXE$GL_TENUSEC, is the number
                                ; of times the following loop will be executed in ten u-seconds.  This is
                                ; done once here to calibrate the loop instead of reading the processor clock.
                                ; The resulting number is used in the system macros TIMEWAIT and TIMEDWAIT.
                                ;
                                ; The first step is to initialize EXE$GL_UBDELAY.  If the bit test instruction
                                ; in the TIMEWAIT macro is executed too rapidly in a loop, it can saturate the
                                ; Unibus.  EXE$GL_UBDELAY is used to introduce a 3 microsecond delay loop into
                                ; the TIMEWAIT bit test loop.
                                ;
                                ; This routine is called only once, from INIT.
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       NONE
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       Time-of-day processor clock.
                                ;       Interval timers.
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       R0 - Destroyed.
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       EXE$GL_TENUSEC - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                                ;                        macros loop for 10 micro-seconds.
                                ;
                                ;       EXE$GL_UBDELAY - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                                ;                        macros loop for 3 micro-seconds in the unibus delay
                                ;                        loop.
                                ;
                                ;--
                                
                                EXE$INI_TIMWAIT::                       ; Initialize time-wait data cells
                                        .ENABLE LSB
                                
                                
                                
 19   00    DA                          MTPR    #0,#PR750$_NICR         ; Initialize next interval count register.
                                
                                
                                
                                
 7E  00004E20 8F   D0                   MOVL    #20000,-(SP)            ; # of times to execute timed loop.
       18   11    DA                   MTPR    #^X11,#PR$_ICCS         ; Start clock, no interrupts.
                                
                                ; * * * start of loop to time * * *
       FD 6E    F5                  10$:    SOBGTR  (SP),10$            ; Delay loop.
                                ; * * * end of loop to time * * *
                                
                                
```

```
                     50   1A   DB  04FE  2215          MFPR    #PR750$_ICR,R0          ; Read total time to execute loop.
                                  0501  2217
                                  0501  2221
                                  0501  2225
                     18   00   DA  0501  2226          MTPR    #0,#PR$_ICCS            ; Shut off clock.
00000000'GF    0000EA60 8F   50   C7  0504  2227          DIVL3   R0,#60000,G^EXE$GL_UBDELAY; Calculate number of times through
                  00000000'GF   D6  0510  2228          INCL    G^EXE$GL_UBDELAY        ; loop to delay 3 microseconds.
                                  0516  2229
                                  0516  2233
                     19   00   DA  0516  2235          MTPR    #0,#PR750$_NICR         ; Initialize next interval count register.
                                  0519  2237
                                  0519  2241
                                  0519  2245
                50   00004E20 8F   D0  0519  2246          MOVL    #20000,R0               ; Number of times to execute test loop
                6E   00000000'GF   D0  0520  2247          MOVL    G^EXE$GL_UBDELAY,(SP)   ; Get delay loop iteration count.
                        18   11   DA  0527  2248          MTPR    #^X11,#PR$_ICCS         ; Start clock, no interrupts
                                  052A  2249
                                  052A  2250 ; **** Start of loop to time
          00000538'EF    8000 8F   B3  052A  2251 20$:     BITW    #^X8000,40$             ; Random BITx instruction to time
                              03   12  0533  2252          BNEQ    40$                     ; Random conditional branch instruction
                           FD 6E   F5  0535  2253 30$:     SOBGTR  (SP),30$                ; Delay 3 microseconds.
                           EF 50   F5  0538  2254 40$:     SOBGTR  R0,20$                  ; Loop
                                  053B  2255 ; **** End of loop to time
                                  053B  2256
                                  053B  2260
                     50   1A   DB  053B  2262          MFPR    #PR750$_ICR,R0          ; Read total time to execute loop.
                                  053E  2264
                                  053E  2268
                     18   00   DA  053E  2272          MTPR    #0,#PR$_ICCS            ; Shut clock off
                              8E   D5  0541  2273          TSTL    (SP)+                   ; Pop delay loop index off stack.
00000000'GF    00030D40 8F   50   C7  0543  2274          DIVL3   R0,#200000,G^EXE$GL_TENUSEC ; Calculate number of times to
                  00000000'GF   D6  054F  2275          INCL    G^EXE$GL_TENUSEC        ; execute the loop to kill 10 u-secs.
                                  0555  2276
                                  0555  2289
                              05  0555  2290          RSB                             ; Return
                                  0556  2291          .DISABLE LSB
```

```
                     0556  2299          .SBTTL  EXE$INIT_TODR  -  SET SYSTEM TIME TO CORRECT VALUE AT STARTUP
                     0556  2300 ;++
                     0556  2301 ; FUNCTIONAL DESCRIPTION:
                     0556  2302 ;
                     0556  2303 ;       EXE$INIT_TODR SOLICITS THE CORRECT TIME FROM THE OPERATOR IF NECESSARY,
                     0556  2304 ;       CONVERTS THE ASCII RESPONSE TO BINARY FORMAT AND CALLS AN INTERNAL
                     0556  2305 ;       ENTRY POINT OF THE $SETIME SYSTEM SERVICE TO SET THE NEW SYSTEM TIME
                     0556  2306 ;       IN MEMORY WITHOUT MODIFYING THE CONTENTS OF THE SYSTEM DISK.
                     0556  2307 ;
                     0556  2308 ;       IF THE TIME WOULD NORMALLY BE SOLICITED FROM AN OPERATOR, BECAUSE
                     0556  2309 ;       THE HARDWARE TIME OF YEAR CLOCK IS ZERO, THEN THE SYSGEN PARAMETER
                     0556  2310 ;       "TPWAIT" IS CHECKED.  IF IT IS ZERO, THEN IT IS ASSUMED THAT NO
                     0556  2311 ;       OPERATOR IS PRESENT AND THE SYSTEM IS BOOTED USING THE LAST TIME
                     0556  2312 ;       RECORDED IN THE SYSTEM IMAGE.  IF THE PARAMETER IS NON ZERO THEN
                     0556  2313 ;       THAT TIME IS USED AS THE MAXIMUM TIME TO WAIT BEFOR ASSUMING THAT
                     0556  2314 ;       THERE IS NO OPERATOR AND BOOTING ANY WAY.  IF THE PARAMETER IS
                     0556  2315 ;       NEGATIVE, THE SYSTEM WILL WAIT FOREVER.
                     0556  2316 ;
                     0556  2317 ;       THIS ROUTINE IS CALLED ONLY ONCE, FROM SYSINIT OR STASYSGEN.
                     0556  2318 ;
                     0556  2319 ; INPUT PARAMETERS:
                     0556  2320 ;
                     0556  2321 ;       NONE
                     0556  2322 ;
                     0556  2323 ; IMPLICIT INPUTS:
                     0556  2324 ;
                     0556  2325 ;       TIME-OF-DAY PROCESSOR CLOCK.
                     0556  2326 ;
                     0556  2327 ; OUTPUT PARAMETERS:
                     0556  2328 ;
                     0556  2329 ;       R0,R1 - DESTROYED
                     0556  2330 ;
                     0556  2331 ; IMPLICIT OUTPUTS:
                     0556  2332 ;
                     0556  2333 ;       EXE$GQ_SYSTIME - SET TO CURRENT TIME IN 100 NANOSECOND UNITS SINCE
                     0556  2334 ;                        17-NOV-1858  00:00:00.
                     0556  2335 ;
                     0556  2336 ;--
                     0556  2337
                     0556  2338 ;
                     0556  2339 ; Stack storage offsets:
                     0556  2340 ;
                00000000  0556  2341 TTCHAN  = ^X00                          ; CHANNEL FOR TERMINAL (LONGWORD)
                00000004  0556  2342 TTNAME  = ^X04                          ; STRING DESCRIPTOR FOR OPERATOR'S TERM
                0000000C  0556  2343 TMPDESC = ^X0C                          ; TEMPORY STRING DESCRIPTOR (QUADWORD)
                00000014  0556  2344 INTIME  = ^X14                          ; INPUT TIME VALUE (QUADWORD)
                0000001C  0556  2345 LINBUF  = ^X1C                          ; INPUT LINE BUFFER (5 LONGWORDS)
                00000014  0556  2346 LINBUFSIZ = ^X14                        ;  (LENGTH OF LINE BUFFER IN BYTES)
                          0556  2347
                          0556  2348 ;
                          0556  2349 ; PURE DATA
                          0556  2350 ;
                          0556  2351 TERM_NAMADR:
                   30 41 50 4F    0556  2352          .ASCII  \OPA0\            ; DEVICE NAME FOR OPERATOR'S TERMINAL
                      00000004    055A  2353 TERM_NAMSIZ = . - TERM_NAMADR
74 61 64 20 64 69 6C 61 76 6E 69 00'  055A  2354 TIMERR: .ASCIC  \invalid date/time\     ;
                   65 6D 69 74 2F 65  0566
```

```
                                 11  055A
                                     056C  2355 TIMEPROMPT:
                                 33' 056C  2356          .BYTE   NPROMPT
54 4E 45 20 45 53 41 45 4C 50 0A 0D  056D  2357          .ASCII  <13><10>/PLEASE ENTER DATE AND TIME (DD-MMM-YYYY  HH:MM)  /
20 44 4E 41 20 45 54 41 44 20 52 45  0579
4D 4D 4D 2D 44 44 28 20 45 4D 49 54  0585
4D 4D 3A 48 48 20 20 59 59 59 59 2D  0591
                              20 20  059D
                          00000033   05A0  2358 NPROMPT=.-TIMEPROMPT-1
                                     05A0  2359
                                     05A0  2360
                                     05A0  2361 EXE$INIT_TODR::                          ; SET CORRECT TIME
                                     05A0  2362          .ENABLE LSB
                        077C 8F   BB 05A0  2363          PUSHR   #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; SAVE REGISTERS
                          5E  30  C2 05A4  2364          SUBL    #4*12,SP                    ; SCRATCH STORAGE
                          56  5E  D0 05A7  2365          MOVL    SP,R6                       ; SAVE ADDRESS OF SCRATCH STORAGE
                    04 A6     04  9A 05AA  2366          MOVZBL  #TERM_NAMSIZ,TTNAME(R6) ; SET SIZE OF OPERATOR'S TERM NAME AND
                08 A6     FFA4 CF  9E 05AE  2367          MOVAB   W^TERM_NAMADR,TTNAME+4(R6) ;  PIC ADDRESS INTO TERM NAME DESC
            1C 00000000'GF    00'  E0 05B4  2368          BBS     S^#EXE$V_SETTIME,G^EXE$GL_FLAGS,READTIME ; BR TO SOLICIT TIME
                                     05BC  2369
                                     05BC  2370
                                     05BC  2374
                        50  1B   DB 05BC  2376          MFPR    #PR750$_TODR,R0             ; GET TIME OF DAY CLOCK VALUE
                                     05BF  2378
                                     05BF  2382
                                     05BF  2386
        59  00000000'GF  50   C3 05BF  2388          SUBL3   R0,G^EXE$GL_TODR,R9         ; GET TOD DELTA TIME (10 MS UNITS)
                          09   1B 05C7  2389          BLEQU   5$                          ; BRANCH IF TIME IS LATER
                0083D600 8F   59   D1 05C9  2390          CMPL    R9,#24*60*60*100            ; CHECK FOR SETBACK OF ONE DAY
                          06   1E 05D0  2391          BGEQU   READTIME                    ; MORE, MUST SOLICIT TIME
                       14 A6   7C 05D2  2396 5$:       CLRQ    INTIME(R6)                  ; NULL ARGUMENT FOR EXE$SETIME_INT
                        00C7   31 05D5  2397          BRW     200$                        ; RETURN TO CALLER
                                     05D8  2398
                                     05D8  2399 READTIME:                                ; SOLICIT TIME
                          59   D4 05D8  2400          CLRL    R9                          ; CLEAR A FLAG
        58  00000000'GF       32 05DA  2401          CVTWL   G^SGN$GW_TPWAIT,R8          ; PICK UP TIMEOUT WAIT INTERVAL
                          14   14 05E1  2402          BGTR    8$                          ; POSITIVE, WAIT THAT PERIOD ONCE
                          0D   19 05E3  2403          BLSS    7$                          ; NEGATIVE IS WAIT FOREVER
                                     05E5  2404 6$:
50      00000000'GF       01   C1 05E5  2406          ADDL3   #1,G^EXE$GL_TODR,R0         ; ZERO, SET TIME-OF-DAY CLOCK TO
                                     05ED  2408
                                     05ED  2412
                    1B    50   DA 05ED  2414          MTPR    R0,#PR750$_TODR             ;   KNOWN VALUE + 10 MSEC AND FINISH UP
                                     05F0  2416
                                     05F0  2420
                                     05F0  2424
                          E0   11 05F0  2426          BRB     5$                          ;
                                     05F2  2428
                                     05F2  2433
                    58    14   D0 05F2  2434 7$:       MOVL    #20,R8                      ; STARTING WAIT
                          59   D6 05F5  2435          INCL    R9                          ; NEGATIVE - WAIT FOREVER
                                     05F7  2436 8$:       $ASSIGN_S       TTNAME(R6),TTCHAN(R6) ; AND ASSIGN TO INPUT DEVICE
                      DD  50   E9 0605  2437          BLBC    R0,6$                       ; ERROR - FALL BACK TO STORED TIME
                52    FF60 CF   9E 0608  2438 10$:     MOVAB   W^TIMEPROMPT,R2             ; GET ADDRESS OF PROMPT STRING
                      53   82   9A 060D  2439          MOVZBL  (R2)+,R3                    ; AND LENGTH
                                     0610  2440          $QIOW_S #0,W^TTCHAN(R6),-           ; PROMPT AND READ TIME
                                     0610  2441                  #<IO$_READPROMPT!IO$M_PURGE!IO$M_TIMED!IO$M_CVTLOW>,-
```

```
                     0610  2442                    TMPDESC(R6),,-                  ;  I/O STATUS BLOCK , NO AST OR PARAM
                     0610  2443                    LINBUF(R6),#LINBUFSIZ,-         ;  BUFFER ADDRESS AND SIZE
                     0610  2444                    R8,#0,-                         ;  TIME OUT
                     0610  2445                    R2,R3                           ;  PROMPT ADDRESS AND SIZE
          AD 50 E9   0635  2446            BLBC    R0,6$                           ; ERROR - FALL BACK TO STORED TIME
     54   0C A6 7D   0638  2447            MOVQ    TMPDESC(R6),R4                  ; GET COMPLETION STATUS
          0D 54 E8   063C  2448            BLBS    R4,20$                          ; CONTINUE IF SUCCESSFUL READ
          A3 59 E9   063F  2449            BLBC    R9,6$                           ; FAILED ON ONE-TIME READ, RETURN
  58   01 A848 9E   0642  2450            MOVAB   1(R8)[R8],R8                    ; (2 * TIMEOUT) + 1
       58   58 3C   0647  2451            MOVZWL  R8,R8                           ; BOUND TIMEOUT
            BC 11   064A  2452            BRB     10$                             ; TRY AGAIN FOR TIME
                    064C  2453 20$:                                               ; SOMETHING WAS INPUT
 0C A6   0E A6 3C   064C  2454            MOVZWL  TMPDESC+2(R6),TMPDESC(R6) ; FORM DESCRIPTOR FOR BUFFER
 10 A6   1C A6 9E   0651  2455            MOVAB   LINBUF(R6),TMPDESC+4(R6) ; SET DESCRIPTOR ADDRESS
                    0656  2456            $BINTIM_S TMPDESC(R6),INTIME(R6) ; CONVERT TO BINARY TIME
          05 50 E9  0663  2457            BLBC    R0,89$                          ; INVALID TIME
          18 A6 D5  0666  2458            TSTL    INTIME+4(R6)                    ; CHECK FOR DELTA TIME
             2A 14  0669  2459            BGTR    100$                            ; BRANCH IF NOT - OK
                    066B  2460 89$:                                               ; INVALID TIME VALUE INPUT
  52   FEEB CF 9E   066B  2461            MOVAB   W^TIMERR,R2                     ; ADDRESS OF ERROR MESSAGE
       53   82 9A   0670  2462            MOVZBL  (R2)+,R3                        ; GET STRING LENGTH
                    0673  2463            $QIOW_S #0,TTCHAN(R6),-                 ; GIVE ERROR MESSAGE
                    0673  2464                    #IO$_WRITEVBLK,-
                    0673  2465                    ,,,                             ;  NO I/O STATUS,AST OR AST PARAM
                    0673  2466                    (R2),R3 ,-                      ;  BUFFER ADDRESS, LENGTH
                    0673  2467                    #0,#32                          ;  SET CARRIAGE CONTROL TO CR/LF
          FF73 31   0692  2468            BRW     10$                             ; AND TRY AGAIN
                    0695  2469 100$:                                              ; EXIT
                    0695  2470            $DASSGN_S TTCHAN(R6)                    ; DE-ASSIGN TERMINAL CHANNEL
          14 A6 7F  069F  2471 200$:      PUSHAQ  INTIME(R6)                      ; SET NEW SYSTEM TIME
00000000'GF   01 FB 06A2  2472            CALLS   #1,G^EXE$SETIME_INT             ; USE TODR CLOCK TO SET SYSTEM TIME
00000000'GF 00000000'GF 7D 06A9  2473     MOVQ    G^EXE$GQ_TODCBASE,G^EXE$GQ_BOOTTIME ; SAVE BOOT TIME
             5E 30 C0   06B4  2474            ADDL    #12*4,SP                        ; CLEAN OFF SCRATCH STORAGE
          077C 8F BA    06B7  2475            POPR    #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; RESTORE REGISTERS
                    06BB  2476
                    06BB  2477 ;
                    06BB  2478 ; Fall through into the deallocate logic.
                    06BB  2479 ;
                    06BB  2480 ;       RSB                                     ; *** This goes in if another piece of
                    06BB  2481                                                 ; *** initialization code is added that
                    06BB  2482                                                 ; *** is executed after EXE$INI_TIMWAIT.
                    06BB  2483         .DISABLE LSB
                    06BB  2484
```

```
                                06BB  2486 DEAL_INIT_CODE:                          ; DEALLOCATE THE INITIALIZATION CODE
                                06BB  2487 ;
                                06BB  2488 ; It is the duty of the last-executed, loadable initialization
                                06BB  2489 ; routine to make itself and all other such routines disappear, i.e.,
                                06BB  2490 ; release the space they occupy to non-paged pool.  Each routine's vector
                                06BB  2491 ; must be disconnected, e.g., be made to point to the symbol, EXE$LOAD_ERROR.
                                06BB  2492 ;
                                06BB  2493 ; NOTE:  This means that new initialization routines should be added
                                06BB  2494 ;         to this module in a particular order, not necessarily at the
                                06BB  2495 ;         end of the module!
                                06BB  2496 ;
                                06BB  2497          .ENABLE LSB
                      7E   52   7D  06BB  2498          MOVQ    R2,-(SP)                ; Save some registers
                                06BE  2499
                                06BE  2500 ;
                                06BE  2501 ; First find the vectors that point to these initialization routines
                                06BE  2502 ; and reset them to point to EXE$LOAD_ERROR.
                                06BE  2503 ;
                     50   0000'CF   9E  06BE  2504          MOVAB   W^SYSL$BEGIN,R0         ; Compute bounds of releasable piece:
                51   50  00000000'8F   C1  06C3  2505          ADDL3   #<STAY_HEADER-SYSL$BEGIN>,R0,R1 ; starting and ending addresses.
                52   00000000'GF   9E  06CB  2506          MOVAB   G^EXE$AL_LOAVEC,R2      ; Get starting address of vectors.
                53   00000000'GF   9E  06D2  2507          MOVAB   G^EXE$LOAD_ERROR,R3     ; Get end of vectors.
                    9F17 8F    62   B1  06D9  2508 10$:     CMPW    (R2),#^X9FT7            ; Is this JMP @# ?
                                1B   13  06DE  2509          BEQL    30$                     ; Br if yes, skip past it.
                       80 8F    03   A2  06E0  2510          CMPB    3(R2),#^X80             ; Is this a system space address
                                16   12  06E5  2511          BNEQ    40$                     ; Br if no, assume it's a HALT instr.
                            50   62   D1  06E7  2512          CMPL    (R2),R0                 ; Is address before the releasable
                                0C   1F  06EA  2513          BLSSU   20$                     ;  piece of memory?  Br on yes.
                            51   62   D1  06EC  2514          CMPL    (R2),R1                 ; Is address after the releasable
                                07   1A  06EF  2515          BGTRU   20$                     ;  piece of memory?  Br on yes.
                62   00000000'GF   9E  06F1  2516          MOVAB   G^EXE$LOAD_ERROR,(R2)   ; Reset this vector.
                            52   02   C0  06F8  2517 20$:     ADDL    #2,R2                   ; Point past this vector.
                                52   D6  06FB  2518 30$:     INCL    R2                      ; Come here to point past JMP @#.
                                52   D6  06FD  2519 40$:     INCL    R2                      ; Come here to point past HALT.
                         53     52   D1  06FF  2520          CMPL    R2,R3                   ; Past the end of the vectors?
                                D5   1F  0702  2521          BLSSU   10$                     ; Keep searching vectors.
                                     0704  2522 ;
                                     0704  2523 ; Now release the memory to non-paged pool.
                                     0704  2524 ;
                50   0000'CF   9E  0704  2525          MOVAB   W^SYSL$BEGIN,R0         ; Point to start of module
                51   0000'8F   3C  0709  2526          MOVZWL  #<STAY_HEADER-SYSL$BEGIN>,R1 ; Length to vaporize
                        F8FB'  31  070E  2527          BRW     50$                     ; Br to code that is not released.
                               0711  2528
                           00000000  2529          .PSECT  $$$INIT__END,PAGE       ; 'PAGE' SINCE 16-BYTE ALIGN IS NOT
                               0000  2530
                               0000  2531 STAY_HEADER:
                00000000 00000000  0000  2532          .LONG   0,0
                           0000'   0008  2533          .WORD   <SYSL$END-STAY_HEADER>
                             62    000A  2534          .BYTE   DYN$C_LOADCODE
                             00    000B  2535          .BYTE   0
                                   000C  2536
                 00000000'9F   16  000C  2537 50$:     JSB     @#EXE$DEANONPGDSIZ      ; Just the smile on the Chesire cat
                     52   8E   7D  0012  2538          MOVQ    (SP)+,R2                ; Restore
                             05  0015  2539          RSB                             ; Return.
                                 0016  2540
                                 0016  2541          .DISABLE LSB
                                 0016  2542          .END
```

```
$$$VMSDEFINED                   = 00000001
$$T1                            = 00000001
ADAPTERS                          00000000 R      02
ADP$B_TYPE                      = 0000000A
ADP$C_CIADPLEN                  = 00000030
ADP$C_DRADPLEN                  = 00000030
ADP$C_MBAADPLEN                 = 00000030
ADP$C_UBAADPLEN                 = 00000258
ADP$L_AVECTOR                   = 0000001C
ADP$L_CRB                       = 00000010
ADP$L_CSR                       = 00000000
ADP$L_DPQFL                     = 00000014
ADP$L_LINK                      = 00000004
ADP$L_MBASCB                    = 00000014
ADP$L_MBASPTE                   = 00000018
ADP$L_MRACTMDRS                 = 0000005C
ADP$L_MRQFL                     = 00000030
ADP$L_UBASCB                    = 00000044
ADP$L_UBASPTE                   = 00000054
ADP$L_VECTOR                    = 00000010
ADP$W_ADPTYPE                   = 0000000E
ADP$W_DPBITMAP                  = 00000060
ADP$W_MRFFENCE                  = 0000015C
ADP$W_MRFREGARY                 = 0000015E
ADP$W_MRNFENCE                  = 00000062
ADP$W_MRNREGARY                 = 00000064
ADP$W_SIZE                      = 00000008
ADP$W_TR                        = 0000000C
ADP$W_UMR_DIS                   = 00000256
ADPLINK                           ********    X   09
ADPTAB_ADPLEN                     00000001
ADPTAB_ATYPE                      00000003
ADPTAB_IDBUNITS                   00000000
ALONPAGD                          00000326 R      09
AT$_CI                          = 00000004
AT$_DR                          = 00000002
AT$_MBA                         = 00000000
AT$_UBA                         = 00000001
BADUMR                            0000030D R      08
BI_BUS_CODE                     = 80000000
BI_CPU                          = 00000000
BI_CSR_LEN                      = 00000002
BI_LIKE                         = 00000000
BLD_CRB                           00000456 R      09
BOO$GB_SYSTEMID                   ********    X   09
BOO$GL_SPTFREH                    ********    X   09
BOO$GL_SPTFREL                    ********    X   09
BOOTVECTOR                        00000000 R      08
BTD$K_CONSOLE                   = 00000040
BTD$K_UDA                       = 00000011
BUS_CODE_OFFSET                 = FFFFFFFC
BUS_CSR_LEN                       00000004 R      08
CI$INITIAL                        ********    X   09
CI$INT                            ********    X   09
CITAB                             00000015 R      08
CONFIG_IOSPACE                    0000005F R      09
CONFREG                           000000A4 R      08
CONFREGL                          000001E4 R      08
CPU_ADPSIZE                       00000019 R      08
CPU_TYPE                        = 00000002
CR                              = 0000000D
CRB$B_TYPE                      = 0000000A
CRB$C_LENGTH                    = 00000048
CRB$L_INTD                      = 00000024
CRB$L_INTD2                     = 00000048
CRB$L_WQBL                      = 00000004
CRB$L_WQFL                      = 00000000
CRB$W_SIZE                      = 00000008
CREATE_ARRAYS                     0000011A R      09
CSR_LEN_OFFSET                  = FFFFFFFB
DDB$T_NAME                      = 00000014
DEAL_INIT_CODE                    000006BB R      09
DIRECT_VEC_NODE_CNT               00000009 R      08
DR$INITIAL                        ********    X   09
DR$INT                            ********    X   09
DRTAB                             00000011 R      08
DYN$C_ADP                       = 00000001
DYN$C_CONF                      = 00000007
DYN$C_CRB                       = 00000005
DYN$C_IDB                       = 00000009
DYN$C_INIT                      = 00000063
DYN$C_LOADCODE                  = 00000062
END_NEXUS                         00000115 R      09
ERROR_HALT                        00000199 R      09
ERROR_HALT_1                      0000019E R      09
EXE$AL_LOAVEC                     ********    X   09
EXE$DEANONPGDSIZ                  ********    X   0A
EXE$GL_CONFREG                    ********    X   09
EXE$GL_CONFREGL                   ********    X   09
EXE$GL_FLAGS                      ********    X   09
EXE$GL_NUMNEXUS                   ********    X   09
EXE$GL_RPB                        ********    X   09
EXE$GL_SCB                        ********    X   09
EXE$GL_TENUSEC                    ********    X   09
EXE$GL_TODR                       ********    X   09
EXE$GL_UBDELAY                    ********    X   09
EXE$GQ_BOOTTIME                   ********    X   09
EXE$GQ_TODCBASE                   ********    X   09
EXE$INIT_TODR                     000005A0 RG     09
EXE$INI_TIMWAIT                   000004EE RG     09
EXE$LOAD_ERROR                    ********    X   09
EXE$MCHK_PRTCT                    ********    X   09
EXE$OUTZSTRING                    ********    X   09
EXE$SETIME_INT                    ********    X   09
EXE$TEST_CSR                      ********    X   09
EXE$UBAERR_INT                    ********    X   09
EXE$V_SETTIME                     ********    X   09
FILL_CRB                          00000463 R      09
GET_GEN_TYPE                      000000B9 R      09
GET_TYPE                          000000A0 R      09
IDB$B_TYPE                      = 0000000A
IDB$C_LENGTH                    = 00000038
IDB$L_ADP                       = 00000014
IDB$L_CSR                       = 00000000
```

```
IDB$W_SIZE                = 00000008
IDB$W_UNITS               = 0000000C
INI$ALLOC_CRB               ********    X   09
INI$ALONONPAGED             ********    X   09
INI$CIADP                   00000341 R      09
INI$CONSOLE                 0000043F RG     09
INI$DRADP                   0000032C R      09
INI$IOMAP                   00000000 RG     09
INI$KDZ11                   0000043E R      09
INI$MBADP                   00000356 R      09
INI$MPMADP                  ********    X   06
INI$UBADP                   000001C4 R      09
INI$UBSPACE                 000001A7 R      09
INIT_ROUTINES               00000000 R      06
INTIME                    = 00000014
IO$M_CVTLOW                 ********    X   09
IO$M_PURGE                  ********    X   09
IO$M_TIMED                  ********    X   09
IO$_READPROMPT              ********    X   09
IO$_WRITEVBLK               ********    X   09
IO750$AL_IOBASE           = 00F20000
IO750$AL_PERNEX           = 00002000
IO750$AL_UBOSP            = 00FC0000
LF                        = 0000000A
LINBUF                    = 0000001C
LINBUFSIZ                 = 00000014
MAP_NEXUS                   000000F9 R      09
MAP_PAGES                   00000173 R      09
MAXNEXUS                  = 00000040
MBA$INITIAL                 ********    X   09
MBA$INT                     ********    X   09
MBATAB                      0000000D R      08
MCHK$M_LOG                = 00000001
MCHK$M_NEXM               = 00000004
MMG$GL_SBICONF              ********    X   09
MMG$GL_SPTBASE              ********    X   09
MMG$SVAPTECHK               ********    X   09
NDT$_BUA                  = 80000102
NDT$_CI                   = 00000038
NDT$_DR32                 = 00000030
NDT$_KDZ11                = 80000105
NDT$_MB                   = 00000020
NDT$_MEM1664NI            = 00000012
NDT$_MEM16I               = 00000011
NDT$_MEM16NI              = 00000010
NDT$_MEM256EIL            = 00000071
NDT$_MEM256EIU            = 00000073
NDT$_MEM256I              = 00000074
NDT$_MEM256NIL            = 00000070
NDT$_MEM256NIU            = 00000072
NDT$_MEM4I                = 00000009
NDT$_MEM4NI               = 00000008
NDT$_MEM64EIL             = 00000069
NDT$_MEM64EIU             = 0000006B
NDT$_MEM64I               = 0000006C
NDT$_MEM64NIL             = 00000068
NDT$_MEM64NIU             = 0000006A
NDT$_MPM0                 = 00000040
NDT$_MPM1                 = 00000041
NDT$_MPM2                 = 00000042
NDT$_MPM3                 = 00000043
NDT$_SCORMEM              = 80000001
NDT$_UB0                  = 00000028
NDT$_UB1                  = 00000029
NDT$_UB2                  = 0000002A
NDT$_UB3                  = 0000002B
NEXUSDESC                   00000020 R      08
NOSPT                       000002E4 R      08
NPROMPT                   = 00000033
NUMUBAVEC                 = 00000080
NUM_PAGES                   00000000 R      04
NXT_NEXUS                   0000006B R      09
PA                        = 00F40000
PR$_CSTD                  = 0000001F
PR$_ICCS                  = 00000018
PR$_SID_TYP730            = 00000003
PR$_SID_TYP750            = 00000002
PR$_SID_TYP780            = 00000001
PR$_SID_TYP790            = 00000004
PR$_SID_TYP8NN            = 00000006
PR$_SID_TYP8SS            = 00000005
PR$_SID_TYPUV1            = 00000007
PR$_TBIS                  = 0000003A
PR750$_ICR                = 0000001A
PR750$_NICR               = 00000019
PR750$_TODR               = 0000001B
PTE$C_RW                  = 10000000
PTE$M_VALID               = 80000000
READTIME                    000005D8 R      09
RPB$B_DEVTYP              = 00000066
RPB$L_ADPPHY              = 0000005C
RPB$L_ADPVIR              = 00000060
RPB$L_BOOTR1              = 00000020
RPB$L_CSRPHY              = 00000054
RPB$L_CSRVIR              = 00000058
RPB$W_ROUBVEC             = 0000001E
SBICONF                     000000E4 R      08
SBI_BUS_CODE              = 00000000
SBI_CPU                   = 00000000
SBI_CSR_LEN               = 00000001
SBI_LIKE                  = 00000001
SGN$GW_TPWAIT               ********    X   09
STAY_HEADER                 00000000 R      0A
SW_BUS_CODE                 00000005 R      08
SYS$ASSIGN                  ********   GX   09
SYS$BINTIM                  ********   GX   09
SYS$DASSGN                  ********   GX   09
SYS$QIOW                    ********   GX   09
SYSL$BEGIN                  ********    X   09
SYSL$END                    ********    X   0A
TERM_NAMADR                 00000556 R      09
TERM_NAMSIZ               = 00000004
TEST_NEXUS                  00000071 R      09
TIMEPROMPT                  0000056C R      09
```

```
TIMERR                          0000055A R     09
TMPDESC                       = 0000000C
TTCHAN                        = 00000000
TTNAME                        = 00000004
UAS$W_IP_CR1                  = 00001464
UBA$INITIAL                     ********   X   09
UBA$INTO                        ********   X   09
UBA$L_MAP                     = 00000800
UBA$UNEXINT                     ********   X   09
UBA1INT                         0000031E R     09
UCB$W_UNIT                    = 00000054
VA$M_SYSTEM                   = 80000000
VEC$C_ADP                     = 00000014
VEC$L_IDB                     = 00000008
VEC$L_INITIAL                 = 0000000C
```

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	00000004 (4.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$INIT$DATA0	00000074 (116.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$INIT$DATA1	00000000 (0.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$INIT$DATA2	0000003A (58.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$INIT$DATA3	00000000 (0.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$INIT$DATA4	00000074 (116.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$INIT$DATA5	00000000 (0.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$INIT$DATA	0000033F (831.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$INIT$CODE	00000711 (1809.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD
$$$INIT__END	00000016 (22.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.04	00:00:02.08
Command processing	109	00:00:00.47	00:00:03.49
Pass 1	531	00:00:13.91	00:00:52.73
Symbol table sort	0	00:00:01.72	00:00:07.34
Pass 2	291	00:00:04.15	00:00:16.06
Symbol table output	29	00:00:00.15	00:00:00.40
Psect synopsis output	3	00:00:00.04	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	994	00:00:20.48	00:01:22.70

```
The working set limit was 2100 pages.
139241 bytes (272 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1656 non-local and 37 local symbols.
2546 source lines were read in Pass 1, producing 39 object records in Pass 2.
47 pages of virtual memory were used to define 45 macros.
```

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

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                        23
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     14
TOTALS (all libraries)                                37

1808 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:INIADP750/OBJ=OBJ$:INIADP750 MSRC$:CPUSW750/UPDATE=(ENH$:CPUSW750)+MSRC$:INIADP/UPDATE=(ENH$:INIADP)+EXECML$/LIB
```

INIADP750
LIS

INIADP730
LIS

INIADP780
LIS

INIADPUVI
LIS

INIADP790
LIS