L 5

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 000000000        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 000000000        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 000000000        AAAAAAAAA
SSS              YYY         YYY  SSS              LLL              000         000  AAA         AAA
SSS              YYY         YYY  SSS              LLL              000         000  AAA         AAA
SSS              YYY         YYY  SSS              LLL              000         000  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              000         000  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              000         000  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              000         000  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              000         000  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              000         000  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              000         000  AAA         AAA
            SSS        YYY                    SSS  LLL              000         000  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              000         000  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              000         000  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              000         000  AAA         AAA
            SSS        YYY                    SSS  LLL              000         000  AAA         AAA
            SSS        YYY                    SSS  LLL              000         000  AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     000000000     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     000000000     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     000000000     AAA         AAA
```

```
_$2

Sym
---
SS$
SS$

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

```
**FILE**ID**INIADP790




  IIIIII    NN      NN    IIIIII      AAAAAA    DDDDDDDD    PPPPPPPP    77777777      999999      000000
  IIIIII    NN      NN    IIIIII      AAAAAA    DDDDDDDD    PPPPPPPP    77777777      999999      000000
    II      NN      NN      II      AA      AA  DD      DD  PP      PP          77  99      99  00      00
    II      NN      NN      II      AA      AA  DD      DD  PP      PP          77  99      99  00      00
    II      NNNN    NN      II      AA      AA  DD      DD  PP      PP          77  99      99  00    0000
    II      NNNN    NN      II      AA      AA  DD      DD  PP      PP          77  99      99  00    0000
    II      NN  NN  NN      II      AA      AA  DD      DD  PPPPPPPP          77      99999999  00  00  00
    II      NN  NN  NN      II      AA      AA  DD      DD  PPPPPPPP          77      99999999  00  00  00
    II      NN    NNNN      II      AAAAAAAAAA  DD      DD  PP              77              99  0000    00
    II      NN    NNNN      II      AAAAAAAAAA  DD      DD  PP              77              99  0000    00
    II      NN      NN      II      AA      AA  DD      DD  PP            77                99  00      00      ....
    II      NN      NN      II      AA      AA  DD      DD  PP            77                99  00      00      ....
  IIIIII    NN      NN    IIIIII    AA      AA  DDDDDDDD    PP          77              999999      000000        ....
  IIIIII    NN      NN    IIIIII    AA      AA  DDDDDDDD    PP          77              999999      000000        ....

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

Table of contents

```
0000            .NLIST  CND
0000
0000
0000
0000            .TITLE  INIADP790 - ADAPTER INITIALIZATION FOR VAX 11//90
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   *
C000    ;*  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 cor:ole device to
0000   ;                    INI$CONSOLE.  When mapping I/O space o  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 CRB 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           $IO790DEF                    ; Define 11/790 I/O space.
0000           $PAMMDEF                     ; Define PAMM register fields.
0000           $SBIADEF                     ; Define SBI adapter register space.
0000           $MCHKDEF                     ; Define machine check masks.
0000           $NDTDEF                      ; Define nexus device types.
0000           $PRDEF                       ; Define IPR numbers.
0000
0000
0000
0000
0000           $PR790DEF                    ; Define 11/790 specific IPR numbers.
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 1 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 RPB$W_ROUBVEC
                0000  ; is used and this array is not referenced at all.  RPB$W_ROUBVEC is set up
                0000  ; by PQDRIVER.  RPB$L_BOOTRO 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$CIADP
          000D
          000D             )APDESC -                        ; KDZ11 Processor
          000D                    ADPTYPES=NDT$_KDZ11, -
          000D                    NUMPAGES=1, -
          000D                    INITRTN=INI$KDZ11
          000D
```

```
                  000D  ;
                  000D  ; TABLES OF ADAPTER-DEPENDENT INFORMATION
                  000D  ;
                  000D  ; THE TABLE OFFSETS ARE:
                  000D  ;
                  000D          $DEFINI ADPTAB
                  0000
        00000001  0000  ADPTAB_IDBUNITS:.BLKB   1                 ; # UNITS TO SET IN IDB
        00000003  0001  ADPTAB_ADPLEN:  .BLKW   1                 ; LENGTH OF ADP
        00000004  0003  ADPTAB_ATYPE:   .BLKB   1                 ; ADP TYPE
                  0004
                  0004          $DEFEND ADPTAB
                  000D
                  000D  ;
                  000D  ; TABLES THEMSELVES:
                  000D  ;
                  000D
                  000D  MBATAB:                                   ; TABLE OF MBA CONSTANTS
              08  000D          .BYTE   8                         ; # UNITS IN MBA IDB
            0030  000E          .WORD   ADP$C_MBAADPLEN           ; # BYTES IN MBA ADP
              00  0010          .BYTE   AT$_MBA                   ; MBA ADAPTER TYPE
                  0011
                  0011  DRTAB:                                    ; TABLE OF DR32 CONSTANTS
              01  0011          .BYTE   1                         ; # UNITS IN DR IDB
            0030  0012          .WORD   ADP$C_DRADPLEN            ; # BYTES IN DR ADP
              02  0014          .BYTE   AT$_DR                    ; DR ADAPTER TYPE
                  0015
                  0015  CITAB:                                    ; TABLE OF CI CONSTANTS
              01  0015          .BYTE   1                         ; # UNITS IN CI IDB
            0030  0016          .WORD   ADP$C_CIADPLEN            ; # BYTES IN CI ADP
              04  0018          .BYTE   AT$_CI                    ; CI ADAPTER TYPE
                  0019
```

```
                                     0019   523         .SBTTL  CPU-specific data structures
                                     0019   524 ;
                                     0019   525 ; To add a new CPU type:
                                     0019   526 ;       1) Create a new nexus descriptor table, using FLOAT_NEXUS and
                                     0019   527 ;          FIXED_NEXUS macros.  Put an END_NEXUSDESC macro at the end.
                                     0019   528 ;
                                     0019   529
                                     0019   552
                                     0019   590
                                     0019   617
                                     0019   619
                                     0019   620 CPU_ADPSIZE:
                               04EC' 0019   621         .WORD   ADP$C_UBAADPLEN+UBINTSZ+<NUMUBAVEC*4>
                                     001B   622
                                     001B   623
                                     001B   624 ;
                                     001B   625 ; Declare the beginning of a nexus-descriptor table.
                                     001B   626 ;
                                     001B   627         NEXUSDESC_TABLE LABEL=NEXUSDESC
                                     0020   628
                                     0020   629
                                     0020   630 ;
                                     0020   631 ; Describe all nexuses on an 11/790 processor.
                                     0020   632 ;
                            00000001  0020   633         SBI_CPU = 1
                            00000000  0020   634         BI_CPU  = 0
                                     0020   635         FLOAT_NEXUS -
                                     0020   636                 PHYSADR=IO790$AL_IOA0, -
                                     0020   637                 NUMNEX=IO790$AL_NNEX, -
                                     0020   638                 PERNEX=IO790$AL_PERNEX
                                     00A0   639         END_NEXUSDESC
                                     00A4   640
                                     00A4   641 ;
                                     00A4   642 ; The following 11/790 data must remain in pool after INIADP is deallocated.
                                     00A4   643 ;
                                     00A4   644         .SAVE
                                 00000000   645         .PSECT  SYSLOA,LONG
                                     0000   646 ;
                                     0000   647 ; These arrays describe the adapters on the 11/790's ABUS.
                                     0000   648 ;
                                     0000   649 ABUS_VA::                               ; Virtual address of ABUS adapter space.
00000000'00000000'00000000'00000000' 0000   650         .LONG   0[4]
                                     0010   651 ABUS_TYPE::
                        00'00'00'00' 0010   652         .BYTE   0[4]                    ; Type code of ABUS adapter.
                                     0014   653 ABUS_INDEX::
                        00'00'00'00' 0014   654         .BYTE   0[4]                    ; If this ABUS adapter is an SBIA, index
                                     0018   655                                         ; into EXE$GL_CONFREGL and MMG$GL_SBICONF
                                     0018   656                                         ; where this SBI's nexus slots start.
                                 000000A4   657         .RESTORE
                                     00A4   659
                                     00A4   660
                                     00A4   682
                                     00A4   706
                                     00A4   707 ;
                                     00A4   708 ; Nexus "descriptor" arrays -- these arrays hold the nexus-device type and
                                     00A4   709 ; virtual address of every adapter on the system.  The arrays, CONFREGL and
                                     00A4   710 ; SBICONF, are allocated enough space to hold the maximum number of adapters
```

```
          00A4  ; 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  MAXNEXUS = 64
          00A4  CONFREG:                                      ; Byte array of nexus-device type codes..
000000E4  00A4                  .BLKB   MAXNEXUS
          00E4  SBICONF:
000001E4  00E4                  .BLKL   MAXNEXUS              ; Longword array of VAs of adapter space.
          01E4  CONFREGL:
000002E4  01E4                  .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   734          .SBTTL  INI$IOMAP, Initialize and map nexuses
                              033F   735 ;++
                              033F   736 ; FUNCTIONAL DESCRIPTION:
                              033F   737 ;        This routine is executed only once, during system initialization.
                              033F   738 ;        It loops through all nexuses on the system, testing for
                              033F   739 ;        adapters.  When it finds an adapter, it maps its I/O space and
                              033F   740 ;        initializes it.
                              033F   741 ;
                              033F   742 ; INPUTS:
                              033F   743 ;        BOO$GL_SPTFREL   - next free VPN
                              033F   744 ;        MMG$GL_SPTVASE   - base of system page table
                              033F   745 ;        EXE$GL_RPB       - address of reboot parameter block
                              033F   747 ;        RPB$L_ADPPHY(RPB) - PFN of boot adapter space
                              033F   749 ;
                              033F   750 ; OUTPUTS:
                              033F   751 ;        R0 - SS$_NORMAL
                              033F   752 ;
                              033F   753 ;        For each adapter found, its accessible I/O space is mapped to virtual
                              033F   754 ;        addresses.  An ADP (Adapter Control Block) is built, and the hardware
                              033F   755 ;        adapter is initialized.
                              033F   756 ;
                              033F   757 ;        The arrays CONFREG (a byte array of nexus-device type codes, defined
                              033F   758 ;        by NDT$_ symbols) and SBICONF (a longword array of
                              033F   759 ;        virtual addresses that map adapter space) are initialized.  Pointers
                              033F   760 ;        to these arrays are stored in EXE$GL_CONFREG  and
                              033F   761 ;        MMG$GL_SBICONF.  The number of entries in these two parallel arrays is
                              033F   762 ;        stored in EXE$GL_NUMNEXUS.
                              033F   763 ;
                              033F   764 ;        Since BI devices have a 16-bit device type code, a new CONFREG array is
                              033F   765 ;        constructed.  This is a longword array called CONFREGL.
                              033F   766 ;
                              033F   767 ;        Several locations in the RPB that describe the boot device are init'ed:
                              033F   768 ;        RPB$L_BOOTR1     - holds index into CONFREG and SBICONF for the boot
                              033F   769 ;                           adapter
                              033F   770 ;        RPB$L_ADPVIR     - holds VA of boot device adapter's register space
                              033F   771 ;        RPB$L_CSRVIR     - holds VA of boot device's register space
                              033F   772 ;--
                              033F   773
                          00000000   774          .PSECT  $$$INIT$CODE,QUAD
                              0000   775 INI$IOMAP::
                              0000   776
             0FFF 8F     BB   0000   777          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                              0004   778 ;
                              0004   779 ; Set up common inputs to CONFIG_IOSPACE subroutine for the CPU-specific code.
                              0004   780 ;
     52   00000000'GF    D0   0004   781          MOVL    G^BOO$GL_SPTFREL,R2      ; Get next available VPN
     53   00000000'GF    D0   000B   782          MOVL    G^MMG$GL_SPTBASE,R3      ; Get base of System Pag. Table.
             53   6342   DE   0012   783          MOVAL   (R3)[R2],R3              ; Compute SVASPT.
          52   52   09   78   0016   784          ASHL    #9,R2,R2                 ; Convert VPN to VA.
     52   80000000 8F    C8   001A   785          BISL    #VA$M_SYSTEM,R2          ; Set system bit.
                    54   D4   0021   786          CLRL    R4                       ; Clear index into CONFREG and SBICONF.
     59   00000000'GF    D0   0023   787          MOVL    G^EXE$GL_RPB,R9          ; Get address of RPB.
   5A   5C A9    F7 8F   78   002A   789          ASHL    #-9,RPB$L_ADPPHY(R9),R10; Get PFN of boot adapter space.
00000000'GF    00E4'CF   DE   0030   791          MOVAL   W^SBICONF,G^MMG$GL_SBICONF  ; Set pointers to local copies
00000000'GF    00A4'CF   DE   0039   792          MOVAL   W^CONFREG,G^EXE$GL_CONFREG  ; of these arrays for init routines.
00000000'GF    01E4'CF   DE   0042   793          MOVAL   W^CONFREGL,G^EXE$GL_CONFREGL ; ...
```

```
                           004B          .SBTTL  INITADP_790
                           004B   ;++
                           004B   ; Configure VENUS I/O Address Space.
                           004B   ;
                           004B   ; ABUS Physical Address Space:
                           004B   ;
                           004B   ;       VENUS's internal I/O bus, or ABUS, has 4 slots on it.   ABUS adapters
                           004B   ; occupy the following ranges of I/O address space:
                           004B   ;
                           004B   ;       ABUS slot           Physical address range
                           004B   ;       ---------           ----------------------
                           004B   ;           0           2000 0000  through  21FF FFFF
                           004B   ;           1           2200 0000  through  23FF FFFF
                           004B   ;           2           2400 0000  through  25FF FFFF
                           004B   ;           3           2600 0000  through  27FF FFFF
                           004B   ;
                           004B   ;
                           004B   ;       For each adapter attached to the ABUS, some adapter register space
                           004B   ; is defined (addresses 2x08 0000 through 2x08 007F).  The first 12 (decimal)
                           004B   ; longwords of the ABUS adapter register space are generically defined for all
                           004B   ; types of ABUS adapters; the remaining register address space is ABUS
                           004B   ; adapter-specific.  The first generic register is the configuration register.
                           004B   ; The low byte of the configuration register identifies the type of ABUS
                           004B   ; adapter, and its revision level.
                           004B   ;
                           004B   ;       Currently, only one adapter is supported on the ABUS: the SBIA, or
                           004B   ; SBI/ABUS adapter.  This adapter allows a standard 780 SBI to be attached to
                           004B   ; VENUS's ABUS, and allows VENUS to support all standard 780 adapters and
                           004B   ; peripherals.
                           004B   ;
                           004B   ;       Search the ABUS slots for SBIA adapters, and configure SBI I/O space
                           004B   ; and SBIA register space for any that are found.
                           004B   ;
                           004B   ;--
                           004B          ASSUME  SBIA$L_CR EQ 0           ; Assume configuration register is first
                           004B                                           ; register in SBIA register space.
  58   C0100400 8F    D0   004B          MOVL    #<<IO790$AL_IOA0+ -
                           0052                   IO790$AL_IOACR>/^X200>,R8      ; Calculate PFN
                           0052                                           ; of first ABUS configuration register.
                  5B  D4   0052          CLRL    R11                      ; Index into ABUS slots.
                           0054
                           0054   ABUS_LOOP:
     51   58   09     78   0054          ASHL    #9,R8,R1                 ; Get physical address back from PFN.
 51   C00FFFFF 8F     CA   0058          BICL    #^C<PAMM$M_PAMADD>,R1    ; Only want bits 29:20 of physical addr.
00000041 8F    5'     DA   005F          MTPR    R1,#PR790$_PAMLOC        ; Request PAMM code for this address.
 51   00000040 8F     DB   0066          MFPR    #PR790$_PAMACC,R1        ; Read PAMM location.
     51   04   00     EF   006D          EXTZV   #PAMM$V_CODE,#PAMM$S_CODE, -    ; Extract PAMM code.
               51          0071                  R1,R1
          0F   51     91   0072          CMPB    R1,#PAMM$C_NEXM          ; Is an ABUS adapter present?
               70     13   0075          BEQL    END_ABUS_LOOP            ; Nothing at this ABUS slot.
     90000000 8F      C9   0077          BISL3   #PTE$M_VALID!PTE$C_KW,- ; Temporarily associate VA in R2 with
          63   58          007D                  R8,(R3)                  ; PFN in R8 via SPTE in R3.
                           007F          $PRTCTINI       B^10$ -          ; Protect against non-existent memory
                           007F                  #<MCHK$M_NEXM!MCHK$M_LOG>       ; machine checks.
          51   62     D0   008B          MOVL    (R2),R1                  ; Read ABUS configuration register.
                           008E          $PRTCTEND       10$              ; End of protected code.
                           008F          INVALID R2                       ; Clear TB of temporary mapping.
```

```
                 52 50   E9  0092   853         BLBC    R0,END_ABUS_LOOP          ; Nothing at this ABUS slot.
                             0095   854 ;
                             0095   855 ; Found an adapter.  See if its an SBIA.
                             0095   856 ;
       51   51   04   04   EF  0095   857         EXTZV   #SBIA$V_TYPE,#SBIA$S_TYPE,R1,R1 ; Get type field in config reg.
          0010'CF4B   51   90  009A   858         MOVB    R1,W^ABUS_TYPE[R11]      ; Save in ABUS type array.
                 51   01   91  00A0   859         CMPB    #IO790$C_SBIA,R1         ; Is this an SBIA?
                      42   12  00A3   860         BNEQ    END_ABUS_LOOP            ; Nope, go to next slot.
                               00A5   861 ;
                               00A5   862 ; Found an SBIA.
                               00A5   863 ; Fill in the 790 nexus descriptor table with the physical addresses
                               00A5   864 ; corresponding to this ABUS slot.
                               00A5   865 ;
                      5B   D5  00A5   866         TSTL    R11                      ; If this is the first ABUS slot,
                      1C   13  00A7   867         BEQL    CONFIG_790               ; table is already set up properly.
                 51   10   D0  00A9   868         MOVL    #IO790$AL_NNEX,R1        ; Get number of nexues on this SBI.
           00000400 8F     C3  00AC   869         SUBL3   #IO790$AL_IOACR/^X200,-  ; Get PFN of SBI nexus space for
                 50   58       00B2   870                 R8,R0                    ; this ABUS slot.
           56   0020'CF    DE  00B4   871         MOVAL   W^NEXUSDESC,R6           ; Get address of 790 nexus table.
                               00B9   872 20$:
                 86   50   D0  00B9   873         MOVL    R0,(R6)+                 ; Put PFN in table.
                 56   04   C0  00BC   874         ADDL2   #4,R6                    ; Step past fixed/floating type code.
                 50   10   C0  00BF   875         ADDL    #IO790$AL_PERNEX/^X200,R0    ; PFN of next nexus on this SBI.
                 F4   51   F5  00C2   876         SOBGTR  R1,20$                   ; Fill in entire table.
                               00C5   877 ;
                               00C5   878 ; Fill in 11/790-specific SCB slots and map SBIA register space, then
                               00C5   879 ; call CONFIG_IOSPACE as a subroutine, to configure this SBI.
                               00C5   880 ;
                               00C5   881 CONFIG_790:
          0000'CF4B   52   D0  00C5   882         MOVL    R2,W^ABUS_VA[R11]        ; Save VA of SBIA register space.
                      58   DD  00CB   883         PUSHL   R8                       ; Save PFN of SBIA register space.
                 51   01   D0  00CD   884         MOVL    #1,R1                    ; Number of pages to map.
                    0140   30  00D0   885         BSBW    MAP_PAGES                ; Map SBIA register space.
                    0183   30  00D3   886         BSBW    INI$SCB                  ; Fill in 790-specific SCB vectors.
           56   0020'CF    DE  00D6   887         MOVAL   W^NEXUSDESC,R6           ; Address of 790 nexus descriptor table.
          0014'CF4B   54   90  00DB   888         MOVB    R4,W^ABUS_INDEX[R11]     ; Save index into CONFREG and SBICONF.
                      1C   10  00E1   889         BSBB    CONFIG_IOSPACE           ; Configure this SBI.
               0100 8F     BA  00E3   890         POPR    #^M<R8>                  ; Restore PFN of SBIA register space.
                               00E7   891
                               00E7   892
                               00E7   893 END_ABUS_LOOP:
   58     00010000 8F      C0  00E7   894         ADDL    #IO790$AL_PERABS/^X200,R8 ; R8 <- PFN of next ABUS adapter's
                               00EE   895                                          ; register space.
   FF60 5B    01   03      F1  00EE   896         ACBL    #3,#1,R11,ABUS_LOOP      ; Branch if more ABUS slots.
                               00F4   909
                    00C3   30  00F4   910         BSBW    CREATE_ARRAYS            ; Create CONFREG and SBICONF arrays.
               0FFF 8F     BA  00F7   911         POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                 50   01   D0  00FB   912         MOVL    #1,R0                    ; Set success status
                      05       00FE   913         RSB                              , Return.
                      00FF    916           .SBTTL  CONFIG_IOSPACE
                      00FF    917   ;
                      00FF    918   ; CONFIG_IOSPACE
                      00FF    919   ;       Given a nexus descriptor table, which describes what "nexuses" or
                      00FF    920   ;       "slots" are available on a system to hold I/O adapters, find and
                      00FF    921   ;       initialize all adapters on the system.
                      00FF    922   ;
                      00FF    923   ; Inputs:
                      00FF    924   ;       R2 - next available virtual address, to be used for mapping I/O space
                      00FF    925   ;       R3 - address of PTE associa ed with VA in R2
                      00FF    926   ;       R4 - Current index into CONFREG and SBICONF arrays (should be 0 the
                      00FF    927   ;            first time CONFIG_IOSPACE is called)
                      00FF    928   ;       R6 - address of nexus descriptor table
                      00FF    929   ;       R9 - address of Restart Parameter Block (RPB)
                      00FF    930   ;       R10 - PFN of boot adapter space
                      00FF    931   ;       R11- page offset from beginning of SCB; tells which page of the SCB
                      00FF    932   ;            to use for this set of nexuses (passed to routines that init ADP)
                      00FF    933   ;
                      00FF    934   ; Outputs:
                      00FF    935   ;       R2,R3,R4 - updated
                      00FF    936   ;       R9,R10,R11 - preserved; all other registers potentially modified
                      00FF    937   ;       CONFREG - initialized with adapter NDT$ code for each nexus
                      00FF    938   ;       SBICONF - initialized with adapter space VA for each nexus
                      00FF    939   ;
                      00FF    940   CONFIG_IOSPACE:
                      00FF    942   ;
                      00FF    943   ; Main loop.  Map and initialize all adapters on system.
                      00FF    944   ;
                      00FF    950
      FB A6     90    00FF    951           MOVB    CSR_LEN_OFFSET(R6),-    ; Move length of adapter type field
    0004'CF           0102    952                   W^BUS_CSR_LEN           ;  in CSR's to static location.
      FC A6     D0    0105    953           MOVL    BUS_CODE_OFFSET(R6),-   ; Move software defined bus type code
    0005'CF           0108    954                   W^SW_BUS_CODE           ;  to static longword.
                      010B    955
                      010B    956   NXT_NEXUS:                              ; For each nexus...
      58   86   D0    010B    957           MOVL    (R6)+,R8                ; Get PFN of nexus.
           01   12    010E    959           BNEQ    TEST_NEXUS              ; If PFN non-zero, go test the slot.
                05    0110    960           RSB                             ; If 0, we've found all nexuses.
                      0111    961   ;
                      0111    962   ; Read configuration register to determine if anything is present at this
                      0111    963   ; nexus.
                      0111    964   ;
                      0111    965   TEST_NEXUS:
90000000 8F     C9    0111    966           BISL3   #PTE$M_VALID!PTE$C_KW,- ; Temporarily associate VA in R2 with
      63   58         0117    967                   R8,(R3)                 ; PFN in R8 via SPTE in R3.
                      0119    968           $PRTCTINI B^10$, -              ; Protect following code from non-
                      0119    969                   #<MCHK$M_NEXM!MCHK$M_LOG>; existent memory machine checks.
      51   62   D0    0125    970           MOVL    (R2),R1                 ; Read adapter configuration register.
                      0128    971           $PRTCTEND 10$                   ; End of protected code.
                      0129    972           INVALID R2                      ; Clear TB of temporary mapping.
      11 50     E8    012C    973           BLBS    R0,GET_TYPE             ; Branch if no machine check occurred.
                      012F    974   ;
                      012F    975   ; No adapter present at this nexus.
                      012F    976   ;
  00A4'CF44     94    012F    977           CLRB    W^CONFREG[R4]           ; Store "unknown" type in CONFREG
  01E4'CF44     D4    0134    978           CLRL    W^CONFREGL[R4]          ; and in CONFREGL also.
         55     D4    0139    979           CLRL    R5                      ; Use general memory type to map
```

```
                          013B   980                                               ; one page of I/O space.
             56   04  C0  013B   981            ADDL2   #4,R6                      ; Step past type code in nexus table.
                  59  11  013E   982            BRB     MAP_NEXUS                  ; Go map I/O space for this nexus.
                          0140   984 ;
                          0140   985 ; Execution continues here if adapter was present.
                          0140   986 ;
                          0140   987 GET_TYPE:
             57   86  D0  0140   988            MOVL    (R6)+,R7                   ; Get nexus-device type from nexus table.
                  14  12  0143   990            BNEQ    GET_GEN_TYPE               ; Branch if fixed slot.
                          0145   991 ;
                          0145   992 ; Floating-type slot.  Use type from configuration register.
                          0145   993 ; Determine if type in configuration register is 8-bits or 16-bits.
                          0145   994 ;
                          0145   995
        0004'CF   01  91  0145   996            CMPB    #1,W^BUS_CSR_LEN           ; Determine length of adapter type
                          014A   997                                               ;  field in CSR contained in R7.
                  05  13  014A   998            BEQL    10$                        ; EQL implies 1 byte (8-bit) field.
             57   51  3C  014C   999            MOVZWL  R1,R7                      ; BI_LIKE, so use word instruction.
                  03  11  014F  1000            BRB     20$                        ; Skip byte instruction.
             57   51  9A  0151  1001 10$:       MOVZBL  R1,R7                      ; Use byte instruction to get type.
                          0154  1002 20$:                                          ;
        57   0005'CF  C8  0154  1003            BISL    W^SW_BUS_CODE,R7           ; Or in software bus code.
                          0159  1005 ;
                          0159  1006 ; Here R7 has hardware adapter code or'ed with software bus code.
                          0159  1007 ; Translate specific nexus device type code into general adapter type code.
                          0159  1008 ;
                          0159  1009 GET_GEN_TYPE:
      00A4'CF44   57  90  0159  1010            MOVB    R7,W^CONFREG[R4]           ; Save nexus-device type in CONFREG.
      01E4'CF44   57  D0  015F  1011            MOVL    R7,W^CONFREGL[R4]          ; CONFREGL also filled in.
                  55  D4  0165  1012            CLRL    R5                         ; Clear loop index.
                          0167  1013 30$:                                          ;
      50   0000'CF45  DE  0167  1014            MOVAL   W^ADAPTERS[R5],R0          ; Get address of adapter type code.
             0000'CF  9F  016D  1015            PUSHAB  W^NUM_PAGES                ; Push addr of end of ADAPTERS array.
             8E   50  D1  0171  1016            CMPL    R0,(SP)+                   ; See if we went beyond array.
                  3F  1E  0174  1017            BGEQU   END_NEXUS                  ; unrecognized adapter, do not map.
             60   57  D1  0176  1018            CMPL    R7,(R0)                    ; Adapter type match?
                  04  13  0179  1019            BEQL    40$                        ; If EQL yes, adapter type match.
                  55  D6  017B  1020            INCL    R5                         ; Increment loop index.
                  E8  11  017D  1021            BRB     30$                        ; Look at next adapter.
                          017F  1022 40$:
                          017F  1023
                          017F  1024 ;
                          017F  1025 ; Store boot parameters.
                          017F  1026 ;
             5A   58  D1  017F  1028            CMPL    R8,R10                     ; Does PFN match boot adapter's PFN?
                  15  12  0182  1029            BNEQ    MAP_NEXUS                  ; No; continue.
          60 A9   52  D0  0184  1031            MOVL    R2,RPB$L_ADPVIR(R9)        ; Store VA of boot adapter space.
          20 A9   54  D0  0188  1032            MOVL    R4,RPB$L_BOOTR1(R9)        ; Store boot adapter nexus number.
    51    54 A9   0D  00  EF  018C  1033        EXTZV   #0,#13, -                  ; Get offset into UNIBUS/QBUS I/O page.
                          0192  1034                    RPB$L_CSRPHY(R9),R1        ;
   58 A9    1000 C241  9E  0192  1035            MOVAB   <8*512>(R2)[R1], -         ; Set VA of UNIBUS/QBUS registers.
                          0199  1036                    RPB$L_CSRVIR(R9)           ;
                          0199  1037
                          0199  1038
                          0199  1039 ;
                          0199  1040 ; R5/ general adapter type; index into "general" adapter arrays.
                          0199  1041 ; For each adapter -
                          0199  1042 ;          Map the # of pages specified in ADAPDESC macro
                          0199  1043 ;          JSB to initialization routine specified in ADAPDESC macro
                          0199  1044 ;
                          0199  1045 MAP_NEXUS:
 00E4'CF44     52    D0   0199  1050          MOVL    R2,W^SBICONF[R4]          ; Save VA of adapter space in SBICONF.
 51   0000'CF45      3C   019F  1051          MOVZWL  W^NUM_PAGES[R5],R1        ; Get number of pages to map.
               6C    10   01A5  1052          BSBB    MAP_PAGES                 ; Map the I/O pages.
 51   0000'CF45      DE   01A7  1053          MOVAL   W^INIT_ROUTINES[R5],R1    ; Get address of initialization routine.
               61    D5   01AD  1054          TSTL    (R1)                      ; Initialization routine specified?
               04    13   01AF  1055          BEQL    END_NEXUS                 ; Branch if none.
         00 B141     16   01B1  1056          JSB     @(RT)[R1]                 ; Call initialization routine.
                          01B5  1057 END_NEXUS:
               54    D6   01B5  1058          INCL    R4                        ; Increment CONFREG and SBICONF index.
             FF51    31   01B7  1060          BRW     NXT_NEXUS                 ; Go do next nexus.
                          01BA  1064
```

```
                              01BA          .SBTTL  CREATE_ARRAYS
                              01BA  ;
                              01BA  ; CREATE_ARRAYS
                              01BA  ;
                              01BA  ;       Move the local CONFREG and SBICONF arrays into non-paged pool.
                              01BA  ;
                              01BA  ; Inputs:
                              01BA  ;       R4 - Number of nexuses on the system.
                              01BA  ;       CONFREG and SBICONF have been initialized.
                              01BA  ;
                              01BA  ; Outputs:
                              01BA  ;       R0 - R5 destroyed
                              01BA  ;       EXE$GL_CONFREG points to a copy of the CONFREG array in non-paged pool
                              01BA  ;       MMG$GL_SBICONF points to a copy of the SBICONF array in non-paged pool
                              01BA  ;       EXE$GL_NUMNEXUS contains the number of nexuses on the system
                              01BA  ;
                              01BA  ;
                              01BA  CREATE_ARRAYS:
  00000000'GF     54    D0    01BA          MOVL    R4,G^EXE$GL_NUMNEXUS    ; Store number of nexuses on system.
      51   0C A444    DE    01C1          MOVAL   12(R4)[R4],R1           ; Allocate n bytes for CONFREG plus
                              01C6                                          ; 4n bytes for SBICONF + header
           51   6144    DE    01C6          MOVAL   (R1)[R4],R1             ; Another 4n bytes for CONFREGL.
                0317    30    01CA          BSBW    ALONPAGD                ; Get pool for CONFREG and SBICONF.
                  82    7C    01CD          CLRQ    (R2)+                   ; Clear out unused
             82   51    B0    01CF          MOVW    R1,(R2)+                ; Set in size
       82   0763 8F    B0    01D2          MOVW    #<DYN$C_CONF@8>!DYN$C_INIT,(R2)+ ; Set type and subtype
  00000000'GF     62    9E    01D7          MOVAB   (R2),G^EXE$GL_CONFREG   ; Store address of system CONFREG.
          51    6244    9E    01DE          MOVAB   (R2)[R4],R1             ; Two steps to CONFREGL, 1st, SBICONF,
  00000000'GF     51    D0    01E2          MOVL    R1,G^MMG$GL_SBICONF     ; Store address of system SBICONF.
00000000'GF     6144    DE    01E9          MOVAL   (R1)[R4],G^EXE$GL_CONFREGL ; And address of system CONFREGL.
                  14    BB    01F1          PUSHR   #^M<R2,R4>              ; Save pool address and nexus count.
   62    00A4'CF  54    28    01F3          MOVC3   R4,W^CONFREG,(R2)       ; Copy CONFREG to pool.
                  14    BA    01F9          POPR    #^M<R2,R4>              ; Retrieve pool address and nexus count.
        51   54   04    C5    01FB          MULL3   #4,R4,R1                ; Number of bytes in SBICONF.
             7E   51    D0    01FF          MOVL    R1,-(SP)                ; Save, SBICONF size = CONFREGL size
 6244    00E4'CF  51    28    0202          MOVC3   R1,W^SBICONF,(R2)[R4]   ; Copy SBICONF to pool.
             51   8E    D0    0209          MOVL    (SP)+,R1                ; Restore size of SBICONF and CONFREGL.
   63    01E4'CF  51    28    020C          MOVC3   R1,W^CONFREGL,(R3)      ; Copy CONFREGL to pool.  R3 is output
                              0212                                          ; from SBICONF MOVC3, so SBICONF and
                              0212                                          ; CONFREGL must be adjacent.
                              0212
                        05    0212          RSB
```

```
                                        0213          .SBTTL  MAP_PAGES
                                        0213   ;++
                                        0213   ; INPUTS:
                                        0213   ;        R1/ Number of pages to map.
                                        0213   ;        R2/ VA of page to map.
                                        0213   ;        R3/ VA of system page table entry to be used.
                                        0213   ;        R8/ PFN of page(s) to map.
                                        0213   ;
                                        0213   ; OUTPUTS:
                                        0213   ;        R2,R3 updated; R1,R8 destroyed; all other registers preserved
                                        0213   ;
                                        0213   ;--
                                        0213
                                        0213   MAP_PAGES:
                                        0213
        83   58   90000000 8F      C9   0213          BISL3   #<PTE$M_VALID!PTE$C_KW>,R8,(R3)+
                                        021B                                                 ; Map a page.
                                58 D6   021B          INCL    R8                             ; Next PFN.
                       52   0200 C2 9E  021D          MOVAB   512(R2),R2                     ; Next VA
                       00000000'GF D6   0222          INCL    G^BOO$GL_SPTFREL               ; Next free entry.
     00000000'GF       00000000'GF D1   0228          CMPL    G^BOO$GL_SPTFREH, -            ; Check for no more system page
                                        0233                  G^BOO$GL_SPTFREL               ; table entries.
                                04 15   0233          BLEQ    ERROR_HALT                     ; Branch if out of SPTEs.
                             DB 51 F5   0235          SOBGTR  R1,MAP_PAGES                   ; Map another page.
                                   05   0238          RSB                                    ; All done.
                                        0239
                                        0239   ERROR_HALT:
                        51   02E4'CF 9E 0239          MOVAB   W^NOSPT,R1                     ; Set error message.
                                        023E   ERROR_HALT_1:
                                5B D4   023E          CLRL    R11                            ; Indicate console terminal.
                       00000000'GF 16   0240          JSB     G^EXE$OUTZSTRING               ; Output error message.
                                   00   0246          HALT                                   ; ***** FATAL ERROR *******
```

```
          0247          .SBTTL  INI$SCB
          0247  ;++
          0247  ;       Fill in 11/790-specific SCB vectors.
          0247  ;
          0247  ; INPUTS:
          0247  ;       R11             - Which SCB page is used by this SBIA
          0247  ;       ABUS_VA[R11]    - Address of this SBIA's register space
          0247  ;
          0247  ; OUTPUTS:
          0247  ;       790-specific SCB vectors are filled in; they will point into a
          0247  ;       JSB table, which transfers control to the appropriate interrupt
          0247  ;       handling routine with the address of the SBIA's register space
          0247  ;       on top of the stack (so the interrupt routine will know which SBIA
          0247  ;       interrupted).  After this routine executes, each 790-specific SCB
          0247  ;       vector is set up as shown below:
          0247  ;
          0247  ;         SCB             SYSLOA790
          0247  ;       +--------+       +---------------+
          0247  ;       |   .    |       |       .       |
          0247  ;       |--------|       |---------------|
          0247  ;       |    +---|------>| JSB int_rtn --|---+
          0247  ;       |--------|       | <SBIA reg adr>|   |
          0247  ;       |   .    |       |---------------|   |
          0247  ;       +--------+       |               |   |
          0247  ;                        |---------------|<--+
          0247  ;                        |    int_rtn    |
          0247  ;                        |---------------|
          0247  ;
          0247  ;       All registers are preserved.
          0247  ;++
          0247
          0247  ;
          0247  ; Macro SCBVEC.
          0247  ;
          0247  ;       This macro defines a table of <SCB vector, interrupt service routine>
          0247  ;       pairs.  This table is used to initialize the 11/790-specific vectors
          0247  ;       in the SCB.
          0247  ;
          0247          .MACRO  SCBVEC  VECNUM,SERVICE_RTN,SYS
          0247
          0247          .WORD   <VECNUM/4>              ; Longword offset from start of SCB page.
          0247          .LONG   <SERVICE_RTN-.>         ; Store self-relative offset to service
          0247                                          ; routines defined in SYSLOA790.
          0247          NUMVECS = NUMVECS + 1           ; Count number of vectors to load.
          0247          .ENDM   SCBVEC
          0247
          0247
          0247  ;
          0247  ; List the 11/790 SCB vectors to be directed to the JSB table.
          0247  ;
00000000  0247          NUMVECS = 0                     ; Number of SCB vectors to load.
          0247  INI$L_SCBVALS:                          ; Define SCB vectors and their
          0247                                          ;  interrupt handling routines.
          0247          SCBVEC  VECNUM=<^X58>,SERVICE_RTN=EXE$INT58
          024D          SCBVEC  VECNUM=<^X5C>,SERVICE_RTN=EXE$INT5C
          0253          SCBVEC  VECNUM=<^X60>,SERVICE_RTN=EXE$INT60
          0259
```

```
                         00000018             .PSECT  SYSLOA,LONG               ; This data stays after INIADP is gone.
                             0018
                             0018             .ALIGN  LONG
                             0018  INI$L_IOAVECS:
                             0018             .REPT   4*NUMVECS                 ; 4 pages of SCB * number of vectors/page.
                             0018             .ALIGN  LONG                      ; SCB routines must be longword aligned.
                             0018             JSB     @#EXE$LOAD_ERROR          ; Init to error halt.
                             0018             .LONG   0                         ; Reserve space for addres, of IOA regs.
                             0018             .ENDR
                             00A6
                         00000259             .PSECT  $$$INIT$CODE
                             0259
                             0259  INI$SCB:
              1F      BB     0259             PUSHR   #^M<R0,R1,R2,R3,R4>       ; Save some registers.
         50   E9 AF   DE     025B             MOVAL   INI$L_SCBVALS,R0          ; Get address of SCB value table.
  51   00000018'EF    DE     025F             MOVAL   INI$L_IOAVECS,R1          ; Get address of JSB table.
     52   5B   24     C5     0266             MULL3   #<NUMVECS*12>,R11,R2      ; Get byte offset into JSB table for
                             026A                                               ; this SCB page.
           51   52    C0     026A             ADDL    R2,R1                     ; R1 points to 1st JSB table entry
                             026D                                               ; for this SCB page.
  52   00000000'GF    D0     026D             MOVL    G^EXE$GL_SCB,R2           ; Address of SCB.
     53   5B   09     78     0274             ASHL    #9,R11,R3                 ; Turn SCB page offset into byte offset.
           52   53    C0     0278             ADDL    R3,R2                     ; R2 points to beginning of correct
                             027B                                               ; SCB page.
           53   03    D0     027B             MOVL    #NUMVECS,R3               ; Get number of vectors to load.
                             027E
                             027E  ;
                             027E  ; R0 walks through a table of the form:
                             027E  ;          .WORD   <longword offset into SCB>
                             027E  ;          .LONG   <self-relative offset to SCB interrupt handling routine>
                             027E  ; where each JSB instruction is longword aligned.
                             027E  ;
                             027E  ; R1 walks through a table of the form:
                             027E  ;          JSB     <@#interrupt handling routine>
                             027E  ;          .LONG   <address of SBIA register space>
                             027E  ;
                             027E  ; R2 points to the beginning of the SCB page for this SBIA.
                             027E  ;
                             027E  ; This loop performs the following functions:
                             027E  ;          (1) Fills in the SCB vector to point to one of the JSB instructions
                             027E  ;              in the IOAVEC table.
                             027E  ;          (2) Fills in the destination of the JSB instruction to point to the
                             027E  ;              correct interrupt handling routine in SYSLOA790.EXE.
                             027E  ;          (3) Fills in the longword after the JSB instruction with the address
                             027E  ;              of register space for this SBIA adapter.
                             027E  ;
                             027E
                             027E  FILL_IN_SCB:
           54   80    B0     027E             MOVW    (R0)+,R4                  ; R4 <- longword offset into SCB page.
     6244   01 A1     9E     0281             MOVAB   1(R1),(R2)[R4]            ; Point SCB vector to JSB instruction
                             0286                                               ; (+1 to execute on interrupt stack).
   02 A1   00 B040    9E     0286             MOVAB   @(R0)[R0],2(R1)           ; Fill in destination of JSB instruction.
  06 A1   0000'CF4B   D0     028C             MOVL    W^ABUS_VA[R11],6(R1)      ; Put address of IOA regs after the JSB.
        50   04 A0    DE     0293             MOVAL   4(R0),R0                  ; Step to next entry in SCB table.
        51   0C A1    DE     0297             MOVAL   12(R1),R1                 ; Step to next entry in JSB table.
             E0 53    F5     029B             SOBGTR  R3,FILL_IN_SCB            ; Repeat for all vectors.
                             029E  ;
```

```
                                029E  ; Take care of a special case: the vector at offset ^x54 into venus' SCB is
                                029E  ; the SBI FAIL vector.  This condition is handled as a powerfail.  This is a
                                029E  ; special case because we don't want the address of the SBIA registers on the
                                029E  ; stack when we vector to the powerfail code.
                                029E  ;
      00000001'GF       9E      029E          MOVAB   G^EXE$POWERFAIL+1,-     ; Load address of powerfail routine.
             64 A2              02A4                  ^X64(R2)
                1F      BA      02A6          POPR    #^M<R0,R1,R2,R3,R4>
                        05      02A8          RSB
                                02A9
```

```
                                      02A9  1269          .SBTTL  INI$UBSPACE
                                      02A9  1270 ;++
                                      02A9  1271 ;        Map UNIBUS space; initialize UNIBUS ADP.
                                      02A9  1272 ;
                                      02A9  1273 ; INPUTS:
                                      02A9  1274 ;        R2 - VA of next free system page
                                      02A9  1275 ;        R3 - VA of system page table entry to be used to map VA in R2
                                      02A9  1276 ;        R4 - nexus identification number of this adapter
                                      02A9  1277 ;   -8(R6) - PFN of this UNIBUS adapter's register space
                                      02A9  1278 ;
                                      02A9  1279 ; OUTPUTS:
                                      02A9  1280 ;        UNIBUS space is mapped.
                                      02A9  1281 ;        INI$UBADP is called to build an ADP block and initialize UNIBUS
                                      02A9  1282 ;        adapter hardware.
                                      02A9  1283 ;
                                      02A9  1284 ;--
                                      02A9  1285
                                      02A9  1286 INI$UBSPACE:
                                      02A9  1287
           58   01E4'CF44       DE    02A9  1290          MOVAL   W^CONFREGL[R4],R8                ; R8 => CONFREGL slot.
       58   68     02    00     EF    02AF  1291          EXTZV   #0,#2,(R8),R8                    ; Get UBA number.
           58   58    09        78    02B4  1292          ASHL    #9,R8,R8                         ; Position UB number.
                                      02B8  1295
                                      02B8  1304
                                      02B8  1309
                                      02B8  1314
                                      02B8  1319
   51   F8 A6   000000FF 8F     CB    02B8  1321          BICL3   #^XFF,-8(R6),R1               ; Get PFN of start of SBI addr space.
           58   09F0 C148       9E    02C1  1322          MOVAB   W^<<IO790$AL_UBOSP+^0760000>/^X200>(R1)[R8],R8
                                      02C7  1323                                                ; Calculate PFN of Unibus I/O page.
                                      02C7  1325
                                      02C7  1330
                 51    10       D0    02C7  1331          MOVL    #16,R1                        ; Number of pages to map (UB/Qbus space).
                     FF46       30    02CA  1332          BSBW    MAP_PAGES                     ; Map I/O pages.
                                      02CD  1333 ;
                                      02CD  1334 ; Call adapter initialization routine.
                                      02CD  1335 ;
                                      02CD  1336 ;        BSBW    INI$UBADP                      ; Init ADP block.
                                      02CD  1337 ;        RSB
```

```
                          02CD        .SBTTL  INI$UBADP - BUILD ADP AND INITIALIZE UBA
                          02CD  ;+
                          02CD  ; INI$UBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
                          02CD  ; DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS.  A CALL IS
                          02CD  ; THEN MADE TO UBA$INITIAL TO INITIALIZE THE ADAPTER HARDWARE.
                          02CD  ;
                          02CD  ; INPUT:
                          02CD  ;       R4 - nexus identification number of this adapter
                          02CD  ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                          02CD  ;-
                          02CD
                          02CD  INI$UBADP:
                          02CD
         01FF 8F     BB   02CD        PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; SAVE R0-R8
                          02D1  ;
                          02D1  ; Allocate and initialize Adapter Control Block (ADP).
                          02D1  ;
     51    0019'CF   3C   02D1        MOVZWL  W^CPU_ADPSIZE,R1            ; PICK UP LENGTH OF ADP
              020B   30   02D6        BSBW    ALONPAGD                    ; ALLOCATE SPACE FOR ADP
      08 A2     51   B0   02D9        MOVW    R1,ADP$W_SIZE(R2)           ; SET SIZE INTO ADP BLOCK
      0A A2     01   90   02DD        MOVB    #DYN$C_ADP, -               ; AND SET TYPE OF BLOCK
                          02E1                ADP$B_TYPE(R2)
      0E A2     01   B0   02E1        MOVW    #AT$_UBA, -                 ; SET TYPE OF ADAPTER
                          02E5                ADP$W_ADPTYPE(R2)
   62    00E4'CF44   D0   02E5        MOVL    W^SBICONF[R4], -            ; SET VA OF CONFIGURATION REG
                          02EB                ADP$L_CSR(R2)
      0C A2     54   B0   02EB        MOVW    R4,ADP$W_TR(R2)             ; SET TR NUMBER FOR ADAPTER
                          02EF
      50     14 A2   DE   02EF        MOVAL   ADP$L_DPQFL(R2),R0          ; ADDRESS OF DATA PATH WAIT QUEUE
         60     50   D0   02F3        MOVL    R0,(R0)                     ; INIT QUEUE HEADER
      04 A0     50   D0   02F6        MOVL    R0,4(R0)                    ;
                          02FA
      50     30 A2   DE   02FA        MOVAL   ADP$L_MRQFL(R2),R0          ; ADDRESS OF MAP WAIT QUEUE
         60     50   D0   02FE        MOVL    R0,(R0)                     ; INIT QUEUE HEADER
      04 A0     50   D0   0301        MOVL    R0,4(R0)                    ;
             04 A2   D4   0305        CLRL    ADP$L_LINK(R2)              ; ZAP ADAPTER CHAIN LINK
             FCF5'   30   0308        BSBW    ADPLINK                     ; LINK ADP TO END OF LIST
                          030B  ;
                          030B  ; Initialize adapter interrupt vectors in System Control Block.
                          030B  ;
   58    00000000'GF     D0   030B    MOVL    G^EXE$GL_SCB,R8             ; GET SCB ADDRESS
                          0312
                          0312
     53    5B    09  78   0312        ASHL    #9,R11,R3                   ; Turn SCB page offset into byte offset.
           58    53  C0   0316        ADDL    R3,R8                       ; Set to beginning of correct SCB page.
                          0319                                            ; Fall into 11/780 code.
                          0319
                          0319
                          0319  ;
                          0319  ; Following ASSUME breaks if the ADP length is not a multiple of 4, thereby
                          0319  ;       causing the vectors to NOT be long word aligned.
                          0319
                          0319        ASSUME  ADP$C_UBAADPLEN/4*4     EQ      ADP$C_UBAADPLEN
                          0319
     53    02EC'C2   9E   0319        MOVAB   ADP$C_UBAADPLEN+UBINTSZ(R2),R3  ; LOCATE VECTORS
       10 A2     53  D0   031E        MOVL    R3,ADP$L_VECTOR(R2)         ; AND RECORD IN ADP
   60 A2    FFFE 8F  B0   0322        MOVW    #^XFFFE,ADP$W_DPBITMAP(R2) ; MARK DATAPATHS 1-15 AVAILABLE
```

```
              53   FF6C'C3   9E  0328  1399          MOVAB   -UBINTSZ(R3),R3           ; BASE OF INTERRUPT CODE
                        3F   BB  032D  1400          PUSHR   #^M<R0,R1,R2,R3,R4,R5>    ; SAVE MOVC REGISTERS
     63   0450'CF   0094'8F  28  032F  1401          MOVC3   #UBINTSZ,W^UBAINTBASE,(R3)       ; COPY INTERRUPT CODE
                        3F   BA  0337  1402          POPR    #^M<R0,R1,R2,R3,R4,R5>    ; RESTORE MOVC REGISTERS
          54   54    04   00 EF  0339  1403          EXTZV   #0,#4,R4,R4               ; Use low 4 bits of nexus number.
             50   0100 C844  DE  033E  1404          MOVAL   ^X100(R8)[R4],R0          ; COMPUTE 1ST VECTOR ADDRESS
                1C A2   50   D0  0344  1405          MOVL    R0,ADP$L_AVECTOR(R2)      ; SAVE ADDR OF ADAPTER SCB VECTORS
                 60   01'A3  9E  0348  1406          MOVAB   B^UBAINT4+1(R3),(R0)      ; STORE VECTOR FOR BR4
              40 A0   21'A3  9E  034C  1407          MOVAB   B^UBAINT5+1(R3),64(R0)    ; STORE VECTOR FOR BR5
            0080 C0   41'A3  9E  0351  1408          MOVAB   B^UBAINT6+1(R3),128(R0)   ; STORE VECTOR FOR BR6
            00C0 C0   61'A3  9E  0357  1409          MOVAB   B^UBAINT7+1(R3),192(R0)   ; STORE VECTOR FOR BR7
                   50   62   D0  035D  1410          MOVL    ADP$L_CSR(R2),R0          ; GET UBACSR ADDRESS
               0A'A3    50   C0  0360  1411          ADDL    R0,B^UBAINT4REL(R3)       ; ADD CSR VA
               2A'A3    50   C0  0364  1412          ADDL    R0,B^UBAINT5REL(R3)       ; TO EACH OF THE
               4A'A3    50   C0  0368  1413          ADDL    R0,B^UBAINT6REL(R3)       ; BICL INSTRUCTIONS
               6A'A3    50   C0  036C  1414          ADDL    R0,B^UBAINT7REL(R3)       ; IN THE INTERRUPT DISPATCHERS
             0089'C3    52   D0  0370  1415          MOVL    R2,UBAINTADP(R3)          ;SET ADDRESS OF ADAPTOR CONTROL BLOCK
                   0000'CF   9E  0375  1416          MOVAB   W^EXE$UBAERR_INT,-
                   0090'C3       0379  1417                   UBAERRADR(R3)            ; SET ADDRESS OF ERROR HANDLER
                     01'A3   9E  037C  1418          MOVAB   B^UBAINT4+1(R3),-
                      44 A2      037F  1419                   ADP$L_UBASCB(R2)         ; SAVE 4 SCB VECTOR CONTENTS
                     21'A3   9E  0381  1420          MOVAB   B^UBAINT5+1(R3),-
                      48 A2      0384  1421                   ADP$L_UBASCB+4(R2)       ; DITTO
                     41'A3   9E  0386  1422          MOVAB   B^UBAINT6+1(R3),-
                      4C A2      0389  1423                   ADP$L_UBASCB+8(R2)       ; DITTO
                     61'A3   9E  038B  1424          MOVAB   B^UBAINT7+1(R3),-
                      50 A2      038E  1425                   ADP$L_UBASCB+12(R2)      ; DITTO
                   54   52   D0  0390  1426          MOVL    R2,R4                     ; COPY ADP ADDRESS
                   52   62   D0  0393  1427          MOVL    ADP$L_CSR(R2),R2          ; VIRTUAL ADDRESS OF ADAPTER
        04 A2   7C000000 8F  D0  0396  1428          MOVL    #^X7C000000,UBA$L_CR(R2); DISABLE ALL UMR'S
                00000000'GF  16  039E  1429          JSB     G^MMG$SVAPTECHK           ; ADDRESS OF SPTE THAT MAPS ADAPTER
                 54 A4   63  D0  03A4  1430          MOVL    (R3),ADP$L_UBASPTE(R4)    ; SAVE CONTENTS OF SPTE MAPPING ADAPTER
            58 A4     20 A3  D0  03A8  1431          MOVL    <8*4>(R3),-               ; CONTENTS OF SPTE MAPPING I/O SPACE
                                 03AD  1432                  ADP$L_UBASPTE+4(R4)
                   52   54   D0  03AD  1433          MOVL    R4,R2                     ; COPY ADP ADDRESS BACK TO R2
          53    00000000'GF  DE  03B0  1434          MOVAL   G^UBA$UNEXINT,R3          ; GET ADDR OF UNEXP INT SERVICE(IN EXEC)
             54    0000'CF   DE  03B7  1435          MOVAL   W^UBA$INT0,R4             ; GET ADDR OF SPECIAL VECTOR 0 ROUTINE
                                 03BC  1436
                                 03BC  1437 ;
                                 03BC  1438 ; INIT UB VECTORS TO UNEXPECTED INTERRUPT SERVICE
                                 03BC  1439 ;
               50     10 A2  D0  03BC  1440          MOVL    ADP$L_VECTOR(R2),R0       ; GET ADDRESS OF VECTORS
                   80   54   D0  03C0  1441          MOVL    R4,(R0)+                  ; SPECIAL CASE FOR VECTOR 0
               51    7F 8F   9A  03C3  1442          MOVZBL  #<NUMUBAVEC-1>,R1         ; REST OF VECTORS
                   80   53   D0  03C7  1443 10$:     MOVL    R3,(R0)+                  ; FILL VECTOR WITH UNEXP INT
                     FA 51   F5  03CA  1444          SOBGTR  R1,10$                    ; FILL ALL VECTORS
                                 03CD  1445
                                 03CD  1447
                                 03CD  1507
                                 03CD  1508
                                 03CD  1536
                                 03CD  1537
                                 03CD  1558
                                 03CD  1559
                                 03CD  1601
                                 03CD  1602
                                 03CD  1651 ;
```

```
                                  03CD  ; Now check for any UNIBUS memory that may be on the adapter. First we must
                                  03CD  ; disable all the UNIBUS Map Registers so that there is no conflict in
                                  03CD  ; which memory will respond.  Then we check all 248Kb of potential memory in
                                  03CD  ; 8Kb chunks, since each disable bit on the 780 UBA represents 16 UMR's or
                                  03CD  ; 8Kb of memory.  The number of registers is stored in the ADP and the
                                  03CD  ; corresponding number withdrawn from the UMR map in the ADP.
                                  03CD  ;
                                  03CD
                    56     62  D0 03CD         MOVL     ADP$L_CSR(R2),R6        ; Pick up adapter pointer
                           51  D4 03D0         CLRL     R1                      ; Zero out number of UMR to disable
  57    08 AE   00000200 8F    C3 03D2         SUBL3    #512,8(SP),R7           ; R7 = VA of last page of UNIBUS
            58  0C AE      04  C3 03D8         SUBL3    #4,12(SP),R8            ; R8 = VA of SPTE mapping (R7)
  54    20 AE   00000200 8F    C3 03E0         SUBL3    #512,32(SP),R4          ; R4 = PFN of first page of UNIBUS
                           68  DD 03E9         PUSHL    (R8)                    ; Save contents of SPTE
                    53     54  D0 03EB         MOVL     R4,R3                   ; Copy starting PFN
                    55     1F  D0 03EE         MOVL     #31,R5                  ; 31 8Kb chunks to test
                                  03F1  50$:   INVALID  R7                      ; Invalidate TB
                90000000 8F    C9 03F4         BISL3    #<PTE$M_VALID!PTE$C_KW>,-
                    68     54     03FA                   R4,(R8)                ; Map each page of UNIBUS
                    50     57  D0 03FC         MOVL     R7,R0                   ; Address to check
                       FBFE'   30 03FF         BSBW     EXE$TEST_CSR            ; Validate it
                     0D 50     E9 0402         BLBC     R0,70$                  ; Not there
                    54     53  D1 0405         CMPL     R3,R4                   ; First time in?
                           04  13 0408         BEQL     60$                     ; Yes, skip next test
                           51  D5 040A         TSTL     R1                      ; Any registers already?
                           3A  13 040C         BEQL     80$                     ; No, memory not start at 0
                51      10 A1  9E 040E  60$:   MOVAB    16(R1),R1               ; Yes, up the count
                54      10 A4  9E 0412  70$:   MOVAB    16(R4),R4               ; Map Next 8Kb (16*512)
                     D8 55     F5 0416         SOBGTR   R5,50$                  ; Loop until done
                        68   8ED0 0419         POPL     (R8)                    ; Restore old contents of SPTE
                                  041C         INVALID  R7                      ; Invalidate TB
            0256 C2        51  B0 041F         MOVW     R1,ADP$W_UMR_DIS(R2)    ; Record number disabled
                                  0424  ;
                                  0424  ; Initialize fields for new UBA map register allocation.  Make it appear
                                  0424  ;       that we have one contiguous array of 496 available map registers.
                                  0424  ;       To do this we set ADP$L_MRACTMDRS to one (the number of active
                                  0424  ;       map register descriptors for distinct contiguous areas),
                                  0424  ;       ADP$W_MRNREGARY(0) to 496 (i.e the number of registers in this
                                  0424  ;       contiguous range) and ADP$FREGARY(0) to 0 (i.e. the first register
                                  0424  ;       in the range is register 0).
                                  0424  ;
            5C A2          01  D0 0424         MOVL      1,ADP$L_MRACTMDRS(R2)  ; 1 active map descriptor
     64 A2  01F0 8F        51  A3 0428         SUBW3    R1,#496,ADP$W_MRNREGARY(R2);  for a range of 496 registers
            015E C2        51  B0 042F         MOVW     R1,ADP$W_MRFREGARY(R2)  ;  starting at register zero.
              62 A2        01  AE 0434         MNEGW    #1,ADP$W_MRNFENCE(R2)   ; Also init "fences" which preceed
            015C C2        01  AE 0438         MNEGW    #1,ADP$W_MRFFENCE(R2)   ;  the two descriptor arrays.
                                  043D  ;
                                  043D  ; Initialize adapter hardware.
                                  043D  ;
                    54     62  D0 043D         MOVL     ADP$L_CSR(R2),R4        ; Get CSR address to init
                       FBBD'   30 0440         BSBW     UBA$INITIAL             ; And initialize adapter
                   01FF 8F     BA 0443         POPR     #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; Restore registers
                               05 0447         RSB                              ; Return
                                  0448
                                  0448  ;
                                  0448  ; Error if UNIBUS memory not start at location 0
                                  0448  ;
```

```
      51   030D'CF   9E  0448  1725 80$:    MOVAB   W^BADUMR,R1             ; Set error message
              FDEE   31  044D  1726         BRW     ERROR_HALT_1            ; Put it out
                         0450  1728
```

```
                                       0450 ;
                                       0450 ; UBA INTERRUPT SERVICE ROUTINES.  ONE COPY OF THESE ROUTINES IS
                                       0450 ; MOVED INTO NONPAGED POOL AND RELOCATED FOR EACH UBA.
                                       0450 ;
                                       0450 ; **** NOTE ****  THE CODING SEQUENCE FOR DISPATCHING ON UBA INTTERUPTS
                                       0450 ; IS ASSUMED IN THE MODULE MCHECK780.MAR.  THE ASSUMPTIONS ARE MADE SO
                                       0450 ; THE MACHINE CHECK HANDLER CAN IDENTIFY A CPU TIMEOUT WHEN THE
                                       0450 ; BICL3 INSTRUCTION IS READING THE  UBA'S BRRVR REGISTER.
                                       0450 ; THE ASSUMPTIONS MADE ARE THAT THE VALUE OF THE VIRTUAL ADDRESS OF THE BRRVR
                                       0450 ; REGISTER IS AT AN OFFSET OF 10. BYTES PAST THE INTERRUPT VECTOR ENTRY POINT,
                                       0450 ; THAT THE PC OF THE INSTRUCTION ACCESSING BRRVR IS 3 BYTES PAST THE INTERRUPT
                                       0450 ; VECTOR ENTRY, AND THAT R4 AND R5 ARE SAVED ON THE STACK AT THAT POINT.
                                       0450 ;
                                       0450
                                       0450         .ENABL  LSB
                                       0450         .ALIGN  QUAD
                                       0450 UBAINTBASE:                                  ; BASE OF UBA INTERRUPT DISPATCHERS
                              00000000 0450 UBAINT4=.-UBAINTBASE                         ; UBA 0 INTERRUPT DISPATCH LEVEL 4
                        7E     54   7D 0450         MOVQ    R4,-(SP)                     ; SAVE REGISTERS
54    00000030 9F   7FFFFE03 8F     CB 0453         BICL3   #^X7FFFFE03,@#UBA$L_BRRVR,R4 ; READ VECTOR  REGISTER AND CLEAR BITS
                              0000000A 045F UBAINT4REL=.-UBAINTBASE-5                    ; OFFSET TO ADD UBACSR VALUE
                        7E     52   7D 045F         MOVQ    R2,-(SP)                     ; SAVE REGISTERS
                   55   E4'AF44     9E 0462         MOVAB   B^VECTAB[R4],R5              ; GET ADDRESS OF INTERRUPT VECTOR
                               65   18 0467         BGEQ    10$                          ; IF GEQ UBA INTERRUPTS
                        7E     50   7D 0469         MOVQ    R0,-(SP)                     ; SAVE REGISTERS
                               95   17 046C         JMP     @(R5)+                       ; DISPATCH INTERRUPT
                                       046E         .ALIGN  QUAD
                              00000020 0470 UBAINT5=.-UBAINTBASE                         ; UBA 0 INTERRUPT DISPATCH LEVEL 5
                        7E     54   7D 0470         MOVQ    R4,-(SP)                     ; SAVE REGISTERS
54    00000034 9F   7FFFFE03 8F     CB 0473         BICL3   #^X7FFFFE03,@#UBA$L_BRRVR+4,R4 ; READ VECTOR REGISTER AND CLEAR BITS
                              0000002A 047F UBAINT5REL=.-UBAINTBASE-5                    ; OFFSET TO ADD UBACSR VALUE
                        7E     52   7D 047F         MOVQ    R2,-(SP)                     ; SAVE REGISTERS
                   55   E4'AF44     9E 0482         MOVAB   B^VECTAB[R4],R5              ; GET ADDRESS OF INTERRUPT VECTOR
                               45   18 0487         BGEQ    10$                          ; IF GEQ UBA INTERRUPTS
                        7E     50   7D 0489         MOVQ    R0,-(SP)                     ; SAVE REGISTERS
                               95   17 048C         JMP     @(R5)+                       ; DISPATCH INTERRUPT
                                       048E         .ALIGN  QUAD
                              00000040 0490 UBAINT6=.-UBAINTBASE                         ; UBA 0 INTERRUPT DISPATCH LEVEL 6
                        7E     54   7D 0490         MOVQ    R4,-(SP)                     ; SAVE REGISTERS
54    00000038 9F   7FFFFE03 8F     CB 0493         BICL3   #^X7FFFFE03,@#UBA$L_BRRVR+8,R4 ; READ VECTOR REGISTER AND CLEAR BITS
                              0000004A 049F UBAINT6REL=.-UBAINTBASE-5                    ; OFFSET TO ADD UBACSR VALUE
                        7E     52   7D 049F         MOVQ    R2,-(SP)                     ; SAVE REGISTERS
                   55   E4'AF44     9E 04A2         MOVAB   B^VECTAB[R4],R5              ; GET ADDRESS OF INTERRUPT VECTOR
                               25   18 04A7         BGEQ    10$                          ; IF GEQ UBA INTERRUPTS
                        7E     50   7D 04A9         MOVQ    R0,-(SP)                     ; SAVE REGISTERS
                               95   17 04AC         JMP     @(R5)+                       ; DISPATCH INTERRUPT
                                       04AE         .ALIGN  QUAD
                              00000060 04B0 UBAINT7=.-UBAINTBASE                         ; UBA 0 INTERRUPT DISPATCH LEVEL 7
                        7E     54   7D 04B0         MOVQ    R4,-(SP)                     ; SAVE REGISTERS
54    0000003C 9F   7FFFFE03 8F     CB 04B3         BICL3   #^X7FFFFE03,@#UBA$L_BRRVR+12,R4 ; READ VECTOR AND CLEAR BITS
                              0000006A 04BF UBAINT7REL=.-UBAINTBASE-5                    ; OFFSET TO ADD UBACSR VALUE
                        7E     52   7D 04BF         MOVQ    R2,-(SP)                     ; SAVE REGISTERS
                   55   E4'AF44     9E 04C2         MOVAB   B^VECTAB[R4],R5              ; GET ADDRESS OF INTERRUPT VECTOR
                               05   18 04C7         BGEQ    10$                          ; IF GEQ UBA INTERRUPTS
                        7E     50   7D 04C9         MOVQ    R0,-(SP)                     ; SAVE REGISTERS
                               95   17 04CC         JMP     @(R5)+                       ; DISPATCH INTERRUPT
                     00 54     1F   E5 04CE 10$:    BBCC    #31,R4,20$                   ;CLEAR ADAPTER ERROR INTERRUPT FLAG (MSB)
```

```
          55   E4'AF44   9E  04D2  1788 20$:     MOVAB   B^VECTAB[R4],R5           ;GET ADDRESS OF INTERRUPT VECTOR
      54   00000000'8F   D0  04D7  1789          MOVL    I^#0,R4                   ;GET ADDRESS OF ADAPTOR CONTROL BLOCK
                 00000089    04DE  1790 UBAINTADP=.-UBAINTBASE-5                   ;OFFSET TO START OF LOADED CODE
           00000000 9F   17  04DE  1791          JMP     @#0                       ;ERROR ROUTINE IN ADPERR780
                 00000090    04E4  1792 UBAERRADR=.-UBAINTBASE-4
                             04E4  1793          .DSABL  LSB
                             04E4  1794
                             04E4  1795          .ALIGN  LONG                      ; LONGWORD ALIGN VECTORS
                             04E4  1796 VECTAB:                                    ; END OF INTERRUPT CODE, START OF VECTORS
                 00000094    04E4  1797 UBINTSZ=.-UBAINTBASE                       ; SIZE OF UBA INTERRUPT CODE
                             04E4  1798
```

```
                        04E4            .SBTTL  INI$MBADP - BUILD ADP AND INITIALIZE MBA
                        04E4            .SBTTL  INI$DRADP - BUILD ADP AND INITIALIZE DR32
                        04E4            .SBTTL  INI$CIADP - BUILD ADP AND INITIALIZE CI
                        04E4    ;+
                        04E4    ; INI$MBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                        04E4    ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                        04E4    ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                        04E4    ; BY CALLING MBA$INITIAL.
                        04E4    ;
                        04E4    ; INI$DRADP IS CALLED AFTER MAPPING THE REGISTERS FOR THE DR32
                        04E4    ; ADAPTER.  THE ADAPTER CONTROL BLOCK, CRB, AND IDB ARE ALLOCATED
                        04E4    ; AND INITIALIZED.  THE ADAPTER HARDWARE IS THEN INITIALIZED BY
                        04E4    ; CALLING DR$INITIAL.
                        04E4    ;
                        04E4    ; INI$MBADP AND INI$DRADP SHARE COMMON CODE AFTER THE TABLE OF ADAPTER
                        04E4    ; SPECIFIC CONSTANTS IS SELECTED AND STORED IN R8.
                        04E4    ;
                        04E4    ; INPUT:
                        04E4    ;       R4 - nexus identification number of this adapter
                        04E4    ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                        04E4    ;
                        04E4    ; OUTPUTS:
                        04E4    ;       ALL REGISTERS PRESERVED
                        04E4    ;-
                        04E4
    00000000'GF     17  04E4    ALONPAGD:JMP    G^INI$ALONONPAGED
                        04EA
                        04EA            .ENABL  LSB
                        04EA
                        04EA    INI$DRADP:                              ; INITIALIZE DR32 DATA STRUCTURES
                        04EA
        07FF 8F     BB  04EA            PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
  58    0011'CF     DE  04EE            MOVAL   W^DRTAB,R8              ; GET DR32 TABLE OF CONSTANTS
  59    0000'CF     9E  04F3            MOVAB   W^DR$INT,R9             ; ADDRESS OF INITERRUPT SERVICE ROUTINE
  5A    0000'CF     9E  04F8            MOVAB   W^DR$INITIAL,R10        ; ADDRESS OF DEVICE INITIALIZATION
             28     11  04FD            BRB     10$                     ; JOIN COMMON CODE
                        04FF
                        04FF    INI$CIADP:                              ; INITIALIZE CI DATA STRUCTURES
                        04FF
        07FF 8F     BB  04FF            PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
  58    0015'CF     DE  0503            MOVAL   W^CITAB,R8              ; GET CI TABLE OF CONSTANTS
  59    0000'CF     9E  0508            MOVAB   W^CI$INT,R9             ; ADDRESS OF INITERRUPT SERVICE ROUTINE
  5A    0000'CF     9E  050D            MOVAB   W^CI$INITIAL,R10        ; ADDRESS OF DEVICE INITIALIZATION
             13     11  0512            BRB     10$                     ; JOIN COMMON CODE
                        0514
                        0514    INI$MBADP:                              ; INIT MBA DATA STRUCTURES
                        0514
        07FF 8F     BB  0514            PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ;
  58    000D'CF     DE  0518            MOVAL   W^MBATAB,R8             ; GET MBA TABLE OF CONSTANTS
  59    0000'CF     9E  051D            MOVAB   W^MBA$INT,R9            ; ADDRESS OF INITERRUPT SERVICE ROUTINE
  5A    0000'CF     9E  0522            MOVAB   W^MBA$INITIAL,R10       ; ADDRESS OF DEVICE INITIALIZATION
                        0527    10$:                                    ;
                        0527    ;
                        0527    ; Allocate and initialize Channel Request Block.
                        0527    ;
  51       0048 8F  3C  0527            MOVZWL  #CRB$C_LENGTH,R1        ; SET SIZE OF CRB
             B6     10  052C            BSBB    ALONPAGD                ; ALLOCATE SPACE FOR CRB
```

```
         08 A2     51   B0  052E   1882          MOVW     R1,CRB$W_SIZE(R2)          ; SET CORRECT SIZE
         0A A2     05   90  0532   1883          MOVB     #DYN$C_CRB,CRB$B_TYPE(R2)  ; SET CORRECT TYPE
            62     62   DE  0536   1884          MOVAL    CRB$L_WQFL(R2),CRB$L_WQFL(R2)   ; INITIALIZE WAIT QUEUE HEADER
         04 A2     62   DE  0539   1885          MOVAL    CRB$L_WQFL(R2),CRB$L_WQBL(R2)   ; FLINK AND BLINK
         50     24 A2   9E  053D   1886          MOVAB    CRB$L_INTD(R2),R0          ; SET ADDRESS OF INTD AREAD
   80    9F163CBB 8F    D0  0541   1887          MOVL     #^X9F163CBB,(R0)+          ; "PUSHR ^M<R2,R3,R4,R5>,JSB @#"
            80     59   D0  0548   1888          MOVL     R9,(R0)+                   ; ADDR OF XXX$INT ROUTINE
                   80   D4  054B   1889          CLRL     (R0)+                      ; CLEAR OUT UNNEEDED AREA
            60     5A   D0  054D   1890          MOVL     R10,(R0)                   ; ADDR OF XXX$INITIAL ROUTINE
            5A     52   D0  0550   1891          MOVL     R2,R10                     ; SAVE CRB ADDRESS
                            0553   1892  ;
                            0553   1893  ; Allocate and initiali.e Interrupt Dispatch Block.
                            0553   1894  ;
            51     68   9A  0553   1895          MOVZBL   ADPTAB_ID^   TS(R8),R1     ; GET # OF IDB UNITS
  51     00000038 9F41  DE  0556   1896          MOVAL    @#IDB$C_LENGTH[R1],R1      ; GET TOTAL SIZE OF IDB
                   84   10  055E   1897          BSBB     ALONPAGD                   ; ALLOCATE SPACE FOR CRB
         08 A2     51   B0  0560   1898          MOVW     R1,IDB$W_SIZE(R2)          ; SET STRUCTURE SIZE
         0A A2     09   90  0564   1899          MOVB     #DYN$C_IDB, -              ; AND TYPE CODE
                            0568   1900                   IDB$B_TYPE(R2.
                   68   9B  0568   1901          MOVZBW   ADPTAB_IDBUNI. .8),-       ; SET COUNT OF UNITS
              0C A2         056A   1902                   IDB$W_UNITS(R2)
      62   00E4'CF44    D0  056C   1903          MOVL     W^SBICONF[R4], -           ; SET CSR ADDRESS TO
                            0572   1904                   IDB$L_CSR(R2)              ;  START OF ADAPTER REG SPACE
         2C AA     52   D0  0572   1905          MOVL     R2, -                      ; SET ADDRESS OF IDB INTO CRB
                            0576   1906                   CRB$L_INTD+VEC$L_IDB(R10)
            59     52   D0  0576   1907          MOVL     R2,R9                      ; SAVE ADDRESS OF IDB
                            0579   1908  ;
                            0579   1909  ; Allocate and initialize Adapter Control Block (ADP).
                            0579   1910  ;
         51     01 A8   3C  0579   1911          MOVZWL   ADPTA3_ADPLEN(R8),R1       ; GET SIZE OF ADAPTER
                 FF64   30  057D   1912          BSBW     ALONPAGD                   ; ALLOCATE SPACE FOR CRB
         08 A2     51   B0  0580   1913          MOVW     R1,ADP$W_SIZE(R2)          ; SET SIZE OF STRUCTURE
         0A A2     01   90  0584   1914          MOVB     #DYN$C_ADP,ADP$B_TYPE(R2); AND TYPE CODE
            62     69   D0  0588   1915          MOVL     IDB$L_CSR(R9),ADP$L_CSR(R2); SET ADDRESS OF CONFIGURATION REGISTER
         0C A2     54   B0  058B   1916          MOVW     R4,ADP$W_TR(R2)            ; SET TR/SLOT-16 NUMBER OF ADAPTER
              03 A8     9B  058F   1917          MOVZBW   ADPTAB_ATYPE(R8),-         ; SET THE ADAPTER TYPE
              0E A2         0592   1918                   ADP$W_ADPTYPE(R2)
         10 A2     5A   D0  0594   1919          MOVL     R10,ADP$L_CRB(R2)          ; POINT ADP TO CRB
                            0598   1920  ;       CMPW     ADP$W_ADPTYPE(R2),#AT$_CI  ; CI?
                            0598   1921  ;       BEQL     20$                        ; YES, DO NOT CONNECT UP VECTORS
                            0598   1922  ;
                            0598   1923  ; Initialize adapter interrupt vectors in System Control Block.
                            0598   1924  ;
   50    00000000'GF    D0  0598   1925          MOVL     G^EXE$GL_SCB,R0            ; GET ADDRESS OF SCB
      55     5B    09   78  059F   1926          ASHL     #9,R11,R5                  ; Turn SCB page offset into byte offset.
            50     55   C0  05A3   1927          ADDL     R5,R0                      ; set to beginning of correct SCB page.
  54     54    04    00   EF  05A6   1928          EXTZV    #0,#4,R4,R4                ; Use low 4 bits of nexus number.
     50     0100 C044   DE  05AB   1929          MOVAL    ^X100(R0)[R4],R0           ; COMPUTE ADDR OF 1ST VECTOR
         1C A2     50   D0  05B1   1930          MOVL     R0,ADP$L_AVECTOR(R2)       ; SAVE ADDR OF ADAPTER'S SCB VECTORS
         60     25 AA   DE  05B5   1931          MOVAL    CRB$L_INTD+1(R10),(R0)     ; CONNECT VECTOR TO CRB CODE
      40 A0     25 AA   DE  05B9   1932          MOVAL    CRB$L_INTD+1(R10),64(R0);  SAME FOR
    0080 C0     25 AA   DE  05BE   1933          MOVAL    CRB$L_INTD+1(R10),128(R0); ALL FOUR
    00C0 C0     25 AA   DE  05C4   1934          MOVAL    CRB$L_INTD+1(R1^),192(R0); VECTORS
                            05CA   1935  ;
                            05CA   1936  ; Continue with ADP initializati.n.
                            05CA   1937  ;
      14 A2     25 AA   DE  05CA   1938  20$:    MOVAL    CRB$L_INTD+1(R10), -       ; SAVE SCB VECTOR CONTENTS IN ADP
```

```
                     05CF            ADP$L_MBASCB(R2)
          0C    BB   05CF    PUSHR   #^M<R2,R3>               ; SAVE SOME REGISTERS
    55    52    D0   05D1    MOVL    R2,R5                    ; COPY ADP ADDRESS
    52    62    D0   05D4    MOVL    ADP$L_CSR(R2),R2         ; VIRTUAL ADDRESS OF ADAPTER
00000000'GF     16   05D7    JSB     G^MMG$SVAPTECHK          ; ADDRESS OF SPTE THAT MAPS ADAPTER
 18 A5    63    D0   05DD    MOVL    (R3),ADP$L_MBASPTE(R5)   ; SAVE CONTENTS OF SPTE
          0C    BA   05E1    POPR    #^M<R2,R3>               ; RESTORE REGISTERS
 38 AA    52    D0   05E3    MOVL    R2,CRB$L_INTD+VEC$L_ADP(R10)     ; SET CRB POINTER TO ADP
 14 A9    52    D0   05E7    MOVL    R2,IDB$L_ADP(R9)         ; AND INTO IDB
       FA12'    30   05EB    BSBW    ADPLINK                  ; LINK ADP TO END OF CHAIN
                     05EE    ;
                     05EE    ; Initialize adapter hardware.
                     05EE    ;
    55    59    D0   05EE    MOVL    R9,R5                    ; ADDRESS OF IDB
    54    65    D0   05F1    MOVL    IDB$L_CSR(R5),R4         ; ADDRESS OF CONFIGURATION REGISTER 0
     30 BA      16   05F4    JSB     @CRB$L_INTD+VEC$L_INITIAL(R10) ; INIT ADAPTER
   07FF 8F      BA   05F7    POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10>; RESTORE ALL REGISTERS
                05   05FB    RSB                              ; RETURN
                     05FC
                     05FC    .DSABL  LSB
          05FC  1997          .SBTTL  INI$KDZ11
          05FC  1998 ;++
          05FC  1999 ;
          05FC  2000 ; INPUTS:
          05FC  2001 ;          R2 - VA of next free system page
          05FC  2002 ;          R3 - VA of system page table entry to be used to map VA in R2
          05FC  2003 ;          R4 - nexus identification number of this adapter
          05FC  2004 ;
          05FC  2005 ; OUTPUTS:
          05FC  2006 ;
          05FC  2007 ;--
          05FC  2008
          05FC  2009 INI$KDZ11:
          05FC  2010
      05  05FC  2029          RSB                                     ; Return to caller.
```

```
                              05FD  2031         .SBTTL  INI$CONSOLE, init data structures for console
                              05FD  2032 ;++
                              05FD  2033 ; FUNCTIONAL DESCRIPTION:
                              05FD  2034 ;
                              05FD  2035 ;       This routine is executed only once, during system initialization.
                              05FD  2036 ;       It initializes the CRB and IDB for boot/console device.
                              05FD  2037 ;
                              05FD  2038 ;       This routine is called from INIT.
                              05FD  2039 ;
                              05FD  2040 ; INPUTS:
                              05FD  2041 ;
                              05FD  2042 ;       R3 -->  DISK [CLASS] DRIVER DDB
                              05FD  2043 ;       R4 -->  DISK [CLASS] DRIVER DPT
                              05FD  2044 ;       R5 -->  DISK [CLASS] DRIVER UCB
                              05FD  2045 ;       R6 -->  RPB
                              05FD  2046 ;       R7 -->  ADP FOR EITHER A REAL DISK OR A PORT
                              05FD  2047 ;       R9 -->  PORT DRIVER DPT (IF PRESENT)
                              05FD  2048 ;       R10-->  PORT DIRVER UCB (IF PRESENT)
                              05FD  2049 ;
                              05FD  2050 ;--
                              05FD  2051
                              05FD  2052 INI$CONSOLE::
                              05FD  2053         .ENABL  LSB
                              05FD  2054
          66 A6         91    05FD  2056         CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
          40 8F               0600  2057                 #BTD$K_CONSOLE          ; STORAGE DEVICE?
             10         12    0602  2058         BNEQ    BLD_CRB                 ; NO
   41534303 8F          D0    0604  2059         MOVL    #^A7CSA/@8+3,-          ; YES, SET DEVICE NAME
          14 A3               060A  2060                 DDB$T_NAME(R3)          ; COUNTED STRING
                              060C  2062
                              060C  2067
             57         D4    060C  2070         CLRL    R7                      ; CLEAR ADP POINTER
      54 A5  01         B0    060E  2071         MOVW    #1,UCB$W_UNIT(R5)       ; SET UNIT NUMBER TO 1
             0D         11    0612  2072         BRB     FILL_CRB
                              0614  2075
                              0614  2076 ;
                              0614  2077 ; NOW BUILD THE AUXILIARY DATA BLOCKS (CRB,IDB)
                              0614  2078 ;
                              0614  2079 BLD_CRB:
      58   10 A7        D0    0614  2080         MOVL    ADP$L_CRB(R7),R8        ; GET ADDRESS OF CRB IF IT EXISTS
      0E A7  01         B1    0618  2081         CMPW    #AT$_UBA,ADP$W_ADPTYPE(R7); IS THIS A UNIBUS ADAPTER?
             03         13    061C  2082         BEQL    FILL_CRB                ; YES, ALLOCATE CRB
           0088         31    061E  2083         BRW     100$                    ; NO, CRB/IDB ALREADY ALLOCATED
                              0621  2084
                              0621  2085 FILL_CRB:
    00000000'9F         16    0621  2086         JSB     @#INI$ALLOC_CRB         ; GO ALLOCATE AND SETUP CRB
24 A2  9F163FBB 8F      D0    0627  2087         MOVL    #^X9F163FBB,CRB$L_INTD(R2) ; SET PUSHR #^M<R0,...R5>
                              062F  2088                                         ;  JSB @#0 INTO INTERRUPT DISPATCH
      38 A2  57         D0    062F  2089         MOVL    R7,CRB$L_INTD+VEC$L_ADP(R2)     ; SET POINTER TO ADP
         58  52         D0    0633  2090         MOVL    R2,R8                   ; SAVE CRB POINTER
    51   0058 8F        3C    0636  2091         MOVZWL  #<IDB$C_LENGTH+<8*4>>,R1; SIZE TO ALLOCATE FOR IDB
    00000000'9F         16    063B  2092         JSB     @#INI$ALONONPAGED       ; ALLOCATE IDB
      08 A2  51         B0    0641  2093         MOVW    R1,IDB$W_SIZE(R2)       ; SET SIZE OF IDB
      0A A2  09         90    0645  2094         MOVB    #DYN$C_IDB,IDB$B_TYPE(R2); AND STRUCTURE TYPE CODE
      2C A8  52         D0    0649  2095         MOVL    R2,CRB$L_INTD+VEC$L_IDB(R8) ; SET IDB INTO CRB
                              064D  2096
          66 A6         91    064D  2099         CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
```

```
                       40 8F       0650  2100                #BTD$K_CONSOLE          ; STORAGE DEVICE?
                          26    12 0652  2101        BNEQ    10$                     ; NO
50    000000F0 8F  00000000'9F  C1 0654  2102        ADDL3   @#EXE$GL_SCB,#^XF0,R0   ; YES, GET ADDRESS OF VECTOR IN SCB
                   80    25 A8  DE 0660  2103        MOVAL   CRB$L_INTD+1(R8),(R0)+  ; SET ADDR IN 1ST VECTOR
                   60    49 A8  DE 0664  2104        MOVAL   CRB$L_INTD2+1(R8),(R0)  ; SET ADDR IN 2ND VECTOR
          48 A8    9F163FBB 8F  D0 0668  2105        MOVL    #^X9F163FBB,CRB$L_INTD2(R8) ; STORE PUSHR #^M<R0...R5>
                                   0670  2106                                        ; JMP @# IN 2ND INT. DISPATCH
                   50 A8     52 D0 0670  2107        MOVL    R2,CRB$L_INTD2+VEC$L_IDB(R8); STORE ADDRESS OF IDB IN CRB
                   2C B8     1F D0 0674  2108        MOVL    #PR$_CSTD, -            ; STORE IPR NUMBER OF CONSOLE INTERFACE
                                   0678  2109                @CRB$L_INTD+8(R8)       ; REGISTER AS DEVICE CSR ADDRESS
                             2F 11 0678  2110        BRB     100$
                                   067A  2113
                   62     58 A6 D0 067A  2114 10$:   MOVL    RPB$L_CSRVIR(R6), -     ; SAVE BOOT DEVICE CSR ADDRESS
                                   067E  2115                IDB$L_CSR(R2)           ; IN INTERRUPT DISPATCH BLOCK
                             11 91 067E  2116        CMPB    #BTD$K_UDA,-            ; LOW ORDER BYTE OF ORIGINAL R0 TELLS
                          66 A6    0680  2117                RPB$B_DEVTYP(R6)        ;  BOOT DEVICE TYPE.
                             08 12 0682  2118        BNEQ    20$                     ; IF NOT BOOTING FROM A UDA BRANCH
                                   0684  2119                                        ;  AROUND.
         00000000'9F      58 A6 D0 0684  2120        MOVL    RPB$L_CSRVIR(R6), -     ; COPY VIRTUAL ADDRESS OF UDA PORT CSR
                                   068C  2121                @#BOO$GB_SYSTEMID       ;  TO LOW ORDER LONGWORD OF SYSTEMID
                                   068C  2122 20$:
                   14 A2     57 D0 068C  2123        MOVL    R7,IDB$L_ADP(R2)        ; POINT IDB TO ADP
                   50     1E A6 3C 0690  2124        MOVZWL  RPB$W_ROUBVEC(R6),R0    ; GET USER SPECIFIED VECTOR
                             0A 12 0694  2125        BNEQ    30$                     ; BRANCH IF VECTOR SPECIFIED
                   50     66 A6 9A 0696  2126        MOVZBL  RPB$B_DEVTYP(R6),R0     ; ELSE GET DEVICE TYPE CODE
               50   FFFE'CF40 3C 069A  2127        MOVZWL  W^BOOTVECTOR-2[R0],R0   ; GET DEFAULT INTERRUPT VECTOR
                  50    10 B740 9E 06A0  2128 30$:   MOVAB   @ADP$L_VECTOR(R7)[R0],R0; COMPUTE ADDRESS OF VECTOR
                   60     26 A8 9E 06A5  2129        MOVAB   CRB$L_INTD+2(R8),(R0)   ; SET ADDR OF INTERRUPT VECTOR
                                   06A9  2130
                                   06A9  2136                                        ;
                                   06A9  2137 100$:
                                05 06A9  2138        RSB                             ; RETURN
                                   06AA  2139        .DISABLE LSB
```

```
                                  06AA          .SBTTL  EXE$INI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
                                  06AA  ;++
                                  06AA  ; FUNCTIONAL DESCRIPTION:
                                  06AA  ;
                                  06AA  ; EXE$INI_TIMWAIT initializes EXE$GL_TENUSEC and EXE$GL_UBDELAY, cells used
                                  06AA  ; in the time-wait macros.  The first data cell, EXE$GL_TENUSEC, is the number
                                  06AA  ; of times the following loop will be executed in ten u-seconds.  This is
                                  06AA  ; done once here to calibrate the loop instead of reading the processor clock.
                                  06AA  ; The resulting number is used in the system macros TIMEWAIT and TIMEDWAIT.
                                  06AA  ;
                                  06AA  ; The first step is to initialize EXE$GL_UBDELAY.  If the bit test instruction
                                  06AA  ; in the TIMEWAIT macro is executed too rapidly in a loop, it can saturate the
                                  06AA  ; Unibus.  EXE$GL_UBDELAY is used to introduce a 3 microsecond delay loop into
                                  06AA  ; the TIMEWAIT bit test loop.
                                  06AA  ;
                                  06AA  ; This routine is called only once, from INIT.
                                  06AA  ;
                                  06AA  ; INPUT PARAMETERS:
                                  06AA  ;
                                  06AA  ;       NONE
                                  06AA  ;
                                  06AA  ; IMPLICIT INPUTS:
                                  06AA  ;
                                  06AA  ;       Time-of-day processor clock.
                                  06AA  ;       Interval timers.
                                  06AA  ;
                                  06AA  ; OUTPUT PARAMETERS:
                                  06AA  ;
                                  06AA  ;       R0 - Destroyed.
                                  06AA  ;
                                  06AA  ; IMPLICIT OUTPUTS:
                                  06AA  ;
                                  06AA  ;       EXE$GL_TENUSEC - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                                  06AA  ;                        macros loop for 10 micro-seconds.
                                  06AA  ;
                                  06AA  ;       EXE$GL_UBDELAY - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                                  06AA  ;                        macros loop for 3 micro-seconds in the unibus delay
                                  06AA  ;                        loop.
                                  06AA  ;
                                  06AA  ;--
                                  06AA
                                  06AA  EXE$INI_TIMWAIT::                       ; Initialize time-wait data cells
                                  06AA          .ENABLE LSB
                                  06AA
                                  06AA
                                  06AA
                                  06AA
              F953'    30         06AA          BSBW    INI$CACHE               ; Initialize and enable cache.
          19    00     DA         06AD          MTPR    #0,#PR790$_NICR         ; Initialize next interval count register.
                                  06B0
   7E   00004E20 8F    D0         06B0          MOVL    #20000,-(SP)            ; # of times to execute timed loop.
          18    11     DA         06B7          MTPR    #^X11,#PR$_ICCS         ; Start clock, no interrupts.
                                  06BA
                                  06BA  ; * * * start of loop to time * * *
             FD 6E     F5         06BA  10$:    SOBGTR  (SP),10$                ; Delay loop.
                                  06BD  ; * * * end of loop to time * * *
                                  06BD
```

```
                                        06BD  2213
                                        06BD  2217
                                        06BD  2221
                       50    1A    DB   06BD  2223          MFPR     #PR790$_ICR,R0          ; Read total time to execute loop.
                                        06C0  2225
                       18    00    DA   06C0  2226          MTPR     #0,#PR$_ICCS            ; Shut off clock.
00000000'GF    0000EA60 8F   50    C7   06C3  2227          DIVL3    R0,#60000,G^EXE$GL_UBDELAY; Calculate number of times through
                    00000000'GF    D6   06CF  2228          INCL     G^EXE$GL_UBDELAY        ; loop to delay 3 microseconds.
                                        06D5  2229
                                        06D5  2233
                                        06D5  2237
                                        06D5  2241
                       19    00    DA   06D5  2243          MTPR     #0,#PR790$_NICR         ; Initialize next interval count register.
                                        06D8  2245
               50   00004E20 8F    D0   06D8  2246          MOVL     #20000,R0               ; Number of times to execute test loop
               6E   00000000'GF    D0   06DF  2247          MOVL     G^EXE$GL_UBDELAY,(SP)   ; Get delay loop iteration count.
                       18    11    DA   06E6  2248          MTPR     #^X11,#PR$_ICCS         ; Start clock, no interrupts
                                        06E9  2249
                                        06E9  2250 ; **** Start of loop to time
       000006F7'EF     8000 8F     B3   06E9  2251 20$:     BITW     #^X8000,40$             ; Random BITx instruction to time
                             03    12   06F2  2252          BNEQ     40$                     ; Random conditional branch instruction
                          FD 6E    F5   06F4  2253 30$:     SOBGTR   (SP),30$                ; Delay 3 microseconds.
                          EF 50    F5   06F7  2254 40$:     SOBGTR   R0,20$                  ; Loop
                                        06FA  2255 ; **** End of loop to time
                                        06FA  2256
                                        06FA  2260
                                        06FA  2264
                                        06FA  2268
                       50    1A    DB   06FA  2270          MFPR     #PR790$_ICR,R0          ; Read total time to execute loop.
                       18    00    DA   06FD  2272          MTPR     #0,#PR$_ICCS            ; Shut clock off
                             8E    D5   0700  2273          TSTL     (SP)+                   ; Pop delay loop index off stack.
00000000'GF    00030D40 8F   50    C7   0702  2274          DIVL3    R0,#200000,G^EXE$GL_TENUSEC ; Calculate number of times to
                    00000000'GF    D6   070E  2275          INCL     G^EXE$GL_TENUSEC        ; execute the loop to kill 10 u-secs.
                                        0714  2276
                                        0714  2278 ;
                                        0714  2279 ; Store the TIMEDWAIT values calculated with cache enabled in the BOOTDRIVR
                                        0714  2280 ; TIMEDWAIT cells.
                                        0714  2281 ;
               50   00000000'GF    D0   0714  2282          MOVL     G^EXE$GL_RPB,R0         ; Get address of RPB.
                     50    34 A0   D0   071B  2283          MOVL     RPB$L_IOVEC(R0),R0      ; Get address of BOOTDRIVR I/O cells.
                    00000000'GF    D0   071F  2284          MOVL     G^EXE$GL_TENUSEC,-      ; Store TENUSEC value in BOOTDRIVR.
                          3E A0         0725  2285                   BQO$L_TENUSEC(R0)
                    00000000'GF    D0   0727  2286          MOVL     G^EXE$GL_UBDELAY,-      ; Store UBDELAY value in BOOTDRIVR.
                          42 A0         072D  2287                   BQO$L_UBDELAY(R0)
                                        072F  2289
                                   05   072F  2290          RSB                              ; Return
                                        0730  2291          .DISABLE LSB
```

```
                              0730  2299          .SBTTL  EXE$INIT_TODR  -  SET SYSTEM TIME TO CORRECT VALUE AT STARTUP
                              0730  2300 ;++
                              0730  2301 ; FUNCTIONAL DESCRIPTION:
                              0730  2302 ;
                              0730  2303 ;        EXE$INIT_TODR SOLICITS THE CORRECT TIME FROM THE OPERATOR IF NECESSARY,
                              0730  2304 ;        CONVERTS THE ASCII RESPONSE TO BINARY FORMAT AND CALLS AN INTERNAL
                              0730  2305 ;        ENTRY POINT OF THE $SETIME SYSTEM SERVICE TO SET THE NEW SYSTEM TIME
                              0730  2306 ;        IN MEMORY WITHOUT MODIFYING THE CONTENTS OF THE SYSTEM DISK.
                              0730  2307 ;
                              0730  2308 ;        IF THE TIME WOULD NORMALLY BE SOLICITED FROM AN OPERATOR, BECAUSE
                              0730  2309 ;        THE HARDWARE TIME OF YEAR CLOCK IS ZERO, THEN THE SYSGEN PARAMETER
                              0730  2310 ;        "TPWAIT" IS CHECKED.  IF IT IS ZERO, THEN IT IS ASSUMED THAT NO
                              0730  2311 ;        OPERATOR IS PRESENT AND THE SYSTEM IS BOOTED USING THE LAST TIME
                              0730  2312 ;        RECORDED IN THE SYSTEM IMAGE.  IF THE PARAMETER IS NON ZERO THEN
                              0730  2313 ;        THAT TIME IS USED AS THE MAXIMUM TIME TO WAIT BEFOR ASSUMING THAT
                              0730  2314 ;        THERE IS NO OPERATOR AND BOOTING ANY WAY.  IF THE PARAMETER IS
                              0730  2315 ;        NEGATIVE, THE SYSTEM WILL WAIT FOREVER.
                              0730  2316 ;
                              0730  2317 ;        THIS ROUTINE IS CALLED ONLY ONCE, FROM SYSINIT OR STASYSGEN.
                              0730  2318 ;
                              0730  2319 ; INPUT PARAMETERS:
                              0730  2320 ;
                              0730  2321 ;        NONE
                              0730  2322 ;
                              0730  2323 ; IMPLICIT INPUTS:
                              0730  2324 ;
                              0730  2325 ;        TIME-OF-DAY PROCESSOR CLOCK.
                              0730  2326 ;
                              0730  2327 ; OUTPUT PARAMETERS:
                              0730  2328 ;
                              0730  2329 ;        R0,R1 - DESTROYED
                              0730  2330 ;
                              0730  2331 ; IMPLICIT OUTPUTS:
                              0730  2332 ;
                              0730  2333 ;        EXE$GQ_SYSTIME - SET TO CURRENT TIME IN 100 NANOSECOND UNITS SINCE
                              0730  2334 ;                         17-NOV-1858  00:00:00.
                              0730  2335 ;
                              0730  2336 ;--
                              0730  2337
                              0730  2338 ;
                              0730  2339 ; Stack storage offsets:
                              0730  2340 ;
                    00000000  0730  2341 TTCHAN  = ^X00                      ; CHANNEL FOR TERMINAL (LONGWORD)
                    00000004  0730  2342 TTNAME  = ^X04                      ; STRING DESCRIPTOR FOR OPERATOR'S TERM
                    0000000C  0730  2343 TMPDESC = ^X0C                      ; TEMPORY STRING DESCRIPTOR (QUADWORD)
                    00000014  0730  2344 INTIME  = ^X14                      ; INPUT TIME VALUE (QUADWORD)
                    0000001C  0730  2345 LINBUF  = ^X1C                      ; INPUT LINE BUFFER (5 LONGWORDS)
                    00000014  0730  2346 LINBUFSIZ = ^X14                    ;  (LENGTH OF LINE BUFFER IN BYTES)
                              0730  2347
                              0730  2348 ;
                              0730  2349 ; PURE DATA
                              0730  2350 ;
                              0730  2351 TERM_NAMADR:
                 30 41 50 4F  0730  2352         .ASCII  \OPA0\              ; DEVICE NAME FOR OPERATOR'S TERMINAL
                    00000004  0734  2353 TERM_NAMSIZ = . - TERM_NAMADR
74 61 64 20 64 69 6C 61 76 6E 69 00' 0734  2354 TIMERR: .ASCIC  \invalid date/time\    ;
           65 6D 69 74 2F 65  0740
```

```
                         11  0734
                             0746   2355 TIMEPROMPT:
                        33'  0746   2356           .BYTE    NPROMPT
54 4E 45 20 45 53 41 45 4C 50 0A 0D  0747   2357           .ASCII   <13><10>/PLEASE ENTER DATE AND TIME (DD-MMM-YYYY  HH:MM)  /
20 44 4E 41 20 45 54 41 44 20 52 45  0753
4D 4D 4D 2D 44 44 28 20 45 4D 49 54  075F
4D 4D 3A 48 48 20 20 59 59 59 59 2D  076B
                     20 20 29  0777
                    00000033  077A   2358 NPROMPT=.-TIMEPROMPT-1
                              077A   2359
                              077A   2360
                              077A   2361 EXE$INIT_TODR::                           ; SET CORRECT TIME
                              077A   2362           .ENABLF LSB
            077C 8F      BB   077A   2363           PUSHR    #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; SAVE REGISTERS
              5E    30   C2   077E   2364           SUBL     #4*12,SP                  ; SCRATCH STORAGE
              56    5E   D0   0781   2365           MOVL     SP,R6                     ; SAVE ADDRESS OF SCRATCH STORAGE
           04 A6    04   9A   0784   2366           MOVZBL   #TERM_NAMSIZ,TTNAME(R6)   ; SET SIZE OF OPERATOR'S TERM NAME AND
        08 A6    FFA4 CF   9E   0788   2367           MOVAB    W^TERM_NAMADR,TTNAME+4(R6) ;  PIC ADDRESS INTO TERM NAME DESC
     1C 00000000'GF    00'  E0   078E   2368           BBS      S^#EXE$V_SETTIME,G^EXE$GL_FLAGS,READTIME ; BR TO SOLICIT TIME
                              0796   2369
                              0796   2370
                              0796   2374
                              0796   2378
                              0796   2382
              50    1B   DB   0796   2384           MFPR     #PR790$_TODR,R0           ; GET TIME OF DAY CLOCK VALUE
                              0799   2386
     59   00000000'GF    50   C3   0799   2388           SUBL3    R0,G^EXE$GL_TODR,R9       ; GET TOD DELTA TIME (10 MS UNITS)
                    09   1B   07A1   2389           BLEQU    5$                        ; BRANCH IF TIME IS LATER
         0083D600 8F    59   D1   07A3   2390           CMPL     R9,#24*60*60*100          ; CHECK FOR SETBACK OF ONE DAY
                    06   1E   07AA   2391           BGEQU    READTIME                  ; MORE, MUST SOLICIT TIME
              14 A6    7C   07AC   2396 5$:       CLRQ     INTIME(R6)                ; NULL ARGUMENT FOR EXE$SETIME_INT
                  00C7   31   07AF   2397           BRW      200$                      ; RETURN TO CALLER
                              07B2   2398
                              07B2   2399 READTIME:                                 ; SOLICIT TIME
                    59   D4   07B2   2400           CLRL     R9                        ; CLEAR A FLAG
       58    00000000'GF    32   07B4   2401           CVTWL    G^SGN$GW_TPWAIT,R8        ; PICK UP TIMEOUT WAIT INTERVAL
                    14   14   07BB   2402           BGTR     8$                        ; POSITIVE, WAIT THAT PERIOD ONCE
                    0D   19   07BD   2403           BLSS     7$                        ; NEGATIVE IS WAIT FOREVER
                              07BF   2404 6$:
     50   00000000'GF    01   C1   07BF   2406           ADDL3    #1,G^EXE$GL_TODR,R0       ; ZERO, SET TIME-OF-DAY CLOCK TO
                              07C7   2408
                              07C7   2412
                              07C7   2416
                              07C7   2420
              1B    50   DA   07C7   2422           MTPR     R0,#PR790$_TODR           ;   KNOWN VALUE + 10 MSEC AND FINISH UP
                              07CA   2424
                    E0   11   07CA   2426           BRB      5$                        ;
                              07CC   2428
                              07CC   2433
              58    14   D0   07CC   2434 7$:       MOVL     #20,R8                    ; STARTING WAIT
                    59   D6   07CF   2435           INCL     R9                        ; NEGATIVE - WAIT FOREVER
                              07D1   2436 8$:       $ASSIGN_S       TTNAME(R6),TTCHAN(R6) ; AND ASSIGN TO INPUT DEVICE
                 DD 50   E9   07DF   2437           BLBC     R0,6$                     ; ERROR - FALL BACK TO STORED TIME
         52   FF60 CF   9E   07E2   2438 10$:      MOVAB    W^TIMEPROMPT,R2           ; GET ADDRESS OF PROMPT STRING
              53    82   9A   07E7   2439           MOVZBL   (R2)+,R3                  ; AND LENGTH
                              07EA   2440           $QIOW_S #0,W^TTCHAN(R6),-          ; PROMPT AND READ TIME
                              07EA   2441                    #<IO$_READPROMPT!IO$M_PURGE!IO$M_TIMED!IO$M_CVTLOW>,-
```

```
                               07EA                      TMPDESC(R6),,,-                 :  I/O STATUS BLOCK , NO AST OR PARAM
                               07EA                      LINBUF(R6),#LINBUFSIZ,-         :  BUFFER ADDRESS AND SIZE
                               07EA                      R8,#0,-                         :  TIME OUT
                               07EA                      R2,R3                           :  PROMPT ADDRESS AND SIZE
                AD 50    E9    080F            BLBC      R0,6$                           : ERROR - FALL BACK TO STORED TIME
         54     0C A6    7D    0812            MOVQ      TMPDESC(R6),R4                  : GET COMPLETION STATUS
                0D 54    E8    0816            BLBS      R4,20$                          : CONTINUE IF SUCCESSFUL READ
                A3 59    E9    0819            BLBC      R9,6$                           : FAILED ON ONE-TIME READ, RETURN
     58      01 A848     9E    081C            MOVAB     1(R8)[R8],R8                    : (2 * TIMEOUT) + 1
             58    58    3C    0821            MOVZWL    R8,R8                           : BOUND TIMEOUT
                   BC    11    0824            BRB       10$                             : TRY AGAIN FOR TIME
                               0826  20$:                                                : SOMETHING WAS INPUT
   0C A6        0E A6    3C    0826            MOVZWL    TMPDESC+2(R6),TMPDESC(R6) ; FORM DESCRIPTOR FOR BUFFER
   10 A6        1C A6    9E    082B            MOVAB     LINBUF(R6),TMPDESC+4(R6) ; SET DESCRIPTOR ADDRESS
                               0830            $BINTIM_S TMPDESC(R6),INTIME(R6) ; CONVERT TO BINARY TIME
                05 50    E9    083D            BLBC      R0,89$                          : INVALID TIME
                18 A6    D5    0840            TSTL      INTIME+4(R6)                    : CHECK FOR DELTA TIME
                   2A    14    0843            BGTR      100$                            : BRANCH IF NOT - OK
                               0845  89$:                                                : INVALID TIME VALUE INPUT
     52      FEEB CF     9E    0845            MOVAB     W^TIMERR,R2                     : ADDRESS OF ERROR MESSAGE
             53    82    9A    084A            MOVZBL    (R2)+,R3                        : GET STRING LENGTH
                               084D            $QIOW_S #0,TTCHAN(R6),-                   : GIVE ERROR MESSAGE
                               084D                      #IO$_WRITEVBLK,-                :
                               084D                      ,,,-                            :  NO I/O STATUS,AST OR AST PARAM
                               084D                      (R2),R3 ,-                      :  BUFFER ADDRESS, LENGTH
                               084D                      #0,#32                          :  SET CARRIAGE CONTROL TO CR/LF
                 FF73    31    086C            BRW       10$                             : AND TRY AGAIN
                               086F  100$:                                               : EXIT
                               086F            $DASSGN_S TTCHAN(R6)                      : DE-ASSIGN TERMINAL CHANNEL
                14 A6    7F    0879  200$:     PUSHAQ    INTIME(R6)                      : SET NEW SYSTEM TIME
        00000000'GF   01    FB    087C         CALLS     #1,G^EXE$SETIME_INT             : USE TODR CLOCK TO SET SYSTEM TIME
00000000'GF  00000000'GF    7D    0883         MOVQ      G^EXE$GQ_TODCBASE,G^EXE$GQ_BOOTTIME ; SAVE BOOT TIME
                5E 30    C0    088E            ADDL      #12*4,SP                        : CLEAN OFF SCRATCH STORAGE
              077C 8F    BA    0891            POPR      #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; RESTORE REGISTERS
                               0895
                               0895  ;
                               0895  ; Fall through into the deallocate logic.
                               0895  ;
                               0895  ;         RSB                                       ; *** This goes in if another piece of
                               0895                                                      ; *** initialization code is added that
                               0895                                                      ; *** is executed after EXE$INI_TIMWAIT.
                               0895            .DISABLE LSB
                               0895
```

```
                                  0895  2486 DEAL_INIT_CODE:                              ; DEALLOCATE THE INITIALIZATION CODE
                                  0895  2487 ;
                                  0895  2488 ; It is the duty of the last-executed, loadable initialization
                                  0895  2489 ; routine to make itself and all other such routines disappear, i.e.,
                                  0895  2490 ; release the space they occupy to non-paged pool.  Each routine's vector
                                  0895  2491 ; must be disconnected, e.g., be made to point to the symbol, EXE$LOAD_ERROR.
                                  0895  2492 ;
                                  0895  2493 ; NOTE:  This means that new initialization routines should be added
                                  0895  2494 ;        to this module in a particular order, not necessarily at the
                                  0895  2495 ;        end of the module!
                                  0895  2496 ;
                                  0895  2497         .ENABLE LSB
                  7E    52    7D  0895  2498         MOVQ    R2,-(SP)                 ; Save some registers
                                  0898  2499
                                  0898  2500 ;
                                  0898  2501 ; First find the vectors that point to these initialization routines
                                  0898  2502 ; and reset them to point to EXE$LOAD_ERROR.
                                  0898  2503 ;
           50   0000'CF       9E  0898  2504         MOVAB   W^SYSL$BEGIN,R0          ; Compute bounds of releasable piece:
     51    50   00000000'8F   C1  089D  2505         ADDL3   #<STAY_HEADER-SYSL$BEGIN>,R0,R1 ; starting and ending addresses.
           52   00000000'GF   9E  08A5  2506         MOVAB   G^EXE$AL_LOAVEC,R2       ; Get starting address of vectors.
           53   00000000'GF   9E  08AC  2507         MOVAB   G^EXE$LOAD_ERROR,R3      ; Get end of vectors.
           9F17 8F      62    B1  08B3  2508 10$:    CMPW    (R2),#^X9FT7             ; Is this JMP @# ?
                        1B    13  08B8  2509         BEQL    30$                      ; Br if yes, skip past it.
           80 8F     03 A2    91  08BA  2510         CMPB    3(R2),#^X80              ; Is this a system space address
                        16    12  08BF  2511         BNEQ    40$                      ; Br if no, assume it's a HALT instr.
                  50    62    D1  08C1  2512         CMPL    (R2),R0                  ; Is address before the releasable
                        0C    1F  08C4  2513         BLSSU   20$                      ;  piece of memory?  Br on yes.
                  51    62    D1  08C6  2514         CMPL    (R2),R1                  ; Is address after the releasable
                        07    1A  08C9  2515         BGTRU   20$                      ;  piece of memory?  Br on yes.
           62   00000000'GF   9E  08CB  2516         MOVAB   G^EXE$LOAD_ERROR,(R2)    ; Reset this vector.
                  52    02    C0  08D2  2517 20$:    ADDL    #2,R2                    ; Point past this vector.
                        52    D6  08D5  2518 30$:    INCL    R2                       ; Come here to point past JMP @#.
                        52    D6  08D7  2519 40$:    INCL    R2                       ; Come here to point past HALT.
                  53    52    D1  08D9  2520         CMPL    R2,R3                    ; Past the end of the vectors?
                        D5    1F  08DC  2521         BLSSU   10$                      ; Keep searching vectors.
                                  08DE  2522 ;
                                  08DE  2523 ; Now release the memory to non-paged pool.
                                  08DE  2524 ;
           50      0000'CF    9E  08DE  2525         MOVAB   W^SYSL$BEGIN,R0          ; Point to start of module
           51      0000'8F    3C  08E3  2526         MOVZWL  #<STAY_HEADER-SYSL$BEGIN>,R1 ; Length to vaporize
                     F721'    31  08E8  2527         BRW     50$                      ; Br to code that is not released.
                                  08EB  2528
                              00000000  2529         .PSECT  $$$INIT__END,PAGE        ; 'PAGE' SINCE 16-BYTE ALIGN IS NOT
                                  0000  2530
                                  0000  2531 STAY_HEADER:
           00000000 00000000      0000  2532         .LONG   0,0
                           0000'  0008  2533         .WORD   <SYSL$END-STAY_HEADER>
                              62  000A  2534         .BYTE   DYN$C_LOADCODE
                              00  000B  2535         .BYTE   0
                                  000C  2536
           00000000'9F        16  000C  2537 50$:    JSB     @#EXE$DEANONPGDSIZ       ; Just the smile on the Chesire cat
                  52    8E    7D  0012  2538         MOVQ    (SP)+,R2                 ; Restore
                              05  0015  2539         RSB                              ; Return.
                                  0016  2540
                                  0016  2541         .DISABLE LSB
                                  0016  2542         .END
```

```
$$$VMSDEFINED                  = 00000001                 BUS_CSR_LEN                      00000004 R       08
$$T1                           = 00000001                 CI$INITIAL                       ********      X  0A
ABUS_INDEX                       00000014 RG      09      CI$INT                           ********      X  0A
ABUS_LOOP                        00000054 R       0A      CITAB                            00000015 R       08
ABUS_TYPE                        00000010 RG      09      CONFIG_790                       000000C5 R       0A
ABUS_VA                          00000000 RG      09      CONFIG_IOSPACE                   000000FF R       0A
ADAPTERS                         00u00000 R       02      CONFREG                          000000A4 R       08
ADP$B_TYPE                     = 0000000A                 CONFREGL                         000001E4 R       08
ADP$C_CIADPLEN                 = 00000030                 CPU_ADPSIZE                      00000019 R       08
ADP$C_DRADPLEN                 = 00000030                 CPU_TYPE                       = 00000004
ADP$C_MBAADPLEN                = 00000030                 CR                             = 0000000D
ADP$C_UBAADPLEN                = 00000258                 CRB$B_TYPE                     = 0000000A
ADP$L_AVECTOR                  = 0000001C                 CRB$C_LENGTH                   = 00000048
ADP$L_CRB                      = 00000010                 CRB$L_INTD                     = 00000024
ADP$L_CSR                      = 00000000                 CRB$L_INTD2                    = 00000048
ADP$L_DPQFL                    = 00000014                 CRB$L_WQBL                     = 00000004
ADP$L_LINK                     = 00000004                 CRB$L_WQFL                     = 00000000
ADP$L_MBASCB                   = 00000014                 CRB$W_SIZE                     = 00000008
ADP$L_MBASPTE                  = 00000018                 CREATE_ARRAYS                    000001BA R       0A
ADP$L_MRACTMDRS                = 0000005C                 CSR_LEN_OFFSET                 = FFFFFFFB
ADP$L_MRQFL                    = 00000030                 DDB$T_NAME                     = 00000014
ADP$L_UBASCB                   = 00000044                 DEAL_INIT_CODE                   00000895 R       0A
ADP$L_UBASPTE                  = 00000054                 DIRECT_VEC_NODE_CNT              00000009 R       08
ADP$L_VECTOR                   = 00000010                 DR$INITIAL                       ********      X  0A
ADP$W_ADPTYPE                  = 0000000E                 DR$INT                           ********      X  0A
ADP$W_DPBITMAP                 = 00000060                 DRTAB                            00000011 R       08
ADP$W_MRFFENCE                 = 0000015C                 DYN$C_ADP                      = 00000001
ADP$W_MRFREGARY                = 0000015E                 DYN$C_CONF                     = 00000007
ADP$W_MRNFENCE                 = 00000062                 DYN$C_CRB                      = 00000005
ADP$W_MRNREGARY                = 00000064                 DYN$C_IDB                      = 00000009
ADP$W_SIZE                     = 00000008                 DYN$C_INIT                     = 00000063
ADP$W_TR                       = 0000000C                 DYN$C_LOADCODE                 = 00000062
ADP$W_UMR_DIS                  = 00000256                 END_ABUS_LOOP                    000000E7 R       0A
ADPLINK                          ********      X  0A      END_NEXUS                        000001B5 R       0A
ADPTAB_ADPLEN                    00000001                 ERROR_HALT                       00000239 R       0A
ADPTAB_ATYPE                     00000003                 ERROR_HALT_1                     0000023E R       0A
ADPTAB_IDBUNITS                  00000000                 EXE$AL_LOAVEC                    ********      X  0A
ALONPAGD                         000004E4 R       0A      EXE$DEANONPGDSIZ                 ********      X  0B
AT$_CI                         = 00000004                 EXE$GL_CONFREG                   ********      X  0A
AT$_DR                         = 00000002                 EXE$GL_CONFREGL                  ********      X  0A
AT$_MBA                        = 00000000                 EXE$GL_FLAGS                     ********      X  0A
AT$_UBA                        = 00000001                 EXE$GL_NUMNEXUS                  ********      X  0A
BADUMR                           0000030D R       08      EXE$GL_RPB                       ********      X  0A
BI_BUS_CODE                    = 80000000                 EXE$GL_SCB                       ********      X  0A
BI_CPU                         = 00000000                 EXE$GL_TENUSEC                   ********      X  0A
BI_CSR_LEN                     = 00000002                 EXE$GL_TODR                      ********      X  0A
BI_LIKE                        = 00000000                 EXE$GL_UBDELAY                   ********      X  0A
BLD_CRB                          00000614 R       0A      EXE$GQ_BOOTTIME                  ********      X  0A
BOO$GB_SYSTEMID                  ********      X  0A      EXE$GQ_TODCBASE                  ********      X  0A
BOO$GL_SPTFREH                   ********      X  0A      EXE$INIT_TODR                    0000077A RG      0A
BOO$GL_SPTFREL                   ********      X  0A      EXE$INI_TIMWAIT                  000006AA RG      0A
BOOTVECTOR                       00000000 R       08      EXE$INT58                        ********      X  0A
BQO$L_TENUSEC                  = 0000003E                 EXE$INT5C                        ********      X  0A
BQO$L_UBDELAY                  = 00000042                 EXE$INT60                        ********      X  0A
BTD$K_CONSOLE                  = 00000040                 EXE$LOAD_ERROR                   ********      X  09
BTD$K_UDA                      = 00000011                 EXE$MCHK_PRTCT                   ********      X  0A
BUS_CODE_OFFSET                = FFFFFFFC                 EXE$OUTZSTRING                   ********      X  0A
```

```
EXE$POWERFAIL                     ********    X    0A        MMG$SVAPTECHK                    ********    X    0A
EXE$SETIME_INT                    ********    X    0A        NDT$_BUA                       = 80000102
EXE$TEST_CSR                      ********    X    0A        NDT$_CI                        = 00000038
EXE$UBAERR_INT                    ********    X    0A        NDT$_DR32                      = 00000030
EXE$V_SETTIME                     ********    X    0A        NDT$_KDZ11                     = 80000105
FILL_CRB                          00000621 R       0A        NDT$_MB                        = 00000020
FILL_IN_SCB                       0000027E R       0A        NDT$_MEM1664NI                 = 00000012
GET_GEN_TYPE                      00000159 R       0A        NDT$_MEM16I                    = 00000011
GET_TYPE                          00000140 R       0A        NDT$_MEM16NI                   = 00000010
IDB$B_TYPE                      = 0000000A                   NDT$_MEM256EIL                 = 00000071
IDB$C_LENGTH                    = 00000038                   NDT$_MEM256EIU                 = 00000073
IDB$L_ADP                       = 00000014                   NDT$_MEM256I                   = 00000074
IDB$L_CSR                       = 00000000                   NDT$_MEM256NIL                 = 00000070
IDB$W_SIZE                      = 00000008                   NDT$_MEM256NIU                 = 00000072
IDB$W_UNITS                     = 0000000C                   NDT$_MEM4I                     = 00000009
INI$ALLOC_CRB                     ********    X    0A        NDT$_MEM4NI                    = 00000008
INI$ALONONPAGED                   ********    X    0A        NDT$_MEM64EIL                  = 00000069
INI$CACHE                         ********    X    0A        NDT$_MEM64EIU                  = 0000006B
INI$CIADP                         000004FF R       0A        NDT$_MEM64I                    = 0000006C
INI$CONSOLE                       000005FD RG      0A        NDT$_MEM64NIL                  = 00000068
INI$DRADP                         000004EA R       0A        NDT$_MEM64NIU                  = 0000006A
INI$IOMAP                         00000000 RG      0A        NDT$_MPM0                      = 00000040
INI$KDZ11                         000005FC R       0A        NDT$_MPM1                      = 00000041
INI$L_IOAVECS                     00000018 R       09        NDT$_MPM2                      = 00000042
INI$L_SCBVALS                     00000247 R       0A        NDT$_MPM3                      = 00000043
INI$MBADP                         00000514 R       0A        NDT$_SCORMEM                   = 80000001
INI$MPMADP                        ********    X    06        NDT$_UB0                       = 00000028
INI$SCB                           00000259 R       0A        NDT$_UB1                       = 00000029
INI$UBADP                         000002CD R       0A        NDT$_UB2                       = 0000002A
INI$UBSPACE                       000002A9 R       0A        NDT$_UB3                       = 0000002B
INIT_ROUTINES                     00000000 R       06        NEXUSDESC                        00000020 R       08
INTIME                          = 00000014                   NOSPT                            000002E4 R       08
IO$M_CVTLOW                       ********    X    0A        NPROMPT                        = 00000033
IO$M_PURGE                        ********    X    0A        NUMUBAVEC                      = 00000080
IO$M_TIMED                        ********    X    0A        NUMVECS                        = 00000003
IO$_READPROMPT                    ********    X    0A        NUM_PAGES                        00000000 R       04
IO$_WRITEVBLK                     ********    X    0A        NXT_NEXUS                        0000010B R       0A
IO790$AL_IOA0                   = 20000000                   PA                             = 20020000
IO790$AL_IOACR                  = 00080000                   PAMM$C_NEXM                    = 0000000F
IO790$AL_NNEX                   = 00000010                   PAMM$M_PAMADD                  = 3FF00000
IO790$AL_PERABS                 = 02000000                   PAMM$S_CODE                    = 00000004
IO790$AL_PERNEX                 = 00002000                   PAMM$V_CODE                    = 00000000
IO790$AL_UBOSP                  = 00100000                   PR$_CSTD                       = 0000001F
IO790$C_SBIA                    = 00000001                   PR$_ICCS                       = 00000018
LF                              = 0000000A                   PR$_SID_TYP730                 = 00000003
LINBUF                          = 0000001C                   PR$_SID_TYP750                 = 00000002
LINBUFSIZ                       = 00000014                   PR$_SID_TYP780                 = 00000001
MAP_NEXUS                         00000199 R       0A        PR$_SID_TYP790                 = 00000004
MAP_PAGES                         00000213 R       0A        PR$_SID_TYP8NN                 = 00000006
MAXNEXUS                        = 00000040                   PR$_SID_TYP8SS                 = 00000u05
MBA$INITIAL                       ********    X    0A        PR$_SID_TYPUV1                 = 00000007
MBA$INT                           ********    X    0A        PR$_TBIS                       = 0000003A
MBATAB                            0000000D R       08        PR790$_ICR                     = 0000001A
MCHK$M_LOG                      = 00000001                   PR790$_NICR                    = 00000019
MCHK$M_NEXM                     = 00000004                   PR790$_PAMACC                  = 00000040
MMG$GL_SBICONF                    ********    X    0A        PR790$_PAMLOC                  = 00000041
MMG$GL_SPTBASE                    ********    X    0A        PR790$_TODR                    = 0000001B
```

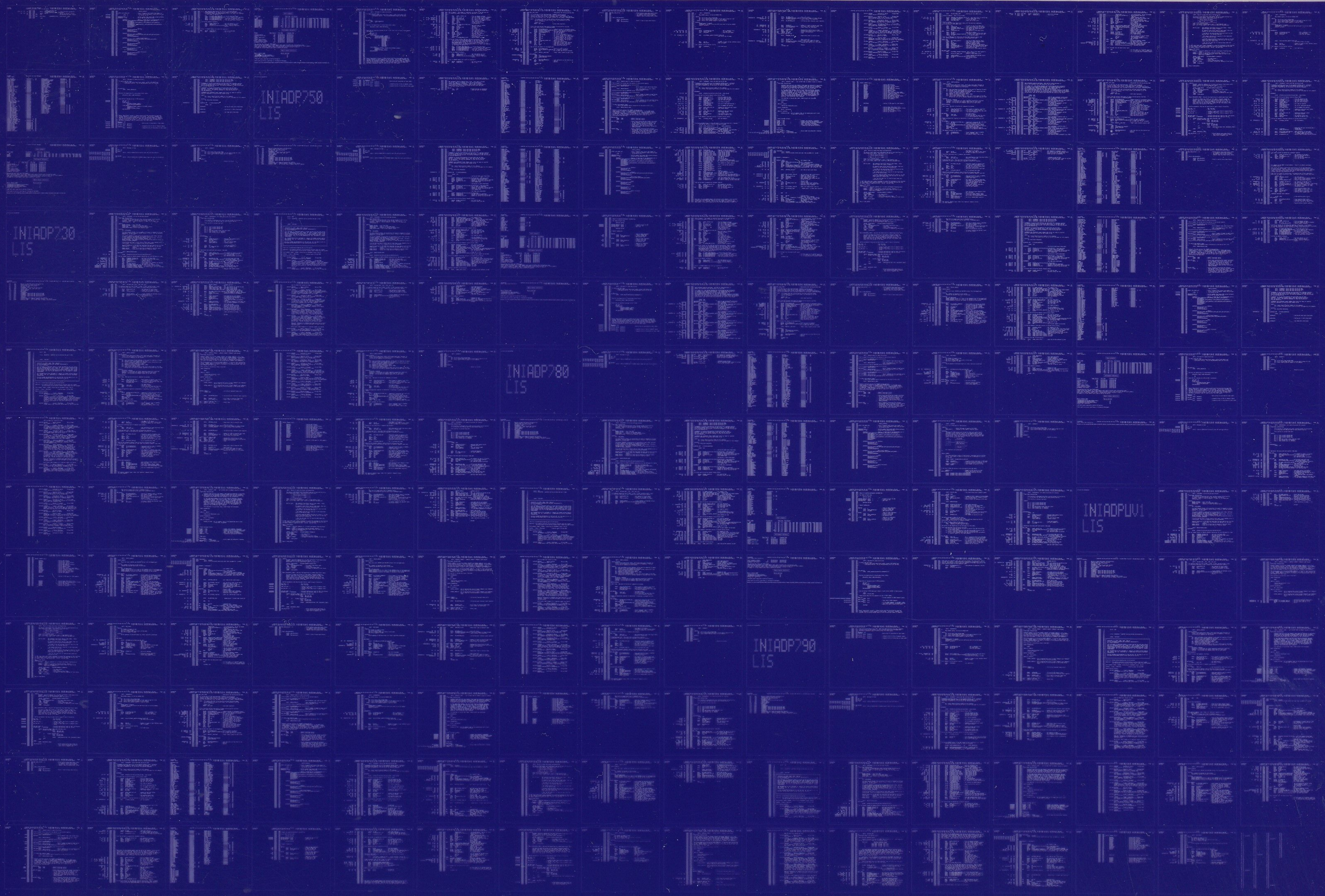
```
PTE$C_KW                    = 10000000                      UBA$L_BRRVR                 = 00000030
PTE$M_VALID                 = 80000000                      UBA$L_CR                    = 00000004
READTIME                      000007B2 R      0A            UBA$UNEXINT                   ********    X   0A
RPB$B_DEVTYP                = 00000066                      UBAERRADR                   = 00000090
RPB$L_ADPPHY                = 0000005C                      UBAINT4                     = 00000000
RPB$L_ADPVIR                = 00000060                      UBAINT4REL                  = 0000000A
RPB$L_BOOTR1                = 00000020                      UBAINT5                     = 00000020
RPB$L_CSRPHY                = 00000054                      UBAINT5REL                  = 0000002A
RPB$L_CSRVIR                = 00000058                      UBAINT6                     = 00000040
RPB$L_IOVEC                 = 00000034                      UBAINT6REL                  = 0000004A
RPB$W_ROUBVEC               = 0000001E                      UBAINT7                     = 00000060
SBIA$C_CR                     00000000                      UBAINT7REL                  = 0000006A
SBIA$L_CSR                    00000004                      UBAINTADP                   = 00000089
SBIA$L_DIAGNOS                0000000C                      UBAINTBASE                    00000450 R      0A
SBIA$L_DMAACA                 00000018                      UBINTSZ                     = 00000094
SBIA$L_DMAAID                 0000001C                      UCB$W_UNIT                  = 00000054
SBIA$L_DMABCA                 00000020                      VA$M_SYSTEM                 = 80000000
SBIA$L_DMABID                 00000024                      VEC$C_ADP                   = 00000014
SBIA$L_DMACCA                 00000028                      VEC$L_IDB                   = 00000008
SBIA$L_DMACID                 0000002C                      VEC$L_INITIAL               = 0000000C
SBIA$L_DMAICA                 00000010                      VECTAB                        000004E4 R      0A
SBIA$L_DMAIID                 00000014
SBIA$L_MAINT                  00000044
SBIA$L_SBIERR                 00000034
SBIA$L_SBIQC                  0000004C
SBIA$L_SBISILO                00000030
SBIA$L_SBISTS                 0000003C
SBIA$L_SILOCMP                00000040
SBIA$L_SUMRY                  00000008
SBIA$L_TMOADDRS               00000038
SBIA$L_UNJAM                  00000048
SBIA$S_TYPE                 = 00000004
SBIA$V_TYPE                 = 00000004
SBICONF                       000000E4 R      08
SBI_BUS_CODE                = 00000000
SBI_CPU                     = 00000001
SBI_CSR_LEN                 = 00000001
SBI_LIKE                    = 00000001
SGN$GW_TPWAIT                 ********    X   0A
STAY_HEADER                   00000000 R      0B
SW_BUS_CODE                   00000005 R      08
SYS$ASSIGN                    ********   GX   0A
SYS$BINTIM                    ********   GX   0A
SYS$DASSGN                    ********   GX   0A
SYS$QIOW                      ********   GX   0A
SYSL$BEGIN                    ********    X   0A
SYSL$END                      ********    X   0B
TERM_NAMADR                   00000730 R      0A
TERM_NAMSIZ                 = 00000004
TEST_NEXUS                    00000111 R      0A
TIMEPROMPT                    00000746 R      0A
TIMERR                        00000734 R      0A
TMPDESC                     = 0000000C
TTCHAN                      = 00000000
TTNAME                      = 00000004
UBA$INITIAL                   ********    X   0A
UBA$INT0                      ********    X   0A
```

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

PSECT name                      Allocation            PSECT No.  Attributes
----------                      ----------            ---------  ----------
.  ABS  .                       00000000  (     0.)   00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000050  (    80.)   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
$$$INIT$DATA                    0000033F  (   831.)   08 (  8.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
SYSLOA                          000000A6  (   166.)   09 (  9.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
$$$INIT$CODE                    000008EB  (  2283.)   0A ( 10.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC QUAD
$$$INIT__END                    00000016  (    22.)   0B ( 11.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC PAGE

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

Phase                      Page faults    CPU Time        Elapsed Time
-----                      -----------    --------        ------------
Initialization                    36      00:00:00.05     00:00:00.88
Command processing               128      00:00:00.60     00:00:03.78
Pass 1                           584      00:00:15.14     00:00:56.71
Symbol table sort                  0      00:00:01.92     00:00:07.68
Pass 2                           342      00:00:04.34     00:00:18.34
Symbol table output               36      00:00:00.19     00:00:00.60
Psect synopsis output              4      00:00:00.03     00:00:00.04
Cross-reference output             0      00:00:00.00     00:00:00.00
Assembler run totals            1132      00:00:22.27     00:01:28.03

The working set limit was 2400 pages.
149851 bytes (293 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1837 non-local and 38 local symbols.
2546 source lines were read in Pass 1, producing 43 object records in Pass 2.
50 pages of virtual memory were used to define 48 macros.

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

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SYSLOA.OBJ]790DEF.MLB;1            2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                 22
_$255$DUA28:[SYSLIB]STARLET.MLB;2              15
TOTALS (all libraries)                         39

1979 GETS were required to define 39 macros.

There were no errors, warnings or information messages.
```

```
MACRO/LIS=LIS$:INIADP790/OBJ=OBJ$:INIADP790 MSRC$:CPUSW790/UPDATE=(ENH$:CPUSW790)+MSRC$:INIADP/UPDATE=(ENH$:INIADP)+EXECML$/LIB+LIB$
```

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



INIADP750
LIS

INIADP730
LIS

INIADP780
LIS

INIADPUV1
LIS

INIADP790
LIS