L 5

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                  OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                  OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  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
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL      OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL      OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL      OOOOOOOOO     AAA         AAA
```

```
_$2
Sym
---
SS$
SS$

SS$
SS$
SS$
SS$
SS$
SS$
SYS
SYS
SYS
TRY
UNL
WRI
```

FILEID**INIADP750

C 4

```
 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
```

IN
VO

Table of contents

```
0000           .NLIST  CND
0000
0000           .TITLE  INIADP750 - ADAPTER INITIALIZATION FOR VAX 11/750
0000
0000
0000
0000
0000
0000           .IDENT  'V04-002'
0000     ;
0000     ;****************************************************************************
0000     ;*                                                                          *
0000     ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     ;*  ALL RIGHTS RESERVED.                                                    *
0000     ;*                                                                          *
0000     ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000     ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000     ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000     ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000     ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000     ;*  TRANSFERRED.                                                            *
0000     ;*                                                                          *
0000     ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000     ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000     ;*  CORPORATION.                                                            *
0000     ;*                                                                          *
0000     ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000     ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000     ;*                                                                          *
0000     ;*                                                                          *
0000     ;****************************************************************************
0000     ;
0000     ; Facility: System bootstrapping and initialization
0000     ;
0000     ; Abstract: This module contains initialization routines that are loaded
0000     ;           during system initialization (rather than linked into the system).
0000     ;
0000     ; Environment: Mode = KERNEL, Executing on INTERRUPT stack, IPL=31
0000     ;
0000     ; Author:  Trudy C. Matthews           Creation date: 22-Jan-1981
0000     ;
0000     ; Modification history:
0000     ;
0000     ;       V04-002 TCM0013         Trudy C. Matthews       10-Sep-1984
0000     ;               Add $BQODEF missing from TCM0012.
0000     ;
0000     ;       V04-001 TCM0012         Trudy C. Matthews       07-Sep-1984
0000     ;               For venus processor:  turn on cache before calibrating
0000     ;               TIMEDWAIT cells (routine EXE$INI_TIMWAIT).  Store the TIMEDWAIT
0000     ;               values calculated after cache is enabled in the boot driver's
0000     ;               TIMEDWAIT cells.  This is because the boot driver initially
0000     ;               has to run with cache off, but after booting will run with
0000     ;               cache on.
0000     ;
0000     ;       V03-024 TCM0011         Trudy C. Matthews       31-Jul-1984
0000     ;               Change venus's CRD interrupt vector back to ^X54 in the SCB,
0000  ;               and its SBIA Fail vector to ^X64.
0000  ;
0000  ;       V03-023 WMC0001         Wayne Cardoza           30-Jul-1984
0000  ;               Add H memory to 780 list.
0000  ;
0000  ;       V03-022 TCM0010         Trudy C. Matthews       25-Jul-1984
0000  ;               Fix a bug in INI$UBSPACE for the 11/790 that caused second
0000  ;               and subsequent unibus adapter spaces to be mapped incorrectly.
0000  ;               Fix bugs in INI$SCB for the 11/790.  Fix conditional
0000  ;               assembly flags in INI$CONSOLE for the 11/790.
0000  ;
0000  ;       V03-021 KDM0100         Kathleen D. Morse       01-May-1984
0000  ;               Correct address of memory CSRs to be past the 8 missing
0000  ;               Qbus adapter pages that do not exist.
0000  ;
0000  ;       V03-020 KDM0099         Kathleen D. Morse       27-Apr-1984
0000  ;               On a MicroVAX I, if the sysgen parameter TIMEDWAIT is set
0000  ;               to request no time-prompting, then use the last recorded
0000  ;               system time instead.  This is found in EXE$GQ_TODCBASE
0000  ;               which can be updated with a SET TIME command.
0000  ;
0000  ;       V03-019 RLRSCORPIO      Robert L. Rappaport     16-Mar-1984
0000  ;               Begin additions (to INI$IOMAP) for Scorpio support.
0000  ;               Also move ADAPDESC to SYSMAR.MAR, changing it to remove
0000  ;               the ADAP_GENERAL array.
0000  ;
0000  ;       V03-018 RLRINIADP       Robert Rappaport        28-Feb-1984
0000  ;               Add refinements to previous update that introduces
0000  ;               longword array CONFREG.  Mainly add logic to allow for
0000  ;               independently assembled invocations of ADAPDESC macro
0000  ;               to be linked into this code.  This provides possible
0000  ;               support of BI as a public bus, with user defined nodes.
0000  ;
0000  ;       V03-017 KPL0100         Peter Lieberwirth       30-Jan-1984
0000  ;               Implement first step towards a longword-array CONFREG to
0000  ;               replace current byte array CONFREG.  INIADP will construct
0000  ;               two confregs, CONFREG and CONFREGL.  CONFREGL will be
0000  ;               a longword array.  The high byte will be a VMS-bus
0000  ;               designation, and the low word will contain the 16-bit
0000  ;               device type.  The BI introduces 16 bit device types.
0000  ;
0000  ;               When all references to CONFREG have been modified to touch
0000  ;               CONFREGL, INIADP will be modified again to stop creating
0000  ;               the byte array.
0000  ;
0000  ;               While here, map 9 pages of CI register space, up from 8.
0000  ;
0000  ;       V03-016 KPL0001         Peter Lieberwirth       17-Jan-1984
0000  ;               Fix bug in V03-015 that caused a failure to boot on 750s.
0000  ;               Specifically, add NDT$_MEM1664NI to ADAPDESC macro.
0000  ;
0000  ;       V03-015 TCM0009         Trudy C. Matthews       12-Dec-1983
0000  ;               Add support for booting from VENUS console device to
0000  ;               INI$CONSOLE.  When mapping I/O space on VENUS, use the
0000  ;               PAMM to determine if any adaptors are present on the
0000  ;               ABUS.
0000  ;
```

```
0000    ;       V03-014 KDM0081         Kathleen D. Morse       13-Sep-1983
0000    ;               Create version for Micro-VAX I.
0000    ;
0000    ;       V03-013 DWT0126         David W. Thiel          30-Aug-1983
0000    ;               Modify EXE$INIT_TODR to set internal time without
0000    ;               modifying the contents of the system disk.
0000    ;
0000    ;       V03-012 KDM0062         Kathleen D. Morse       18-Jul-1983
0000    ;               Add loadable, cpu-dependent routine for initializing
0000    ;               the time-wait loop data cells, EXE$INI_TIMWAIT.
0000    ;
0000    ;       V03-011 KDM0057         Kathleen D. Morse       15-Jul-1983
0000    ;               Added loadable, cpu-dependent routine for initializing
0000    ;               the system time, EXE$INIT_TODR.
0000    ;
0000    ;       V03-010 KTA3071         Kerbey T. Altmann       12-Jul-1983
0000    ;               Include CPU-specific console init code.
0000    ;
0000    ;       V03-009 TCM0008         Trudy C. Matthews       10-Jan-1983
0000    ;               Change PSECT of 11/790 data that must stick around after
0000    ;               INIADP is deleted.  Build arrays ABUS_VA, ABUS_TYPE, and
0000    ;               ABUS_INDEX that describe the 11/790 ABUS configuration.
0000    ;
0000    ;       V03-008 MSH0002         Maryann Hinden          08-Dec-1982
0000    ;               Add powerfail support for DW750.
0000    ;
0000    ;       V03-007 ROW0142         Ralph O. Weber          24-NOV-1982
0000    ;               Change UBA interrupt services routines prototype so that
0000    ;               UBAERRADR is correctly computed as an offset from UBAINTBASE.
0000    ;
0000    ;       V03-006 TCM0007         Trudy C. Matthews       10-Nov-1982
0000    ;               Add 11/790-specific initialization of SCB.
0000    ;
0000    ;       V03-005 TCM0006         Trudy C. Matthews       8-Nov-1982
0000    ;               Initialize field ADP$L_AVECTOR with the address of
0000    ;               each adapter's first SCB vector.
0000    ;
0000    ;       V03-004 KTA3018         Kerbey T. Altmann       30-Oct-1982
0000    ;               Move from INILOA facility, rename from INITADP,
0000    ;               put in conditional assembly, rewrite some routines.
0000    ;
0000    ;       V03-003 MSH0001         Maryann Hinden          24-Sep-1982
0000    ;               Change EXE$DW780_INT to EXE$UBAERR_INT.
0000    ;
0000    ;       V03-002 TCM0005         Trudy C. Matthews       10-Aug-1982
0000    ;               Added support for 11/790 processor.
0000    ;
0000    ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    ;               Added $DCDEF.
0000    ;
0000    ;--
```

```
0000   ;
0000   ; MACRO LIBRARY CALLS
0000   ;
0000            $ADPDEF                          ; Define ADP offsets.
0000            $BIICDEF                         ; Define BIIC offsets.
0000            $BQODEF                          ; Define boot vector offsets.
0000            $BTDDEF                          ; Define boot devices
0000            $BUADEF                          ; Define BUA Register offsets.
0000            $CRBDEF                          ; Define `RB offsets.
0000            $DCDEF                           ; Define adapter types
0000            $DDBDEF                          ; Define DDB offsets
0000            $DYNDEF                          ; Define data structure type codes.
0000            $IDBDEF                          ; Define interrupt dispatcher offsets.
0000            $IO750DEF                        ; Define 11/750 I/O space.
0000            $UASDEF                          ; Define DW750 IPEC registers.
0000            $MCHKDEF                         ; Define machine check masks.
0000            $NDTDEF                          ; Define nexus device types.
0000            $PRDEF                           ; Define IPR numbers.
0000
0000
0000            $PR750DEF                        ; Define 11/750 specific IPR numbers.
0000
0000
0000
0000
0000
0000            $PTEDEF                          ; Define Page Table Entry bits.
0000            $RPBDEF                          ; Define Restart Parameter Block fields.
0000            $UBADEF                          ; Define UBA register offsets.
0000            $UCBDEF                          ; Define UCB offsets.
0000            $VADEF                           ; Define virtual address fields.
0000            $VECDEF                          ; Define vec offsets.
```

```
0000            .SBTTL  Macros to describe nexus configurations
0000    ;
0000    ;       The macros FLOAT_NEXUS and FIXED_NEXUS add one or more entries to a
0000    ;       nexus descriptor table.  Each entry is of the form:
0000    ;               +--------------------------------+
0000    ;               :   PFN of nexus I/O space       :
0000    ;               +-------+-------+----------------+
0000    ;               :  bus  :   0   :      type      :
0000    ;               +-------+-------+----------------+
0000    ;       type = 0 -> floating nexus
0000    ;       type = non-zero -> fixed nexus; type = fixed adapter type
0000    ;       bus  = 0, if SBI; %x80 if BI  (this is a VMS-only designation)
0000    ;
0000    ;
0000    ;       device_type:    SBI adapters have 8-bit device type codes.  These
0000    ;                       device types are simple integers.
0000    ;
0000    ;                       BI adapters have 16-bit device type codes, that are
0000    ;                       subject to the following interpretation:
0000    ;
0000    ;                       - the MSB of the device-type field will be 0 for DEC
0000    ;                       devices and ' for non-DEC devices,
0000    ;
0000    ;                       - DEC memory devices will have 0s in the high-order
0000    ;                       byte of the device type,
0000    ;
0000    ;                       - non-DEC supplied memory devices will have a 1 in the
0000    ;                       MSB of the high-order byte, and the rest of the high
0000    ;                       order byte will contain 0s.
0000    ;
0000    ;                       - The ''all 0s'' and ''all 1s'' device-type codes are
0000    ;                       reserved for DEC.
0000    ;
0000    ; If SBI type codes were simply expanded to a word for purposes of the routines
0000    ; in this module, there would be possible conflicts between SBI devices and
0000    ; BI memory adapters supplied by DEC.  Voila:  the bus type.
0000    ;
0000    ; Macro FLOAT_NEXUS.
0000    ; INPUTS:
0000    ;       PHYSADR -- physical address of 1 or more contiguous floating nexus
0000    ;                  slots
0000    ;       NUMNEX -- number of contiguous floating nexuses, default = 1
0000    ;       PERNEX -- amount of address space per nexus (does not have to be
0000    ;                 specified if NUMNEX = 1)
0000    ;
0000            .MACRO  FLOAT_NEXUS     PHYSADR,NUMNEX=1,PERNEX=0
0000            PA = PHYSADR
0000            .REPEAT NUMNEX          ; For each nexus...
0000            .LONG   <PA/^X200>      ; Store PFN.
0000            .LONG   0               ; Store floating nexus type.
0000            PA = PA + PERNEX        ; Increment to physical address of next nexus.
0000            .ENDR
0000            .ENDM   FLOAT_NEXUS
0000
0000    ;
0000    ; Macro FIXED_NEXUS.
0000    ;
```

```
                    0000    ; INPUTS:
                    0000    ;       PHYSADR - physical address of 1 or more contiguous fixed nexus slots
                    0000    ;       PERNEX - amount of address space per nexus
                    0000    ;       NEXUSTYPES - a list of fixed nexus types, enclosed in <>
                    0000    ;
                    0000            .MACRO  FIXED_NEXUS     PHYSADR,PERNEX=0,NEXUSTYPES
                    0000            PA = PHYSADR
                    0000            .IRP    TYPECODE,NEXUSTYPES     ; For each fixed nexus type...
                    0000            .LONG   <PA/^X200>              ; Store PFN.
                    0000            .LONG   TYPECODE                ; Store fixed nexus type.
                    0000            PA = PA + PERNEX                ; Increment to address of next nexus.
                    0000            .ENDR
                    0000            .ENDM   FIXED_NEXUS
                    0000
                    0000    ;
                    0000    ; Macro NEXUSDESC_TABLE - declare the beginning of a NEXUS descriptor table
                    0000    ;
                    0000    ;       1st byte in table (at offset -5 from label) contains length of
                    0000    ;       adapter type code field in CSR's on this bus. [Note for SBI like
                    0000    ;       busses, this is 1.]  The next longword (at offset -4) in the
                    0000    ;       table contains the Software defined bus type byte defined in the
                    0000    ;       high order byte of the longword.  [Note for SBI like busses, this
                    0000    ;       value is 0, for the BI it is ^x80.]
                    0000    ;
                    0000
                    0000    ; Define parameters that may be specified or used in macro invocation.
                    0000
        00000000    0000    BI_LIKE  = 0                    ; BI like bus.
        00000001    0000    SBI_LIKE = 1                    ; SBI like bus.
                    0000
        00000001    0000    SBI_CSR_LEN = 1                 ; Length of type code field in adapter CSR's
                    0000                                    ;  on SBI, CMI, etc.
        00000002    0000    BI_CSR_LEN  = 2                 ; Length of type code field in adapter CSR's
                    0000                                    ;  on BI.
                    0000
        00000000    0000    SBI_BUS_CODE = 0                ; Software defined bus code for SBI like busses.
        80000000    0000    BI_BUS_CODE  = ^x80000000       ; Software defined bus code for the BI.
                    0000
                    0000            .MACRO  NEXUSDESC_TABLE LABEL,BUS_TYPE=SBI_LIKE
                    0000            .IF     EQ,BUS_TYPE-SBI_LIKE
                    0000                                    .BYTE   SBI_CSR_LEN
                    0000                                    .LONG   SBI_BUS_CODE
                    0000            .IFF
                    0000                    .IF     EQ,BUS_TYPE-BI_LIKE
                    0000                                    .BYTE   BI_CSR_LEN
                    0000                                    .LONG   BI_BUS_CODE
                    0000                    .IFF
                    0000                                    .ERROR  ; UNRECOGNIZED BUS TYPE, NEXUSDESC_TABLE;
                    0000                    .ENDC
                    0000            .ENDC
                    0000
                    0000    LABEL:
                    0000            .ENDM   NEXUSDESC_TABLE
                    0000
        FFFFFFFB    0000    CSR_LEN_OFFSET  = -5                    ; Offset before nexus descriptor of
                    0000                                            ;  byte containing length of adapter
                    0000                                            ;  type field in adapter CSR.
```

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

```
                     0000          .SBTTL  Adapter-specific data structures
                     0000  ;
                     0000  ; Put a symbol for arrays built by macros in the correct psects.
                     0000  ;
                     0000  ;***************** ADAPTERS array *************
                 00000000          .PSECT  $$$INIT$DATA0
                     0000  ADAPTERS:                              ; Build adapter type code arrays here.
                     0000
                 00000000          .PSECT  $$$INIT$DATA1          ; User contributions in this .PSECT.
                     0000                                         ; End of ADAPTERS array.
                     0000  ;***************** End of ADAPTERS array *************
                     0000
                     0000  ;***************** 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.
                     0000  ;***************** End of NUM_PAGESarray *************
                     0000
                     0000  ;***************** 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.
                     0000  ;***************** End of INIT_ROUTINES array *************
                     0000
                     0000  ;
                     0000  ; To add a new adapter type:
                     0000  ;       1) Add a new ADAPDESC macro invocation to the end of this list.
                     0000  ;
                 00000000          .PSECT  $$$INIT$DATA,LONG
                     0000
                     0000  ;
                     0000  ; Default interupt vectors for UNIBUS system devices
                     0000  ; (This array is indexed by the RPB field RPB$B_DEVTYP, if the RPB field
                     0000  ; RPB$W_ROUBVEC is zero.  If RPB$W_ROUBVEC is not zero, then RP[illegible]ROUBVEC
                     0000  ; is used and this array is not referenced at all.  RPB$W_ROUBVE[illegible] is set up
                     0000  ; by PQDRIVER.  RPB$L_BOOTR0 is set by VMB to contain the device name in
                     0000  ; ASCII, not the vector number and device type, as it does on full
                     0000  ; architecture VAX machines.
                     0000  ;
                     0000  BOOTVECTOR:
               0088  0000          .WORD   ^X88           ; RK06/7 Interrupt vector
               0070  0002          .WORD   ^X70           ; RL01/2 Interrupt vector
                     0004
                     0004  BUS_CSR_LEN:                   ; Static byte containing the length (in bytes)
                 00  0004          .BYTE   0              ;  of the adapter type field in the CSR's of
                     0005                                 ;  the bus currently being configured.  The
                     0005                                 ;  proper value for the bus of interest is
                     0005                                 ;  copied here, from the current nexus
                     0005                                 ;  descriptor table, when we enter subroutine
                     0005                                 ;  CONFIG_IOSPACE.
                     0005
                     0005  SW_BUS_CODE:                   ; Static longword containing the software
           00000000  0005          .LONG   0              ;  defined bus type, of the bus currently being
                     0009                                 ;  configured, in the high order byte.  The
                     0009                                 ;  proper value for the bus of current interest
                     0009                                 ;  is copied here, from the nexus descriptor
                     0009                                 ;  table, when we enter subroutine
```

```
          0009                                          ;  CONFIG_IOSPACE.
          0009
          0009  DIRECT_VEC_NODE_CNT:                    ; Static longword that counts the number of
          0009                                          ;  direct vectoring adpater nodes that we have
00000000  0009          .LONG   0                       ;  run across so far.
          000D
00000001  000D  $$$VMSDEFINED = 1                       ; Define symbol that means VMS system software.
00000080  000D  NUMUBAVEC = 128                         ; ALLOW FOR 128 UNIBUS VECTORS
          000D
          000D          ADAPDESC -                      ; Memory. ** MUST BE 1ST IN DESCRIPTOR LIST **
          000D          ADPTYPES=<NDT$_MEM1664NI,NDT$_MEM4NI,NDT$_MEM4I,NDT$_MEM16NI, -
          000D                  NDT$_MEM16I, -
          000D                  NDT$_MEM64NIL,NDT$_MEM64EIL,NDT$_MEM64NIU,NDT$_MEM64EIU,  -
          000D                  NDT$_MEM64I, -
          000D                  NDT$_MEM256NIL,NDT$_MEM256EIL,NDT$_MEM256NIU,NDT$_MEM256EIU,  -
          000D                  NDT$_MEM256I, -
          000D                  NDT$_SCORMEM> -
          000D                  NUMPAGES=1
          000D
          000D          ADAPDESC -                      ; MASSbus.
          000D                  ADPTYPES=NDT$_MB, -
          000D                  NUMPAGES=8, -
          000D                  INITRTN=INI$MBADP
          000D
          000D          ADAPDESC -                      ; UNIbus.
          000D                  ADPTYPES=<NDT$_UB0,NDT$_UB1,NDT$_UB2,NDT$_UB3,NDT$_BUA>, -
          000D                  NUMPAGES=8, -
          000D                  INITRTN=INI$UBSPACE
          000D
          000D          ADAPDESC -                      ; Multi-port memory.
          000D                  ADPTYPES=<NDT$_MPM0,NDT$_MPM1,NDT$_MPM2,NDT$_MPM3>, -
          000D                  NUMPAGES=1, -
          000D                  INITRTN=INI$MPMADP
          000D
          000D          ADAPDESC -                      ; DR32.
          000D                  ADPTYPES=NDT$_DR32, -
          000D                  NUMPAGES=4, -
          000D                  INITRTN=INI$DRADP
          000D
          000D          ADAPDESC -                      ; CI780
          000D                  ADPTYPES=NDT$_CI, -
          000D                  NUMPAGES=9, -
          000D                  INITRTN=INI$CIADP
          000D
          000D          ADAPDESC -                      ; KDZ11 Processor
          000D                  ADPTYPES=NDT$_KDZ11, -
          000D                  NUMPAGES=1, -
          000D                  INITRTN=INI$KDZ11
          000D
```

```
                   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              .SBTTL  CPU-specific data structures
                 0019      ;
                 0019      ; To add a new CPU type:
                 0019      ;       1) Create a new nexus descriptor table, using FLOAT_NEXUS and
                 0019      ;          FIXED_NEXUS macros.  Put an END_NEXUSDESC macro at the end.
                 0019      ;
                 0019
                 0019
                 0019
                 0019      CPU_ADPSIZE:
           0258  0019              .WORD   ADP$C_UBAADPLEN
                 001B
                 001B
                 001B      ;
                 001B      ; Declare the beginning of a nexus-descriptor table.
                 001B      ;
                 001B              NEXUSDESC_TABLE LABEL=NEXUSDESC
                 0020
                 0020
                 0020      ;
                 0020      ; Describe all possible nexuses on an 11/750 (the first 10 have fixed adapter
                 0020      ; assignments).
                 0020      ;
       00000000  0020              SBI_CPU = 0
       00000000  0020              BI_CPU  = 0
                 0020              FIXED_NEXUS -
                 0020                      PHYSADR=IO750$AL_IOBASE, -
                 0020                      PERNEX=IO750$AL_PERNEX, -
                 0020                      NEXUSTYPES=<NDT$_MEM1664NI, -
                 0020                                  NDT$_MPM0, -
                 0020                                  NDT$_MPM1, -
                 0020                                  NDT$_MPM2, -
                 0020                                  NDT$_MB, -
                 0020                                  NDT$_MB, -
                 0020                                  NDT$_MB, -
                 0020                                  NDT$_MB, -
                 0020                                  NDT$_UB0, -
                 0020                                  NDT$_UB1>
                 0070              FLOAT_NEXUS -
                 0070                      PHYSADR=IO750$AL_IOBASE+<10*IO750$AL_PERNEX>, -
                 0070                      NUMNEX=6, -
                 0070                      PERNEX=IO750$AL_PERNEX
                 00A0              END_NEXUSDESC
                 00A4
                 00A4
                 00A4
                 00A4
                 00A4
                 00A4
                 00A4      ;
                 00A4      ; Nexus "descriptor" arrays -- these arrays hold the nexus-device type and
                 00A4      ; virtual address of every adapter on the system.  The arrays, CONFREGL and
                 00A4      ; SBICONF, are allocated enough space to hold the maximum number of adapters
                 00A4      ; that can be attached to any CPU.  When the code discovers how many adapters
                 00A4      ; actually exist on the system, it will allocate space from non-paged pool
                 00A4      ; and move a permanent copy of these arrays into that space.
                 00A4      ;
```

```
00000040  00A4   715 MAXNEXUS = 64
          00A4   716 CONFREG:                                   ; Byte array of nexus-device type codes..
000000E4  00A4   717          .BLKB    MAXNEXUS
          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
```

```
                                  033F          .SBTTL  INI$IOMAP, Initialize and map nexuses
                                  033F   ;++
                                  033F   ; FUNCTIONAL DESCRIPTION:
                                  033F   ;      This routine is executed only once, during system initialization.
                                  033F   ;      It loops through all nexuses on the system, testing for
                                  033F   ;      adapte  .  When it finds an adapter, it maps its I/O space and
                                  033F   ;      initia.izes it.
                                  033F   ;
                                  033F   ; INPUTS:
                                  033F   ;      BOO$GL_SPTFREL   - next free VPN
                                  033F   ;      MMG$GL_SPTVASE   - base of system page table
                                  033F   ;      EXE$GL_RPB       - address of reboot parameter block
                                  033F   ;      RPB$L_ADPPHY(RPB) - PFN of boot adapter space
                                  033F   ;
                                  033F   ; OUTPUTS:
                                  033F   ;      R0 - SS$_NORMAL
                                  033F   ;
                                  033F   ;      For each adapter found, its accessible I/O space is mapped to virtual
                                  033F   ;      addresses.  An ADP (Adapter Control Block) is built, and the hardware
                                  033F   ;      adapter is initialized.
                                  033F   ;
                                  033F   ;      The arrays CONFREG (a byte array of nexus-device type codes, defined
                                  033F   ;      by NDT$_ symbols) and SBICONF (a longword array of
                                  033F   ;      virtual addresses that map adapter space) are initialized.  Pointers
                                  033F   ;      to these arrays are stored in EXE$GL_CONFREG  and
                                  033F   ;      MMG$GL_SBICONF.  The number of entries in these two parallel arrays is
                                  033F   ;      stored in EXE$GL_NUMNEXUS.
                                  033F   ;
                                  033F   ;      Since BI devices have a 16-bit device type code, a new CONFREG array is
                                  033F   ;      constructed.  This is a longword array called CONFREGL.
                                  033F   ;
                                  033F   ;      Several locations in the RPB that describe the boot device are init'ed:
                                  033F   ;      RPB$L_BOOTR1     - holds index into CONFREG and SBICONF for the boot
                                  033F   ;                         adapter
                                  033F   ;      RPB$L_ADPVIR     - holds VA of boot device adapter's register space
                                  033F   ;      RPB$L_CSRVIR     - holds VA of boot device's register space
                                  033F   ;--
                                  033F
                              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.
```

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

```
                           009B    980                                         ; one page of I/O space.
         56    04    C0    009B    981          ADDL2   #4,R6                  ; Step past type code in nexus table.
               59    11    009E    982          BRB     MAP_NEXUS              ; Go map I/O space for this nexus.
                           00A0    984  ;
                           00A0    985  ; Execution continues here if adapter was present.
                           00A0    986  ;
                           00A0    987  GET_TYPE:
         57    86    D0    00A0    988          MOVL    (R6)+,R7               ; Get nexus-device type from nexus table.
               14    12    00A3    990          BNEQ    GET_GEN_TYPE           ; Branch if fixed slot.
                           00A5    991  ;
                           00A5    992  ; Floating-type slot.  Use type from configuration register.
                           00A5    993  ; Determine if type in configuration register is 8-bits or 16-bits.
                           00A5    994  ;
                           00A5    995
    0004'CF    01    91    00A5    996          CMPB    #1,W^BUS_CSR_LEN       ; Determine length of adapter type
                           00AA    997                                         ;  field in CSR contained in R7.
               05    13    00AA    998          BEQL    10$                    ; EQL implies 1 byte (8-bit) field.
         57    51    3C    00AC    999          MOVZWL  R1,R7                  ; BI_LIKE, so use word instruction.
               03    11    00AF   1000          BRB     20$                    ; Skip byte instruction.
         57    51    9A    00B1   1001  10$:    MOVZBL  R1,R7                  ; Use byte instruction to get type.
                           00B4   1002  20$:                                   ;
   57   0005'CF    C8      00B4   1003          BISL    W^SW_BUS_CODE,R7       ; Or in software bus code.
                           00B9   1005  ;
                           00B9   1006  ; Here R7 has hardware adapter code or'ed with software bus code.
                           00B9   1007  ; Translate specific nexus device type code into general adapter type code.
                           00B9   1008  ;
                           00B9   1009  GET_GEN_TYPE:
  00A4'CF44    57    90    00B9   1010          MOVB    R7,W^CONFREG[R4]       ; Save nexus-device type in CONFREG.
  01E4'CF44    57    D0    00BF   1011          MOVL    R7,W^CONFREGL[R4]      ; CONFREGL also filled in.
               55    D4    00C5   1012          CLRL    R5                     ; Clear loop index.
                           00C7   1013  30$:                                   ;
   50   0000'CF45    DE    00C7   1014          MOVAL   W^ADAPTERS[R5],R0      ; Get address of adapter type code.
         0000'CF     9F    00CD   1015          PUSHAB  W^NUM_PAGES            ; Push addr of end of ADAPTERS array.
         8E    50    D1    00D1   1016          CMPL    R0,(SP)+               ; See if we went beyond array.
               3F    1E    00D4   1017          BGEQU   END_NEXUS              ; unrecognized adapter, do not map.
         60    57    D1    00D6   1018          CMPL    R7,(R0)                ; Adapter type match?
               04    13    00D9   1019          BEQL    40$                    ; If EQL yes, adapter type match.
               55    D6    00DB   1020          INCL    R5                     ; Increment loop index.
               E8    11    00DD   1021          BRB     30$                    ; Look at next adapter.
                           00DF   1022  40$:
                           00DF   1023
                           00DF   1024  ;
                           00DF   1025  ; Store boot parameters.
                           00DF   1026  ;
         5A    58    D1    00DF   1028          CMPL    R8,R10                 ; Does PFN match boot adapter's PFN?
               15    12    00E2   1029          BNEQ    MAP_NEXUS              ; No; continue.
      60 A9    52    D0    00E4   1031          MOVL    R2,RPB$L_ADPVIR(R9)    ; Store VA of boot adapter space.
      20 A9    54    D0    00E8   1032          MOVL    R4,RPB$L_BOOTR1(R9)    ; Store boot adapter nexus number.
51   54 A9   0D    00    EF    00EC   1033          EXTZV   #0,#13, -              ; Get offset into UNIBUS/QBUS I/O page.
                           00F2   1034                  RPB$L_CSRPHY(R9),R1    ;
   58 A9    1000 C241    9E    00F2   1035          MOVAB   <8*512>(R2)[R1], -     ; Set VA of UNIBUS/QBUS registers.
                           00F9   1036                  RPB$L_CSRVIR(R9)       ;
                           00F9   1037
                           00F9   1038
                           00F9   1039  ;
                           00F9   1040  ; R5/ general adapter type; index into "general" adapter arrays.
                           00F9   1041  ; For each adapter -
```

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

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

```
                                        0173          .SBTTL  MAP_PAGES
                                        0173  ;++
                                        0173  ; INPUTS:
                                        0173  ;       R1/ Number of pages to map.
                                        0173  ;       R2/ VA of page to map.
                                        0173  ;       R3/ VA of system page table entry to be used.
                                        0173  ;       R8/ PFN of page(s) to map.
                                        0173  ;
                                        0173  ; OUTPUTS:
                                        0173  ;       R2,R3 updated; R1,R8 destroyed; all other registers preserved
                                        0173  ;
                                        0173  ;--
                                        0173
                                        0173  MAP_PAGES:
                                        0173
     83   58   90000000 8F        C9    0173          BISL3   #<PTE$M_VALID!PTE$C_KW>,R8,(R3)+
                                        017B                                                ; Map a page.
                              58  D6    017B          INCL    R8                            ; Next PFN.
                    52   0200 C2  9E    017D          MOVAB   512(R2),R2                    ; Next VA.
                       00000000'GF  D6  0182          INCL    G^BOO$GL_SPTFREL              ; Next free entry.
        00000000'GF  00000000'GF  D1    0188          CMPL    G^BOO$GL_SPTFREH, -           ; Check for no more system page
                                        0193                  G^BOO$GL_SPTFREL              ; table entries.
                              04  15    0193          BLEQ    ERROR_HALT                    ; Branch if out of SPTEs.
                           DB 51  F5    0195          SOBGTR  R1,MAP_PAGES                  ; Map another page.
                              05        0198          RSB                                   ; All done.
                                        0199
                                        0199  ERROR_HALT:
                  51   02E4'CF    9E    0199          MOVAB   W^NOSPT,R1                    ; Set error message.
                                        019E  ERROR_HALT_1:
                              5B  D4    019E          CLRL    R11                           ; Indicate console terminal.
                       00000000'GF  16  01A0          JSB     G^EXE$OUTZSTRING              ; Output error message.
                              00        01A6          HALT                                  ; ***** FATAL ERROR *******
```

```
                                01A7          .SBTTL  INI$UBSPACE
                                01A7    ;++
                                01A7    ;
                                01A7    ;       Map UNIBUS space; initialize UNIBUS ADP.
                                01A7    ;
                                01A7    ; INPUTS:
                                01A7    ;       R2 - VA of next free system page
                                01A7    ;       R3 - VA of system page table entry to be used to map VA in R2
                                01A7    ;       R4 - nexus identification number of this adapter
                                01A7    ;    -8(R6) - PFN of this UNIBUS adapter's register space
                                01A7    ;
                                01A7    ; OUTPUTS:
                                01A7    ;       UNIBUS space is mapped.
                                01A7    ;       INI$UBADP is called to build an ADP block and initialize UNIBUS
                                01A7    ;       adapter hardware.
                                01A7    ;
                                01A7    ;--
                                01A7
                                01A7    INI$UBSPACE:
                                01A7
          58   01E4'CF44    DE  01A7          MOVAL   W^CONFREGL[R4],R8                 ; R8 => CONFREGL slot.
       58   68     02   00  EF  01AD          EXTZV   #0,#2,(R8),R8                     ; Get UBA number.
          58     58     09  78  01B2          ASHL    #9,R8,R8                          ; Position UB number.
                                01B6
                                01B6
                                01B6
   58   00007FF0 8F     58  C3  01B6          SUBL3   R8,#<IO750$AL_UB0SP+^0760000/^X200>,R8
                                01BE                                            ; Get PFN of UB I/O page.
                                01BE
                                01BE
                                01BE
                                01BE
                 51     10  D0  01BE          MOVL    #16,R1                    ; Number of pages to map (UB/Qbus space).
                      FFAF  30  01C1          BSBW    MAP_PAGES                 ; Map I/O pages.
                                01C4    ;
                                01C4    ; Call adapter initialization routine.
                                01C4    ;
                                01C4    ;       BSBW    INI$UBADP               ; Init ADP block.
                                01C4    ;       RSB
```

```
                                  01C4         .SBTTL  INI$UBADP - BUILD ADP AND INITIALIZE UBA
                                  01C4    ;+
                                  01C4    ; INI$UBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
                                  01C4    ; DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS.  A CALL IS
                                  01C4    ; THEN MADE TO UBA$INITIAL TO INITIALIZE THE ADAPTER HARDWARE.
                                  01C4    ;
                                  01C4    ; INPUT:
                                  01C4    ;       R4 - nexus identification number of this adapter
                                  01C4    ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                                  01C4    ;-
                                  01C4
                                  01C4    INI$UBADP:
                                  01C4
              01FF 8F       BB    01C4          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; SAVE R0-R8
                                  01C8    ;
                                  01C8    ; Allocate and initialize Adapter Control Block (ADP).
                                  01C8    ;
     51    0019'CF          3C    01C8          MOVZWL  W^CPU_ADPSIZE,R1         ; PICK UP LENGTH OF ADP
                0156        30    01CD          BSBW    ALONPAGD                 ; ALLOCATE SPACE FOR ADP
        08 A2    51         B0    01D0          MOVW    R1,ADP$W_SIZE(R2)        ; SET SIZE INTO ADP BLOCK
        0A A2    01         90    01D4          MOVB    #DYN$C_ADP, -            ; AND SET TYPE OF BLOCK
                                  01D8                  ADP$B_TYPE(R2)
        0E A2    01         B0    01D8          MOVW    #AT$_UBA, -              ; SET TYPE OF ADAPTER
                                  01DC                  ADP$W_ADPTYPE(R2)
     62    00E4'CF44        D0    01DC          MOVL    W^SBICONF[R4], -         ; SET VA OF CONFIGURATION REG
                                  01E2                  ADP$L_CSR(R2)
        0C A2    54         B0    01E2          MOVW    R4,ADP$W_TR(R2)          ; SET TR NUMBER FOR ADAPTER
                                  01E6
     50    14 A2            DE    01E6          MOVAL   ADP$L_DPQFL(R2),R0       ; ADDRESS OF DATA PATH WAIT QUEUE
           60    50         D0    01EA          MOVL    R0,(R0)                  ; INIT QUEUE HEADER
        04 A0    50         D0    01ED          MOVL    R0,4(R0)                 ;
                                  01F1
     50    30 A2            DE    01F1          MOVAL   ADP$L_MRQFL(R2),R0       ; ADDRESS OF MAP WAIT QUEUE
           60    50         D0    01F5          MOVL    R0,(R0)                  ; INIT QUEUE HEADER
        04 A0    50         D0    01F8          MOVL    R0,4(R0)                 ;
              04 A2         D4    01FC          CLRL    ADP$L_LINK(R2)           ; ZAP ADAPTER CHAIN LINK
               FDFE'        30    01FF          BSBW    ADPLINK                  ; LINK ADP TO END OF LIST
                                  0202    ;
                                  0202    ; Initialize adapter interrupt vectors in System Control Block.
                                  0202    ;
  58    00000000'GF         D0    0202          MOVL    G^EXE$GL_SCB,R8          ; GET SCB ADDRESS
                                  0209
                                  0209
                                  0209
                                  0209
                  28        DD    0209          PUSHL   #NDT$_UB0                ; ASSUME UB0
             0200 C8        DE    020B          MOVAL   ^X200(R8),-              ; GET VECTOR SPACE FOR UB0
                10 A2             020F                  ADP$L_VECTOR(R2)         ;
     28    01E4'CF44        D1    0211          CMPL    W^CONFREGL[R4],#NDT$_UB0; IS DEVICE TYPE = UB0?
                  0B        13    0217          BEQL    10$                      ; BRANCH IF SO
             6E   29        D0    0219          MOVL    #NDT$_UB1,(SP)           ; INDICATE UB1
 10 A2   00000200 8F        C0    021C          ADDL    #^X200,ADP$L_VECTOR(R2)  ; STEP TO ITS VECTOR SPACE
          60 A2   0E        B0    0224    10$:  MOVW    #^XE,ADP$W_DPBITMAP(R2)  ; MARK DATAPATHS 1-3 AVAILABLE
             53   62        D0    0228          MOVL    ADP$L_CSR(R2),R3         ; VIRTUAL ADDRESS OF ADAPTER
     53    0800 C3          9E    022B          MOVAB   UBA$L_MAP(R3),R3         ; POINT TO MAPPING REGISTERS
     54    01F0 8F          3C    0230          MOVZWL  #496,R4                  ; NUMBER OF UMR TO DISABLE
                  83        D4    0235    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  ; 8Kb of memory.  The number of registers is stored in the ADP and the
                                  029B  ; corresponding number withdrawn from the UMR map in the ADP.
                                  029B  ;
                                  029B
                 56    62    D0   029B          MOVL    ADP$L_CSR(R2),R6         ; Pick up adapter pointer
                       51    D4   029E          CLRL    R1                       ; Zero out number of UMR to disable
  57   08 AE  00000200 8F    C3   02A0          SUBL3   #512,8(SP),R7            ; R7 = VA of last page of UNIBUS
          58   0C AE   04    C3   02A9          SUBL3   #4,12(SP),R8             ; R8 = VA of SPTE mapping (R7)
  54   20 AE  00000200 8F    C3   02AE          SUBL3   #512,32(SP),R4           ; R4 = PFN of first page of UNIBUS
                       68    DD   02B7          PUSHL   (R8)                     ; Save contents of SPTE
                 53    54    D0   02B9          MOVL    R4,R3                    ; Copy starting PFN
                 55    1F    D0   02BC          MOVL    #31,R5                   ; 31 8Kb chunks to test
                                  02BF  50$:    INVALID R7                       ; Invalidate TB
              90000000 8F    C9   02C2          BISL3   #<PTE$M_VALID!PTE$C_KW>,-
                 68    54         02C8           R4,(R8)                         ; Map each page of UNIBUS
                 50    57    D0   02CA          MOVL    R7,R0                    ; Address to check
                   FD30'     30   02CD          BSBW    EXE$TEST_CSR             ; Validate it
                 0D    50    E9   02D0          BLBC    R0,70$                   ; Not there
                 54    53    D1   02D3          CMPL    R3,R4                    ; First time in?
                       04    13   02D6          BEQL    60$                      ; Yes, skip next test
                       51    D5   02D8          TSTL    R1                       ; Any registers already?
                       3A    13   02DA          BEQL    80$                      ; No, memory not start at 0
            51    10 A1      9E   02DC  60$:    MOVAB   16(R1),R1                ; Yes, up the count
            54    10 A4      9E   02E0  70$:    MOVAB   16(R4),R4                ; Map Next 8Kb (16*512)
                    D8 55    F5   02E4          SOBGTR  R5,50$                   ; Loop until done
                    68 8ED0       02E7          POPL    (R8)                     ; Restore old contents of SPTE
                                  02EA          INVALID R7                       ; Invalidate TB
          0256 C2      51    B0   02ED          MOVW    R1,ADP$W_UMR_DIS(R2)     ; Record number disabled
                                  02F2  ;
                                  02F2  ; Initialize fields for new UBA map register allocation.  Make it appear
                                  02F2  ;       that we have one contiguous array of 496 available map registers.
                                  02F2  ;       To do this we set ADP$L_MRACTMDRS to one (the number of active
                                  02F2  ;       map register descriptors for distinct contiguous areas),
                                  02F2  ;       ADP$W_MRNREGARY(0) to 496 (i.e the number of registers in this
                                  02F2  ;       contiguous range) and ADP$FREGARY(0) to 0 (i.e. the first register
                                  02F2  ;       in the range is register 0).
                                  02F2  ;
            5C A2      01    D0   02F2          MOVL    #1,ADP$L_MRACTMDRS(R2)   ; 1 active map descriptor
     64 A2   01F0 8F   51    A3   02F6          SUBW3   R1,#496,ADP$W_MRNREGARY(R2);  for a range of 496 registers
          015E C2      51    B0   02FD          MOVW    R1,ADP$W_MRFREGARY(R2)   ;  starting at register zero.
            62 A2      01    AE   0302          MNEGW   #1,ADP$W_MRNFENCE(R2)    ; Also init "fences" which preceed
          015C C2      01    AE   0306          MNEGW   #1,ADP$W_MRFFENCE(R2)    ;  the two descriptor arrays.
                                  030B  ;
                                  030B  ; Initialize adapter hardware.
                                  030B  ;
                 54    62    D0   030B          MOVL    ADP$L_CSR(R2),R4         ; Get CSR address to init
                   FCEF'     30   030E          BSBW    UBA$INITIAL              ; And initialize adapter
               01FF 8F       BA   0311          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; Restore registers
                             05   0315          RSB                              ; Return
                                  0316
                                  0316  ;
                                  0316  ; Error if UNIBUS memory not start at location 0
                                  0316  ;
            51    030D'CF    9E   0316  80$:    MOVAB   W^BADUMR,R1              ; Set error message
                     FE80    31   031B          BRW     ERROR_HALT_1             ; Put it out
                                  031E
```

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

```
                          0326          .SBTTL  INI$MBADP - BUILD ADP AND INITIALIZE MBA
                          0326          .SBTTL  INI$DRADP - BUILD ADP AND INITIALIZE DR32
                          0326          .SBTTL  INI$CIADP - BUILD ADP AND INITIALIZE CI
                          0326   ;+
                          0326   ; INI$MBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                          0326   ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                          0326   ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                          0326   ; BY CALLING MBA$INITIAL.
                          0326   ;
                          0326   ; INI$DRADP IS CALLED AFTER MAPPING THE REGISTERS FOR THE DR32
                          0326   ; ADAPTER.  THE ADAPTER CONTROL BLOCK, CRB, AND IDB ARE ALLOCATED
                          0326   ; AND INITIALIZED.  THE ADAPTER HARDWARE IS THEN INITIALIZED BY
                          0326   ; CALLING DR$INITIAL.
                          0326   ;
                          0326   ; INI$MBADP AND INI$DRADP SHARE COMMON CODE AFTER THE TABLE OF ADAPTER
                          0326   ; SPECIFIC CONSTANTS IS SELECTED AND STORED IN R8.
                          0326   ;
                          0326   ; INPUT:
                          0326   ;       R4 - nexus identification number of this adapter
                          0326   ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                          0326   ;
                          0326   ; OUTPUTS:
                          0326   ;       ALL REGISTERS PRESERVED
                          0326   ;-
                          0326
   00000000'GF       17   0326   ALONPAGD:JMP   G^INI$ALONONPAGED
                          032C
                          032C          .ENABL  LSB
                          032C
                          032C   INI$DRADP:                                  ; INITIALIZE DR32 DATA STRUCTURES
                          032C
      07FF 8F        BB   032C          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
58    0011'CF        DE   0330          MOVAL   W^DRTAB,R8                  ; GET DR32 TABLE OF CONSTANTS
59    0000'CF        9E   0335          MOVAB   W^DR$INT,R9                 ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A    0000'CF        9E   033A          MOVAB   W^DR$INITIAL,R10            ; ADDRESS OF DEVICE INITIALIZATION
           28        11   033F          BRB     10$                         ; JOIN COMMON CODE
                          0341
                          0341   INI$CIADP:                                  ; INITIALIZE CI DATA STRUCTURES
                          0341
      07FF 8F        BB   0341          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
58    0015'CF        DE   0345          MOVAL   W^CITAB,R8                  ; GET CI TABLE OF CONSTANTS
59    0000'CF        9E   034A          MOVAB   W^CI$INT,R9                 ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A    0000'CF        9E   034F          MOVAB   W^CI$INITIAL,R10            ; ADDRESS OF DEVICE INITIALIZATION
           13        11   0354          BRB     10$                         ; JOIN COMMON CODE
                          0356
                          0356   INI$MBADP:                                  ; INIT MBA DATA STRUCTURES
                          0356
      07FF 8F        BB   0356          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ;
58    000D'CF        DE   035A          MOVAL   W^MBATAB,R8                 ; GET MBA TABLE OF CONSTANTS
59    0000'CF        9E   035F          MOVAB   W^MBA$INT,R9                ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A    0000'CF        9E   0364          MOVAB   W^MBA$INITIAL,R10           ; ADDRESS OF DEVICE INITIALIZATION
                          0369   10$:                                        ;
                          0369   ;
                          0369   ; Allocate and initialize Channel Request Block.
                          0369   ;
51    0048 8F        3C   0369          MOVZWL  #CRB$C_LENGTH,R1            ; SET SIZE OF CRB
           B6        10   036E          BSBB    ALONPAGD                    ; ALLOCATE SPACE FOR CRB
           08 A2    51   B0  0370           MOVW    R1,CRB$W_SIZE(R2)          ; SET CORRECT SIZE
           0A A2    05   90  0374           MOVB    #DYN$C_CRB,CRB$B_TYPE(R2)  ; SET CORRECT TYPE
              62    62   DE  0378           MOVAL   CRB$L_WQFL(R2),CRB$L_WQFL(R2)    ; INITIALIZE WAIT QUEUE HEADER
           04 A2    62   DE  037B           MOVAL   CRB$L_WQFL(R2),CRB$L_WQBL(R2)    ; FLINK AND BLINK
           50    24 A2   9E  037F           MOVAB   CRB$L_INTD(R2),R0          ; SET ADDRESS OF INTD AREAD
  80     9F163CBB 8F     D0  0383           MOVL    #^X9F163CBB,(R0)+          ; "PUSHR ^M<R2,R3,R4,R5>,JSB @#"
                 80 59   D0  038A           MOVL    R9,(R0)+                   ; ADDR OF XXX$INT ROUTINE
                    80   D4  038D           CLRL    (R0)+                      ; CLEAR OUT UNNEEDED AREA
              60    5A   D0  038F           MOVL    R10,(R0)                   ; ADDR OF XXX$INITIAL ROUTINE
              5A    52   D0  0392           MOVL    R2,R10                     ; SAVE CRB ADDRESS
                             0395      ;
                             0395      ; Allocate and initialize Interrupt Dispatch Block.
                             0395      ;
              51    68   9A  0395           MOVZBL  ADPTAB_IDBUNITS(R8),R1     ; GET # OF IDB UNITS
  51     00000038 9F41   DE  0398           MOVAL   @#IDB$C_LENGTH[R1],R1      ; GET TOTAL SIZE OF IDB
                    84   10  03A0           BSBB    ALONPAGD                   ; ALLOCATE SPACE FOR CRB
           08 A2    51   B0  03A2           MOVW    R1,IDB$W_SIZE(R2)          ; SET STRUCTURE SIZE
           0A A2    09   90  03A6           MOVB    #DYN$C_IDB, -              ; AND TYPE CODE
                             03AA                   IDB$B_TYPE(R2)
                    68   9B  03AA           MOVZBW  ADPTAB_IDBUNITS(R8),-      ; SET COUNT OF UNITS
                 0C A2       03AC                   IDB$W_UNITS(R2)
  62       00E4'CF44     D0  03AE           MOVL    W^SBICONF[R4], -           ; SET CSR ADDRESS TO
                             03B4                   IDB$L_CSR(R2)              ;  START OF ADAPTER REG SPACE
           2C AA    52   D0  03B4           MOVL    R2, -                      ; SET ADDRESS OF IDB INTO CRB
                             03B8                   CRB$L_INTD+VEC$L_IDB(R10)
              59    52   D0  03B8           MOVL    R2,R9                      ; SAVE ADDRESS OF IDB
                             03BB      ;
                             03BB      ; Allocate and initialize Adapter Control Block (ADP).
                             03BB      ;
           51    01 A8   3C  03BB           MOVZWL  ADPTAB_ADPLEN(R8),R1       ; GET SIZE OF ADAPTER
                  FF64   30  03BF           BSBW    ALONPAGD                   ; ALLOCATE SPACE FOR CRB
           08 A2    51   B0  03C2           MOVW    R1,ADP$W_SIZE(R2)          ; SET SIZE OF STRUCTURE
           0A A2    01   90  03C6           MOVB    #DYN$C_ADP,ADP$B_TYPE(R2); AND TYPE CODE
              62    69   D0  03CA           MOVL    IDB$L_CSR(R9),ADP$L_CSR(R2); SET ADDRESS OF CONFIGURATION REGISTER
           0C A2    54   B0  03CD           MOVW    R4,ADP$W_TR(R2)            ; SET TR/SLOT-16 NUMBER OF ADAPTER
                 03 A8   9B  03D1           MOVZBW  ADPTAB_ATYPE(R8),-         ; SET THE ADAPTER TYPE
                 0E A2       03D4                   ADP$W_ADPTYPE(R2)
           10 A2    5A   D0  03D6           MOVL    R10,ADP$L_CRB(R2)          ; POINT ADP TO CRB
                             03DA      ;    CMPW    ADP$W_ADPTYPE(R2),#AT$_CI  ; CI?
                             03DA      ;    BEQL    20$                        ; YES, DO NOT CONNECT UP VECTORS
                             03DA      ;
                             03DA      ; Initialize adapter interrupt vectors in System Control Block.
                             03DA      ;
  50     00000000'GF     D0  03DA           MOVL    G^EXE$GL_SCB,R0            ; GET ADDRESS OF SCB
     55     5B      09   78  03E1           ASHL    #9,R11,R5                  ; Turn SCB page offset into byte offset.
              50    55   C0  03E5           ADDL    R5,R0                      ; set to beginning of correct SCB page.
  54    54  04      00   EF  03E8           EXTZV   #0,#4,R4,R4                ; Use low 4 bits of nexus number.
     50     0100 C044    DE  03ED           MOVAL   ^X100(R0)[R4],R0           ; COMPUTE ADDR OF 1ST VECTOR
          1C A2     50   D0  03F3           MOVL    R0,ADP$L_AVECTOR(R2)       ; SAVE ADDR OF ADAPTER'S SCB VECTORS
            60     25 AA DE  03F7           MOVAL   CRB$L_INTD+1(R10),(R0)     ; CONNECT VECTOR TO CRB CODE
         40 A0     25 AA DE  03FB           MOVAL   CRB$L_INTD+1(R10),64(R0);  SAME FOR
       0080 C0     25 AA DE  0400           MOVAL   CRB$L_INTD+1(R10),128(R0); ALL FOUR
       00C0 C0     25 AA DE  0406           MOVAL   CRB$L_INTD+1(R10),192(R0); VECTORS
                             040C      ;
                             040C      ; Continue with ADP initialization.
                             040C      ;
        14 A2      25 AA DE  040C      20$: MOVAL   CRB$L_INTD+1(R10), -       ; SAVE SCB VECTOR CONTENTS IN ADP
```

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

```
      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          .SBTTL  INI$CONSOLE, init data structures for console
                         043F  ;++
                         043F  ; FUNCTIONAL DESCRIPTION:
                         043F  ;
                         043F  ;       This routine is executed only once, during system initialization.
                         043F  ;       It initializes the CRB and IDB for boot/console device.
                         043F  ;
                         043F  ;       This routine is called from INIT.
                         043F  ;
                         043F  ; INPUTS:
                         043F  ;
                         043F  ;       R3 -->  DISK [CLASS] DRIVER DDB
                         043F  ;       R4 -->  DISK [CLASS] DRIVER DPT
                         043F  ;       R5 -->  DISK [CLASS] DRIVER UCB
                         043F  ;       R6 -->  RPB
                         043F  ;       R7 -->  ADP FOR EITHER A REAL DISK OR A PORT
                         043F  ;       R9 -->  PORT DRIVER DPT (IF PRESENT)
                         043F  ;       R10-->  PORT DIRVER UCB (IF PRESENT)
                         0.3F  ;
                         043F  ;--
                         043F
                         043F  INI$CONSOLE::
                         043F          .ENABL  LSB
                         043F
           66 A6    91   043F          CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
           40 8F         0442                  #BTD$K_CONSOLE          ; STORAGE DEVICE?
              10    12   0444          BNEQ    BLD_CRB                 ; NO
     41534303 8F    D0   0446          MOVL    #^A7CSA/@8+3,-          ; YES, SET DEVICE NAME
           14 A3         044C                  DDB$T_NAME(R3)          ; COUNTED STRING
                         044E
                         044E
              57    D4   044E          CLRL    R7                      ; CLEAR ADP POINTER
        54 A5 01    B0   0450          MOVW    #1,UCB$W_UNIT(R5)       ; SET UNIT NUMBER TO 1
              0D    11   0454          BRB     FILL_CRB
                         0456
                         0456  ;
                         0456  ; NOW BUILD THE AUXILIARY DATA BLOCKS (CRB,IDB)
                         0456  ;
                         0456  BLD_CRB:
        58 10 A7    D0   0456          MOVL    ADP$L_CRB(R7),R8        ; GET ADDRESS OF CRB IF IT EXISTS
        0E A7 01    B1   045A          CMPW    #AT$_UBA,ADP$W_ADPTYPE(R7); IS THIS A UNIBUS ADAPTER?
              03    13   045E          BEQL    FILL_CRB                ; YES, ALLOCATE CRB
            008A    31   0460          BRW     100$                    ; NO, CRB/IDB ALREADY ALLOCATED
                         0463
                         0463  FILL_CRB:
     00000000'9F    16   0463          JSB     @#INI$ALLOC_CRB         ; GO ALLOCATE AND SETUP CRB
24 A2 9F163FBB 8F   D0   0469          MOVL    #^X9F163FBB,CRB$L_INTD(R2) ; SET PUSHR #^M<R0,...R5>
                         0471                                          ; JSB @#0 INTO INTERRUPT DISPATCH
        38 A2 57    D0   0471          MOVL    R7,CRB$L_INTD+VEC$L_ADP(R2)     ; SET POINTER TO ADP
           58 52    D0   0475          MOVL    R2,R8                   ; SAVE CRB POINTER
   51 0058 8F       3C   0478          MOVZWL  #<IDB$C_LENGTH+<8*4>>,R1; SIZE TO ALLOCATE FOR IDB
     00000000'9F    16   047D          JSB     @#INI$ALONONPAGED       ; ALLOCATE IDB
        08 A2 51    B0   0483          MOVW    R1,IDB$W_SIZE(R2)       ; SET SIZE OF IDB
        0A A2 09    90   0487          MOVB    #DYN$C_IDB,IDB$B_TYPE(R2); AND STRUCTURE TYPE CODE
        2C A8 52    D0   048B          MOVL    R2,CRB$L_INTD+VEC$L_IDB(R8) ; SET IDB INTO CRB
                         048F
           66 A6    91   048F          CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
```

```
                     40 8F         0492                    #BTD$K_CONSOLE               ; STORAGE DEVICE?
                        26    12   0494            BNEQ    10$                          ; NO
50   000000F0 8F   00000000'9F    C1   0496            ADDL3   @#EXE$GL_SCB,#^XF0,R0        ; YES, GET ADDRESS OF VECTOR IN SCB
                 80    25 A8    DE   04A2            MOVAL   CRB$L_INTD+1(R8),(R0)+       ; SET ADDR IN 1ST VECTOR
                 60    49 A8    DE   04A6            MOVAL   CRB$L_INTD2+1(R8),(R0)       ; SET ADDR IN 2ND VECTOR
        48 A8   9F163FBB 8F    D0   04AA            MOVL    #^X9F163FBB,CRB$L_INTD2(R8) ; STORE PUSHR #^M<R0...R5>
                                     04B2                                                 ; JMP @# IN 2ND INT. DISPATCH
                 50 A8    52    D0   04B2            MOVL    R2,CRB$L_INTD2+VEC$L_IDB(R8); STORE ADDRESS OF IDB IN CRB
                 2C B8    1F    D0   04B6            MOVL    #PR$_CSTD, -                 ; STORE IPR NUMBER OF CONSOLE INTERFACE
                                     04BA                    @CRB$L_INTD+8(R8)            ; REGISTER AS DEVICE CSR ADDRESS
                          31    11   04BA            BRB     100$
                                     04BC
                 62    58 A6    D0   04BC    10$:    MOVL    RPB$L_CSRVIR(R6), -          ; SAVE BOOT DEVICE CSR ADDRESS
                                     04C0                    IDB$L_CSR(R2)                ; IN INTERRUPT DISPATCH BLOCK
                          11    91   04C0            CMPB    #BTD$K_UDA,-                 ; LOW ORDER BYTE OF ORIGINAL R0 TELLS
                       66 A6         04C2                    RPB$B_DEVTYP(R6)             ;  BOOT DEVICE TYPE.
                          08    12   04C4            BNEQ    20$                          ; IF NOT BOOTING FROM A UDA BRANCH
                                     04C6                                                 ;  AROUND.
        00000000'9F    58 A6    D0   04C6            MOVL    RPB$L_CSRVIR(R6), -          ; COPY VIRTUAL ADDRESS OF UDA PORT CSR
                                     04CE                    @#BOO$GB_SYSTEMID            ;  TO LOW ORDER LONGWORD OF SYSTEMID
                                     04CE    20$:
                 14 A2    57    D0   04CE            MOVL    R7,IDB$L_ADP(R2)             ; POINT IDB TO ADP
                 50    1E A6    3C   04D2            MOVZWL  RPB$W_ROUBVEC(R6),R0         ; GET USER SPECIFIED VECTOR
                          0A    12   04D6            BNEQ    30$                          ; BRANCH IF VECTOR SPECIFIED
                 50    66 A6    9A   04D8            MOVZBL  RPB$B_DEVTYP(R6),R0          ; ELSE GET DEVICE TYPE CODE
             50   FFFE'CF40     3C   04DC            MOVZWL  W^BOOTVECTOR-2[R0],R0        ; GET DEFAULT INTERRUPT VECTOR
              50    10 B740     9E   04E2    30$:    MOVAB   @ADP$L_VECTOR(R7)[R0],R0;    COMPUTE ADDRESS OF VECTOR
                 60    26 A8    9E   04E7            MOVAB   CRB$L_INTD+2(R8),(R0)        ; SET ADDR OF INTERRUPT VECTOR
                                     04EB
                          60    D7   04EB            DECL    (R0)                         ;  BACK TWO BYTES TO PUSHR, +1 TO
                                     04ED                                                 ;
                                     04ED    100$:
                                05   04ED            RSB                                  ; RETURN
                                     04EE            .DISABLE LSB
                          04EE         .SBTTL  EXE$INI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
                          04EE  ;++
                          04EE  ; FUNCTIONAL DESCRIPTION:
                          04EE  ;
                          04EE  ; EXE$INI_TIMWAIT initializes EXE$GL_TENUSEC and EXE$GL_UBDELAY, cells used
                          04EE  ; in the time-wait macros.  The first data cell, EXE$GL_TENUSEC, is the number
                          04EE  ; of times the following loop will be executed in ten u-seconds.  This is
                          04EE  ; done once here to calibrate the loop instead of reading the processor clock.
                          04EE  ; The resulting number is used in the system macros TIMEWAIT and TIMEDWAIT.
                          04EE  ;
                          04EE  ; The first step is to initialize EXE$GL_UBDELAY.  If the bit test instruction
                          04EE  ; in the TIMEWAIT macro is executed too rapidly in a loop, it can saturate the
                          04EE  ; Unibus.  EXE$GL_UBDELAY is used to introduce a 3 microsecond delay loop into
                          04EE  ; the TIMEWAIT bit test loop.
                          04EE  ;
                          04EE  ; This routine is called only once, from INIT.
                          04EE  ;
                          04EE  ; INPUT PARAMETERS:
                          04EE  ;
                          04EE  ;       NONE
                          04EE  ;
                          04EE  ; IMPLICIT INPUTS:
                          04EE  ;
                          04EE  ;       Time-of-day processor clock.
                          04EE  ;       Interval timers.
                          04EE  ;
                          04EE  ; OUTPUT PARAMETERS:
                          04EE  ;
                          04EE  ;       R0 - Destroyed.
                          04EE  ;
                          04EE  ; IMPLICIT OUTPUTS:
                          04EE  ;
                          04EE  ;       EXE$GL_TENUSEC - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                          04EE  ;                        macros loop for 10 micro-seconds.
                          04EE  ;
                          04EE  ;       EXE$GL_UBDELAY - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                          04EE  ;                        macros loop for 3 micro-seconds in the unibus delay
                          04EE  ;                        loop.
                          04EE  ;
                          04EE  ;--
                          04EE
                          04EE  EXE$INI_TIMWAIT::                       ; Initialize time-wait data cells
                          04EE          .ENABLE LSB
                          04EE
                          04EE
          19    00    DA  04EE          MTPR    #0,#PR750$_NICR         ; Initialize next interval count register.
                          04F1
                          04F1
                          04F1
7E  00004E20 8F    D0     04F1          MOVL    #20000,-(SP)            ; # of times to execute timed loop.
          18    11    DA  04F8          MTPR    #^X11,#PR$_ICCS         ; Start clock, no interrupts.
                          04FB
                          04FB  ; * * * start of loop to time * * *
             FD 6E    F5  04FB  10$:    SOBGTR  (SP),10$                ; Delay loop.
                          04FE  ; * * * end of loop to time * * *
                          04FE
                          04FE
```

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

```
                              0556         .SBTTL  EXE$INIT_TODR  -  SET SYSTEM TIME TO CORRECT VALUE AT STARTUP
                              0556  ;++
                              0556  ; FUNCTIONAL DESCRIPTION:
                              0556  ;
                              0556  ;      EXE$INIT_TODR SOLICITS THE CORRECT TIME FROM THE OPERATOR IF NECESSARY,
                              0556  ;      CONVERTS THE ASCII RESPONSE TO BINARY FORMAT AND CALLS AN INTERNAL
                              0556  ;      ENTRY POINT OF THE $SETIME SYSTEM SERVICE TO SET THE NEW SYSTEM TIME
                              0556  ;      IN MEMORY WITHOUT MODIFYING THE CONTENTS OF THE SYSTEM DISK.
                              0556  ;
                              0556  ;      IF THE TIME WOULD NORMALLY BE SOLICITED FROM AN OPERATOR, BECAUSE
                              0556  ;      THE HARDWARE TIME OF YEAR CLOCK IS ZERO, THEN THE SYSGEN PARAMETER
                              0556  ;      "TPWAIT" IS CHECKED.  IF IT IS ZERO, THEN IT IS ASSUMED THAT NO
                              0556  ;      OPERATOR IS PRESENT AND THE SYSTEM IS BOOTED USING THE LAST TIME
                              0556  ;      RECORDED IN THE SYSTEM IMAGE.  IF THE PARAMETER IS NON ZERO THEN
                              0556  ;      THAT TIME IS USED AS THE MAXIMUM TIME TO WAIT BEFOR ASSUMING THAT
                              0556  ;      THERE IS NO OPERATOR AND BOOTING ANY WAY.  IF THE PARAMETER IS
                              0556  ;      NEGATIVE, THE SYSTEM WILL WAIT FOREVER.
                              0556  ;
                              0556  ;      THIS ROUTINE IS CALLED ONLY ONCE, FROM SYSINIT OR STASYSGEN.
                              0556  ;
                              0556  ; INPUT PARAMETERS:
                              0556  ;
                              0556  ;      NONE
                              0556  ;
                              0556  ; IMPLICIT INPUTS:
                              0556  ;
                              0556  ;      TIME-OF-DAY PROCESSOR CLOCK.
                              0556  ;
                              0556  ; OUTPUT PARAMETERS:
                              0556  ;
                              0556  ;      R0,R1 - DESTROYED
                              0556  ;
                              0556  ; IMPLICIT OUTPUTS:
                              0556  ;
                              0556  ;      EXE$GQ_SYSTIME - SET TO CURRENT TIME IN 100 NANOSECOND UNITS SINCE
                              0556  ;                       17-NOV-1858  00:00:00.
                              0556  ;
                              0556  ;--
                              0556
                              0556  ;
                              0556  ; Stack storage offsets:
                              0556  ;
                    00000000  0556  TTCHAN  = ^X00                        ; CHANNEL FOR TERMINAL (LONGWORD)
                    00000004  0556  TTNAME  = ^X04                        ; STRING DESCRIPTOR FOR OPERATOR'S TERM
                    0000000C  0556  TMPDESC = ^X0C                        ; TEMPORY STRING DESCRIPTOR (QUADWORD)
                    00000014  0556  INTIME  = ^X14                        ; INPUT TIME VALUE (QUADWORD)
                    0000001C  0556  LINBUF  = ^X1C                        ; INPUT LINE BUFFER (5 LONGWORDS)
                    00000014  0556  LINBUFSIZ = ^X14                      ;  (LENGTH OF LINE BUFFER IN BYTES)
                              0556
                              0556  ;
                              0556  ; PURE DATA
                              0556  ;
                              0556  TERM_NAMADR:
                 30 41 50 4F  0556          .ASCII  \OPA0\               ; DEVICE NAME FOR OPERATOR'S TERMINAL
                    00000004  055A  TERM_NAMSIZ = . - TERM_NAMADR
74 61 64 20 64 69 6C 61 76 6E 69 00' 055A  TIMERR: .ASCIC  \invalid date/time\     ;
           65 6D 69 74 2F 65  0566
```

```
                                 11  055A
                                     056C  TIMEPROMPT:
                                33'  056C          .BYTE   NPROMPT
54 4E 45 20 45 53 41 45 4C 50 0A 0D  056D          .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 29  059D
                           00000033  05A0  NPROMPT=.-TIMEPROMPT-1
                                     05A0
                                     05A0
                                     05A0  EXE$INIT_TODR::                          ; SET CORRECT TIME
                                     05A0          .ENABLE LSB
                      077C 8F    BB  05A0          PUSHR   #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; SAVE REGISTERS
                        5E 30    C2  05A4          SUBL    #4*12,SP                 ; SCRATCH STORAGE
                        56 5E    D0  05A7          MOVL    SP,R6                    ; SAVE ADDRESS OF SCRATCH STORAGE
                     04 A6 04    9A  05AA          MOVZBL  #TERM_NAMSIZ,TTNAME(R6)  ; SET SIZE OF OPERATOR'S TERM NAME AND
                08 A6   FFA4 CF  9E  05AE          MOVAB   W^TERM_NAMADR,TTNAME+4(R6) ;  PIC ADDRESS INTO TERM NAME DESC
            1C 00000000'GF   00' E0  05B4          BBS     S^#EXE$V_SETTIME,G^EXE$GL_FLAGS,RFADTIME ; BR TO SOLICIT TIME
                                     05BC
                                     05BC
                                     05BC
                        50 1B    DB  05BC          MFPR    #PR750$_TODR,R0          ; GET TIME OF DAY CLOCK VALUE
                                     05BF
                                     05BF
                                     05BF
         59   00000000'GF   50   C3  05BF          SUBL3   R0,G^EXE$GL_TODR,R9      ; GET TOD DELTA TIME (10 MS UNITS)
                            09   1B  05C7          BLEQU   5$                       ; BRANCH IF TIME IS LATER
              0083D600 8F   59   D1  05C9          CMPL    R9,#24*60*60*100         ; CHECK FOR SETBACK OF ONE DAY
                            06   1E  05D0          BGEQU   READTIME                 ; MORE, MUST SOLICIT TIME
                         14 A6   7C  05D2  5$:     CLRQ    INTIME(R6)               ; NULL ARGUMENT FOR EXE$SETIME_INT
                          00C7   31  05D5          BRW     200$                     ; RETURN TO CALLER
                                     05D8
                                     05D8  READTIME:                                ; SOLICIT TIME
                            59   D4  05D8          CLRL    R9                       ; CLEAR A FLAG
         58      00000000'GF     32  05DA          CVTWL   G^SGN$GW_TPWAIT,R8       ; PICK UP TIMEOUT WAIT INTERVAL
                            14   14  05E1          BGTR    8$                       ; POSITIVE, WAIT THAT PERIOD ONCE
                            0D   19  05E3          BLSS    7$                       ; NEGATIVE IS WAIT FOREVER
                                     05E5  6$:
         50   00000000'GF   01   C1  05E5          ADDL3   #1,G^EXE$GL_TODR,R0      ; ZERO, SET TIME-OF-DAY CLOCK TO
                                     05ED
                                     05ED
                        1B  50   DA  05ED          MTPR    R0,#PR750$_TODR          ;   KNOWN VALUE + 10 MSEC AND FINISH UP
                                     05F0
                                     05F0
                                     05F0
                            E0   11  05F0          BRB     5$                       ;
                                     05F2
                                     05F2
                        58  14   D0  05F2  7$:     MOVL    #20,R8                   ; STARTING WAIT
                            59   D6  05F5          INCL    R9                       ; NEGATIVE - WAIT FOREVER
                                     05F7  8$:     $ASSIGN_S      TTNAME(R6),TTCHAN(R6) ; AND ASSIGN TO INPUT DEVICE
                        DD  50   E9  0605          BLBC    R0,6$                    ; ERROR - FALL BACK TO STORED TIME
             52     FF60 CF      9E  0608  10$:    MOVAB   W^TIMEPROMPT,R2          ; GET ADDRESS OF PROMPT STRING
                        53  82   9A  060D          MOVZBL  (R2)+,R3                 ; AND LENGTH
                                     0610          $QIOW_S #0,W^TTCHAN(R6),-        ; PROMPT AND READ TIME
                                     0610                  #<IO$_READPROMPT!IO$M_PURGE!IO$M_TIMED!IO$M_CVTLOW>,-
                                 0610                        TMPDESC(R6),,,-              ;  I/O STATUS BLOCK , NO AST OR PARAM
                                 0610                        LINBUF(R6),#LINBUFSIZ,-      ;  BUFFER ADDRESS AND SIZE
                                 0610                        R8,#0,-                      ;  TIME OUT
                                 0610                        R2,R3                        ;  PROMPT ADDRESS AND SIZE
                AD 50   E9       0635               BLBC     R0,6$                        ; ERROR - FALL BACK TO STORED TIME
         54     0C A6   7D       0638               MOVQ     TMPDESC(R6),R4               ; GET COMPLETION STATUS
                0D 54   E8       063C               BLBS     R4,20$                       ; CONTINUE IF SUCCESSFUL READ
                A3 59   E9       063F               BLBC     R9,6$                        ; FAILED ON ONE-TIME READ, RETURN
         58   01 A848   9E       0642               MOVAB    1(R8)[R8],R8                 ; (2 * TIMEOUT) + 1
              58    58  3C       0647               MOVZWL   R8,R8                        ; BOUND TIMEOUT
                    BC  11       064A               BRB      10$                          ; TRY AGAIN FOR TIME
                                 064C   20$:                                              ; SOMETHING WAS INPUT
       0C A6    0E A6   3C       064C               MOVZWL   TMPDESC+2(R6),TMPDESC(R6)  ; FORM DESCRIPTOR FOR BUFFER
       10 A6    1C A6   9E       0651               MOVAB    LINBUF(R6),TMPDESC+4(R6)   ; SET DESCRIPTOR ADDRESS
                                 0656               $BINTIM_S TMPDESC(R6),INTIME(R6)  ; CONVERT TO BINARY TIME
                05 50   E9       0663               BLBC     R0,89$                       ; INVALID TIME
                18 A6   D5       0666               TSTL     INTIME+4(R6)                 ; CHECK FOR DELTA TIME
                   2A   14       0669               BGTR     100$                         ; BRANCH IF NOT - OK
                                 066B   89$:                                              ; INVALID TIME VALUE INPUT
       52    FEEB CF    9E       066B               MOVAB    W^TIMERR,R2                  ; ADDRESS OF ERROR MESSAGE
          53    82      9A       0670               MOVZBL   (R2)+,R3                     ; GET STRING LENGTH
                                 0673               $QIOW_S  #0,TTCHAN(R6),-              ; GIVE ERROR MESSAGE
                                 0673                        #IO$_WRITEVBLK,-             ;
                                 0673                        ,,,-                         ;  NO I/O STATUS,AST OR AST PARAM
                                 0673                        (R2),R3 ,-                   ;  BUFFER ADDRESS, LENGTH
                                 0673                        #0,#32                       ;  SET CARRIAGE CONTROL TO CR/LF
                 FF73   31       0692               BRW      10$                          ; AND TRY AGAIN
                                 0695   100$:                                             ; EXIT
                                 0695               $DASSGN_S TTCHAN(R6)                  ; DE-ASSIGN TERMINAL CHANNEL
                14 A6   7F       069F   200$:       PUSHAQ   INTIME(R6)                   ; SET NEW SYSTEM TIME
      00000000'GF    01 FB       06A2               CALLS    #1,G^EXE$SETIME_INT          ; USE TODR CLOCK TO SET SYSTEM TIME
00000000'GF  00000000'GF 7D      06A9               MOVQ     G^EXE$GQ_TODCBASE,G^EXE$GQ_BOOTTIME ; SAVE BOOT TIME
                5E   30 C0       06B4               ADDL     #12*4,SP                     ; CLEAN OFF SCRATCH STORAGE
             077C 8F    BA       06B7               POPR     #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; RESTORE REGISTERS
                                 06BB
                                 06BB   ;
                                 06BB   ; Fall through into the deallocate logic.
                                 06BB   ;
                                 06BB   ;           RSB                                   ; *** This goes in if another piece of
                                 06BB                                                     ; *** initialization code is added that
                                 06BB                                                     ; *** is executed after EXE$INI_TIMWAIT.
                                 06BB               .DISABLE LSB
                                 06BB
```

```
                             06BB  DEAL_INIT_CODE:                                ; DEALLOCATE THE INITIALIZATION CODE
                             06BB  ;
                             06BB  ; It is the duty of the last-executed, loadable initialization
                             06BB  ; routine to make itself and all other such routines disappear, i.e.,
                             06BB  ; release the space they occupy to non-paged pool.  Each routine's vector
                             06BB  ; must be disconnected, e.g., be made to point to the symbol, EXE$LOAD_ERROR.
                             06BB  ;
                             06BB  ; NOTE:  This means that new initialization routines should be added
                             06BB  ;        to this module in a particular order, not necessarily at the
                             06BB  ;        end of the module!
                             06BB  ;
                             06BB          .ENABLE LSB
              7E    52   7D  06BB          MOVQ    R2,-(SP)                       ; Save some registers
                             06BE
                             06BE  ;
                             06BE  ; First find the vectors that point to these initialization routines
                             06BE  ; and reset them to point to EXE$LOAD_ERROR.
                             06BE  ;
         50   0000'CF    9E  06BE          MOVAB   W^SYSL$BEGIN,R0                ; Compute bounds of releasable piece:
  51  50   00000000'8F   C1  06C3          ADDL3   #<STAY_HEADER-SYSL$BEGIN>,R0,R1 ; starting and ending addresses.
      52   00000000'GF   9E  06CB          MOVAB   G^EXE$AL_LOAVEC,R2             ; Get starting address of vectors.
      53   00000000'GF   9E  06D2          MOVAB   G^EXE$LOAD_ERROR,R3            ; Get end of vectors.
        9F17 8F     62   B1  06D9  10$:    CMPW    (R2),#^X9F17                   ; Is this JMP @# ?
                    1B   13  06DE          BEQL    30$                            ; Br if yes, skip past it.
       80 8F    03 A2    91  06E0          CMPB    3(R2),#^X80                    ; Is this a system space address
                    16   12  06E5          BNEQ    40$                            ; Br if no, assume it's a HALT instr.
              50    62   D1  06E7          CMPL    (R2),R0                        ; Is address before the releasable
                    0C   1F  06EA          BLSSU   20$                            ;  piece of memory?  Br on yes.
              51    62   D1  06EC          CMPL    (R2),R1                        ; Is address after the releasable
                    07   1A  06EF          BGTRU   20$                            ;  piece of memory?  Br on yes.
      62   00000000'GF   9E  06F1          MOVAB   G^EXE$LOAD_ERROR,(R2)          ; Reset this vector.
              52    02   C0  06F8  20$:    ADDL    #2,R2                          ; Point past this vector.
                    52   D6  06FB  30$:    INCL    R2                             ; Come here to point past JMP @#.
                    52   D6  06FD  40$:    INCL    R2                             ; Come here to point past HALT.
              53    52   D1  06FF          CMPL    R2,R3                          ; Past the end of the vectors?
                    D5   1F  0702          BLSSU   10$                            ; Keep searching vectors.
                             0704  ;
                             0704  ; Now release the memory to non-paged pool.
                             0704  ;
         50   0000'CF    9E  0704          MOVAB   W^SYSL$BEGIN,R0                ; Point to start of module
         51   0000'8F    3C  0709          MOVZWL  #<STAY_HEADER-SYSL$BEGIN>,R1   ; Length to vaporize
                 F8FB'   31  070E          BRW     50$                            ; Br to code that is not released.
                             0711
                         00000000          .PSECT  $$$INIT__END,PAGE              ; 'PAGE' SINCE 16-BYTE ALIGN IS NOT
                             0000
                             0000  STAY_HEADER:
         00000000 00000000   0000          .LONG   0,0
                        0000'  0008        .WORD   <SYSL$END-STAY_HEADER>
                         62  000A          .BYTE   DYN$C_LOADCODE
                         00  000B          .BYTE   0
                             000C
          00000000'9F    16  000C  50$:    JSB     @#EXE$DEANONPGDSIZ             ; Just the smile on the Chesire cat
              52    8E   7D  0012          MOVQ    (SP)+,R2                       ; Restore
                         05  0015          RSB                                    ; Return.
                             0016
                             0016          .DISABLE LSB
                             0016          .END
```

```
                                                          C 7
INIADP750                      - ADAPTER INITIALIZATION FOR VAX 11/750  16-SEP-1984 00:46:01  VAX/VMS Macro V04-00       Page  38         IN
Symbol table                                                            11-SEP-1984 16:29:18  [SYSLOA.SRC]INIADP.MAR;3          (17)         V0

$$$VMSDEFINED                  = 00000001                  CONFREGL                       000001E4 R      08
$$T1                           = 00000001                  CPU_ADPSIZE                    00000019 R      08
ADAPTERS                         00000000 R      02        CPU_TYPE                     = 00000002
ADP$B_TYPE                     = 0000000A                  CR                           = 0000000D
ADP$C_CIADPLEN                 = 00000030                  CRB$B_TYPE                   = 0000000A
ADP$C_DRADPLEN                 = 00000030                  CRB$C_LENGTH                 = 00000048
ADP$C_MBAADPLEN                = 00000030                  CRB$L_INTD                   = 00000024
ADP$C_UBAADPLEN                = 00000258                  CRB$L_INTD2                  = 00000048
ADP$L_AVECTOR                  = 0000001C                  CRB$L_WQBL                   = 00000004
ADP$L_CRB                      = 00000010                  CRB$L_WQFL                   = 00000000
ADP$L_CSR                      = 00000000                  CRB$W_SIZE                   = 00000008
ADP$L_DPQFL                    = 00000014                  CREATE_ARRAYS                  0000011A R      09
ADP$L_LINK                     = 00000004                  CSR_LEN_OFFSET               = FFFFFFFB
ADP$L_MBASCB                   = 00000014                  DDB$T_NAME                   = 00000014
ADP$L_MBASPTE                  = 00000018                  DEAL_INIT_CODE                 000006B8 R      09
ADP$L_MRACTMDRS                = 0000005C                  DIRECT_VEC_NODE_CNT            00000009 R      08
ADP$L_MRQFL                    = 00000030                  DR$INITIAL                     ********   X   09
ADP$L_UBASCB                   = 00000044                  DR$INT                         ********   X   09
ADP$L_UBASPTE                  = 00000054                  DRTAB                          00000011 R      08
ADP$L_VECTOR                   = 00000010                  DYN$C_ADP                    = 00000001
ADP$W_ADPTYPE                  = 0000000E                  DYN$C_CONF                   = 00000007
ADP$W_DPBITMAP                 = 00000060                  DYN$C_CRB                    = 00000005
ADP$W_MRFFENCE                 = 0000015C                  DYN$C_IDB                    = 00000009
ADP$W_MRFREGARY                = 0000015E                  DYN$C_INIT                   = 00000063
ADP$W_MRNFENCE                 = 00000062                  DYN$C_LOADCODE               = 00000062
ADP$W_MRNREGARY                = 00000064                  END_NEXUS                      00000115 R      09
ADP$W_SIZE                     = 00000008                  ERROR_HALT                     00000199 R      09
ADP$W_TR                       = 0000000C                  ERROR_HALT_1                   0000019E R      09
ADP$W_UMR_DIS                  = 00000256                  EXE$AL_LOAVEC                  ********   X   09
ADPLINK                          ********   X   09         EXE$DEANONPGDSIZ               ********   X   0A
ADPTAB_ADPLEN                    00000001                  EXE$GL_CONFREG                 ********   X   09
ADPTAB_ATYPE                     00000003                  EXE$GL_CONFREGL                ********   X   09
ADPTAB_IDBUNITS                  00000000                  EXE$GL_FLAGS                   ********   X   09
ALONPAGD                         00000326 R      09        EXE$GL_NUMNEXUS                ********   X   09
AT$_CI                         = 00000004                  EXE$GL_RPB                     ********   X   09
AT$_DR                         = 00000002                  EXE$GL_SCB                     ********   X   09
AT$_MBA                        = 00000000                  EXE$GL_TENUSEC                 ********   X   09
AT$_UBA                        = 00000001                  EXE$GL_TODR                    ********   X   09
BADUMR                           0000030D R      08        EXE$GL_UBDELAY                 ********   X   09
BI_BUS_CODE                    = 80000000                  EXE$GQ_BOOTTIME                ********   X   09
BI_CPU                         = 00000000                  EXE$GQ_TODCBASE                ********   X   09
BI_CSR_LEN                     = 00000002                  EXE$INIT_TODR                  000005A0 RG     09
BI_LIKE                        = 00000000                  EXE$INI_TIMWAIT                000004EE RG     09
BLD_CRB                          00000456 R      09        EXE$LOAD_ERROR                 ********   X   09
BOO$GB_SYSTEMID                  ********   X    09        EXE$MCHK_PRTCT                 ********   X   09
BOO$GL_SPTFREH                   ********   X    09        EXE$OUTZSTRING                 ********   X   09
BOO$GL_SPTFREL                   ********   X    09        EXE$SETIME_INT                 ********   X   09
BOOTVECTOR                       00000000 R      08        EXE$TEST_CSR                   ********   X   09
BTD$K_CONSOLE                  = 00000040                  EXE$UBAERR_INT                 ********   X   09
BTD$K_UDA                      = 00000011                  EXE$V_SETTIME                  ********   X   09
BUS_CODE_OFFSET                = FFFFFFFC                  FILL_CRB                       00000463 R      09
BUS_CSR_LEN                      00000004 R      08        GET_GEN_TYPE                   000000B9 R      09
CI$INITIAL                       ********   X    09        GET_TYPE                       000000A0 R      09
CI$INT                           ********   X    09        IDB$B_TYPE                   = 0000000A
CITAB                            00000015 R      08        IDB$C_LENGTH                 = 00000038
CONFIG_IOSPACE                   0000005F R      09        IDB$L_ADP                    = 00000014
CONFREG                          000000A4 R      08        IDB$L_CSR                    = 00000000
```

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

0396 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

INIADP750
LIS

INIADP730
LIS

INIADP780
LIS

INIADPUV1
LIS

INIADP790
LIS