L 5

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

```
IIIIII    NN       NN    IIIIII      AAAAAA    DDDDDDDD    PPPPPPPP    77777777      333333      000000
IIIIII    NN       NN    IIIIII      AAAAAA    DDDDDDDD    PPPPPPPP    77777777      333333      000000
  II      NN       NN      II       AA    AA   DD     DD   PP     PP          77  33     33  00    00
  II      NN       NN      II       AA    AA   DD     DD   PP     PP          77  33     33  00    00
  II      NNNN     NN      II       AA    AA   DD     DD   PP     PP          77          33  00  0000
  II      NNNN     NN      II       AA    AA   DD     DD   PP     PP          77          33  00  0000
  II      NN NN    NN      II       AA    AA   DD     DD   PPPPPPPP        77          33  00 00 00
  II      NN NN    NN      II       AA    AA   DD     DD   PPPPPPPP        77          33  00 00 00
  II      NN  NNNN NN      II       AAAAAAAAAA DD     DD   PP            77          33  0000  00
  II      NN  NNNN NN      II       AAAAAAAAAA DD     DD   PP            77          33  0000  00
  II      NN       NN      II       AA    AA   DD     DD   PP          77      33     33  00    00
  II      NN       NN      II       AA    AA   DD     DD   PP          77      33     33  00    00
IIIIII    NN       NN    IIIIII     AA    AA   DDDDDDDD    PP        77          333333      000000
IIIIII    NN       NN    IIIIII     AA    AA   DDDDDDDD    PP        77          333333      000000
```

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

Table of contents

```
0000     1          .NLIST  CND
0000     5
0000     9
0000    11          .TITLE  INIADP730 - ADAPTER INITIALIZATION FOR VAX 11/730
0000    13
0000    17
0000    21
0000    25
0000    26          .IDENT  'V04-002'
0000    27  ;
0000    28  ;*****************************************************************************
0000    29  ;*                                                                           *
0000    30  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    31  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    32  ;*  ALL RIGHTS RESERVED.                                                     *
0000    33  ;*                                                                           *
0000    34  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    35  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    36  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    37  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    38  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    39  ;*  TRANSFERRED.                                                             *
0000    40  ;*                                                                           *
0000    41  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    42  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    43  ;*  CORPORATION.                                                             *
0000    44  ;*                                                                           *
0000    45  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    46  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    47  ;*                                                                           *
0000    48  ;*                                                                           *
0000    49  ;*****************************************************************************
0000    50  ;
0000    51  ; Facility: System bootstrapping and initialization
0000    52  ;
0000    53  ; Abstract: This module contains initialization routines that are loaded
0000    54  ;           during system initialization (rather than linked into the system).
0000    55  ;
0000    56  ; Environment: Mode = KERNEL, Executing on INTERRUPT stack, IPL=31
0000    57  ;
0000    58  ; Author:  Trudy C. Matthews            Creation date: 22-Jan-1981
0000    59  ;
0000    60  ; Modification history:
0000    61  ;
0000    62  ;       V04-002 TCM0013         Trudy C. Matthews       10-Sep-1984
0000    63  ;               Add $BQODEF missing from TCM0012.
0000    64  ;
0000    65  ;       V04-001 TCM0012         Trudy C. Matthews       07-Sep-1984
0000    66  ;               For venus processor:  turn on cache before calibrating
0000    67  ;               TIMEDWAIT cells (routine EXE$INI_TIMWAIT).  Store the TIMEDWAIT
0000    68  ;               values calculated after cache is enabled in the boot driver's
0000    69  ;               TIMEDWAIT cells.  This is because the boot driver initially
0000    70  ;               has to run with cache off, but after booting will run with
0000    71  ;               cache on.
0000    72  ;
0000    73  ;       V03-024 TCM0011         Trudy C. Matthews       31-Jul-1984
0000    74  ;               Change venus's CRD interrupt vector back to ^X54 in the SCB.
```

```
0000     75 ;                  and its SBIA Fail vector to ^X64.
0C00     76 ;
0000     77 ;        V03-023 WMC0001         Wayne Cardoza           30-Jul-1984
0000     78 ;                Add H memory to 780 list.
0000     79 ;
0000     80 ;        V03-022 TCM0010         Trudy C. Matthews       25-Jul-1984
0000     81 ;                Fix a bug in INI$UBSPACE for the 11/790 that caused second
0000     82 ;                and subsequent unibus adapter spaces to be mapped incorrectly.
0000     83 ;                Fix bugs in INI$SCB for the 11/790.  Fix conditional
0000     84 ;                assembly flags in INI$CONSOLE for the 11/790.
0000     85 ;
0000     86 ;        V03-021 KDM0100         Kathleen D. Morse       01-May-1984
0000     87 ;                Correct address of memory CSRs to be past the 8 missing
0000     88 ;                Qbus adapter pages that do not exist.
0000     89 ;
0000     90 ;        V03-020 KDM0099         Kathleen D. Morse       27-Apr-1984
0000     91 ;                On a MicroVAX I, if the sysgen parameter TIMEDWAIT is set
0000     92 ;                to request no time-prompting, then use the last recorded
0000     93 ;                system time instead.  This is found in EXE$GQ_TODCBASE
0000     94 ;                which can be updated with a SET TIME command.
0000     95 ;
0000     96 ;        V03-019 RLRSCORPIO      Robert L. Rappaport     16-Mar-1984
0000     97 ;                Begin additions (to INI$IOMAP) for Scorpio support.
0000     98 ;                Also move ADAPDESC to SYSMAR.MAR, changing it to remove
0000     99 ;                the ADAP_GENERAL array.
0000    100 ;
0000    101 ;        V03-018 RLRINIADP       Robert Rappaport        28-Feb-1984
0000    102 ;                Add refinements to previous update that introduces
0000    103 ;                longword array CONFREG.  Mainly add logic to allow for
0000    104 ;                independently assembled invocations of ADAPDESC macro
0000    105 ;                to be linked into this code.  This provides possible
0000    106 ;                support of BI as a public bus, with user defined nodes.
0000    107 ;
0000    108 ;        V03-017 KPL0100         Peter Lieberwirth       30-Jan-1984
0000    109 ;                Implement first step towards a longword-array CONFREG to
0000    110 ;                replace current byte array CONFREG.  INIADP will construct
0000    111 ;                two confregs, CONFREG and CONFREGL.  CONFREGL will be
0000    112 ;                a longword array.  The high byte will be a VMS-bus
0000    113 ;                designation, and the low word will contain the 16-bit
0000    114 ;                device type.  The BI introduces 16 bit device types.
0000    115 ;
0000    116 ;                When all references to CONFREG have been modified to touch
0000    117 ;                CONFREGL, INIADP will be modified again to stop creating
0000    118 ;                the byte array.
0000    119 ;
0000    120 ;                While here, map 9 pages of CI register space, up from 8.
0000    121 ;
0000    122 ;        V03-016 KPL0001         Peter Lieberwirth       17-Jan-1984
0000    123 ;                Fix bug in V03-015 that caused a failure to boot on 750s.
0000    124 ;                Specifically, add NDT$_MEM1664NI to ADAPDESC macro.
0000    125 ;
0000    126 ;        V03-015 TCM0009         Trudy C. Matthews       12-Dec-1983
0000    127 ;                Add support for booting from VENUS console device to
0000    128 ;                INI$CONSOLE.  When mapping I/O space on VENUS, use the
0000    129 ;                PAMM to determine if any adaptors are present on the
0000    130 ;                ABUS.
0000    131 ;
```

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

```
                        ;
                        ; MACRO LIBRARY CALLS
                        ;
                                $ADPDEF                 ; Define ADP offsets.
                                $BIICDEF                ; Define BIIC offsets.
                                $BOOTDEF                ; Define boot vector offsets.
                                $BTDDEF                 ; Define boot devices
                                $BUADEF                 ; Define BUA Register offsets.
                                $CRBDEF                 ; Define CRB offsets.
                                $DCDEF                  ; Define adapter types
                                $DDBDEF                 ; Define DDB offsets
                                $DYNDEF                 ; Define data structure type codes.
                                $IDBDEF                 ; Define interrupt dispatcher offsets.
                                $IO730DEF               ; Define 11/730 I/O space.
                                $MCHKDEF                ; Define machine check masks.
                                $NDTDEF                 ; Define nexus device types.
                                $PRDEF                  ; Define IPR numbers.


                                $PR730DEF               ; Define 11/730 specific IPR numbers.


                                $PTEDEF                 ; Define Page Table Entry bits.
                                $RPBDEF                 ; Define Restart Parameter Block fields.
                                $UBADEF                 ; Define UBA register offsets.
                                $UCBDEF                 ; Define UCB offsets.
                                $VADEF                  ; Define virtual address fields.
                                $VECDEF                 ; Define vec offsets.
```

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

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

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

```
                        .SBTTL  Adapter-specific data structures
                ;
                ; Put a symbol for arrays built by macros in the correct psects.
                ;
                ;****************** ADAPTERS array *************
00000000                .PSECT  $$$INIT$DATA0
0000            ADAPTERS:                               ; Build adapter type code arrays here.
                
00000000                .PSECT  $$$INIT$DATA1           ; User contributions in this .PSECT.
                                                        ; End of ADAPTERS array.
                ;****************** End of ADAPTERS array *************
                
                ;****************** NUM_PAGES array *************
00000000                .PSECT  $$$INIT$DATA2
0000            NUM_PAGES:                              ; Build "number of pages to map" array.
00000000                .PSECT  $$$INIT$DATA3           ; User contributions in this .PSECT.
                ;****************** End of NUM_PAGESarray *************
                
                ;****************** INIT_ROUTINES array *************
00000000                .PSECT  $$$INIT$DATA4
0000            INIT_ROUTINES:                          ; Build "address of init routine" array.
00000000                .PSECT  $$$INIT$DATA5           ; User contributions in this .PSECT.
                ;****************** End of INIT_ROUTINES array *************
                
                ;
                ; To add a new adapter type:
                ;       1) Add a new ADAPDESC macro invocation to the end of this list.
                ;
00000000                .PSECT  $$$INIT$DATA,LONG
                
                ;
                ; Default interupt vectors for UNIBUS system devices
                ; (This array is indexed by the RPB field RPB$B_DEVTYP, if the RPB field
                ; RPB$W_ROUBVEC is zero.  If RPB$W_ROUBVEC is not zero, then RPB$W_ROUBVEC
                ; is used and this array is not referenced at all.  RPB$W_ROUBVEC is set up
                ; by PQDRIVER.  RPB$L_BOOTRQ is set by VMB to contain the device name in
                ; ASCII, not the vector number and device type, as it does on full
                ; architecture VAX machines.
                ;
0000            BOOTVECTOR:
0088  0000                .WORD   ^X88                    ; RK06/7 Interrupt vector
0070  0002                .WORD   ^X70                    ; RL01/2 Interrupt vector
                
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
                                                        ;  the bus currently being configured.  The
                                                        ;  proper value for the bus of interest is
                                                        ;  copied here, from the current nexus
                                                        ;  descriptor table, when we enter subroutine
                                                        ;  CONFIG_IOSPACE.
                
0005            SW_BUS_CODE:                            ; Static longword containing the software
00000000  0005                .LONG   0                 ;  defined bus type, of the bus currently being
                                                        ;  configured, in the high order byte.  The
                                                        ;  proper value for the bus of current interest
                                                        ;  is copied here, from the nexus descriptor
                                                        ;  table, when we enter subroutine
```

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

```
                     000D   523          .SBTTL  CPU-specific data structures
                     000D   524 ;
                     000D   525 ; To add a new CPU type:
                     000D   526 ;       1) Create a new nexus descriptor table, using FLOAT_NEXUS and
                     000D   527 ;          FIXED_NEXUS macros.  Put an END_NEXUSDESC macro at the end.
                     000D   528 ;
                     000D   529
                     000D   552
                     000D   590
                     000D   592
                     000D   593 CPU_ADPSIZE:
                0258 000D   594          .WORD   ADP$C_UBAADPLEN
                     000F   595
                     000F   596
                     000F   597 ;
                     000F   598 ; Declare the beginning of a nexus-descriptor table.
                     000F   599 ;
                     000F   600          NEXUSDESC_TABLE LABEL=NEXUSDESC
                     0014   601
                     0014   602
                     0014   603 ;
                     0014   604 ; Describe all nexuses on an 11/730 processor.
                     0014   605 ;
            00000000 0014   606          SBI_CPU = 0
            00000000 0014   607          BI_CPU  = 0
                     0014   608          FIXED_NEXUS -
                     0014   609                  PHYSADR=I0730$AL_IOBASE, -
                     0014   610                  NEXUSTYPES=NDT$_MEM64NIL
                     001C   611          FLOAT_NEXUS -
                     001C   612                  PHYSADR=I0730$AL_IOBASE+<1*I0730$AL_PERNEX>, -
                     001C   613                  NUMNEX=15, -
                     001C   614                  PERNEX=I0730$AL_PERNEX
                     0094   615          END_NEXUSDESC
                     0098   617
                     0098   659
                     0098   660
                     0098   682
                     0098   706
                     0098   707 ;
                     0098   708 ; Nexus "descriptor" arrays -- these arrays hold the nexus-device type and
                     0098   709 ; virtual address of every adapter on the system.  The arrays, CONFREGL and
                     0098   710 ; SBICONF, are allocated enough space to hold the maximum number of adapters
                     0098   711 ; that can be attached to any CPU.  When the code discovers how many adapters
                     0098   712 ; actually exist on the system, it will allocate space from non-paged pool
                     0098   713 ; and move a permanent copy of these arrays into that space.
                     0098   714 ;
            00000040 0098   715 MAXNEXUS = 64
                     0098   716 CONFREG:                                    ; Byte array of nexus-device type codes..
            000000D8 0098   717          .BLKB   MAXNEXUS
                     00D8   718 SBICONF:
            000001D8 00D8   719          .BLKL   MAXNEXUS                   ; Longword array of VAs of adapter space.
                     01D8   720 CONFREGL:
            000002D8 01D8   721          .BLKL   MAXNEXUS                   ; Longword array of nexus-device type codes
```

```
                                                02D8   723          .SBTTL  Message strings
                                                02D8   724
                                      0000000D  02D8   725 CR = 13
                                      0000000A  02D8   726 LF = 10
                                                02D8   727 NOSPT:
2D 54 49 4E 49 43 45 58 45 25 0A 0D             02D8   728          .ASCIZ  <CR><LF>/%EXECINIT-F-Insufficient SPT entries/<CR><LF>
65 69 63 69 66 66 75 73 6E 49 2D 46             02E4
69 72 74 6E 65 20 54 50 53 20 74 6E             02F0
                  00 0A 0D 73 65                02FC
                                                0301   730 BADUMR:
2D 54 49 4E 49 43 45 58 45 25 0A 0D             0301   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             030D
74 6F 6E 20 73 65 6F 64 20 79 72 6F             0319
0D 30 20 74 61 20 74 72 61 74 73 20             0325
                        00 0A                   0331
```

```
                                0333   734          .SBTTL  INI$IOMAP, Initialize and map nexuses
                                0333   735 ;++
                                0333   736 ; FUNCTIONAL DESCRIPTION:
                                0333   737 ;       This routine is executed only once, during system initialization.
                                0333   738 ;       It loops through all nexuses on the system, testing for
                                0333   739 ;       adapters.  When it finds an adapter, it maps its I/O space and
                                0333   740 ;       initializes it.
                                0333   741 ;
                                0333   742 ; INPUTS:
                                0333   743 ;       BOO$GL_SPTFREL  - next free VPN
                                0333   744 ;       MMG$GL_SPTVASE  - base of system page table
                                0333   745 ;       EXE$GL_RPB      - address of reboot parameter block
                                0333   747 ;       RPB$L_ADPPHY(RPB) - PFN of boot adapter space
                                0333   749 ;
                                0333   750 ; OUTPUTS:
                                0333   751 ;       R0 - SS$_NORMAL
                                0333   752 ;
                                0333   753 ;       For each adapter found, its accessible I/O space is mapped to virtual
                                0333   754 ;       addresses.  An ADP (Adapter Control Block) is built, and the hardware
                                0333   755 ;       adapter is initialized.
                                0333   756 ;
                                0333   757 ;       The arrays CONFREG (a byte array of nexus-device type codes, defined
                                0333   758 ;       by NDT$_ symbols) and SBICONF (a longword array of
                                0333   759 ;       virtual addresses that map adapter space) are initialized.  Pointers
                                0333   760 ;       to these arrays are stored in EXE$GL_CONFREG  and
                                0333   761 ;       MMG$GL_SBICONF.  The number of entries in these two parallel arrays is
                                0333   762 ;       stored in EXE$GL_NUMNEXUS.
                                0333   763 ;
                                0333   764 ;       Since BI devices have a 16-bit device type code, a new CONFREG array is
                                0333   765 ;       constructed.  This is a longword array called CONFREGL.
                                0333   766 ;
                                0333   767 ;       Several locations in the RPB that describe the boot device are init'ed:
                                0333   768 ;       RPB$L_BOOTR1    - holds index into CONFREG and SBICONF for the boot
                                0333   769 ;                         adapter
                                0333   770 ;       RPB$L_ADPVIR    - holds VA of boot device adapter's register space
                                0333   771 ;       RPB$L_CSRVIR    - holds VA of boot device's register space
                                0333   772 ;--
                                0333   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 Page 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$L_ADPPHY(R9),R10; Get PFN of boot adapter space.
00000000'GF    00D8'CF          DE  0030   791          MOVAL   W^SBICONF,G^MMG$GL_SBICONF  ; Set pointers to local copies
00000000'GF    0098'CF          DE  0039   792          MOVAL   W^CONFREG,G^EXE$GL_CONFREG  ; of these arrays for init routines.
00000000'GF    01D8'CF          DE  0042   793          MOVAL   W^CONFREGL,G^EXE$GL_CONFREGL ; ...
```

```
                     004B   899          .SBTTL  INITADP_780, _750, _730, and _UV1
                     004B   900 ;
                     004B   901 ; I/O address space for the 11/780, 11/750, 11/730, and Micro-VAX I cpus
                     004B   902 ; is statically defined in their respective nexus descriptor tables.
                     004B   903 ;
          56 0014'CF DE    004B   904          MOVAL   W^NEXUSDESC,R6          ; Get address of nexus table.
                5B   D4    0050   905          CLRL    R11                     ; Signal use 1st page of SCB.
                0B   10    0052   906          BSBB    CONFIG_IOSPACE          ; Configure processor I/O space.
                           0054   907
                           0054   909
              00C3   30    0054   910          BSBW    CREATE_ARRAYS           ; Create CONFREG and SBICONF arrays.
          0FFF 8F   BA     0057   911          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          50   01   D0     005B   912          MOVL    #1,R0                   ; Set success status
                    05     005E   913          RSB                             ; Return.
```

```
                     005F   916          .SBTTL  CONFIG_IOSPACE
                     005F   917 ;
                     005F   918 ; CONFIG_IOSPACE
                     005F   919 ;       Given a nexus descriptor table, which describes what "nexuses" or
                     005F   920 ;       "slots" are available on a system to hold I/O adapters, find and
                     005F   921 ;       initialize all adapters on the system.
                     005F   922 ;
                     005F   923 ; Inputs:
                     005F   924 ;       R2 - next available virtual address, to be used for mapping I/O space
                     005F   925 ;       R3 - address of PTE associated with VA in R2
                     005F   926 ;       R4 - Current index into CONFREG and SBICONF arrays (should be 0 the
                     005F   927 ;            first time CONFIG_IOSPACE is called)
                     005F   928 ;       R6 - address of nexus descriptor table
                     005F   929 ;       R9 - address of Restart Parameter Block (RPB)
                     005F   930 ;       R10 - PFN of boot adapter space
                     005F   931 ;       R11- page offset from beginning of SCB; tells which page of the SCB
                     005F   932 ;            to use for this set of nexuses (passed to routines that init ADP)
                     005F   933 ;
                     005F   934 ; Outputs:
                     005F   935 ;       R2,R3,R4 - updated
                     005F   936 ;       R9,R10,R11 - preserved; all other registers potentially modified
                     005F   937 ;       CONFREG - initialized with adapter NDT$ code for each nexus
                     005F   938 ;       SBICONF - initialized with adapter space VA for each nexus
                     005F   939 ;
                     005F   940 CONFIG_IOSPACE:
                     005F   942 ;
                     005F   943 ; Main loop.  Map and initialize all adapters on system.
                     005F   944 ;
                     005F   950
      FB A6   90     005F   951          MOVB    CSR_LEN_OFFSET(R6),-    ; Move length of adapter type field
0004'CF                0062   952                  W^BOS_CSR_LEN           ;  in CSR's to static location.
      FC A6   D0       0065   953          MOVL    BUS_CODE_OFFSET(R6),-   ; Move software defined bus type code
0005'CF                0068   954                  W^SW_BUS_CODE           ;  to static longword.
                       006B   955
                       006B   956 NXT_NEXUS:                              ; For each nexus...
   58   86   D0       006B   957          MOVL    (R6)+,R8                ; Get PFN of nexus.
        01   12       006E   959          BNEQ    TEST_NEXUS              ; If PFN non-zero, go test the slot.
             05       0070   960          RSB                             ; If 0, we've found all nexuses.
                       0071   961 ;
                       0071   962 ; Read configuration register to determine if anything is present at this
                       0071   963 ; nexus.
                       0071   964 ;
                       0071   965 TEST_NEXUS:
90000000 8F     C9     0071   966          BISL3   #PTE$M_VALID!PTE$C_KW,- ; Temporarily associate VA in R2 with
   63   58             0077   967                  R8,(R3)                 ; PFN in R8 via SPTE in R3.
                       0079   968          $PRTCTINI B^10$, -              ; Protect following code from non-
                       0079   969                  #<MCHK$M_NEXM!MCHK$M_LOG>; existent memory machine checks.
   51   62   D0       0085   970          MOVL    (R2),R1                 ; Read adapter configuration register.
                       0088   971          $PRTCTEND 10$                   ; End of protected code.
                       0089   972          INVALID R2                      ; Clear TB of temporary mapping.
     11 50   E8       008C   973          BLBS    R0,GET_TYPE             ; Branch if no machine check occurred.
                       008F   974 ;
                       008F   975 ; No adapter present at this nexus.
                       008F   976 ;
0098'CF44   94         008F   977          CLRB    W^CONFREG[R4]           ; Store "unknown" type in CONFREG
01D8'CF44   D4         0094   978          CLRL    W^CONFREGL[R4]          ; and in CONFREGL also.
        55  D4         0099   979          CLRL    R5                      ; Use general memory type to map
```

```
                        009B   980                                                    ; one page of I/O space.
         56   04   C0   009B   981          ADDL2   #4,R6                             ; Step past type code in nexus table.
              59   11   009E   982          BRB     MAP_NEXUS                         ; Go map I/O space for this nexus.
                        00A0   984 ;
                        00A0   985 ; Execution continues here if adapter was present.
                        00A0   986 ;
                        00A0   987 GET_TYPE:
         57   86   D0   00A0   988          MOVL    (R6)+,R7                          ; Get nexus-device type from nexus table.
              14   12   00A3   990          BNEQ    GET_GEN_TYPE                      ; Branch if fixed slot.
                        00A5   991 ;
                        00A5   992 ; Floating-type slot.  Use type from configuration register.
                        00A5   993 ; Determine if type in configuration register is 8-bits or 16-bits.
                        00A5   994 ;
                        00A5   995
    0004'CF   01   91   00A5   996          CMPB    #1,W^BUS_CSR_LEN                  ; Determine length of adapter type
                        00AA   997                                                    ;  field in CSR contained in R7.
              05   13   00AA   998          BEQL    10$                               ; EQL implies 1 byte (8-bit) field.
         57   51   3C   00AC   999          MOVZWL  R1,R7                             ; BI_LIKE, so use word instruction.
              03   11   00AF  1000          BRB     20$                               ; Skip byte instruction.
         57   51   9A   00B1  1001 10$:     MOVZBL  R1,R7                             ; Use byte instruction to get type.
                        00B4  1002 20$:                                               ;
    57    0005'CF  C8   00B4  1003          BISL    W^SW_BUS_CODE,R7                  ; Or in software bus code.
                        00B9  1005 ;
                        00B9  1006 ; Here R7 has hardware adapter code or'ed with software bus code.
                        00B9  1007 ; Translate specific nexus device type code into general adapter type code.
                        00B9  1008 ;
                        00B9  1009 GET_GEN_TYPE:
  0098'CF44   57   90   00B9  1010          MOVB    R7,W^CONFREG[R4]                  ; Save nexus-device type in CONFREG.
  01D8'CF44   57   D0   00BF  1011          MOVL    R7,W^CONFREGL[R4]                 ; CONFREGL also filled in.
              55   D4   00C5  1012          CLRL    R5                                ; Clear loop index.
                        00C7  1013 30$:                                               ;
  50    0000'CF45  DE   00C7  1014          MOVAL   W^ADAPTERS[R5],R0                 ; Get address of adapter type code.
          0000'CF  9F   00CD  1015          PUSHAB  W^NUM_PAGES                       ; Push addr of end of ADAPTERS array.
         8E   50   D1   00D1  1016          CMPL    R0,(SP)+                          ; See if we went beyond array.
              3F   1E   00D4  1017          BGEQU   END_NEXUS                         ; unrecognized adapter, do not map.
         60   57   D1   00D6  1018          CMPL    R7,(R0)                           ; Adapter type match?
              04   13   00D9  1019          BEQL    40$                               ; If EQL yes, adapter type match.
              55   D6   00DB  1020          INCL    R5                                ; Increment loop index.
              E8   11   00DD  1021          BRB     30$                               ; Look at next adapter.
                        00DF  1022 40$:
                        00DF  1023
                        00DF  1024 ;
                        00DF  1025 ; Store boot parameters.
                        00DF  1026 ;
         5A   58   D1   00DF  1028          CMPL    R8,R10                            ; Does PFN match boot adapter's PFN?
              15   12   00E2  1029          BNEQ    MAP_NEXUS                         ; No; continue.
      60 A9   52   D0   00E4  1031          MOVL    R2,RPB$L_ADPVIR(R9)               ; Store VA of boot adapter space.
      20 A9   54   D0   00E8  1032          MOVL    R4,RPB$L_BOOTR1(R9)               ; Store boot adapter nexus number.
  51   54 A9   0D   00 EF   00EC  1033      EXTZV   #0,#13, -                         ; Get offset into UNIBUS/QBUS I/O page.
                        00F2  1034                  RPB$L_CSRPHY(R9),R1               ;
   58 A9   1000 C241   9E   00F2  1035      MOVAB   <8*512>(R2)[R1], -                ; Set VA of UNIBUS/QBUS registers.
                        00F9  1036                  RPB$L_CSRVIR(R9)                  ;
                        00F9  1037
                        00F9  1038
                        00F9  1039 ;
                        00F9  1040 ; R5/ general adapter type; index into "general" adapter arrays.
                        00F9  1041 ; For each adapter -
```

```
                                00F9  1042 ;       Map the # of pages specified in ADAPDESC macro
                                00F9  1043 ;       JSB to initialization routine specified in ADAPDESC macro
                                00F9  1044 ;
                                00F9  1045 MAP_NEXUS:
00D8'CF44    52    D0           00F9  1050         MOVL    R2,W^SBICONF[R4]          ; Save VA of adapter space in SBICONF.
51    0000'CF45    3C           00FF  1051         MOVZWL  W^NUM_PAGES[R5],R1        ; Get number of pages to map.
                6C    10        0105  1052         BSBB    MAP_PAGES                 ; Map the I/O pages.
51    0000'CF45    DE           0107  1053         MOVAL   W^INIT_ROUTINES[R5],R1    ; Get address of initialization routine.
                61    D5        010D  1054         TSTL    (R1)                      ; Initialization routine specified?
                04    13        010F  1055         BEQL    END_NEXUS                 ; Branch if none.
          00 B141    16          0111  1056         JSB     @(R1)[R1]                 ; Call initialization routine.
                                0115  1057 END_NEXUS:
                54    D6        0115  1058         INCL    R4                        ; Increment CONFREG and SBICONF index.
              FF51    31        0117  1060         BRW     NXT_NEXUS                 ; Go do next nexus.
                                011A  1064
```

```
                                011A  1066          .SBTTL  CREATE_ARRAYS
                                011A  1067 ;
                                011A  1068 ; CREATE_ARRAYS
                                011A  1069 ;
                                011A  1070 ;       Move the local CONFREG and SBICONF arrays into non-paged pool.
                                011A  1071 ;
                                011A  1072 ; Inputs:
                                011A  1073 ;       R4 - Number of nexuses on the system.
                                011A  1074 ;       CONFREG and SBICONF have been initialized.
                                011A  1075 ;
                                011A  1076 ; Outputs:
                                011A  1077 ;       R0 - R5 destroyed
                                011A  1078 ;       EXE$GL_CONFREG points to a copy of the CONFREG array in non-paged pool
                                011A  1079 ;       MMG$GL_SBICONF points to a copy of the SBICONF array in non-paged pool
                                011A  1080 ;       EXE$GL_NUMNEXUS contains the number of nexuses on the system
                                011A  1081 ;
                                011A  1082 ;
                                011A  1083 CREATE_ARRAYS:
  00000000'GF    54    D0      011A  1084          MOVL    R4,G^EXE$GL_NUMNEXUS    ; Store number of nexuses on system.
      51    0C A444    DE      0121  1085          MOVAL   12(R4)[R4],R1           ; Allocate n bytes for CONFREG plus
                                0126  1086                                          ; 4n bytes for SBICONF + header
          51    6144    DE      0126  1087          MOVAL   (R1)[R4],R1             ; Another 4n bytes for CONFREGL.
                01A8    30      012A  1088          BSBW    ALONPAGD                ; Get pool for CONFREG and SBICONF.
                  82    7C      012D  1089          CLRQ    (R2)+                   ; Clear out unused
              82    51    B0      012F  1090          MOVW    R1,(R2)+                ; Set in size
       82    0763 8F    B0      0132  1091          MOVW    #<DYN$C_CONF@8>!DYN$C_INIT,(R2)+ ; Set type and subtype
  00000000'GF    62    9E      0137  1092          MOVAB   (R2),G^EXE$GL_CONFREG   ; Store address of system CONFREG.
          51    6244    9E      013E  1093          MOVAB   (R2)[R4],R1             ; Two steps to CONFREGL, 1st, SBICONF,
  00000000'GF    51    D0      0142  1094          MOVL    R1,G^MMG$GL_SBICONF     ; Store address of system SBICONF.
00000000'GF    6144    DE      0149  1095          MOVAL   (R1)[R4],G^EXE$GL_CONFREGL ; And address of system CONFREGL.
                  14    BB      0151  1096          PUSHR   #^M<R2,R4>              ; Save pool address and nexus count.
 62    0098'CF    54    28      0153  1097          MOVC3   R4,W^CONFREG,(R2)       ; Copy CONFREG to pool.
                  14    BA      0159  1098          POPR    #^M<R2,R4>              ; Retrieve pool address and nexus count.
          51    54    04      C5    015B  1099          MULL3   #4,R4,R1                ; Number of bytes in SBICONF.
                7E    51      D0    015F  1100          MOVL    R1,-(SP)                ; Save, SBICONF size = CONFREGL size
 6244    00D8'CF    51      28    0162  1101          MOVC3   R1,W^SBICONF,(R2)[R4]   ; Copy SBICONF to pool.
                51    8E      D0    0169  1102          MOVL    (SP)+,R1                ; Restore size of SBICONF and CONFREGL.
  63    01D8'CF    51      28    016C  1103          MOVC3   R1,W^CONFREGL,(R3)      ; Copy CONFREGL to pool.  R3 is output
                                0172  1104                                          ; from SBICONF MOVC3, so SBICONF and
                                0172  1105                                          ; CONFREGL must be adjacent.
                                0172  1106
                          05    0172  1107          RSB
```

```
                                0173  1109          .SBTTL  MAP_PAGES
                                0173  1110 ;++
                                0173  1111 ; INPUTS:
                                0173  1112 ;       R1/ Number of pages to map.
                                0173  1113 ;       R2/ VA of page to map.
                                0173  1114 ;       R3/ VA of system page table entry to be used.
                                0173  1115 ;       R8/ PFN of page(s) to map.
                                0173  1116 ;
                                0173  1117 ; OUTPUTS:
                                0173  1118 ;       R2,R3 updated; R1,R8 destroyed; all other registers preserved
                                0173  1119 ;
                                0173  1120 ;--
                                0173  1121
                                0173  1122 MAP_PAGES:
                                0173  1123
      83    58    90000000 8F   C9  0173  1124          BISL3   #<PTE$M_VALID!PTE$C_KW>,R8,(R3)+
                                017B  1125                                          ; Map a page.
                          58    D6  017B  1126          INCL    R8                  ; Next PFN.
                    52    0200 C2   9E  017D  1127          MOVAB   512(R2),R2          ; Next VA.
                     00000000'GF    D6  0182  1128          INCL    G^BOO$GL_SPTFREL    ; Next free entry.
    00000000'GF      00000000'GF    D1  0188  1129          CMPL    G^BOO$GL_SPTFREH, - ; Check for no more system page
                                0193  1130                  G^BOO$GL_SPTFREL    ; table entries.
                          04    15  0193  1131          BLEQ    ERROR_HALT          ; Branch if out of SPTEs.
                       DB 51    F5  0195  1132          SOBGTR  R1,MAP_PAGES        ; Map another page.
                                05  0198  1133          RSB                         ; All done.
                                0199  1134
                                0199  1135 ERROR_HALT:
                    51    02D8'CF   9E  0199  1136          MOVAB   W^NOSPT,R1          ; Set error message.
                                019E  1137 ERROR_HALT_1:
                          5B    D4  019E  1138          CLRL    R11                 ; Indicate console terminal.
                     00000000'GF    16  01A0  1139          JSB     G^EXE$OUTZSTRING    ; Output error message.
                                00  01A6  1140          HALT                        ; ***** FATAL ERROR *******
```

```
                                01A7  1269          .SBTTL  INI$UBSPACE
                                01A7  1270 ;++
                                01A7  1271 ;       Map UNIBUS space; initialize UNIBUS ADP.
                                01A7  1272 ;
                                01A7  1273 ; INPUTS:
                                01A7  1274 ;       R2 - VA of next free system page
                                01A7  1275 ;       R3 - VA of system page table entry to be used to map VA in R2
                                01A7  1276 ;       R4 - nexus identification number of this adapter
                                01A7  1277 ;     -8(R6) - PFN of this UNIBUS adapter's register space
                                01A7  1278 ;
                                01A7  1279 ; OUTPUTS:
                                01A7  1280 ;       UNIBUS space is mapped.
                                01A7  1281 ;       INI$UBADP is called to build an ADP block and initialize UNIBUS
                                01A7  1282 ;       adapter hardware.
                                01A7  1283 ;
                                01A7  1284 ;--
                                01A7  1285
                                01A7  1286 INI$UBSPACE:
                                01A7  1287
      58    01D8'CF44    DE    01A7  1290          MOVAL   W^CONFREGL[R4],R8               ; R8 => CONFREGL slot.
   58   68    02    00    EF    01AD  1291          EXTZV   #0,#2,(R8),R8                   ; Get UBA number.
      58    58    09    78    01B2  1292          ASHL    #9,R8,R8                        ; Position UB number.
                                01B6  1295
                                01B6  1304
                                01B6  1309
                                01B6  1314
   58   00007FF0 8F    58    C3    01B6  1316          SUBL3   R8,#<I0730$AL_UB0SP+^0760000/^X200>,R8
                                01BE  1317                                              ; Get PFN of UB I/O page.
                                01BE  1319
                                01BE  1325
                                01BE  1330
   51    10    D0    01BE  1331          MOVL    #16,R1                          ; Number of pages to map (UB/Qbus space).
      FFAF    30    01C1  1332          BSBW    MAP_PAGES                       ; Map I/O pages.
                                01C4  1333 ;
                                01C4  1334 ; Call adapter initialization routine.
                                01C4  1335 ;
                                01C4  1336 ;       BSBW    INI$UBADP               ; Init ADP block.
                                01C4  1337 ;       RSB
```

```
                                01C4  1339          .SBTTL  INI$UBADP - BUILD ADP AND INITIALIZE UBA
                                01C4  1340 ;+
                                01C4  1341 ; INI$UBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
                                01C4  1342 ; DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS.  A CALL IS
                                01C4  1343 ; THEN MADE TO UBA$INITIAL TO INITIALIZE THE ADAPTER HARDWARE.
                                01C4  1344 ;
                                01C4  1345 ; INPUT:
                                01C4  1346 ;       R4 - nexus identification number of this adapter
                                01C4  1347 ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                                01C4  1348 ;-
                                01C4  1349
                                01C4  1350 INI$UBADP:
                                01C4  1351
          01FF 8F    BB         01C4  1352          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; SAVE R0-R8
                                01C8  1353 ;
                                01C8  1354 ; Allocate and initialize Adapter Control Block (ADP).
                                01C8  1355 ;
    51   000D'CF    3C         01C8  1356          MOVZWL  W^CPU_ADPSIZE,R1        ; PICK UP LENGTH OF ADP
            0105    30         01CD  1357          BSBW    ALONPAGD                ; ALLOCATE SPACE FOR ADP
       08 A2   51   B0         01D0  1358          MOVW    R1,ADP$W_SIZE(R2)       ; SET SIZE INTO ADP BLOCK
       0A A2   01   90         01D4  1359          MOVB    #DYN$C_ADP,-            ; AND SET TYPE OF BLOCK
                               01D8  1360                  ADP$B_TYPE(R2)
       0E A2   01   B0         01D8  1361          MOVW    #ATA_UBA,-              ; SET TYPE OF ADAPTER
                               01DC  1362                  ADP$W_ADPTYPE(R2)
    62   00D8'CF44   D0         01DC  1363          MOVL    W^SBICONF[R4],-         ; SET VA OF CONFIGURATION REG
                               01E2  1364                  ADP$L_CSR(R2)
       0C A2   54   B0         01E2  1365          MOVW    R4,ADP$W_TR(R2)         ; SET TR NUMBER FOR ADAPTER
                               01E6  1366
       50   14 A2   DE         01E6  1367          MOVAL   ADP$L_DPQFL(R2),R0      ; ADDRESS OF DATA PATH WAIT QUEUE
          60   50   D0         01EA  1368          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
       04 A0   50   D0         01ED  1369          MOVL    R0,4(R0)                ;
                               01F1  1370
       50   30 A2   DE         01F1  1371          MOVAL   ADP$L_MRQFL(R2),R0      ; ADDRESS OF MAP WAIT QUEUE
          60   50   D0         01F5  1372          MOVL    R0,(R0)                 ; INIT QUEUE HEADER
       04 A0   50   D0         01F8  1373          MOVL    R0,4(R0)                ;
             04 A2   D4         01FC  1374          CLRL    ADP$L_LINK(R2)          ; ZAP ADAPTER CHAIN LINK
              FDFE'  30         01FF  1375          BSBW    ADPLINK                 ; LINK ADP TO END OF LIST
                               0202  1376 ;
                               0202  1377 ; Initialize adapter interrupt vectors in System Control Block.
                               0202  1378 ;
 58   00000000'GF   D0         0202  1379          MOVL    G^EXE$GL_SCB,R8         ; GET SCB ADDRESS
                               0209  1380
                               0209  1387
                               0209  1447
                               0209  1507
                               0209  1508
                               0209  1510                                          ; REMAINING ADP INIT FOR 11/730:
         0200 C8   DE         0209  1511          MOVAL   ^X200(R8),-             ; ASSUME UB0
           10 A2              020D  1512                  ADP$L_VECTOR(R2)        ;  VECTOR SPACE
    28   01D8'CF44   D1         020F  1513          CMPL    W^CONFREGL[R4],#NDT$_UB0; IS DEVICE TYPE = UB0?
                08   13         0215  1514          BEQL    10$                     ; BRANCH IF SO
10 A2   00000200 8F   C0         0217  1515          ADDL    #^X200,ADP$L_VECTOR(R2) ; ELSE STEP TO UB1 VECTOR SPACE
       60 A2   0E   B0         021F  1516 10$:     MOVW    #^XE,ADP$W_DPBITMAP(R2) ; MARK DATAPATHS 1-3 AVAILABLE
           53   62   D0         0223  1517          MOVL    ADP$L_CSR(R2),R3        ; VIRTUAL ADDRESS OF ADAPTER
       53   0800 C3   9E         0226  1518          MOVAB   UBA$L_MAP(R3),R3        ; POINT TO MAPPING REGISTERS
       54   01F0 8F   3C         022B  1519          MOVZWL  #496,R4                 ; NUMBER OF UMR TO DISABLE
                83   D4         0230  1520 20$:     CLRL    (R3)+                   ; DISABLE A UNIBUS MAP REGISTER
```

```
                FB 54    F5  0232  1521          SOBGTR  R4,20$                      ; LOOP THRU THEM ALL
      53  00000001'GF    DE  0235  1522          MOVAL   G^UBA$UNEXINT+1,R3          ; GET ADDR OF UNEXP INT SERVICE
                         0239  1523                                                  ;  (+1 MEANS HANDLE ON INT STACK)
          54  0001'CF    DE  023C  1524          MOVAL   W^UBA$INT0+1,R4             ; SPECIAL CASE TO COUNT PASSIVE RELEASE
                         0241  1525
                         0241  1526 ;
                         0241  1527 ; INIT UB VECTORS TO UNEXPECTED INTERRUPT SERVICE
                         0241  1528 ;
           50   10 A2    D0  0241  1529          MOVL    ADP$L_VECTOR(R2),R0         ; GET ADDRESS OF VECTORS
             80   54    D0  0245  1530          MOVL    R4,(R0)+                    ; SPECIAL CASE FOR VECTOR 0
           51   7F 8F    9A  0248  1531          MOVZBL  #<NUMUBAVEC-1>,R1           ; REST OF VECTORS
             80   53    D0  024C  1532 30$:     MOVL    R3,(R0)+                    ; FILL VECTOR WITH UNEXP INT
                FA 51    F5  024F  1533          SOBGTR  R1,30$                      ; FILL ALL VECTORS
                         0252  1534
                         0252  1536
                         0252  1537
                         0252  1558
                         0252  1559
                         0252  1601
                         0252  1602
                         0252  1651 ;
                         0252  1652 ; Now check for any UNIBUS memory that may be on the adapter. First we must
                         0252  1653 ; disable all the UNIBUS Map Registers so that there is no conflict in
                         0252  1654 ; which memory will respond.  Then we check all 248Kb of potential memory in
                         0252  1655 ; 8Kb chunks, since each disable bit on the 780 UBA represents 16 UMR's or
                         0252  1656 ; 8Kb of memory.  The number of registers is stored in the ADP and the
                         0252  1657 ; corresponding number withdrawn from the UMR map in the ADP.
                         0252  1658 ;
                         0252  1659
              56   62    D0  0252  1661          MOVL    ADP$L_CSR(R2),R6            ; Pick up adapter pointer
                51   D4    D4  0255  1662          CLRL    R1                          ; Zero out number of UMR to disable
57  08 AE    00000200    8F  C3  0257  1664          SUBL3   #512,8(SP),R7               ; R7 = VA of last page of UNIBUS
      58   0C AE    04  C3  0260  1665          SUBL3   #4,12(SP),R8                ; R8 = VA of SPTE mapping (R7)
54  20 AE    00000200    8F  C3  0265  1666          SUBL3   #512,32(SP),R4              ; R4 = PFN of first page of UNIBUS
                68   DD  026E  1667          PUSHL   (R8)                        ; Save contents of SPTE
              53   54    D0  0270  1668          MOVL    R4,R3                       ; Copy starting PFN
              55   1F    D0  0273  1669          MOVL    #31,R5                      ; 31 8Kb chunks to test
                         0276  1670 50$:     INVALID R7                          ; Invalidate TB
          90000000 8F    C9  0279  1671          BISL3   #<PTE$M_VALID!PTE$C_KW>,-
              68   54        027F  1672                   R4,(R8)                   ; Map each page of UNIBUS
              50   57    D0  0281  1673          MOVL    R7,R0                       ; Address to check
                 FD79'   30  0284  1674          BSBW    EXE$TEST_CSR                ; Validate it
                0D 50    E9  0287  1675          BLBC    R0,70$                      ; Not there
              54   53    D1  028A  1676          CMPL    R3,R4                       ; First time in?
                   04    13  028D  1677          BEQL    60$                         ; Yes, skip next test
                   51    D5  028F  1678          TSTL    R1                          ; Any registers already?
                   3A    13  0291  1679          BEQL    80$                         ; No, memory not start at 0
           51   10 A1    9E  0293  1680 60$:     MOVAB   16(R1),R1                   ; Yes, up the count
           54   10 A4    9E  0297  1681 70$:     MOVAB   16(R4),R4                   ; Map Next 8Kb (16*512)
                D8 55    F5  029B  1682          SOBGTR  R5,50$                      ; Loop until done
                   68 8ED0  029E  1683          POPL    (R8)                        ; Restore old contents of SPTE
                         02A1  1684          INVALID R7                          ; Invalidate TB
          0256 C2    51    B0  02A4  1686          MOVW    R1,ADP$W_UMR_DIS(R2)        ; Record number disabled
                         02A9  1688 ;
                         02A9  1689 ; Initialize fields for new UBA map register allocation.  Make it appear
                         02A9  1690 ;       that we have one contiguous array of 496 available map registers.
                         02A9  1691 ;       To do this we set ADP$L_MRACTMDRS to one (the number of active
```

```
                                02A9  1692 ;       map register descriptors for distinct contiguous areas),
                                02A9  1693 ;       ADP$W_MRNREGARY(0) to 496 (i.e the number of registers in this
                                02A9  1694 ;       contiguous range) and ADP$FREGARY(0) to 0 (i.e. the first register
                                02A9  1695 ;       in the range is register 0).
                                02A9  1696 ;
          5C A2    01    D0   02A9  1697         MOVL    #1,ADP$L_MRACTMDRS(R2)  ; 1 active map descriptor
64 A2   01F0 8F    51    A3   02AD  1698         SUBW3   R1,#496,ADP$W_MRNREGARY(R2);  for a range of 496 registers
        015E C2    51    B0   02B4  1710         MOVW    R1,ADP$W_MRFREGARY(R2)  ;  starting at register zero.
          62 A2    01    AE   02B9  1711         MNEGW   #1,ADP$W_MRNFENCE(R2)   ; Also init "fences" which preceed
        015C C2    01    AE   02BD  1712         MNEGW   #1,ADP$W_MRFFENCE(R2)   ;  the two descriptor arrays.
                                02C2  1713 ;
                                02C2  1714 ; Initialize adapter hardware.
                                02C2  1715 ;
             54    62    D0   02C2  1716         MOVL    ADP$L_CSR(R2),R4        ; Get CSR address to init
                FD38'    30   02C5  1717         BSBW    UBA$INITIAL             ; And initialize adapter
             01FF 8F    BA   02C8  1718         POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; Restore registers
                         05   02CC  1719         RSB                             ; Return
                                02CD  1720
                                02CD  1721 ;
                                02CD  1723 ; Error if UNIBUS memory not start at location 0
                                02CD  1724 ;
             51  0301'CF    9E   02CD  1725 80$:    MOVAB   W^BADUMR,R1             ; Set error message
                    FEC9    31   02D2  1726         BRW     ERROR_HALT_1            ; Put it out
                                02D5  1728
```

```
                     02D5  1815          .SBTTL  INI$MBADP - BUILD ADP AND INITIALIZE MBA
                     02D5  1816          .SBTTL  INI$DRADP - BUILD ADP AND INITIALIZE DR32
                     02D5  1817          .SBTTL  INI$CIADP - BUILD ADP AND INITIALIZE CI
                     02D5  1818 ;+
                     02D5  1819 ; INI$MBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
                     02D5  1820 ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED.  A CRB AND IDB ARE
                     02D5  1821 ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
                     02D5  1822 ; BY CALLING MBA$INITIAL.
                     02D5  1823 ;
                     02D5  1824 ; INI$DRADP IS CALLED AFTER MAPPING THE REGISTERS FOR THE DR32
                     02D5  1825 ; ADAPTER.  THE ADAPTER CONTROL BLOCK, CRB, AND IDB ARE ALLOCATED
                     02D5  1826 ; AND INITIALIZED.  THE ADAPTER HARDWARE IS THEN INITIALIZED BY
                     02D5  1827 ; CALLING DR$INITIAL.
                     02D5  1828 ;
                     02D5  1829 ; INI$MBADP AND INI$DRADP SHARE COMMON CODE AFTER THE TABLE OF ADAPTER
                     02D5  1830 ; SPECIFIC CONSTANTS IS SELECTED AND STORED IN R8.
                     02D5  1831 ;
                     02D5  1832 ; INPUT:
                     02D5  1833 ;       R4 - nexus identification number of this adapter
                     02D5  1834 ;       R11- offset from beginning of SCB to correct SCB page for this adapter
                     02D5  1835 ;
                     02D5  1836 ; OUTPUTS:
                     02D5  1837 ;       ALL REGISTERS PRESERVED
                     02D5  1838 ;-
                     02D5  1839
00000000'GF    17    02D5  1840 ALONPAGD:JMP    G^INI$ALONONPAGED
                     02DB  1841
                     02DB  1842          .ENABL  LSB
                     02DB  1843
                     02DB  1844 INI$DRADP:                              ; INITIALIZE DR32 DATA STRUCTURES
                     02DB  1845
                     02DB  1855
                     02DB  1856 INI$CIADP:                              ; INITIALIZE CI DATA STRUCTURES
                     02DB  1857
                     02DB  1867
                     02DB  1868 INI$MBADP:                              ; INIT MBA DATA STRUCTURES
                     02DB  1869
```

```
                                .SBTTL  INI$KDZ11
                        ;++
                        ;
                        ; INPUTS:
                        ;       R2 - VA of next free system page
                        ;       R3 - VA of system page table entry to be used to map VA in R2
                        ;       R4 - nexus identification number of this adapter
                        ;
                        ; OUTPUTS:
                        ;
                        ;--
                        
                        INI$KDZ11:
                        
05                              RSB                             ; Return to caller.
```

```
                        02DC  2031          .SBTTL  INI$CONSOLE, init data structures for console
                        02DC  2032 ;++
                        02DC  2033 ; FUNCTIONAL DESCRIPTION:
                        02DC  2034 ;
                        02DC  2035 ;       This routine is executed only once, during system initialization.
                        02DC  2036 ;       It initializes the CRB and IDB for boot/console device.
                        02DC  2037 ;
                        02DC  2038 ;       This routine is called from INIT.
                        02DC  2039 ;
                        02DC  2040 ; INPUTS:
                        02DC  2041 ;
                        02DC  2042 ;       R3 -->  DISK [CLASS] DRIVER DDB
                        02DC  2043 ;       R4 -->  DISK [CLASS] DRIVER DPT
                        02DC  2044 ;       R5 -->  DISK [CLASS] DRIVER UCB
                        02DC  2045 ;       R6 -->  RPB
                        02DC  2046 ;       R7 -->  ADP FOR EITHER A REAL DISK OR A PORT
                        02DC  2047 ;       R9 -->  PORT DRIVER DPT (IF PRESENT)
                        02DC  2048 ;       R10-->  PORT DIRVER UCB (IF PRESENT)
                        02DC  2049 ;
                        02DC  2050 ;--
                        02DC  2051
                        02DC  2052 INI$CONSOLE::
                        02DC  2053          .ENABL  LSB
                        02DC  2054
          66 A6   91    02DC  2056          CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
          40 8F         02DF  2057                  #BTD$K_CONSOLE          ; STORAGE DEVICE?
             10   12    02E1  2058          BNEQ    BLD_CRB                 ; NO
41534303 8F   D0    02E3  2059          MOVL    #^A7CSA/@8+3,-          ; YES, SET DEVICE NAME
          14 A3         02E9  2060                  DDB$T_NAME(R3)          ; COUNTED STRING
                        02EB  2062
                        02EB  2067
             57   D4    02EB  2070          CLRL    R7                      ; CLEAR ADP POINTER
          54 A5   01    B0    02ED  2071          MOVW    #1,UCB$W_UNIT(R5)       ; SET UNIT NUMBER TO 1
                  0D    11    02F1  2072          BRB     FILL_CRB
                        02F3  2075
                        02F3  2076 ;
                        02F3  2077 ; NOW BUILD THE AUXILIARY DATA BLOCKS (CRB,IDB)
                        02F3  2078 ;
                        02F3  2079 BLD_CRB:
          58    10 A7   D0    02F3  2080          MOVL    ADP$L_CRB(R7),R8        ; GET ADDRESS OF CRB IF IT EXISTS
          0E A7   01    B1    02F7  2081          CMPW    #AT$_UBA,ADP$W_ADPTYPE(R7); IS THIS A UNIBUS ADAPTER?
                  03    13    02FB  2082          BEQL    FILL_CRB                ; YES, ALLOCATE CRB
                008A    31    02FD  2083          BRW     100$                    ; NO, CRB/IDB ALREADY ALLOCATED
                        0300  2084
                        0300  2085 FILL_CRB:
          00000000'9F   16    0300  2086          JSB     @#INI$ALLOC_CRB         ; GO ALLOCATE AND SETUP CRB
24 A2     9F163FBB 8F   D0    0306  2087          MOVL    #^X9F163FBB,CRB$L_INTD(R2) ; SET PUSHR #^M<R0,...R5>
                        030E  2088                                              ;  JSB @#0 INTO INTERRUPT DISPATCH
          38 A2   57    D0    030E  2089          MOVL    R7,CRB$L_INTD+VEC$L_ADP(R2)     ; SET POINTER TO ADP
             58   52    D0    0312  2090          MOVL    R2,R8                   ; SAVE CRB POINTER
       51    0058 8F    3C    0315  2091          MOVZWL  #<IDB$C_LENGTH+<8*4>>,R1; SIZE TO ALLOCATE FOR IDB
        00000000'9F     16    031A  2092          JSB     @#INI$ALLONONPAGED      ; ALLOCATE IDB
          08 A2   51    B0    0320  2093          MOVW    R1,IDB$W_SIZE(R2)       ; SET SIZE OF IDB
          0A A2   09    90    0324  2094          MOVB    #DYN$C_IDB,IDB$B_TYPE(R2); AND STRUCTURE TYPE CODE
          2C A8   52    D0    0328  2095          MOVL    R2,CRB$L_INTD+VEC$L_IDB(R8) ; SET IDB INTO CRB
                        032C  2096
                66 A6   91    032C  2099          CMPB    RPB$B_DEVTYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
```

```
                     40 8F      032F  2100                    #BTD$K_CONSOLE          ; STORAGE DEVICE?
                        26  12  0331  2101            BNEQ    10$                     ; NO
50   000000F0 8F   00000000'9F  C1  0333  2102            ADDL3   @#EXE$GL_SCB,#^XF0,R0  ; YES, GET ADDRESS OF VECTOR IN SCB
                     80   25 A8  DE  033F  2103            MOVAL   CRB$L_INTD+1(R8),(R0)+  ; SET ADDR IN 1ST VECTOR
                     60   49 A8  DE  0343  2104            MOVAL   CRB$L_INTD2+1(R8),(R0)  ; SET ADDR IN 2ND VECTOR
          48 A8   9F163FBB 8F    D0  0347  2105            MOVL    #^X9F163FBB,CRB$L_INTD2(R8) ; STORE PUSHR #^M<R0...R5>
                                     034F  2106                                            ; JMP @# IN 2ND INT. DISPATCH
                  50 A8   52    D0  034F  2107            MOVL    R2,CRB$L_INTD2+VEC$L_IDB(R8); STORE ADDRESS OF IDB IN CRB
                  2C B8   1F    D0  0353  2108            MOVL    #PR$_CSTD, -            ; STORE IPR NUMBER OF CONSOLE INTERFACE
                                     0357  2109                    @CRB$L_INTD+8(R8)       ; REGISTER AS DEVICE CSR ADDRESS
                        31  11  0357  2110            BRB     100$
                                     0359  2113
                  62   58 A6    D0  0359  2114 10$:     MOVL    RPB$L_CSRVIR(R6), -     ; SAVE BOOT DEVICE CSR ADDRESS
                                     035D  2115                    IDB$L_CSR(R2)           ; IN INTERRUPT DISPATCH BLOCK
                        11  91  035D  2116            CMPB    #BTD$K_UDA,-            ; LOW ORDER BYTE OF ORIGINAL R0 TELLS
                     66 A6          035F  2117                    RPB$B_DEVTYP(R6)        ;  BOOT DEVICE TYPE.
                        08  12  0361  2118            BNEQ    20$                     ; IF NOT BOOTING FROM A UDA BRANCH
                                     0363  2119                                            ;  AROUND.
       00000000'9F   58 A6    D0  0363  2120            MOVL    RPB$L_CSRVIR(R6), -     ; COPY VIRTUAL ADDRESS OF UDA PORT CSR
                                     036B  2121                    @#BOO$GB_SYSTEMID       ;  TO LOW ORDER LONGWORD OF SYSTEMID
                                     036B  2122 20$:
                  14 A2   57    D0  036B  2123            MOVL    R7,IDB$L_ADP(R2)        ; POINT IDB TO ADP
                  50   1E A6    3C  036F  2124            MOVZWL  RPB$W_R0OBVEC(R6),R0    ; GET USER SPECIFIED VECTOR
                        0A  12  0373  2125            BNEQ    30$                     ; BRANCH IF VECTOR SPECIFIED
                  50   66 A6    9A  0375  2126            MOVZBL  RPB$B_DEVTYP(R6),R0     ; ELSE GET DEVICE TYPE CODE
            50    FFFE'CF40    3C  0379  2127            MOVZWL  W^BOOTVECTOR-2[R0],R0   ; GET DEFAULT INTERRUPT VECTOR
              50    10 B740    9E  037F  2128 30$:     MOVAB   @ADP$L_VECTOR(R7)[R0],R0; COMPUTE ADDRESS OF VECTOR
                  60   26 A8    9E  0384  2129            MOVAB   CRB$L_INTD+2(R8),(R0)   ; SET ADDR OF INTERRUPT VECTOR
                                     0388  2130
                        60  D7  0388  2133            DECL    (R0)                    ;  BACK TWO BYTES TO PUSHR, +1 TO
                                     038A  2136                                            ;
                                     038A  2137 100$:
                        05  038A  2138            RSB                             ; RETURN
                                     038B  2139            .DISABLE LSB
```

```
                     038B  2141          .SBTTL  EXE$INI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
                     038B  2142 ;++
                     038B  2143 ; FUNCTIONAL DESCRIPTION:
                     038B  2144 ;
                     038B  2145 ; EXE$INI_TIMWAIT initializes EXE$GL_TENUSEC and EXE$GL_UBDELAY, cells used
                     038B  2146 ; in the time-wait macros.  The first data cell, EXE$GL_TENUSEC, is the number
                     038B  2147 ; of times the following loop will be executed in ten u-seconds.  This is
                     038B  2148 ; done once here to calibrate the loop instead of reading the processor clock.
                     038B  2149 ; The resulting number is used in the system macros TIMEWAIT and TIMEDWAIT.
                     038B  2150 ;
                     038B  2151 ; The first step is to initialize EXE$GL_UBDELAY.  If the bit test instruction
                     038B  2152 ; in the TIMEWAIT macro is executed too rapidly in a loop, it can saturate the
                     038B  2153 ; Unibus.  EXE$GL_UBDELAY is used to introduce a 3 microsecond delay loop into
                     038B  2154 ; the TIMEWAIT bit test loop.
                     038B  2155 ;
                     038B  2156 ; This routine is called only once, from INIT.
                     038B  2157 ;
                     038B  2158 ; INPUT PARAMETERS:
                     038B  2159 ;
                     038B  2160 ;       NONE
                     038B  2161 ;
                     038B  2162 ; IMPLICIT INPUTS:
                     038B  2163 ;
                     038B  2164 ;       Time-of-day processor clock.
                     038B  2165 ;       Interval timers.
                     038B  2166 ;
                     038B  2167 ; OUTPUT PARAMETERS:
                     038B  2168 ;
                     038B  2169 ;       R0 - Destroyed.
                     038B  2170 ;
                     038B  2171 ; IMPLICIT OUTPUTS:
                     038B  2172 ;
                     038B  2173 ;       EXE$GL_TENUSEC - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                     038B  2174 ;                        macros loop for 10 micro-seconds.
                     038B  2175 ;
                     038B  2176 ;       EXE$GL_UBDELAY - set to appropriate value to make TIMEWAIT and TIMEDWAIT
                     038B  2177 ;                        macros loop for 3 micro-seconds in the unibus delay
                     038B  2178 ;                        loop.
                     038B  2179 ;
                     038B  2180 ;--
                     038B  2181
                     038B  2182 EXE$INI_TIMWAIT::                           ; Initialize time-wait data cells
                     038B  2184          .ENABLE LSB
                     038B  2185
                     038B  2189
                     038B  2193
          19   00   DA 038B  2195          MTPR    #0,#PR730$_NICR           ; Initialize next interval count register.
                     038E  2197
                     038E  2202
7E   00004E20 8F   D0 038E  2203          MOVL    #20000,-(SP)              ; # of times to execute timed loop.
          18   11   DA 0395  2204          MTPR    #^X11,#PR$_ICCS           ; Start clock, no interrupts.
                     0398  2205
                     0398  2206 ; * * * start of loop to time * * *
            FD 6E    F5 0398  2207 10$:     SOBGTR  (SP),10$                  ; Delay loop.
                     039B  2208 ; * * * end of loop to time * * *
                     039B  2209
                     039B  2213
```

```
                                        039B  2217
                      50    1A    DB    039B  2219          MFPR    #PR730$_ICR,R0          ; Read total time to execute loop.
                                        039E  2221
                                        039E  2225
                      18    00    DA    039E  2226          MTPR    #0,#PR$_ICCS            ; Shut off clock.
00000000'GF   0000EA60 8F    50    C7    03A1  2227          DIVL3   R0,#60000,G^EXE$GL_UBDELAY; Calculate number of times through
                 00000000'GF    D6    03AD  2228          INCL    G^EXE$GL_UBDELAY        ; loop to delay 3 microseconds.
                                        03B3  2229
                                        03B3  2233
                                        03B3  2237
                      19    00    DA    03B3  2239          MTPR    #0,#PR730$_NICR         ; Initialize next interval count register.
                                        03B6  2241
                                        03B6  2245
              50      00004E20 8F    D0    03B6  2246          MOVL    #20000,R0               ; Number of times to execute test loop
              6E      00000000'GF    D0    03BD  2247          MOVL    G^EXE$GL_UBDELAY,(SP)   ; Get delay loop iteration count.
                      18    11    DA    03C4  2248          MTPR    #^X11,#PR$_ICCS         ; Start clock, no interrupts
                                        03C7  2249
                                        03C7  2250 ; **** Start of loop to time
        000003D5'EF   8000 8F    B3    03C7  2251 20$:     BITW    #^X8000,40$             ; Random BITx instruction to time
                            03    12    03D0  2252          BNEQ    40$                     ; Random conditional branch instruction
                          FD 6E    F5    03D2  2253 30$:     SOBGTR  (SP),30$                ; Delay 3 microseconds.
                          EF 50    F5    03D5  2254 40$:     SOBGTR  R0,20$                  ; Loop
                                        03D8  2255 ; **** End of loop to time
                                        03D8  2256
                                        03D8  2260
                                        03D8  2264
                      50    1A    DB    03D8  2266          MFPR    #PR730$_ICR,R0          ; Read total time to execute loop.
                                        03DB  2268
                      18    00    DA    03DB  2272          MTPR    #0,#PR$_ICCS            ; Shut clock off
                            8E    D5    03DE  2273          TSTL    (SP)+                   ; Pop delay loop index off stack.
00000000'GF   00030D40 8F    50    C7    03E0  2274          DIVL3   R0,#200000,G^EXE$GL_TENUSEC ; Calculate number of times to
                 00000000'GF    D6    03EC  2275          INCL    G^EXE$GL_TENUSEC        ; execute the loop to kill 10 u-secs.
                                        03F2  2276
                                        03F2  2289
                                  05    03F2  2290          RSB                             ; Return
                                        03F3  2291          .DISABLE LSB
```

```
                        03F3  2299          .SBTTL  EXE$INIT_TODR  -  SET SYSTEM TIME TO CORRECT VALUE AT STARTUP
                        03F3  2300 ;++
                        03F3  2301 ; FUNCTIONAL DESCRIPTION:
                        03F3  2302 ;
                        03F3  2303 ;       EXE$INIT_TODR SOLICITS THE CORRECT TIME FROM THE OPERATOR IF NECESSARY,
                        03F3  2304 ;       CONVERTS THE ASCII RESPONSE TO BINARY FORMAT AND CALLS AN INTERNAL
                        03F3  2305 ;       ENTRY POINT OF THE $SETIME SYSTEM SERVICE TO SET THE NEW SYSTEM TIME
                        03F3  2306 ;       IN MEMORY WITHOUT MODIFYING THE CONTENTS OF THE SYSTEM DISK.
                        03F3  2307 ;
                        03F3  2308 ;       IF THE TIME WOULD NORMALLY BE SOLICITED FROM AN OPERATOR, BECAUSE
                        03F3  2309 ;       THE HARDWARE TIME OF YEAR CLOCK IS ZERO, THEN THE SYSGEN PARAMETER
                        03F3  2310 ;       "TPWAIT" IS CHECKED.  IF IT IS ZERO, THEN IT IS ASSUMED THAT NO
                        03F3  2311 ;       OPERATOR IS PRESENT AND THE SYSTEM IS BOOTED USING THE LAST TIME
                        03F3  2312 ;       RECORDED IN THE SYSTEM IMAGE.  IF THE PARAMETER IS NON ZERO THEN
                        03F3  2313 ;       THAT TIME IS USED AS THE MAXIMUM TIME TO WAIT BEFOR ASSUMING THAT
                        03F3  2314 ;       THERE IS NO OPERATOR AND BOOTING ANY WAY.  IF THE PARAMETER IS
                        03F3  2315 ;       NEGATIVE, THE SYSTEM WILL WAIT FOREVER.
                        03F3  2316 ;
                        03F3  2317 ;       THIS ROUTINE IS CALLED ONLY ONCE, FROM SYSINIT OR STASYSGEN.
                        03F3  2318 ;
                        03F3  2319 ; INPUT PARAMETERS:
                        03F3  2320 ;
                        03F3  2321 ;       NONE
                        03F3  2322 ;
                        03F3  2323 ; IMPLICIT INPUTS:
                        03F3  2324 ;
                        03F3  2325 ;       TIME-OF-DAY PROCESSOR CLOCK.
                        03F3  2326 ;
                        03F3  2327 ; OUTPUT PARAMETERS:
                        03F3  2328 ;
                        03F3  2329 ;       R0,R1 - DESTROYED
                        03F3  2330 ;
                        03F3  2331 ; IMPLICIT OUTPUTS:
                        03F3  2332 ;
                        03F3  2333 ;       EXE$GQ_SYSTIME - SET TO CURRENT TIME IN 100 NANOSECOND UNITS SINCE
                        03F3  2334 ;                        17-NOV-1858  00:00:00.
                        03F3  2335 ;
                        03F3  2336 ;--
                        03F3  2337
                        03F3  2338 ;
                        03F3  2339 ; Stack storage offsets:
                        03F3  2340 ;
                00000000  03F3  2341 TTCHAN  = ^X00                          ; CHANNEL FOR TERMINAL (LONGWORD)
                00000004  03F3  2342 TTNAME  = ^X04                          ; STRING DESCRIPTOR FOR OPERATOR'S TERM
                0000000C  03F3  2343 TMPDESC = ^X0C                          ; TEMPORY STRING DESCRIPTOR (QUADWORD)
                00000014  03F3  2344 INTIME  = ^X14                          ; INPUT TIME VALUE (QUADWORD)
                0000001C  03F3  2345 LINBUF  = ^X1C                          ; INPUT LINE BUFFER (5 LONGWORDS)
                00000014  03F3  2346 LINBUFSIZ = ^X14                        ;  (LENGTH OF LINE BUFFER IN BYTES)
                        03F3  2347
                        03F3  2348 ;
                        03F3  2349 ; PURE DATA
                        03F3  2350 ;
                        03F3  2351 TERM_NAMADR:
                  30 41 50 4F  03F3  2352          .ASCII  \OPA0\              ; DEVICE NAME FOR OPERATOR'S TERMINAL
                00000004  03F7  2353 TERM_NAMSIZ = . - TERM_NAMADR
74 61 64 20 64 69 6C 61 76 6E 69 00'  03F7  2354 TIMERR: .ASCIC  \invalid date/time\      ;
                  65 6D 69 74 2F 65    0403
```

```
                                 11  03F7
                                     0409  2355 TIMEPROMPT:
                                 33' 0409  2356         .BYTE   NPROMPT
54 4E 45 20 45 53 41 45 4C 50 0A 0D  040A  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  0416
4D 4D 4D 2D 44 44 28 20 45 4D 49 54  0422
4D 4D 3A 48 48 20 20 59 59 59 59 2D  042E
                        20 20 29     043A
                     00000033        043D  2358 NPROMPT=.-TIMEPROMPT-1
                                     043D  2359
                                     043D  2360
                                     043D  2361 EXE$INIT_TODR::                          ; SET CORRECT TIME
                                     043D  2362         .ENABLE LSB
                   077C  8F    BB    043D  2363         PUSHR   #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; SAVE REGISTERS
                   5E    30    C2    0441  2364         SUBL    #4*12,SP                    ; SCRATCH STORAGE
                   56    5E    D0    0444  2365         MOVL    SP,R6                       ; SAVE ADDRESS OF SCRATCH STORAGE
                04 A6    04    9A    0447  2366         MOVZBL  #TERM_NAMSIZ,TTNAME(R6) ; SET SIZE OF OPERATOR'S TERM NAME AND
             08 A6    FFA4 CF    9E    044B  2367         MOVAB   W^TERM_NAMADR,TTNAME+4(R6) ;  PIC ADDRESS INTO TERM NAME DESC
          1C 00000000'GF    00'  E0    0451  2368         BBS     S^#EXE$V_SETTIME,G^EXE$GL_FLAGS,READTIME ; BR TO SOLICIT TIME
                                     0459  2369
                                     0459  2370
                                     0459  2374
                                     0459  2378
                   50    1B    DB    0459  2380         MFPR    #PR730$_TODR,R0             ; GET TIME OF DAY CLOCK VALUE
                                     045C  2382
                                     045C  2386
       59    00000000'GF    50    C3    045C  2388         SUBL3   R0,G^EXE$GL_TODR,R9      ; GET TOD DELTA TIME (10 MS UNITS)
                          09    1B    0464  2389         BLEQU   5$                          ; BRANCH IF TIME IS LATER
            0083D600 8F    59    D1    0466  2390         CMPL    R9,#24*60*60*100            ; CHECK FOR SETBACK OF ONE DAY
                          06    1E    046D  2391         BGEQU   READTIME                    ; MORE, MUST SOLICIT TIME
                     14 A6    7C    046F  2396 5$:       CLRQ    INTIME(R6)                  ; NULL ARGUMENT FOR EXE$SETIME_INT
                      00C7    31    0472  2397         BRW     200$                        ; RETURN TO CALLER
                                     0475  2398
                                     0475  2399 READTIME:                                  ; SOLICIT TIME
                          59    D4    0475  2400         CLRL    R9                          ; CLEAR A FLAG
       58    00000000'GF    32    0477  2401         CVTWL   G^SGN$GW_TPWAIT,R8          ; PICK UP TIMEOUT WAIT INTERVAL
                          14    14    047E  2402         BGTR    8$                          ; POSITIVE, WAIT THAT PERIOD ONCE
                          0D    19    0480  2403         BLSS    7$                          ; NEGATIVE IS WAIT FOREVER
                                     0482  2404 6$:
 50    00000000'GF    01    C1    0482  2406         ADDL3   #1,G^EXE$GL_TODR,R0         ; ZERO, SET TIME-OF-DAY CLOCK TO
                                     048A  2408
                                     048A  2412
                                     048A  2416
                   1B    50    DA    048A  2418         MTPR    R0,#PR730$_TODR             ;   KNOWN VALUE + 10 MSEC AND FINISH UP
                                     048D  2420
                                     048D  2424
                         E0    11    048D  2426         BRB     5$                          ;
                                     048F  2428
                                     048F  2433
                   58    14    D0    048F  2434 7$:       MOVL    #20,R8                      ; STARTING WAIT
                         59    D6    0492  2435         INCL    R9                          ; NEGATIVE - WAIT FOREVER
                                     0494  2436 8$:       $ASSIGN_S       TTNAME(R6),TTCHAN(R6) ; AND ASSIGN TO INPUT DEVICE
                      DD 50    E9    04A2  2437         BLBC    R0,6$                       ; ERROR - FALL BACK TO STORED TIME
             52    FF60 CF    9E    04A5  2438 10$:      MOVAB   W^TIMEPROMPT,R2             ; GET ADDRESS OF PROMPT STRING
                   53    82    9A    04AA  2439         MOVZBL  (R2)+,R3                    ; AND LENGTH
                                     04AD  2440         $QIOW_S #0,W^TTCHAN(R6),-           ; PROMPT AND READ TIME
                                     04AD  2441                 #<IO$_READPROMPT!IO$M_PURGE!IO$M_TIMED!IO$M_CVTLOW>,-
```

```
                                04AD  2442                TMPDESC(R6),,-                  ;  I/O STATUS BLOCK , NO AST OR PARAM
                                04AD  2443                LINBUF(R6),#LINBUFSIZ,-         ;  BUFFER ADDRESS AND SIZE
                                04AD  2444                R8,#0,-                         ;  TIME OUT
                                04AD  2445                R2,R3                           ;  PROMPT ADDRESS AND SIZE
                      AD 50     E9  04D2  2446          BLBC    R0,6$                     ; ERROR - FALL BACK TO STORED TIME
                54    0C A6     7D  04D5  2447          MOVQ    TMPDESC(R6),R4            ; GET COMPLETION STATUS
                      0D 54     E8  04D9  2448          BLBS    R4,20$                    ; CONTINUE IF SUCCESSFUL READ
                      A3 59     E9  04DC  2449          BLBC    R9,6$                     ; FAILED ON ONE-TIME READ, RETURN
             58    01 A848      9E  04DF  2450          MOVAB   1(R8)[R8],R8              ; (2 * TIMEOUT) + 1
                   58   58      3C  04E4  2451          MOVZWL  R8,R8                     ; BOUND TIMEOUT
                        BC      11  04E7  2452          BRB     10$                       ; TRY AGAIN FOR TIME
                                04E9  2453 20$:                                           ; SOMETHING WAS INPUT
          0C A6      0E A6      3C  04E9  2454          MOVZWL  TMPDESC+2(R6),TMPDESC(R6) ; FORM DESCRIPTOR FOR BUFFER
          10 A6      1C A6      9E  04EE  2455          MOVAB   LINBUF(R6),TMPDESC+4(R6)  ; SET DESCRIPTOR ADDRESS
                                04F3  2456          $BINTIM_S TMPDESC(R6),INTIME(R6)  ; CONVERT TO BINARY TIME
                      05 50     E9  0500  2457          BLBC    R0,89$                    ; INVALID TIME
                      18 A6     D5  0503  2458          TSTL    INTIME+4(R6)              ; CHECK FOR DELTA TIME
                         2A     14  0506  2459          BGTR    100$                      ; BRANCH IF NOT - OK
                                0508  2460 89$:                                          ; INVALID TIME VALUE INPUT
             52    FEEB CF     9E  0508  2461          MOVAB   W^TIMERR,R2               ; ADDRESS OF ERROR MESSAGE
                   53   82     9A  050D  2462          MOVZBL  (R2)+,R3                  ; GET STRING LENGTH
                                0510  2463          $QIOW_S #0,TTCHAN(R6),-           ; GIVE ERROR MESSAGE
                                0510  2464                  #IO$_WRITEVBLK,-
                                0510  2465                  ,,,-                      ;  NO I/O STATUS,AST OR AST PARAM
                                0510  2466                  (R2),R3 ,-                ;  BUFFER ADDRESS, LENGTH
                                0510  2467                  #0,#32                    ;  SET CARRIAGE CONTROL TO CR/LF
                         FF73   31  052F  2468          BRW     10$                       ; AND TRY AGAIN
                                0532  2469 100$:                                         ; EXIT
                                0532  2470          $DASSGN_S TTCHAN(R6)               ; DE-ASSIGN TERMINAL CHANNEL
                      14 A6     7F  053C  2471 200$:    PUSHAQ  INTIME(R6)              ; SET NEW SYSTEM TIME
          00000000'GF   01     FB  053F  2472          CALLS   #1,G^EXE$SETIME_INT       ; USE TODR CLOCK TO SET SYSTEM TIME
00000000'GF   00000000'GF     7D  0546  2473          MOVQ    G^EXE$GQ_TODCBASE,G^EXE$GQ_BOOTTIME ; SAVE BOOT TIME
                   5E   30     C0  0551  2474          ADDL    #12*4,SP                  ; CLEAN OFF SCRATCH STORAGE
                   077C 8F     BA  0554  2475          POPR    #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; RESTORE REGISTERS
                                0558  2476
                                0558  2477 ;
                                0558  2478 ; Fall through into the deallocate logic.
                                0558  2479 ;
                                0558  2480 ;       RSB                               ; *** This goes in if another piece of
                                0558  2481                                          ; *** initialization code is added that
                                0558  2482                                          ; *** is executed after EXE$INI_TIMWAIT.
                                0558  2483          .DISABLE LSB
                                0558  2484
```

```
                                0558  2486 DEAL_INIT_CODE:                         ; DEALLOCATE THE INITIALIZATION CODE
                                0558  2487 ;
                                0558  2488 ; It is the duty of the last-executed, loadable initialization
                                0558  2489 ; routine to make itself and all other such routines disappear, i.e.,
                                0558  2490 ; release the space they occupy to non-paged pool.  Each routine's vector
                                0558  2491 ; must be disconnected, e.g., be made to point to the symbol, EXE$LOAD_ERROR.
                                0558  2492 ;
                                0558  2493 ; NOTE:  This means that new initialization routines should be added
                                0558  2494 ;       to this module in a particular order, not necessarily at the
                                0558  2495 ;       end of the module!
                                0558  2496 ;
                                0558  2497         .ENABLE LSB
                      7E   52   7D  0558  2498         MOVQ    R2,-(SP)             ; Save some registers
                                055B  2499
                                055B  2500 ;
                                055B  2501 ; First find the vectors that point to these initialization routines
                                055B  2502 ; and reset them to point to EXE$LOAD_ERROR.
                                055B  2503 ;
               50    0000'CF   9E  055B  2504         MOVAB   W^SYSL$BEGIN,R0      ; Compute bounds of releasable piece:
          51   50  00000000'8F   C1  0560  2505         ADDL3   #<STAY_HEADER-SYSL$BEGIN>,R0,R1 ; starting and ending addresses.
               52  00000000'GF   9E  0568  2506         MOVAB   G^EXE$AL_LOAVEC,R2   ; Get starting address of vectors.
               53  00000000'GF   9E  056F  2507         MOVAB   G^EXE$LOAD_ERROR,R3  ; Get end of vectors.
                 9F17 8F    62   B1  0576  2508 10$:     CMPW    (R2),#^X9F17         ; Is this JMP @# ?
                                18   13  057B  2509         BEQL    30$                  ; Br if yes, skip past it.
                    80 8F   03   A2  057D  2510         CMPB    3(R2),#^X80          ; Is this a system space address
                                16   12  0582  2511         BNEQ    40$                  ; Br if no, assume it's a HALT instr.
                           50   62   D1  0584  2512         CMPL    (R2),R0              ; Is address before the releasable
                                0C   1F  0587  2513         BLSSU   20$                  ;  piece of memory?  Br on yes.
                           51   62   D1  0589  2514         CMPL    (R2),R1              ; Is address after the releasable
                                07   1A  058C  2515         BGTRU   20$                  ;  piece of memory?  Br on yes.
          62  00000000'GF   9E  058E  2516         MOVAB   G^EXE$LOAD_ERROR (R2)  ; Reset this vector.
                      52   02   C0  0595  2517 20$:     ADDL    #2,R2                ; Point past this vector.
                           52   D6  0598  2518 30$:     INCL    R2                   ; Come here to point past JMP @#.
                           52   D6  059A  2519 40$:     INCL    R2                   ; Come here to point past HALT.
                      53   52   D1  059C  2520         CMPL    R2,R3                ; Past the end of the vectors?
                           D5   1F  059F  2521         BLSSU   10$                  ; Keep searching vectors.
                                05A1  2522 ;
                                05A1  2523 ; Now release the memory to non-paged pool.
                                05A1  2524 ;
               50    0000'CF   9E  05A1  2525         MOVAB   W^SYSL$BEGIN,R0      ; Point to start of module
               51    0000'8F   3C  05A6  2526         MOVZWL  #<STAY_HEADER-SYSL$BEGIN>,R1 ; Length to vaporize
                        FA5E'   31  05AB  2527         BRW     50$                  ; Br to code that is not released.
                                05AE  2528
                                00000000  2529         .PSECT  $$$INIT__END,PAGE    ; 'PAGE' SINCE 16-BYTE ALIGN IS NOT
                                0000  2530
                                0000  2531 STAY_HEADER:
                00000000 00000000  0000  2532         .LONG   0,0
                               0000'  0008  2533         .WORD   <SYSL$END-STAY_HEADER>
                                  62   000A  2534         .BYTE   DYN$C_LOADCODE
                                  00   000B  2535         .BYTE   0
                                000C  2536
           00000000'9F    16   000C  2537 50$:     JSB     @#EXE$DEANONPGDSIZ   ; Just the smile on the Chesire cat
                      52   8E   7D  0012  2538         MOVQ    (SP)+,R2             ; Restore
                           05   0015  2539         RSB                          ; Return.
                                0016  2540
                                0016  2541         .DISABLE LSB
                                0016  2542         .END
```

```
$$$VMSDEFINED                   =  00000001
$$T1                            =  00000001
ADAPTERS                           00000000 R      02
ADP$B_TYPE                      =  0000000A
ADP$C_UBAADPLEN                 =  00000258
ADP$L_CRB                       =  00000010
ADP$L_CSR                       =  00000000
ADP$L_DPQFL                     =  00000014
ADP$L_LINK                      =  00000004
ADP$L_MRACTMDRS                 =  0000005C
ADP$L_MRQFL                     =  00000030
ADP$L_VECTOR                    =  00000010
ADP$W_ADPTYPE                   =  0000000E
ADP$W_DPBITMAP                  =  00000060
ADP$W_MRFFENCE                  =  0000015C
ADP$W_MRFREGARY                 =  0000015E
ADP$W_MRNFENCE                  =  00000062
ADP$W_MRNREGARY                 =  00000064
ADP$W_SIZE                      =  00000008
ADP$W_TR                        =  0000000C
ADP$W_UMR_DIS                   =  00000256
ADPLINK                            ********   X    09
ALONPAGD                           000002D5 R      09
ATS_UBA                         =  00000001
BADUMR                             00000301 R      08
BI_BUS_CODE                     =  80000000
BI_CPU                          =  00000000
BI_CSR_LEN                      =  00000002
BI_LIKE                         =  00000000
BLD_CRB                            000002F3 R      09
BOO$GB_SYSTEMID                    ********   X    09
BOO$GL_SPTFREH                     ********   X    09
BOO$GL_SPTFREL                     ********   X    09
BOOTVECTOR                         00000000 R      08
BTD$K_CONSOLE                   =  00000040
BTD$K_UDA                       =  00000011
BUS_CODE_OFFSET                 =  FFFFFFFC
BUS_CSR_LEN                        00000004 R      08
CONFIG_IOSPACE                     0000005F R      09
CONFREG                            00000098 R      08
CONFREGL                           000001D8 R      08
CPU_ADPSIZE                        0000000D R      08
CPU_TYPE                        =  00000003
CR                              =  0000000D
CRB$L_INTD                      =  00000024
CRB$L_INTD2                     =  00000048
CREATE_ARRAYS                      0000011A R      09
CSR_LEN_OFFSET                  =  FFFFFFFB
DDB$T_NAME                      =  00000014
DEAL_INIT_CODE                     00000558 R      09
DIRECT_VEC_NODE_CNT                00000009 R      08
DYN$C_ADP                       =  00000001
DYN$C_CONF                      =  00000007
DYN$C_IDB                       =  00000009
DYN$C_INIT                      =  00000063
DYN$C_LOADCODE                  =  00000062
END_NEXUS                          00000115 R      09
ERROR_HALT                         00000199 R      09
ERROR_HALT_1                       0000019E R      09
EXE$AL_LOADVEC                     ********     X  09
EXE$DEANONPGDSIZ                   ********     X  0A
EXE$GL_CONFREG                     ********     X  09
EXE$GL_CONFREGL                    ********     X  09
EXE$GL_FLAGS                       ********     X  09
EXE$GL_NUMNEXUS                    ********     X  09
EXE$GL_RPB                         ********     X  09
EXE$GL_SCB                         ********     X  09
EXE$GL_TENUSEC                     ********     X  09
EXE$GL_TODR                        ********     X  09
EXE$GL_UBDELAY                     ********     X  09
EXE$GQ_BOOTTIME                    ********     X  09
EXE$GQ_TODCBASE                    ********     X  09
EXE$INIT_TODR                      0000043D RG     09
EXE$INI_TIMWAIT                    0000038B RG     09
EXE$LOAD_ERROR                     ********   X    09
EXE$MCHK_PRTCT                     ********   X    09
EXE$OUTZSTRING                     ********   X    09
EXE$SETIME_INT                     ********   X    09
EXE$TEST_CSR                       ********   X    09
EXE$V_SETTIME                      ********   X    09
FILL_CRB                           00000300 R      09
GET_GEN_TYPE                       000000B9 R      09
GET_TYPE                           000000A0 R      09
IDB$B_TYPE                      =  0000000A
IDB$C_LENGTH                    =  00000038
IDB$L_ADP                       =  00000014
IDB$L_CSR                       =  00000000
IDB$W_SIZE                      =  00000008
INI$ALLOC_CRB                      ********     X  09
INI$ALONONPAGED                    ********     X  09
INI$CIADP                          000002DB R      09
INI$CONSOLE                        000002DC RG     09
INI$DRADP                          000002DB R      09
INI$IOMAP                          00000000 RG     09
INI$KDZ11                          000002DB R      09
INI$MBADP                          000002DB R      09
INI$MPMADP                         ********     X  06
INI$UBADP                          000001C4 R      09
INI$UBSPACE                        000001A7 R      09
INIT_ROUTINES                      00000000 R      06
INTIME                          =  00000014
IO$M_CVTLOW                        ********     X  09
IO$M_PURGE                         ********     X  09
IO$M_TIMED                         ********     X  09
IO$_READPROMPT                     ********     X  09
IO$_WRITEVBLK                      ********     X  09
IO730$AL_IOBASE                 =  00F20000
IO730$AL_PERNEX                 =  00002000
IO730$AL_UBOSP                  =  00FC0000
LF                              =  0000000A
LINBUF                          =  0000001C
LINBUFSIZ                       =  00000014
MAP_NEXUS                          000000F9 R      09
MAP_PAGES                          00000173 R      09
```

```
MAXNEXUS                =  00000040
MCHK$M_LOG              =  00000001
MCHK$M_NEXM             =  00000004
MMG$GL_SBICONF             ********    X   09
MMG$GL_SPTBASE             ********    X   09
NDT$_BOA                =  80000102
NDT$_CI                 =  00000038
NDT$_DR32               =  00000030
NDT$_KDZ11              =  80000105
NDT$_MB                 =  00000020
NDT$_MEM1664NI          =  00000012
NDT$_MEM16I             =  00000011
NDT$_MEM16NI            =  00000010
NDT$_MEM256EIL          =  00000071
NDT$_MEM256EIU          =  00000073
NDT$_MEM256I            =  00000074
NDT$_MEM256NIL          =  00000070
NDT$_MEM256NIU          =  00000072
NDT$_MEM4I              =  00000009
NDT$_MEM4NI             =  00000008
NDT$_MEM64EIL           =  00000069
NDT$_MEM64EIU           =  0000006B
NDT$_MEM64I             =  0000006C
NDT$_MEM64NIL           =  00000068
NDT$_MEM64NIU           =  0000006A
NDT$_MPM0               =  00000040
NDT$_MPM1               =  00000041
NDT$_MPM2               =  00000042
NDT$_MPM3               =  00000043
NDT$_SCORMEM            =  80000001
NDT$_UB0                =  00000028
NDT$_UB1                =  00000029
NDT$_UB2                =  0000002A
NDT$_UB3                =  0000002B
NEXUSDESC                  00000014 R       08
NOSPT                      000002D8 R       08
NPROMPT                 =  00000033
NUMUBAVEC               =  00000080
NUM_PAGES                  00000000 R       04
NXT_NEXUS                  0000006B R       09
PA                      =  00F40000
PR$_CSTD                =  0000001F
PR$_ICCS                =  00000018
PR$_SID_TYP730          =  00000003
PR$_SID_TYP750          =  00000002
PR$_SID_TYP780          =  00000001
PR$_SID_TYP790          =  00000004
PR$_SID_TYP8NN          =  00000006
PR$_SID_TYP8SS          =  00000005
PR$_SID_TYPUV1          =  00000007
PR$_TBIS                =  0000003A
PR730$_ICR              =  0000001A
PR730$_NICR             =  00000019
PR730$_TODR             =  0000001B
PTE$C_RW                =  10000000
PTE$M_VALID             =  80000000
READTIME                   00000475 R       09
RPB$B_DEVTYP            =  00000066
RPB$L_ADPPHY            =  0000005C
RPB$L_ADPVIR            =  00000060
RPB$L_BOOTR1            =  00000020
RPB$L_CSRPHY            =  00000054
RPB$L_CSRVIR            =  00000058
RPB$W_ROUBVEC           =  0000001E
SBICONF                    000000D8 R       08
SBI_BUS_CODE            =  00000000
SBI_CPU                 =  00000000
SBI_CSR_LEN             =  00000001
SBI_LIKE                =  00000001
SGN$GW_TPWAIT              ********    X   09
STAY_HEADER                00000000 R       0A
SW_BUS_CODE                00000005 R       08
SYS$ASSIGN                 ********  GX    09
SYS$BINTIM                 ********  GX    09
SYS$DASSGN                 ********  GX    09
SYS$QIOW                   ********  GX    09
SYSL$BEGIN                 ********    X   09
SYSL$END                   ********    X   0A
TERM_NAMADR                000003F3 R       09
TERM_NAMSIZ             =  00000004
TEST_NEXUS                 00000071 R       09
TIMEPROMPT                 00000409 R       09
TIMERR                     000003F7 R       09
TMPDESC                 =  0000000C
TTCHAN                  =  00000000
TTNAME                  =  00000004
UBA$INITIAL                ********    X   09
UBA$INTO                   ********    X   09
UBA$L_MAP               =  00000800
UBA$UNEXINT                ********    X   09
UCB$W_UNIT              =  00000054
VA$M_SYSTEM             =  80000000
VEC$L_ADP               =  00000014
VEC$L_IDB               =  00000008
```

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	00000000 (0.)	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
$$$INIT$DATA	00000333 (819.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$INIT$CODE	000005AE (1454.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD
$$$INIT__END	00000016 (22.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.05	00:00:00.41
Command processing	108	00:00:00.46	00:00:02.29
Pass 1	524	00:00:13.26	00:00:50.33
Symbol table sort	0	00:00:01.68	00:00:08.24
Pass 2	257	00:00:03.91	00:00:19.16
Symbol table output	25	00:00:00.12	00:00:00.12
Psect synopsis output	4	00:00:00.04	00.00:00.19
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	951	00:00:19.52	00:01:20.74

The working set limit was 1800 pages.
136142 bytes (266 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1619 non-local and 34 local symbols.
2546 source lines were read in Pass 1, producing 37 object records in Pass 2.
45 pages of virtual memory were used to define 43 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	22
_255DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	36

1782 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:INIADP730/OBJ=OBJ$:INIADP730 MSRC$:CPUSW730/UPDATE=(ENH$:CPUSW730)+MSRC$:INIADP/UPDATE=(ENH$:INIADP)+EXECML$/LIB
```

INIADP750
LIS

INIADP730
LIS

INIADP780
LIS

INIADPUVI
LIS

INIADP790
LIS